

# Equational Security Proofs of Oblivious Transfer Protocols\*

Baiyu Li<sup>†</sup>

Daniele Micciancio<sup>‡</sup>

January 9, 2018

## Abstract

We exemplify and evaluate the use of the equational framework of Micciancio and Tessaro (ITCS 2013) by analyzing 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.

## 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 [4] to present security definitions and proofs of standard cryptographic functions. But even the detailed framework of [4] 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 [5] (or one of its many technical variants [1, 2, 7, 9, 16, 17, 21],) 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<sup>1</sup> makes it rather hard to use in practice, when specifying and analyzing concrete protocols.<sup>2</sup> These limitations are widely recognized, and have prompted researchers to explore several variants, simplifications and specialization of the general UC security model [6, 19, 22, 30]. In this perspective, a very interesting line of work is represented by the “abstract cryptography” framework of [25], which calls for an axiomatic approach to the description and analysis of cryptographic primitives/protocols, and the “constructive cryptography” [24] and “equational security” [26] frameworks, which can be thought of as logical models of the axioms put forward in [25].

---

\*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.

<sup>†</sup>University of California, San Diego, USA. E-mail: [baiyu@cs.ucsd.edu](mailto:baiyu@cs.ucsd.edu)

<sup>‡</sup>University of California, San Diego, USA. E-mail: [daniele@cs.ucsd.edu](mailto:daniele@cs.ucsd.edu)

<sup>1</sup>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.

<sup>2</sup>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.

In this work we examine the equational security framework of [26], 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 [12], 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 [12, 27], and an important (in fact, complete) building block for the construction of arbitrary security protocols [11, 14, 18, 20, 23, 32]. 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 [10].

We remark that the primary goal of our work is to exemplify and evaluate the usability of the equational security framework of [26], 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 [10] can be proved secure according to *neither* the classic *nor* the revised OT definitions considered above.

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  $\text{OT}((m_0, m_1), b) = m_b$  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  $\kappa$ ,
2. Use these seeds as keys to encrypt the two messages using a private-key encryption scheme,<sup>3</sup> and send both ciphertexts to the receiver over a standard (authenticated, but insecure to eavesdropping) communication channel.

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

<sup>3</sup>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  $m_i$ .

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 her 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  $\text{OT}((m_0, m_1), b) = (f(b), m_b)$ , 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 Oblivious Transfer in the Random Oracle Model.

In [10], Chou and Orlandi propose a new OT protocol achieving UC security in the random oracle model [3]. 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 insecure against both corrupted senders and corrupted receivers. For the case of corrupted senders, the failure of simulation is due to the fact that in a real protocol execution the sender learns if and when the receiver provides her selection bit  $b$ , which is not available to the simulator. For the case of corrupted receivers, the problem is that in a real protocol execution the receiver can delay its random oracle query until after seeing the sender’s ciphertexts, but in the ideal protocol execution, if the simulator has to output the ciphertexts before seeing the receiver’s random oracle query, then it must be able to guess an external random bit correctly before seeing any inputs, which is impossible to achieve with high probability. However, unlike the case of the OT length extension transformation, these problems are not the only weakness of the protocol, and security cannot be proved by switching to the revised OT definition given above.

## 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 wonder 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 [10], our primary aim is to illustrate the use of the equational framework to analyze a protocol in the random oracle model. We are certainly not concerned about whether the protocol is making a “morally correct” use of the random oracle, or if a “global” random oracle definition [8] should be used instead. We simply use the equational framework to model and analyze the protocol as described in the original paper [10]. Our analysis shows that the protocol is not secure according to the original OT definition (seemingly used in [10],) but even using a revised OT definition still does not allow to prove security in the equational framework, in the technical sense that for any simulator (expressible in the equational framework) there is an environment that distinguishes between the real and the ideal systems.

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. Within the context of our work, the equational framework of [26] 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 us to easily explore different definitional variants and put them to good use to spot potential bugs in cryptographic protocols. Exploring the applicability of abstract frameworks along the lines of [24–26] 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 Notation

In this section we review the equational framework of [26], and define the notation used in this paper. For completeness, we will first recall some background on the (standard) theory that gives a precise meaning to systems of equations as used in [26] 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 computational models and notational conventions.

### 2.1 Domain Theoretical Background

The mathematical foundation of the equational framework is provided by domain theory. Here we give just enough background to describe the systems studied in this paper, and refer the reader to [15, 28, 29] 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 in this paper are complete partial orders (CPOs), i.e., any (possibly empty) chain  $x_1 < x_2 < \dots$  has a least upper bound  $\sup_i x_i$  in  $X$ . The Cartesian product  $X \times Y$  of two CPOs is also a CPO with the component-wise partial order  $(x_1, y_1) \leq (x_2, y_2) \iff x_1 \leq x_2 \wedge y_1 \leq y_2$ . 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  $C$ . A set is *open* if its complement is closed. 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  $[X \rightarrow Y]$ . 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  $\perp = \sup \emptyset$ , called the *bottom*, which satisfies  $\perp \leq x$  for all  $x \in X$ .

For any set  $A$ , we can always construct a *flat* CPO  $A_\perp = A \cup \{\perp\}$  by including a unique bottom element  $\perp$ . The partial order in  $A_\perp$  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  $A_\perp$ . Functions  $f: A \rightarrow B$  between sets can be lifted to strict functions  $f: A_\perp \rightarrow B_\perp$  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.

For any CPO  $X$ , every continuous functions  $f: X \rightarrow X$  admits a *least fixed point*, denoted as  $\text{fix}(f)$ , 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  $\perp, f(\perp), f^2(\perp), \dots$ . A system of mutually recursive equations can be solved via least fixed point computation. Such a solution describes the final outputs of interactive computations between nodes in a network. By Bekič’s theorem [31], the least fixed point of such a system can be computed one component at a time: For example, the system  $(x, y) = (f(x, y), g(x, y))$  can be solved by computing first  $\hat{x} = \text{fix}(\lambda x. f(x, \text{fix}(\lambda y. g(x, y))))$  and then  $\hat{y} = \text{fix}(\lambda y. g(\hat{x}, y))$ , and the least fixed point of the system is  $(\hat{x}, \hat{y})$ .

We can also model probabilistic behaviors in equational framework. A *probability distribution* on a CPO  $X$  is a function  $p: X \rightarrow [0, 1]$  such that<sup>4</sup>  $p(A) + p(B) = p(A \cup B)$  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*.<sup>5</sup> 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  $p \leq q$  (in  $D(X)$ ) if and only if  $p(A) \leq q(A)$  for any open subset  $A \subseteq X$ . *Probabilistic functions* are just (continuous) functions between sets of distributions with respect to this ordering relation.

## 2.2 Computational Model

We recall that the execution model of [26] 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 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  $M^* = \{(m_1, \dots, m_k) : k \geq 0, \forall i. m_i \in M\}$  of messages from a ground set  $M$ , ordered according to the prefix partial order. By combining the set  $M^\infty$  of infinite sequences of messages from  $M$ , we get a CPO  $M^\omega$ . Another common example, 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 bottom 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  $F: X \rightarrow Y$  from the incoming channels to the outgoing channels, satisfying the natural monotonicity requirement that for any  $h_1 \leq h_2$  in  $X$ , we have  $F(h_1) \leq F(h_2)$  in  $Y$ . Informally, monotonicity captures the intuition that once a party transmits a message, it cannot go back in time and take it back. A probabilistic computational unit can be modeled as a function of type  $X \rightarrow \mathcal{D}(Y)$ , where  $\mathcal{D}$  is the probability monad. We may also consider units with limited computational power in the monadic approach, which is an important extension to the equational framework. However, as all the protocols considered in this paper run in constant time, for simplicity we do not formalize computational cost (e.g. running time, space, etc) in our analysis.

