Analysis Of The Simulatability Of An Oblivious Transfer

Bing Zeng[a]

[a]*School of Software Engineering, South China University of Technology, Guangzhou, 510006, China*

Abstract

In the Journal of Cryptology (25(1): 158-193. 2012), Shai Halevi and Yael Kalai proposed a general framework for constructing two-message oblivious transfer protocols using smooth projective hashing. The authors asserts that this framework gives a simulation-based security guarantee when the sender is corrupted. Later this work has been believed to be half-simulatable in literatures. In this paper, we show that the assertion is not true and present our ideas to construct a fully-simulatable oblivious transfer framework.

Keywords: oblivious transfer, secure multiparty computation, malicious adversaries, smooth projective hashing.

1. Introduction

Oblivious transfer (OT), introduced in [1], is a fundamental cryptographic primitive allowing the secure multiparty computation (SMPC) of *any* computable function [2]. Beside this completeness result by Kilian, OT exhibits interests on its own since this primitive is generally used as a building block in a variety of cryptographic protocols [3] [4] [5] [6] [7, 8] [9].

1-out-of-2 oblivious transfer (OT_1^2) deals with the scenario where a sender holds 2 private values m_1, m_2 and a receiver possesses a private indexes $i \in \{1, 2\}$. The receiver expects to get the values m_i without leaking any information about which one were chosen. On the other hand, the sender does not want the receiver to know anything but the value queried about.

A standard way of proving the security of a SMPC protocol is to use the ideal/real model paradigm. In this context, adversaries in the real world are demonstrated to be equivalent to enemies in the ideal world where the computation is executed by an incorruptible entity named *ideal functionality*. As the SMPC protocol (in the real world) simulates the ideal world, the security is said to be *fully-simulatable*. For any non-simulatable OT, it is harder to use it as a building block in cryptographic protocols. A concrete attack on non-simulatable OT shown by [7] is selective-failure attack where the sender causes a failure depending on the receiver's selection. What is worse, there possibly exists attacks that is unknown at present but will be carried out in the future.

In [10], Shai Halevi and Yael Kalai proposes a OT_1^2 based on smooth projective hashing. This work is remarkable among known protocols, because it is the most efficient protocol for OT_1^2 against malicious adversaries in plain model and it is a generally realizable framework. In [10] the authors also discuss the possibility of simulation-based security of their work. In the case that the receiver is corrupted, they admit that their work does not give a simulation-based security guarantee, while in the case that the sender is corrupted, they assert that their work gives a simulation-based one. Indeed Halevi-Kalai OT_1^2 has been believed to be half-simulatable in literatures, e.g., [11, 12, 13]. In this paper, we show that this assertion is not true and present our ideas to construct a fully-simulatable framework based on Halevi-Kalai OT_1^2.

2. Preliminaries

2.1. OT_1^2 Under the Ideal/Real Simulation Paradigm

In OT_1^2 functionality $f((m_1, m_2), i) = (\lambda, m_i)$, the sender $\mathcal{S}$ privately holds the input (m_1, m_2) and receives no output λ while the receiver $\mathcal{R}$ privately owns its choice i and receives m_i.

Email address: `zeng.bing.zb@gmail.com` (Bing Zeng)

In the ideal world, there is an incorruptible *trusted third party* (TTP). $\mathcal{S}$ and $\mathcal{R}$ send their inputs to TTP. If a party is corrupted by a malicious adversary $\mathcal{A}$, then then its input is sent by $\mathcal{A}$ and may be altered by $\mathcal{A}$. Denote the inputs received by the TTP by (y_1, y_2). The TTP computes $f(y_1, y_2)$ and sends the results to $\mathcal{A}$ and the honest party.

In the real world, there is a protocol Π instead of the TTP. Computing f is done via interactions between the sender and the receiver. The honest party strictly follows the prescribed protocol Π. The corrupted party follows $\mathcal{A}$'s instructions and may arbitrarily deviate from Π.

At the end of the executions, the corrupted party outputs nothing λ. The adversary $\mathcal{A}$ outputs any arbitrary function of the information it gathers. The honest party in the ideal world outputs the message obtained from the TTP while in the real world outputs what Π instructs. The output Ideal (Real, respectively) of the execution in the ideal world (the real world, respectively) is a 3-entry vector consisting of outputs of malicious adversary $\mathcal{A}$, the sender $\mathcal{S}$ and the receiver $\mathcal{R}$ in order.

