“The Simplest Protocol for Oblivious Transfer” Revisited

Ziya Alper Genç[1], Vincenzo Iovino[1,*], Alfredo Rial[1]

Abstract

In 2015, Chou and Orlandi presented an oblivious transfer protocol that already drew a lot of attention both from theorists and practitioners due to its extreme simplicity and high efficiency.

Chou and Orlandi claimed that their protocol is universally composable secure (UC-secure) in the random oracle model under dynamic corruptions. UC-security is a very strong security guarantee that assures that, not only the protocol in itself is secure, but can be also used safely in larger protocols. Unfortunately, in this work we point out a flaw in their security proof for the case of a corrupt sender.

In more detail, we define a decisional problem and we prove that, if a correct security proof for the Chou and Orlandi's protocol is provided, then this problem can be solved correctly with overwhelming probability. Therefore, the protocol of Chou and Orlandi cannot be instantiated securely with groups for which our decisional problem cannot be solved correctly with overwhelming probability. Consequently, the protocol of Chou and Orlandi cannot be instantiated with *all* groups $\mathbb{G}$ in which the CDH problem is intractable, but only with groups in which both the CDH problem is intractable and our decisional problem can be solved with overwhelming probability.

After the appearance of our work, Chou and Orlandi acknowledged the problems we pointed out in their security proof and subsequent works showed additional issues, removing the claims of UC security of their protocol.

Keywords: oblivious transfer, universal composability

1. Introduction

Oblivious Transfer. In an oblivious transfer (OT) protocol, a sender receives as input messages $M_1, \ldots M_N$ and a receiver receives as input indices $\sigma_1, \ldots, \sigma_k \in [1, N]$. At the end of the protocol, the receiver outputs $M_{\sigma_1}, \ldots, M_{\sigma_k}$ and learns nothing about the other messages. The sender does not learn anything about the indices.

[*] Corresponding author

Email addresses: ziya.genc@uni.lu (Ziya Alper Genç), vinciovino@gmail.com (Vincenzo Iovino), alfredo.rial@uni.lu (Alfredo Rial)

OT was introduced by Rabin [15] and generalized by Even, Goldreich and Lempel [9] and Brassard, Crépeau and Robert [2]. (The notion of OT was also developed independently by Wiesner in the 1970's but published only later [16].) OT has a lot of applications and it is at the core of multi-party computation [17, 11, 13].

Chou and Orlandi's OT Protocol. Chou and Orlandi (CO) in [6] present a novel OT protocol and claim that it is universally composable (UC) [5] under dynamic corruptions. Their protocol has the advantages of being extremely simple and efficient. The work of CO has already gained some popularity both from theorists and practitioners and has so far been cited 76 times according to Google Scholar.

CO present a 1-out-of-2 OT protocol and extend it to a 1-out-of-n OT protocol in a straightforward manner. For the purpose of this work, which focuses on negative results about the security of the CO's protocol, it suffices to analyze the 1-out-of-2 OT protocol. We note that our negative results also apply to the 1-out-of-n OT protocol.

The 1-out-of-2 OT protocol of CO is depicted in Figure 1. To run the protocol, Alice (the sender) and Bob (the receiver) have first to agree on a group $\mathbb{G}$ and a generator g of prime order p. In the first message, Alice samples a random element a in $\mathbb{Z}_p$ and sends $A = g^a$ to Bob. Bob picks random b in $\mathbb{Z}_p$ and, depending on his index $c \in \{0, 1\}$, sends either $B = g^b$ or $B = Ag^b$ to Alice. Then, Alice derives two keys k_0 and k_1 from $(B)^a$ and $(B/A)^a$ respectively. Alice encrypts the messages M_0 and M_1 by using the keys k_0 and k_1 respectively. Bob can derive the key k_R from A^b, which allows Bob to obtain M_c. However, it is computationally hard for him to compute the key that allows the obtention of M_{1-c}.

The protocol uses as building block a symmetric-key encryption scheme given by two algorithms Enc and Dec. In [6], it is claimed that UC-security against a corrupt sender holds in the random oracle model if the scheme (Enc, Dec) is robust. In this paper, we do not analyze the security in the case of a corrupt receiver; see 1.1.1 for references to related works pointing additional composability issues for the case of a corrupt receiver.

1.1. Our Results

We show a mistake in the security proof given by CO in [6] for the case of a corrupt sender. Namely, in their security proof, their simulator extracts incorrectly the messages M_0 and M_1 that are sent to the ideal functionality.

We also define a decisional problem in the group $\mathbb{G}$ and we prove that, if a correct simulator is provided for the case of a corrupt sender, then this problem can be solved with overwhelming probability. Therefore, the protocol of CO cannot be instantiated securely with groups $\mathbb{G}$ in which our decisional problem cannot be solved with overwhelming probability.