Computation units can be connected to a communication network  $N$  to form a system, where  $N$  is also a monotone function. Such a system is again a monotone function mapping external input channels to external output channels of all the units, and it is modeled as a composition of functions describing all the units and the network. Syntactically, function compositions can be simplified by substitution and variable elimination, and, when recursive definition is involved, by using fixed point operations. In general, we use the notation  $(F|G)$  to denote the system composed by functions  $F$  and  $G$ , where the composition operator “ $|$ ” is associative. The main advantage of the equational framework is that it has a mathematically clean and well defined semantics, where functions can be completely described by mathematical equations (specifying the relation between the input and the output of the units), and composition simply combines equations together. The equational approach also provides a simple and precise way to reason about relations between systems. For example, equivalent components (in the sense of having equivalent equations) can be replaced by

<sup>4</sup>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.

<sup>5</sup>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.

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.

### 2.3 Security

The definition of security in the equational framework follows the well-accepted simulation-based security paradigm. In this paper we consider only OT protocols, which are two-party protocols between a sender program and a receiver program. An ideal functionality  $F$  is a function from  $X = X_0 \times X_1$  to  $Y = Y_0 \times Y_1$ , where  $X_i$  ( $Y_i$ ) is the external input (output) of party  $P_i$ . An environment is a function  $\text{Env} : Y^\omega \rightarrow X^\omega \times \{\top\}_\perp$  such that it takes as input the output history (as a sequence of evolving messages) of a system, and it produces a sequence of evolving inputs to the system and a decision bit  $t$ . Here a sequence of messages  $x_0x_1\dots$  over  $X$  is *evolving* if  $x_i \leq_X x_{i+1}$  for all  $i$ , where  $x_i \in X$  and  $\leq_X$  is the partial order of  $X$ . An experiment between an environment  $\text{Env}$  and a system  $S$ , is executed as follows:  $\text{Env}$  generates an evolving sequence of input  $x_0x_1\dots$  to  $S$  such that  $S$  outputs  $y_i = S(x_i)$  for each  $x_i$ ,  $\text{Env}$  takes as input the sequence  $y_0y_1\dots$ , and it eventually produces an external decision bit  $t$ . We write  $\text{Env}[S]$  for the output (distribution)  $t$  of this experiment. When all parties are honest, the real system is a composition of the network  $N$  and two parties  $P_0$  and  $P_1$ , denoted as  $(P_0|P_1|N)$ , and it must be equivalent to the ideal functionality  $F$ . When a party  $P_i$  is *corrupted*, the real system is composed by the remaining honest party and the network, and the ideal system is composed by  $F$  and a monotone simulator  $\text{Sim}$ . We say that a protocol is secure against the corruption of  $P_i$  if there exists a simulator  $\text{Sim}$  as a computation unit such that the systems  $(N|P_{(1-i)})$  and  $(\text{Sim}|F)$  are indistinguishable by any environment that produces a decision bit in polynomial time in the output length of the system and the security parameter.

A distinctive feature of the equational framework is the ability to specify fully asynchronous systems. An environment might not provide a complete input to a system at once, that is, the input to certain channels might be  $\perp$ . So we must consider such asynchronous environments when analyzing the security of a protocol.

It is an very interesting and important open question to compare the equational framework (with the full extension of computational security) with the UC model and its variants (for example, the simplified models of [6, 30].) Due to space limitation, we do not address such a problem in the current paper and we will study it in future work.

### 2.4 Notation

Now we briefly mention our notational conventions. In this paper we mainly use flat CPOs, i.e., partially ordered sets  $\mathbb{X}$  with a bottom element  $\perp \in \mathbb{X}$  such that  $x_1 \leq x_2$  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  $\mathbb{X}^{\times 2} = \{(x, y) : x, y \in \mathbb{X}, x \neq \perp, y \neq \perp\}_\perp$  for the CPO of *strict* pairs over  $\mathbb{X}$  and  $\perp$ . 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  $x, y \neq \perp$ . For any set  $A$ , we write  $x \leftarrow A_\perp$  for the operation of selecting an element  $x \neq \perp$  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  $\odot$ ,

$$z = \langle z[0], z[1] \rangle \tag{1}$$

$$f(\langle x_0, x_1 \rangle[i]) = \langle f(x_0), f(x_1) \rangle[i] \tag{2}$$

$$\langle x_0, x_1 \rangle[i] \odot \langle y_0, y_1 \rangle[i] = \langle x_0 \odot y_0, x_1 \odot y_1 \rangle[i] \tag{3}$$

The followings are 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  $\mathbb{M}_n = \{0, 1\}_\perp^n$  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 notation introduced so far, we can describe the ideal (1-out-of-2) OT functionality by the equations in Fig. 1. (Notice that this functionality is parameterized by a message space  $\mathbb{M}$ .) The first line specifies the names of the functionality ( $\text{OT}$ ), 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.

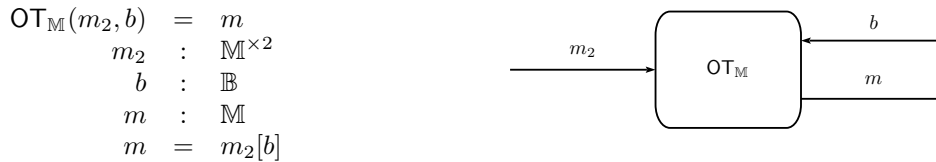


Figure 1: A naive OT functionality: the receiver gets the selected message  $m = m_2[b]$ , and the sender does not get anything at all.

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.

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.

### 3 Oblivious Transfer Length Extension: a first attempt

As an abbreviation, when the message space  $\mathbb{M} = \{0, 1\}_\perp^n$  is the set of all bitstrings of length  $n$ , we write  $\text{OT}_n$  instead of  $\text{OT}_{\mathbb{M}}$ . Consider the following OT *length extension* problem: given an  $\text{OT}_n$  channel for messages of some (sufficiently large) length  $n$ , build an OT functionality  $\text{OT}_\ell$  for messages of length  $\ell > n$ . The goal is to implement  $\text{OT}_\ell$  making a single use of the basic  $\text{OT}_n$  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,<sup>6</sup> we model the communication channel as a functionality  $\text{Net}_\ell$  that copies its input of length  $\ell$  to the output of the same length:



The OT length extension protocol is specified by a pair of **Sender** and **Receiver** programs, which are interconnected (using the  $\text{OT}_n$  and  $\text{Net}_{2\ell}$  functionalities) as shown in Fig. 2. Notice how the external input/output interface of the system corresponding to a real execution of the protocol in Fig. 2 is the same as that of the ideal functionality  $\text{OT}_\ell(m'_2, b') = m'$  the protocol is trying to implement.

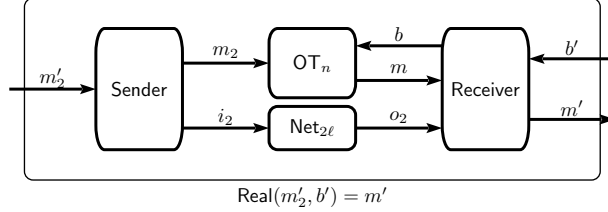


Figure 2: A real execution of a candidate OT length extension protocol. The protocol consists of a **Sender** and a **Receiver** programs that communicate using  $\text{OT}_n$  and  $\text{Net}_{2\ell}$  functionalities.