Loosely speaking, protocol Π *securely computes OT^2_1 functionality f* in the presence of malicious adversaries, if and only if for any non-uniform probabilistic polynomial time adversary $\mathcal{A}$ in the real world, there exists a non-uniform probabilistic expected polynomial-time adversary $\mathfrak{S}$ in the ideal world such that the two outputs Ideal and Real of the two worlds are computationally indistinguishable. The adversary $\mathfrak{S}$ is called *the simulator* of the adversary $\mathcal{A}$.

2.2. Smooth Projective Hash

Smooth projective hash (SPH) was introduced to design chosen-ciphertext secure encryption schemes [14] and Halevi and Tauman Kalai applied a variant of this cryptographic primitive to construct a protocol for OT.

Definition 1 ([10])**.** *A hash family $\mathcal{H}$ is defined by means of the following PPT algorithms $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{KG}, \mathsf{Hash}, \mathsf{pHash})$:*

- *Parameter generator* PG*: it takes a security parameter k as input and returns a hash parameter Λ: i.e. $\Lambda \leftarrow \mathsf{PG}(1^k)$.*
- *Instance sampler* IS*: it takes a security parameter k and a hash parameter Λ as input and returns a triple, i.e., $(\dot{x}, \dot{w}, \ddot{x}) \leftarrow \mathsf{IS}(1^k, \Lambda)$), where $\dot{x}$ is a projective instance, $\dot{w}$ is one of its witnesses, $\ddot{x}$ is a smooth instance.*
- *Instance-testing algorithm* IT*: it tests the parameters Λ and two strings x_0, x_1, i.e., $\mathsf{IT}(\Lambda, x_0, x_1) \in \{0, 1\}$. The intent is to test that at least one of x_0, x_1 is a smooth instance.*
- *Key generator* KG*: it takes a security parameter k, a hash parameter Λ and an instance x as input and outputs a hash-projection key pair (hk, pk): i.e., $(hk, pk) \leftarrow \mathsf{KG}(1^k, \Lambda)$.*
- *Hash algorithm* Hash*: it takes a security parameter k, a hash parameter Λ, an instance x and a hash key hk as input and outputs a value y: i.e., $y \leftarrow \mathsf{Hash}(1^k, \Lambda, x, hk)$.*
- *Projection algorithm* pHash*: it takes a security parameter k, a hash parameter Λ, an instance x, a projection key pk and a witness w of x as input and outputs a value y: i.e., $y \leftarrow \mathsf{pHash}(1^k, \Lambda, x, pk, w)$.*

The *smoothness* requires that for any $\ddot{x}$, its projection key and hash value are almost uniformly distributed. The *projection* requires that for any $\dot{x}$ and any its hash-projection key pair (hk, pk), its hash value equals its projection value. The *verifiable smoothness* requires that if $\mathsf{IT}(\Lambda, x_0, x_1) = 1$, then at least one of x_0, x_1 is a smooth instance. The *hard subset membership* requires the smooth instances $\ddot{x}$ and projective instances $\dot{x}$ are computationally indistinguishable.

3. A Brief Review of Halevi-Kalai OT

Halevi-Kalai OT^2_1 [10] proceeds as follows:

- R1 (Receiver's step): $\mathcal{R}$ generates the hashing parameters Λ and samples random instances $(\dot{x}, \ddot{x}, w)$, where $\dot{x}$ is projective and w is its witness. $\mathcal{R}$ sets $x_i \leftarrow \dot{x}$ and $x_{3-i} \leftarrow \ddot{x}$ $(i \in \{1, 2\})$. $\mathcal{R}$ sends (Λ, x_i, x_{3-i}).
- S1 (Sender's step): $\mathcal{S}$ verifies that at least one instance of (x_1, x_2) is smooth. If the test fails then the sender aborts. Otherwise the sender encrypts each message m_i via XOR-ing it with hash value of x_i. $\mathcal{S}$ sends ciphertext c_i along projection key of x_i.

- R2 (Receiver's step): $\mathcal{R}$ XOR-es ciphertext c_i with projection value of x_i and gets message m_i.