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

$$\begin{aligned} \text{Sender}(m'_2) &= (m_2, i_2) \\ m_2 &\leftarrow \mathbb{M}_n^{2} \\ i_2[0] &= m'_2[0] \oplus \mathcal{G}(m_2[0]) \\ i_2[1] &= m'_2[1] \oplus \mathcal{G}(m_2[1]) \end{aligned} \qquad \begin{aligned} \text{Receiver}(m, o_2, b') &= (b, m') \\ b &= b' \\ m' &= o_2[b'] \oplus \mathcal{G}(m) \end{aligned}$$



In words, these programs work as follows:

- The sender picks a pair  $m_2$  of two random seeds, and passes (one of) them to the receiver using the  $\text{OT}_n$  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  $\text{Net}_{2\ell}$ .
- The receiver selects one of the two seeds from the  $\text{OT}_n$  functionality, expands it using the pseudorandom generator, and uses the result to “unmask” the corresponding message from  $\text{Net}_{2\ell}$ .

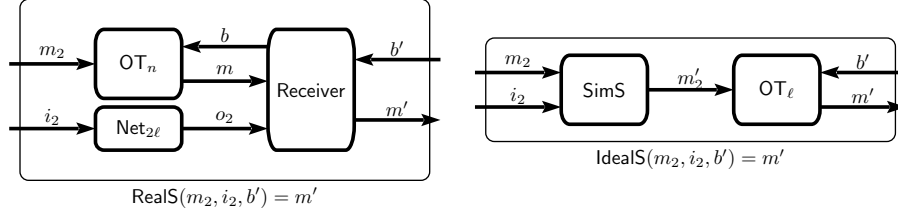
It is easy to show that the protocol is correct, in the sense that combining the equations of  $\text{OT}_n$ ,  $\text{Net}_{2\ell}$ , **Sender** and **Receiver** as shown in Fig. 2 results in a system  $\text{Real}(m'_2, b') = m'$  that is perfectly equivalent

<sup>6</sup>This corresponds to a perfectly secure communication channel. More complex/realistic communication channels are discussed at the end of this section.

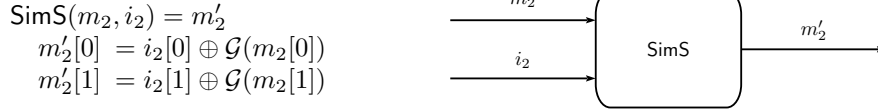


to the defining equation  $m' = m'_2[b']$  of the ideal functionality  $\text{OT}_\ell$ . Intuitively, the protocol also seems secure because only one of the two seeds can be recovered by the receiver, and the unselected message is protected by an unpredictable pseudorandom pad. But security of cryptographic 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 Fig. 2. Following the simulation paradigm, security requires exhibiting an efficient simulator program **SimS** (interacting, as a sender, with the ideal functionality  $\text{OT}_\ell$ ) such that the following real and ideal systems are computationally indistinguishable:

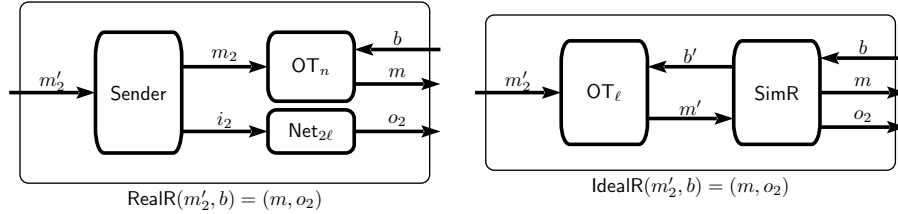


Security is easily proved by defining the following simulator:



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:

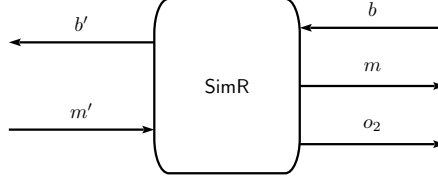


Of course, this time we can only aim at proving computational security, i.e., coming up with a simulator such that **RealR** and **IdealR** are computationally indistinguishable. We begin by writing down explicitly the equations that define the real system execution. Combining the equations for **Sender**,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

$$\begin{aligned} \text{RealR}(m'_2, b) &= (m, o_2) \\ m_2 &\leftarrow \mathbb{M}_n^{\times 2} \\ o_2[0] &= m'_2[0] \oplus \mathcal{G}(m_2[0]) \\ o_2[1] &= m'_2[1] \oplus \mathcal{G}(m_2[1]) \\ m &= m_2[b] \end{aligned}$$

So, the simulator may proceed by picking  $m_0, m_1$  at random on its own, and set  $m = m_2[b]$  just as in the real execution. However, the simulator cannot compute  $o_2$  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:

$$\begin{aligned}
\text{SimR}(m', b) &= (b', m, o_2) \\
b' &= b \\
m_2 &\leftarrow \mathbb{M}_n^{\times 2} \\
m &= m_2[b] \\
o_2[0] &= m' \oplus \mathcal{G}(m_2[0]) \\
o_2[1] &= m' \oplus \mathcal{G}(m_2[1])
\end{aligned}$$



Combining  $\text{SimR}$  with  $\text{OT}_\ell$  results in the ideal system:

$$\begin{aligned}
\text{IdealR}(m'_2, b) &= (m, o_2) \\
m_2 &\leftarrow \mathbb{M}_n^{\times 2} \\
o_2[0] &= m'_2[b] \oplus \mathcal{G}(m_2[0]) \\
o_2[1] &= m'_2[b] \oplus \mathcal{G}(m_2[1]) \\
m &= m_2[b]
\end{aligned}$$

As expected, the two systems  $\text{IdealR}$ ,  $\text{RealR}$  are indistinguishable for both  $b = 0$  and  $b = 1$ . For example,  $\text{RealR}(m'_2, 0)$  and  $\text{IdealR}(m'_2, 0)$  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  $\text{RealR}$  and  $\text{IdealR}$  are equivalent, but they are not: they can be easily distinguished by an environment that sets  $m'_2 \neq \perp$  and  $b = \perp$ . In fact,  $\text{IdealR}(m'_2, \perp) = (\perp, \perp)$ , but  $\text{RealR}(m'_2, \perp) = (\perp, o_2)$ , where  $o_2 \neq \perp$ . So,  $\text{IdealR}$  and  $\text{RealR}$  are not equivalent, and the simulator  $\text{SimR}$  is not valid.

**Insecurity in general** By generalizing the above idea, we can show that, for any simulator  $\text{SimR}$  there is an environment  $\text{Env}$  that can distinguish the two systems  $\text{RealR}$  and  $\text{IdealR}$  with nonnegligible probability. We build  $\text{Env}$  that works in two stages:

$$\begin{aligned}
\text{Env}_0(m, o_2) &= (b, m'_2, t) \text{ where} \\
&\quad b = \perp, m'_2 \leftarrow \mathbb{M}_n^{\times 2}, t = (o_2 > \perp) \\
\text{Env}_1(m, o_2) &= (b, m'_2, t) \text{ where} \\
&\quad b \leftarrow \{0, 1\}, m'_2 \leftarrow \mathbb{M}_n^{\times 2}, t = (\mathcal{G}(m) + o_2[b] = m'_2[b])
\end{aligned}$$

Notice that the output of the ideal system  $\text{IdealR}(m'_2, b) = (m, o_2)$  is defined by  $(b', m, o_2) \leftarrow \text{SimR}(m'_2[b'], b)$ , where  $b'$  is an internal channel. Since  $b'$  ranges over a flat CPO, and  $m'_2[\perp] = \perp$ , the value of  $b'$  resulting from a least fixed point computation is given by  $(b', -, -) = \text{SimR}(\perp, b)$ . In particular,  $b'$  may depend only on the external input  $b$ . We denote using  $\text{SimR}(b)^{b'}$  the random variable  $b'$  computed on input  $b$ .

Let  $p = \Pr\{\text{SimR}(\perp)^{b'} = \perp\}$  and  $q = \Pr\{\text{SimR}(\perp, \perp)^{o_2} = \perp\}$ . It is clear that  $\Pr\{\text{Env}_i[\text{RealR}] = \top\} = 1$  for all  $i \in \{1, 2\}$ . For the ideal system, we have

$$\begin{aligned}
\Pr\{\text{Env}_0[\text{IdealR}] = \top\} &= \Pr\{\text{SimR}(\perp, \perp)^{o_2} > \perp\} \cdot p \\
&\quad + \Pr\{\text{SimR}(\perp, m'_2[b'])^{o_2} > \perp\} \cdot (1 - p) \\
&= (1 - q)p + \Pr\{\text{SimR}(\perp, m'_2[b'])^{o_2} > \perp\} \cdot (1 - p).
\end{aligned}$$

Since  $\Pr\{\text{Env}_0[\text{RealR}] = \top\} = 1$ ,  $\Pr\{\text{Env}_0[\text{IdealR}] = \top\}$  must be overwhelming; and since  $\Pr\{\text{SimR}(\perp, \perp)^{o_2} > \perp\} \leq \Pr\{\text{SimR}(\perp, m'_2[b'])^{o_2} > \perp\}$ ,  $p$  must be negligible. Finally, notice that

$$\begin{aligned}
\Pr\{\text{Env}_1[\text{IdealR}] = \top\} &= \Pr\{\mathcal{G}(m) + o_2[b] = m'_2[b] \mid \text{SimR}(\perp)^{b'} = \perp\} \cdot p \\
&\quad + \Pr\{\mathcal{G}(m) + o_2[b] = m'_2[b] \mid \text{SimR}(\perp)^{b'} > \perp\} \cdot (1 - p).
\end{aligned}$$

If  $\text{SimR}(\perp)^{b'} > \perp$ , then  $\Pr\{b' = b\} = \frac{1}{2}$  and so

$$\Pr\{\mathcal{G}(m) + o_2[b] = m'_2[b] \mid \text{SimR}(\perp)^{b'} > \perp\} = \frac{1}{2} \left(1 + \frac{1}{2^\ell}\right).$$

This implies that  $\Pr\{\text{Env}_2[\text{IdealR}] = \top\} = \frac{1}{2} + \epsilon$  for some negligible  $\epsilon > 0$ , and so  $\text{Env}$  can distinguish the two systems.

The discrepancy between the two systems as shown above highlights a subtle timing bug in the protocol: in order to carry out the simulation, the transmission of  $i_2$  should be delayed until after the receiver has selected her 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  $\text{Net}_{\mathbb{M}}$  with an authenticated channel  $\text{AuthNet}_{\mathbb{M}}(i, e_i) = (o, e_o)$  that also takes an input  $e_i : \mathbb{T}$  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 the OT variants 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  $m = e_i!m_2[b]$ . This idea is similar to the “message header” in the UC models proposed in [6, 30].

We remark that none of these modifications affect the analysis presented in this section. In particular, considering a perfectly secure communication channel  $\text{Net}$  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  $i_2$  until  $b > \perp$ . So, it is not enough to provide this information to the environment. The timing signal  $b!\top$  needs to be provided as an input to the honest sender.

## 4 OT Length Extension

We have seen that the “standard” OT definition is inadequate even to model and analyze a simple OT length-extension protocol. In Fig. 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.

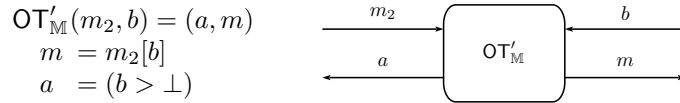
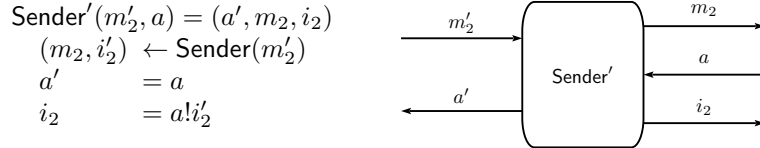


Figure 3: A revised OT functionality.

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 **Receiver** program as defined in Section 3, but modifies **Sender** 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  $\text{OT}'$  definition:



The **Sender** and **Receiver** programs are interconnected using  $\text{OT}'_n$  and  $\text{Net}_{2\ell}$  as shown in Fig. 4. As in the previous section, it is easy to check that the protocol is correct, i.e., combining and simplifying all the

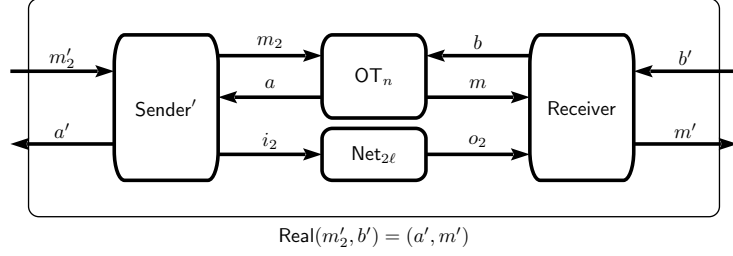
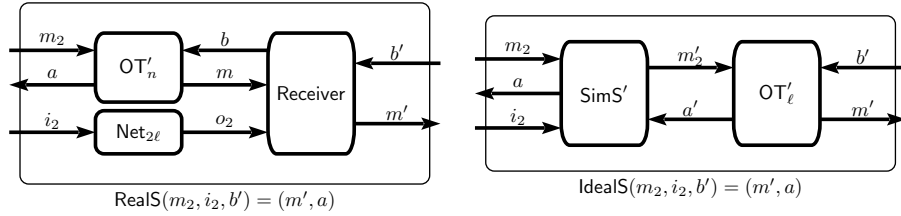
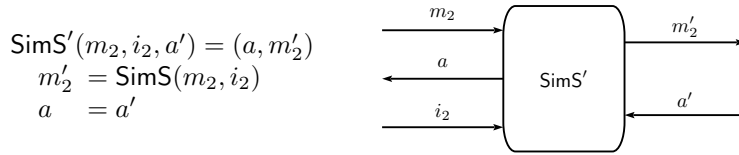


Figure 4: A normal execution of the OT Length Extension protocol.

equations from the real system in Fig. 4 produces a set of equations identical to the revised definition of the ideal functionality  $\text{OT}'(m'_2, b') = (a', m')$ . Security when the sender is corrupted is also similar to before. The real and ideal systems in this case are given by

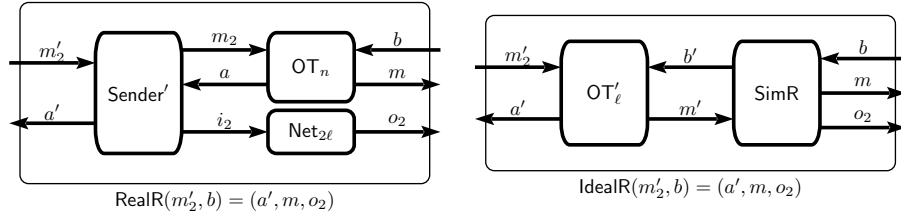


We see that this time  $\text{SimS}'$  has an additional input  $a'$  and output  $a$ . We adapt the simulator from the previous section simply by adding an equation that forwards the  $a'$  signal from  $\text{OT}'$  to the external environment:



$\text{RealS}(m_2, i_2, b')$  and  $\text{Ideal}(m_2, i_2, b')$  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  $\text{OT}'$  definition.