Halevi-Kalai OT_1^2 protocol meets privacy-based definition, which is a weaker notion than simulation-based definition. In this definition, no malicious adversary should be able to distinguish two views which are generated on distinct sets of inputs for the honest party but yield the same output.

Definition 2 ([10]). *A protocol is said to* privately implement oblivious transfer OT_1^2 *if the following conditions are satisfied:*

- *Receiver's Privacy: Denoted by $\mathcal{R}(1^n, i)$ the message sent by the honest receiver with input $(1^n, i)$. Then the ensembles $\{\mathcal{R}(1^n, 1)\}_{n\in\mathbb{N}}$ and $\{\mathcal{R}(1^n, 2)\}_{n\in N}$ are computationally indistinguishable; $\{\mathcal{R}(1^n, 1)\}_{n\in\mathbb{N}} \overset{c}{=} \{\mathcal{R}(1^n, 2)\}_{n\in N}$.*
- *Sender's Privacy: Denote by $\mathcal{S}(1^n, m_1, m_2, q)$ the response of the honest sender with input $(1^n, m_1, m_2)$ when the receiver' s first message is q. Then there is a negligible function μ such that for any $n > 0$, any three messages $m_1, m_2, m' \in \{0, 1\}^{l(n)}$, and any message $q \in \{0, 1\}^*$, it holds that*

$$\mathcal{S}(1^n, m_1, m_2, q) \overset{s}{=} \mathcal{S}(1^n, m_1, m', q) \vee \quad or$$
$$\mathcal{S}(1^n, m_1, m_2, q) \overset{s}{=} \mathcal{S}(1^n, m', m_2, q),$$

  where $\overset{s}{=}$ means statistically indistinguishability.

4. Analysis of the Simulatability of Halevi-Kalai OT

In the case that the receiver is corrupted, Halevi and Kalai admit that their work does not give a simulation-based security guarantee, while in the case that the sender is corrupted, they assert that Definition 2 gives a simulation-based guarantee. Let us focus on the latter case. Following their ideas [10, Sec. 3], the simulator $\mathfrak{S}$ should be constructed as follows.

1. The simulator $\mathfrak{S}$ invokes the adversary $\mathcal{A}$ as a subroutine.
2. Simulator $\mathfrak{S}$ plays the role of an honest receiver with private input 1, and extracts message m_1.
3. $\mathfrak{S}$ rewinds $\mathcal{A}$, plays the role of an honest receiver with private input 2, and extracts message m_2.
4. $\mathfrak{S}$ sends (m_1, m_2) to the TTP, and outputs what $\mathcal{A}$ outputs.

Since $\{\mathcal{R}(1^n, 1)\}_{n\in N} \overset{s}{=} \{\mathcal{R}(1^n, 2)\}_{n\in N}$, the views of adversary $\mathcal{A}$ in the real world and ideal world are statistically indistinguishable too. Combining with the fact that the sender $\mathcal{S}$ outputs nothing in both the real world and the ideal world, one may conclude that the two worlds are computationally indistinguishable. However, this simulation ignores two subtle problems.

P1 $\mathcal{A}$ may not always gives responses to $\mathfrak{S}$.

P2 (m_1, m_2) is not extracted in one shot.

To illustrate P1, consider the following. $\mathcal{A}$ may gives responses in the first extraction and refuses to respond in the second extraction. This is possible, because $\mathcal{A}$ receives distinct messages in two distinct extractions. If this is the case, $\mathfrak{S}$ can not extract m_2 and fails.

To illustrate P2, consider the following. Since $\mathcal{A}$ is malicious, it may choose distinct values in distinct extractions. For example, the adversary $\mathcal{A}$ follows the following strategy:

- in each execution, $\mathcal{A}$ chooses two random values $m_1, m_2 \in_U \{0, 1\}^*$ as its real input and finally outputs them.