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



No change to the simulator are required: we use exactly the same “candidate” simulator  $\text{SimR}$  as defined in Section 3. Combining and simplifying the equations, gives the following real and ideal systems:

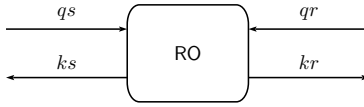
$$\begin{array}{ll}
\mathbf{RealR}(m'_2, b) = (a', m, o_2) & \mathbf{IdealR}(m'_2, b) = (a', m, o_2) \\
m_2 \leftarrow \mathbb{M}_n^{\times 2} & m_2 \leftarrow \mathbb{M}_n^{\times 2} \\
c_0 = m'_2[0] \oplus \mathcal{G}(m_2[0]) & c_0 = m'_2[b] \oplus \mathcal{G}(m_2[0]) \\
c_1 = m'_2[1] \oplus \mathcal{G}(m_2[1]) & c_1 = m'_2[b] \oplus \mathcal{G}(m_2[1]) \\
o_2 = b!(c_0, c_1) & o_2 = \langle c_0, c_1 \rangle \\
m = m_2[b] & m = m_2[b] \\
a' = (b > \perp) & a' = (b > \perp)
\end{array}$$

Now, when  $b = \perp$ , we have  $\mathbf{RealR}(m'_2, \perp) = \mathbf{IdealR}(m'_2, \perp) = (\perp, \perp, \perp)$ . So, no adversary can distinguish the two systems by not setting  $b$ . On the other hand, when  $b \neq \perp$ ,  $\mathbf{RealR}$  and  $\mathbf{IdealR}$  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 [10]. 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 the 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 [10] using the equational framework.

**The Random Oracle model** The protocol of [10] is designed and analyzed in the random oracle model [3]. So, both parties have access to an ideal functionality  $\mathbf{RO}$  implementing a random function with appropriately chosen domain  $Q$  and range  $K$ . Queries from the sender and receiver are answered consistently, and, in general,  $\mathbf{RO}$  can receive multiple (adaptively chosen) queries from both parties. Formally, the random oracle is modeled by the following functionality, where  $f^*(x_1, x_2, \dots) = (f(x_1), f(x_2), \dots)$  is the standard extension of  $f$  to sequences:

$$\begin{array}{l}
\mathbf{RO}_{Q,K}(qs, qr) = (ks, kr) \\
qs, qr : Q^* \\
ks, kr : K^* \\
f \leftarrow [Q \rightarrow K] \\
ks = f^*(qs) \\
kr = f^*(qr)
\end{array}$$


The random oracle starts by picking a function  $f: Q \rightarrow K$  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  $\mathbf{RO}$  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 Receiver programs from the protocol of [10] 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  $\mathbf{RO}(q_2, q) = (k_2, k)$  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.<sup>7</sup> 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 [8], 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 [8] 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 [10] 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, for simplicity, we still consider the traditional random oracle model as used in [10].

**The protocol** In order to facilitate a comparison with the original paper, we use as far as possible the same notation as [10]. Let  $G = \langle B \rangle$  be a group generated by an element  $B$  of prime order  $p$ . Following [10], we use additive group notation, so that the group elements are written as  $xB$  for  $x = 0, \dots, p-1$ .<sup>8</sup> In [10] 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  $\text{RO}(q_2, q) = (k_2, k)$  for functions with domain  $Q = G^2 \times G$  and range  $K = \{0, 1\}^n$ , which receives two (parallel) queries  $q_2 = \langle q_0, q_1 \rangle \in Q_{\perp}^{\times 2}$  from the sender and one query  $q \in Q_{\perp}$  from the receiver.

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

1. Non-committing: There exist PPT  $\mathcal{S}_1, \mathcal{S}_2$  such that, for all  $m \in \mathbb{M}_n$ , the following distributions are identical:<sup>9</sup>

$$\begin{aligned} \{(e, k) & : k \leftarrow K, e \leftarrow \mathbf{E}(k, m)\} \\ \{(e, k) & : e \leftarrow \mathcal{S}_1, k \leftarrow \mathcal{S}_2(e, m)\} \end{aligned}$$

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  $V_{S,e} = \{k \in S \mid \mathbf{D}(k, e) \neq \perp\}$  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  $\mathbf{E}(m, k) = (m, 0^n) \oplus k$ , i.e., padding the message with a string of zeros for redundancy, and masking the result with a one-time pad.

The protocol of [10] can be described by the equations in Fig. 5, and its execution is depicted in Fig. 6. We briefly explain the normal protocol execution: Sender first samples a random group element  $X$  and sends

<sup>7</sup>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.

<sup>8</sup>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 [10] and facilitate the comparison with the original protocol description.

<sup>9</sup>In fact, computational indistinguishability is enough, but it is easy to achieve perfect security.

it to Receiver; once it receives  $Y$  from 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 ciphertext pair  $c_2$  to Receiver. On the other hand, Receiver first samples a random group element  $yB$ , and upon receiving  $X$  from Sender it computes  $Y = bX + yB$  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  $m$ .

$$\begin{array}{ll}
 \text{Sender}(m_2, k_2, Y) = (q_2, X, c_2) & \text{Receiver}(k, X, c_2, b) = (q, Y, m) \\
 x \leftarrow \mathbb{Z}_p^* & y \leftarrow \mathbb{Z}_p^* \\
 X = xB & Y = bX + yB \\
 q_2[0] = ((X, Y), xY) & q = ((X, Y), yX) \\
 q_2[1] = ((X, Y), xY - xX) & m = \text{D}(k, c_2[b]) \\
 c_2[0] \leftarrow \mathbf{E}(k_2[0], m_2[0]) & \\
 c_2[1] \leftarrow \mathbf{E}(k_2[1], m_2[1]) &
 \end{array}$$

Figure 5: The OT protocol of Chau and Orlandi.

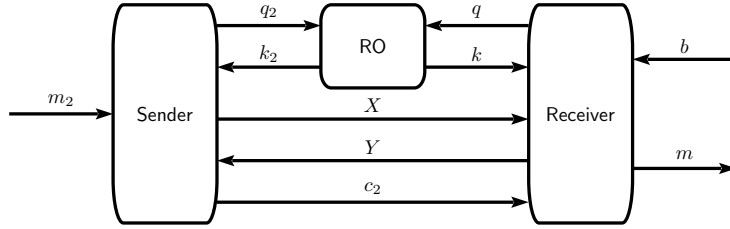
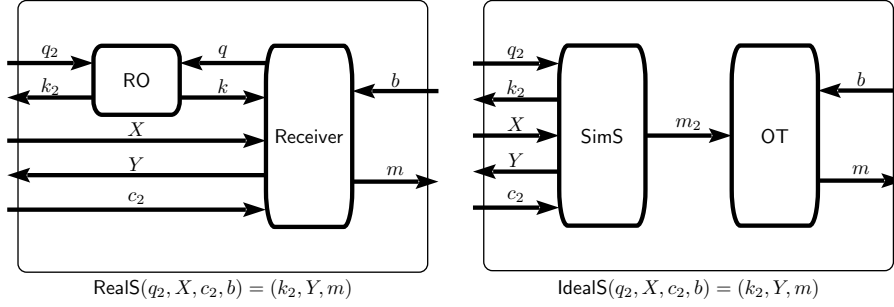


Figure 6: A normal execution of the OT protocol of Chou and Orlandi.