More concretely, let us assume that the real input of $\mathcal{A}$ in the first interaction and the second interaction are (a_1, a_2) and (b_1, b_2) respectively. We also assume that honest receiver R takes $i = 1$ as its input. Then we know,

$$\mathsf{Real} = ((a_1, a_2), \lambda, a_1) \quad \mathsf{Ideal} = ((b_1, b_2), \lambda, a_1)$$

where $(a_1, a_2), (b_1, b_2)$ are outputs of adversary $\mathcal{A}$ and simulator $\mathfrak{S}$, respectively. Considering the random choices of a_1, a_2, b_1, b_2, it holds with overwhelming probability that

$$a_1 \notin \{b_1, b_2\}.$$

That is, in the ideal world, the receiver get a value that is highly probable to be inconsistent with the output of $\mathcal{S}$. This distinguishes the real world from the ideal world.

Theorem 3. *Halevi-Kalai OT does not provide a simulation-based security guarantee when the sender or the receiver is corrupted.*

More generally, we have Lemma 4. Since the proof is similar to the discussion above, we omit the details.

Lemma 4. *Let Π be a protocol that is supposed to implement a two-party functionality $f(x, y)$ such that the first party receives $f_1(x, y)$ and the second party receives $f_2(x, y)$. Let $\mathfrak{S}$ be the simulator of the case that the malicious adversary corrupts the first party. If simulator $\mathfrak{S}$ does not extract real input of the adversary in one shot, and there exists a value v such that makes $f_2(\cdot, v)$ injective, then $\mathfrak{S}$ does not provide blackbox-simulation-based security.*

5. Our Ideas To Achieve A Fully-Simulatable Framework for $\mathbf{OT}_1^2$

First, let us consider the case that the receiver is corrupted. As pointed out by the Halevi and Kalai, the reason why their OT is non-simulatable is that the simulator $\mathfrak{S}$ can not extract the choice i of the adversary $\mathcal{A}$. We note that the receiver $\mathcal{R}$ learns the value m_i it queried about via a projective instance $\dot{x}$. For a smooth-projective instance pair $(\dot{x}, \ddot{x})$, if the witness $\dot{w}$ is available, $\mathfrak{S}$ can identify the projective instance, and hence $\mathfrak{S}$ can learn which value $\mathcal{A}$ chooses (i.e. it learns the adversary's real input i). Employing a cut-and-choose technique as in [15, 16], $\mathfrak{S}$ can see the witnesses by rewinding $\mathcal{A}$'s computation.

Naturally, our idea is for the receiver $\mathcal{R}$ to "cut" some instance pairs (where each one contains a projective instance and a smooth instance) and for the sender $\mathcal{S}$ to "choose" some instance pairs at random to check their *legalities* (i.e., $\mathcal{S}$ checks that each pair indeed contains at least one smooth instance). The receiver then sends the chosen instance pairs' witnesses. Combining the previous analysis, we can see that for a protocol constructed following this idea, $\mathfrak{S}$ can extract $\mathcal{A}$'s real input, and hence simulation-based security can be gained.

Second, let us consider the case that only the sender is corrupted. As pointed out in Section 4, the main problem is that the input extraction is not completed in one shot. To solve this problem, we let both parties commonly choose instance pairs to open via a coin-tossing protocol. This gives opportunities to $\mathfrak{S}$ to know the choice of malicious adversary $\mathcal{A}$ and to bias the common choices. Then $\mathfrak{S}$ can cheat $\mathcal{A}$ and extract its input in one shot.

To summarize our idea, a fully-simulatable framework can be depicted at a high level as follows:

1. Let K be a predetermined positive integer. The receiver $\mathcal{R}$ generates a hash family parameter and "cuts" K instance pairs where each pair contains a projective instance and a smooth instance. It shuffles each pair and sends the parameter and the shuffled pairs to the sender $\mathcal{S}$.
2. The sender $\mathcal{S}$ checks that the hash family parameter is legal. Then, both parties commonly run a coin-tossing protocol to "choose" instance pairs to check the legalities.
3. To prove the chosen instance pairs' legalities, the receiver $\mathcal{R}$ sends their witness to the sender $\mathcal{S}$.
4. Based on its private input i, the receiver $\mathcal{R}$ chooses a rearrangement indicator for each non-chosen pair.
5. According to the rearrangement indicators, the sender $\mathcal{S}$ reorders each non-chosen pair so that the i-th entry of the resultant pair is projective. Then, $\mathcal{S}$ encrypts its private 2 values by XOR-ing them with the hash values of the non-chosen instance pairs. Finally, $\mathcal{S}$ sends the encryptions and projection keys of non-chosen instance pairs to the receiver.
6. The receiver $\mathcal{R}$ computes the projection values of the non-chosen instances pairs and it XOR-es the projection values and the encryptions to gain the value it sought.