In the following subsections, we show that this protocol is insecure, both according to the classic OT definition given in Fig. 1, and according to our revised  $\text{OT}'$  definition of Fig. 3 that includes the signal  $a = (b > \perp)$  to the sender. Specifically, first, in Subsections 5.1 and 5.2 we show that if the definition from Fig. 1 is used, then the protocol is insecure against corrupted senders and corrupted receivers. The sender insecurity is for reasons very similar to those leading to the failure simulation in Section 3. Unlike the case of OT length extension, when considering the revised  $\text{OT}'$  definition and modifying the sender program accordingly, we show in Subsection 5.3 that the modified protocol is still insecure against corrupted senders and corrupted receivers.

## 5.1 Corrupted sender

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



For the protocol to be secure, the two systems should be computationally indistinguishable (for some simulator program  $\text{SimS}$ .) Just like the case of OT length extension, there exists an environment that can distinguish the two systems. We now describe an environment  $\text{Env}$  that works in two stages  $\text{Env}_0$  and  $\text{Env}_1$ , and show that for any  $\text{SimS}$ , at least one of  $\text{Env}_0$  and  $\text{Env}_1$  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  $t \in \{\perp, \top\}$ . The distinguishing advantage of  $\text{Env}_i$  is given by

$$\text{Adv}[\text{Env}_i] = |\Pr\{\text{Env}_i[\text{RealS}] = \top\} - \Pr\{\text{Env}_i[\text{IdealS}] = \top\}|.$$

The two stages of the distinguisher work as follows:

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

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

$$\Pr\{\text{Env}_0[\text{IdealS}] = \top\} = \Pr\{\text{Env}_1[\text{IdealS}] = \top\} \quad (4)$$

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

Intuitively, this environment can distinguish the real and the ideal systems because a corrupted sender (interacting with the real system  $\text{RealS}$ ), learns when the receiver sets  $b > \perp$  by observing the incoming message  $Y > \perp$ , but in the ideal system this timing information is not passed to the simulator.

## 5.2 Corrupted receiver

We have seen that when using the standard OT definition, the protocol is not secure against corrupted senders. Now we turn to analyzing the protocol against corrupted receivers with respect to the standard OT definition. The real and ideal system in this case are shown in Fig. 7.

Security requires that the real and the ideal systems are indistinguishable for some simulator program  $\text{SimR}$ . Unfortunately, as we are about to show, no such simulator exists.

**Proposition 1.** *For the OT protocol in Fig. 5, when the receiver is corrupted, for any receiver simulator  $\text{SimR}$ , there is an environment that distinguishes the two systems with nonnegligible probability.*

*Proof.* We build an environment that works in three stages, denoted by  $\text{Env}_i$  for  $i \in \{0, 1, 2\}$ :



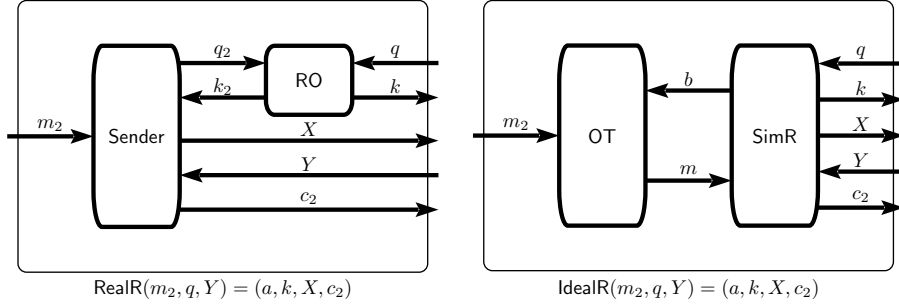


Figure 7: The real and ideal systems when receiver is corrupted.

$$\begin{aligned}
\text{Env}_0(k, X, c_2) &= (m_2, q, Y, t) \text{ where} \\
& d \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_p^*, m_2 = \perp, Y = dX + yB, q = \perp, t = (c_2 = \perp) \\
\text{Env}_1(k, X, c_2) &= (m_2, q, Y, t) \text{ where} \\
& d \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_p^*, m_2 \leftarrow \mathbb{M}_n^{\times 2}, Y = dX + yB, q = \perp, t = (c_2 > \perp) \\
\text{Env}_2(k, X, c_2) &= (m_2, q, Y, t) \text{ where} \\
& d \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_p^*, m_2 \leftarrow \mathbb{M}_n^{\times 2}, Y = dX + yB, q = ((X, Y), yX), \\
& t = (\mathsf{D}(k, c_2[d]) = m_2[d])
\end{aligned}$$

Assume there exists a receiver simulator  $\text{SimR}$ . With the real system,  $\text{Env}_i$  outputs  $t = \top$  with probability 1 for all  $i \in \{0, 1, 2\}$ . So  $\Pr\{\text{Env}_i[(\text{OT}|\text{SimR})] = \top\}$  must be overwhelming for all  $i \in \{0, 1, 2\}$ .

Notice that in the ideal system both  $b$  and  $m$  are internal channels such that  $m = m_2[b]$ , and we can simplify the output of the ideal system as  $(k, X, c_2) \leftarrow \text{SimR}(m_2[b], q, Y)$ . For  $i = 0, 1, 2$ , let  $u_i$  denote the (random variable of) the input to  $\text{SimR}$  when working with  $\text{Env}_i$ , and let  $\text{SimR}(u_i)^b$  denote the (random variable of) the value of  $b$  given input  $u_i$ . The external input channels to  $\text{SimR}$  are  $q$  and  $Y$ , and their values are  $\perp$  in both  $\text{Env}_0$  and  $\text{Env}_1$ . If  $\text{SimR}$  sets  $b = \perp$  when  $q = \perp$  and  $Y = \perp$ , then it cannot tell the difference between  $\text{Env}_0$  and  $\text{Env}_1$ , and thus at least one of  $\text{Env}_0$  and  $\text{Env}_1$  has a nonnegligible distinguishing advantage. So  $\Pr\{\text{SimR}(u_0)^b > \perp\}$  must be overwhelming. Since  $\text{SimR}$  is a monotone function,  $\Pr\{\text{SimR}(u_i)^b > \perp\}$  is also overwhelming for  $i \in \{1, 2\}$ . In particular, let  $\epsilon = \frac{1}{2} \Pr\{\text{SimR}(u_1)^b = \perp\}$ , then  $\epsilon$  is negligible.

Now consider  $\text{Env}_1$ , which sets  $q = \perp$  and samples  $Y$  from the distribution  $\{dX + yB \mid y \leftarrow \mathbb{Z}_p^*\} \equiv \{yB \mid y \leftarrow \mathbb{Z}_p^*\}$ . So  $q$  and  $Y$  are independent of  $d$ , and thus  $\Pr\{\text{SimR}(u_1)^b = d\} = \Pr\{\text{SimR}(u_1)^b = 1 - d\} = \frac{1}{2} - \epsilon$ .

Finally, when working with  $\text{Env}_2$  we have

$$\begin{aligned}
& \Pr\{\text{Env}_2[(\text{OT}|\text{SimR})] = \top\} = \Pr\{\mathsf{D}(k, c_2[d]) = m_2[d]\} \\
&= \Pr\{\mathsf{D}(k, c_2[d]) = m_2[d] \mid \text{SimR}(u_2)^b = d\} \Pr\{\text{SimR}(u_2)^b = d\} \\
&\quad + \Pr\{\mathsf{D}(k, c_2[d]) = m_2[d] \mid \text{SimR}(u_2)^b = 1 - d\} \Pr\{\text{SimR}(u_2)^b = 1 - d\} \\
&\quad + \Pr\{\mathsf{D}(k, c_2[d]) = m_2[d] \mid \text{SimR}(u_2)^b = \perp\} \Pr\{\text{SimR}(u_2)^b = \perp\}
\end{aligned}$$

Since  $\text{SimR}$  is monotone,  $\frac{1}{2} - \epsilon = \Pr\{\text{SimR}(u_1)^b = 1 - d\} \leq \Pr\{\text{SimR}(u_2)^b = 1 - d\}$ , and thus  $\Pr\{\text{SimR}(u_2)^b = d\} \leq \frac{1}{2} + \epsilon$ . On the other hand, when  $\text{SimR}(u_2)^b = 1 - d$ , it holds that  $\text{SimR}(u_i)^b \in \{1 - d\}_\perp$  for  $i \in \{0, 1\}$  and thus  $\text{SimR}$  has no access to  $m_2[d]$ , and since  $m_2[d]$  is independently sampled from  $\mathbb{M}_n$ ,  $\text{SimR}$  cannot guess it correctly with probability more than  $\frac{1}{2^n}$ . So we can bound the probability

$$\Pr\{\text{Env}_2[(\text{OT}|\text{SimR})] = \top\} \leq \frac{1}{2} + \epsilon + \frac{1}{2^n} + 2\epsilon,$$

which is close to  $\frac{1}{2}$ . Therefore the environment can distinguish the real and the ideal systems with nonnegligible probability.  $\square$

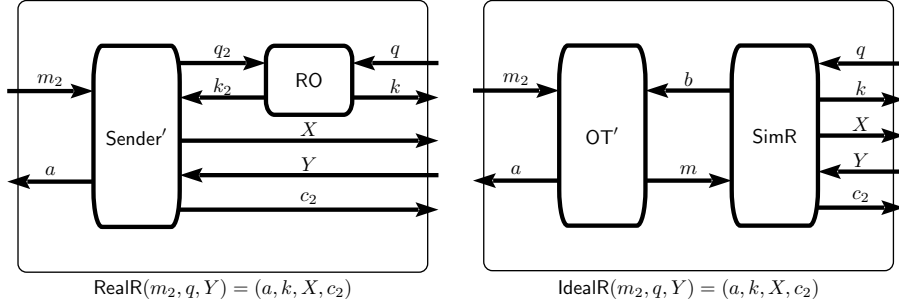


Figure 8: The real and ideal systems when receiver is corrupted, under revised OT definition.

### 5.3 Revised OT definition

The timing issue with a corrupted sender is similar to the one for OT length extension that is fixed by adding an acknowledgment signal. So it is natural to ask if the insecurity problems can be resolved by modifying the protocol according to the revised functionality  $\text{OT}'$ . Clearly, changing the definition requires also modifying the sender program to output a signal  $a$  in order to match  $\text{OT}'$ . 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:

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

We leave it to the reader to verify that a real protocol execution  $(\text{Sender}' \mid \text{RO} \mid \text{Receiver}): (m_2, b) \mapsto (a, m)$  is equivalent to the ideal functionality  $\text{OT}': (m_2, b) \mapsto (a, m)$ .

For security, we start with the case when the receiver is corrupted. The real and ideal systems are depicted in Fig. 8. Notice that the additional bit  $a$  is not provided to the simulator but is instead given to the environment. So any receiver simulator  $\text{SimR}$  that connects to  $\text{OT}'$  to form the ideal system in the revised OT definition has the same interface as a receiver simulator in the standard OT definition. Thus we obtain the same result as in Proposition 1 that the modified protocol is insecure against corrupted receivers.

When the sender is corrupted, the sender simulator is now provided with an additional bit  $a = (b > \perp)$ , as shown in Fig. 9. This small modification is the key to prove security for the OT length extension protocol, so one might speculate, as we did in the previous version of this paper, that security could also hold for the current protocol in the case of sender corruption. On the contrary, this modification is not enough. As we are exploring the useability of the equational framework, we show in the following why the natural simulation strategy that takes advantage of the signal  $a$  fails at proving security.

The speculated simulator is shown below. As we are presenting negative results, we limit the power of a corrupted sender such that it can send at most one pair of RO queries  $q_2$  and it obtains at most one pair of keys  $k_2$ .

$$\begin{aligned} \text{SimS}(q_2, X, a, c_2) &= (k_2, Y, m_2) \\ f &\leftarrow [(G^2 \times G) \rightarrow K] \\ k_2 &= f^*(q_2) \\ y &\leftarrow \mathbb{Z}_p^* \\ Y &= X!a!yB \\ m_2[0] &= \text{if } (\exists i. q_2[i] = ((X, Y), \dots)) \text{ then } D(k_2[i], c_2[0]) \\ m_2[1] &= \text{if } (\exists i. q_2[i] = ((X, Y), \dots)) \text{ then } D(k_2[i], c_2[1]) \end{aligned}$$

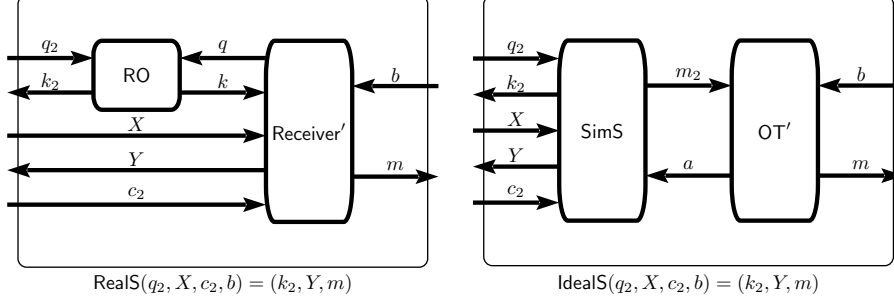


Figure 9: The real and ideal systems when sender is corrupted, under revised OT definition.

Let us derive an equation for  $m$ . In the real system  $\text{RealS}$ , the message  $m$  satisfies the equation

$$m = \mathsf{D}(f((X, bX + yB), yX), c_2[b]), \quad (5)$$

where  $y$  is sampled uniformly at random from  $\mathbb{Z}_p^*$  by the honest receiver. In the ideal system  $\text{IdealS} = (\text{SimS}|\text{OT}')$ , notice that  $a = (b > \perp)$ , and so

$$m = \mathsf{D}(f((X, X!b!yB), W), c_2[b]), \quad (6)$$

where  $y$  is sampled uniformly at random from  $\mathbb{Z}_p^*$  by the simulator and  $W$  is some element of  $G$  chosen by the environment. In both equations (5) and (6),  $c_2[b]$  is an input to the system given by the environment. By a careful examination, we can see that the value of  $m$  as computed in these two equations could be different if the environment sets  $W$  to be distinct from  $yX$ . We follow this idea to construct the following environment:

$$\begin{aligned} \text{Env}(k_2, Y, m) &= (q_2, X, c_2, b, t) \text{ where} \\ x &\leftarrow \mathbb{Z}_p^*, X = xB, w \leftarrow \mathbb{Z}_p^*, W = wB, b \leftarrow \{0, 1\}, \\ \text{For } i \in \{0, 1\}: \\ q_2[i] &= ((X, Y), W), c_2[i] \leftarrow \mathsf{E}(k_2[i], 0), \\ t &= (m > \perp) \end{aligned}$$