See [17] for the detailed framework and the formal proof which is highly complicated.

References

[1] M. O. Rabin, How to exchange secrets by oblivious transfer, Tech. Rep. Technical Report TR-81, Aiken Computation Lab, Harvard University (1981).

[2] J. Kilian, Founding cryptography on oblivious transfer, in: 20th Annual ACM Symposium on Theory of Computing (STOC'88), ACM Press, Chicago, USA, 1988, pp. 20 – 31.

[3] N. Mohammed, D. Alhadidi, B. C. M. Fung, M. Debbabi, Secure two-party differentially private data release for vertically partitioned data, Dependable and Secure Computing, IEEE Transactions on 11 (1) (2014) 59–71.

[4] C. Hazay, K. Nissim, Efficient set operations in the presence of malicious adversaries, Journal of Cryptology 25 (3) (2012) 383–433.

[5] B. Aiello, Y. Ishai, O. Reingold, Priced oblivious transfer: How to sell digital goods, in: B. Pfitzmann (Ed.), Advances in Cryptology - Eurocrypt'01, Vol. 2045 of Lecture Notes in Computer Science, Springer - Verlag, Innsbruck, Austria, 2001, pp. 119 – 135.

[6] Y. Lindell, B. Pinkas, Privacy preserving data mining, Journal of Cryptology 15 (2002) 177–206.

[7] M. Naor, B. Pinkas, Computationally secure oblivious transfer, Journal of Cryptology 18 (1) (2005) 1–35.

[8] M. Green, S. Hohenberger, Practical adaptive oblivious transfer from simple assumptions, in: Y. Ishai (Ed.), 8th Theory of Cryptography Conference (TCC'11), Vol. 6597, Springer - Verlag, Providence, USA, 2011, pp. 347 – 363.

[9] W. Ogata, K. Kurosawa, Oblivious keyword search, Journal of complexity 20 (2) (2004) 356–371.

[10] S. Halevi, Y. Tauman Kalai, Smooth projective hashing and two-message oblivious transfer, Journal of Cryptology 25 (1) (2012) 158–193.

[11] C. Peikert, V. Vaikuntanathan, B. Waters, A framework for efficient and composable oblivious transfer, in: D. Wagner (Ed.), Advances in Cryptology-CRYPTO'2008, Springer-Verlag Berlin, Santa Barbara, CA, 2008, pp. 554–571.

[12] J. Camenisch, G. Neven, abhi shelat, Simulatable adaptive oblivious transfer, in: M. Naor (Ed.), Advances in Cryptology - Eurocrypt'07, Vol. 4515 of Lecture Notes in Computer Science, Springer - Verlag, Barcelona, Spain, 2007, pp. 573 – 590.

[13] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, H. Wang, Adaptive oblivious transfer with access control from lattice assumptions, Advances in Cryptology – ASIACRYPT 2017, Springer International Publishing, 2017, pp. 533–563.

[14] R. Cramer, V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in: L. R. Knudsen (Ed.), Advances in Cryptology - Eurocrypt'02, Vol. 2332 of Lecture Notes in Computer Science, Springer - Verlag, Amsterdam, The Netherlands, 2002, pp. 45 – 64.

[15] A. Y. Lindell, Efficient fully-simulatable oblivious transfer, in: T. Malkin (Ed.), Topics in Cryptology - CT-RSA 2008, Vol. 4964 of Lecture Notes in Computer Science, Springer - Verlag, San Francisco, USA, 2008, pp. 52 – 70.

[16] Y. Lindell, B. Pinkas, An efficient protocol for secure two-party computation in the presence of malicious adversaries, Journal of Cryptology 28 (2) (2015) 312–350. doi:10.1007/s00145-014-9177-x.
URL http://dx.doi.org/10.1007/s00145-014-9177-x

[17] B. Zeng, Founding cryptography on smooth projective hashing, Cryptology ePrint Archive, Report 2018/444, https://eprint.iacr.org/2018/444 (2018).