In the real system,  $\text{Env}$  outputs  $t = \top$  only in two cases: either the key  $k = f((X, Y), yX)$  obtained by the receiver is same as the key  $k_2[b] = f((X, Y), W)$  used by  $\text{Env}$  to encrypt  $m_2[b]$  in the ciphertext  $c_2[b]$ , where  $f$  is a random function sampled by  $\text{RO}$ , or the decryption succeeds when  $k \neq k_2[b]$ . For a sufficiently large key space  $\mathbb{K}_n$ , since  $yX = yxB$  and  $W = wB$  are independently sampled and uniformly distributed, the probability  $\epsilon$  that  $k = k_2[b]$  is negligible. Since  $(\mathsf{E}, \mathsf{D})$  is a robust encryption scheme, when  $k \neq k_2[b]$  the decryption can succeed with only a negligible probability  $\delta$ . So  $\text{Env}$  outputs  $t = \top$  with a negligible probability  $\epsilon + (1 - \epsilon)\delta$ . But in the ideal system, the decryption always succeeds and thus we get  $m = 0 > \perp$ , which implies that  $\text{Env}$  outputs  $t = \top$  with probability 1. Therefore  $\text{Env}$  has a nonnegligible distinguishing advantage.

We remark that, if the above simulator  $\text{SimS}$  has access to a DDH oracle  $\mathsf{O}$  that answers on input  $(X, Y, W)$  whether  $W = yxB$  for  $X = xB$  and  $Y = yB$ , then we can modify the equations for  $m_2$  in  $\text{SimS}$  to prove sender security with respect to the revised OT definition:

$$\begin{aligned} m_2[0] &= \text{if } (\exists i. q_2[i] = ((X, Y), W) \text{ and } \mathsf{O}(X, Y, W) = \top) \text{ then } \mathsf{D}(k_2[i], c_2[0]) \\ m_2[1] &= \text{if } (\exists i. q_2[i] = ((X, Y), W) \text{ and } \mathsf{O}(X, Y, W) = \top) \text{ then } \mathsf{D}(k_2[i], c_2[1]) \end{aligned}$$

That is, if a  $\text{RO}$  query contains a triple of group elements satisfying the DDH condition, then  $\text{SimS}$  uses the corresponding key to decrypt both  $c_2[0]$  and  $c_2[1]$  and assigns the resulting plaintext to  $m_2[0]$  and  $m_2[1]$ , respectively. As already noted by Genç, Iovino, and Rial [13], sender security holds with certain gap-DH groups in which the CDH problem is hard but the DDH problem is easy to solve.

## 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 [10]. 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 allow to salvage the OT protocol of Chou and Orlandi. Overall, the equational framework proved to be a convenient formalism to carry out rigorous, yet concise, security analysis of cryptographically interesting protocols.

## References

- [1] Backes, M., Pfitzmann, B., Waidner, M.: The reactive simulatability (RSIM) framework for asynchronous systems. *Inf. Comput.* 205(12), 1685–1720 (2007)
- [2] Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: 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. pp. 186–195 (2004)
- [3] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73. CCS ’93, ACM, New York, NY, USA (1993)
- [4] Bellare, M., Rogaway, P.: 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. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006)
- [5] Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on. pp. 136–145 (Oct 2001)
- [6] Canetti, R., Cohen, A., Lindell, Y.: 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. pp. 3–22 (2015)
- [7] Canetti, R., Dodis, Y., Pass, R., Walfish, S.: 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. pp. 61–85 (2007)
- [8] Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 597–608. CCS ’14, ACM, New York, NY, USA (2014)
- [9] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing. pp. 494–503. STOC ’02, ACM, New York, NY, USA (2002)
- [10] Chou, T., Orlandi, C.: The simplest protocol for oblivious transfer. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) Progress in Cryptology – LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings. pp. 40–58. Springer International Publishing (2015)

- [11] Crépeau, C., Graaf, J., Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Coppersmith, D. (ed.) *Advances in Cryptology — CRYPTO’ 95: 15th Annual International Cryptology Conference* Santa Barbara, California, USA, August 27–31, 1995 Proceedings. pp. 110–123. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
- [12] Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. *Commun. ACM* 28(6), 637–647 (1985)
- [13] Genç, Z.A., Iovino, V., Rial, A.: “The simplest protocol for oblivious transfer” revisited. *IACR Cryptology ePrint Archive* 2017, 370 (2017), <http://eprint.iacr.org/2017/370>
- [14] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. pp. 218–229. STOC ’87, ACM, New York, NY, USA (1987)
- [15] Gunter, C.A., Scott, D.S.: *Handbook of theoretical computer science (vol. b)*. chap. Semantic Domains, pp. 633–674. MIT Press, Cambridge, MA, USA (1990)
- [16] Hofheinz, D., Shoup, V.: GNUC: A new universal composability framework. *J. Cryptology* 28(3), 423–508 (2015)
- [17] Hofheinz, D., Unruh, D., Müller-Quade, J.: Polynomial runtime and composability. *J. Cryptology* 26(3), 375–441 (2013)
- [18] Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) *Advances in Cryptology – CRYPTO 2008: 28th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 17–21, 2008. Proceedings. pp. 572–591. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
- [19] Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Universally composable synchronous computation. In: *TCC*. pp. 477–498 (2013)
- [20] Kilian, J.: Founding cryptography on oblivious transfer. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. pp. 20–31. STOC ’88, ACM, New York, NY, USA (1988)
- [21] Küsters, R.: Simulation-based security with inexhaustible interactive turing machines. In: *Computer Security Foundations Workshop, CSFW-19 ’06*. pp. 309–320. IEEE Computer Society (2006)
- [22] Küsters, R., Tuengerthal, M.: The IITM model: a simple and expressive model for universal composability. *IACR Cryptology ePrint Archive* 2013, 25 (2013), <http://eprint.iacr.org/2013/025>
- [23] Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. *Journal of Cryptology* 25(4), 680–722 (2011)
- [24] Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) *Theory of Security and Applications: Joint Workshop, TOSCA 2011*, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers. pp. 33–56. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
- [25] Maurer, U., Renner, R.: Abstract cryptography. In: *Innovations in Computer Science - ICS 2010*, Tsinghua University, Beijing, China, January 7–9, 2011. Proceedings. pp. 1–21 (2011)
- [26] Micciancio, D., Tessaro, S.: An equational approach to secure multi-party computation. In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. pp. 355–372. ITCS ’13, ACM, New York, NY, USA (2013)
- [27] Rabin, M.O.: How to exchange secrets with oblivious transfer (1981), technical Report, TR-81 edn. Aiken Computation Lab, Harvard University

- [28] Scott, D.S.: Domains for denotational semantics. In: Proceedings of the 9th Colloquium on Automata, Languages and Programming. pp. 577–613. Springer-Verlag, London, UK, UK (1982)
- [29] Stoltenberg-Hansen, V., Lindström, I., Griffor, E.R.: Mathematical Theory of Domains. Cambridge University Press, New York, NY, USA (1994)
- [30] Wikström, D.: Simplified universal composability framework. In: Theory of Cryptography - TCC 2016-A. LNCS, vol. 9562, pp. 566–595. Springer (2016)
- [31] Winskel, G.: The Formal Semantics of Programming Languages: An Introduction. MIT Press, Cambridge, MA, USA (1993)
- [32] Yao, A.C.: How to generate and exchange secrets (extended abstract). In: Foundations of Computer Science, Proceedings of FOCS'86. pp. 162–167. IEEE Computer Society (1986)