Consequently, the protocol of CO cannot be instantiated with *all* groups $\mathbb{G}$ in which the CDH problem is intractable (as originally claimed), but only with groups in which both the CDH problem is intractable and our decisional

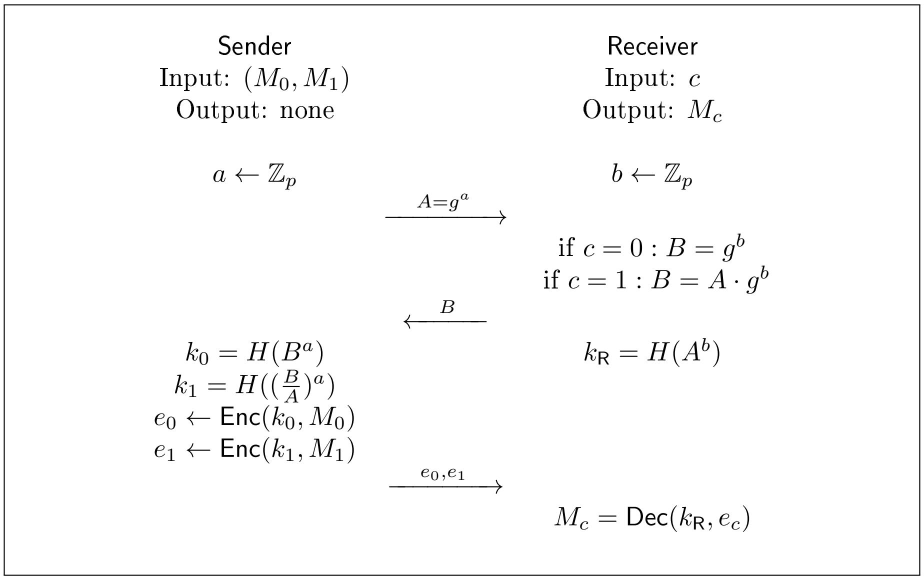
Sender
Input: (M_0, M_1)
Output: none
Receiver
Input: c
Output: M_c
$a \leftarrow \mathbb{Z}_p$
$b \leftarrow \mathbb{Z}_p$
$\xrightarrow{A=g^a}$
if $c = 0 : B = g^b$
if $c = 1 : B = A \cdot g^b$
$\xleftarrow{B}$
$k_0 = H(B^a)$
$k_1 = H((\frac{B}{A})^a)$
$e_0 \leftarrow \mathsf{Enc}(k_0, M_0)$
$e_1 \leftarrow \mathsf{Enc}(k_1, M_1)$
$k_{\mathsf{R}} = H(A^b)$
$\xrightarrow{e_0,e_1}$
$M_c = \mathsf{Dec}(k_{\mathsf{R}}, e_c)$

Figure 1: Chou and Orlandi's 1-out-of-2 OT Protocol.

problem can be solved with overwhelming probability. Our decisional problem can be solved with overwhelming probability when the DDH problem is easy in $\mathbb{G}$. Therefore, it seems likely that the protocol of CO can be instantiated securely with gap-DH groups, that is groups in which the CDH problem is hard and the DDH problem is easy. Examples of such groups are for instance the bilinear groups [1]. In this case, the proof of UC-security in the case of a corrupt sender would work. Unfortunately, after the appearance of our work, subsequent works pointed out additional issues in the composable security of the CO's protocol; see Section 1.1.1.

Our results also entail techniques to show that non-trivial natural protocols cannot be proven UC-secure, which can be of independent interest.

Outline of the Paper. In Section 2, we describe an ideal functionality for OT, which we use in our negative result of Section 4. This ideal functionality takes into account an observation made by Li and Micciancio [14] on the definition of ideal functionalities for OT that the protocol of CO realizes. In Section 3, we describe the flaw in the simulator of CO for the case of sender corruption. In Section 4, we define a decisional problem and we prove that the CO's protocol cannot be instantiated securely with groups $\mathbb{G}$ where this problem cannot be solved with overwhelming probability. We conclude in Section 5.

1.1.1. Differences with the previous versions and related work

The original work of CO appeared in Latincrypt 2015 [6] and was posted in the same year on the IACR eprint archive [7]. In 2017, we posted an earlier

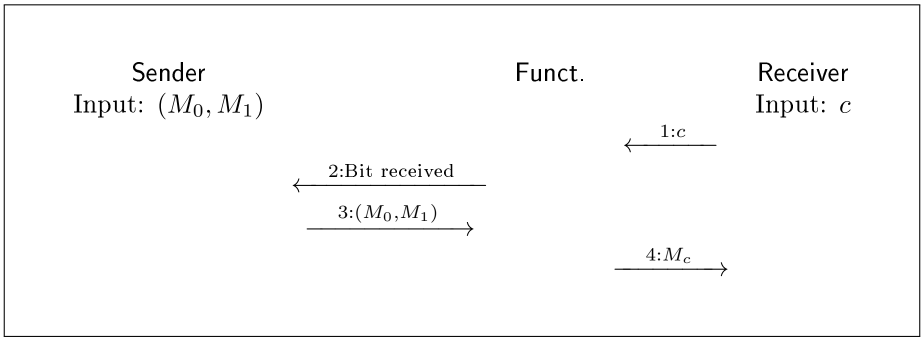

Figure 2: Ideal protocol for 1-out-of-2 OT between the functionality, the sender and the receiver. Li and Micciancio [14] detected that message number 2 to inform the sender that the receiver has sent his input bit is required for the OT functionality to be realizable.

version of this work on the IACR eprint archive [10]. In 2018, CO updated the aforementioned IACR eprint version of their work [7] acknowledging the problems in the security proof of their protocol that we and others had found, and removing the claims of UC-security.

Already in the first revision of our IACR ePrint paper, it was pointed out how to fix the issue we had found in the UC-security proof of the CO's protocol using gap-DH groups. After the appearance of our IACR ePrint work, Hauck and Loss [12] independently fixed the problem in the UC-security of the CO's protocol using, as suggested by us, the gap-DH assumption, and also proposed a different protocol based on the CDH assumption only.

Li and Micciancio [14] showed that the CO's protocol is not simulatable in the equational framework. Byali, Patra, Ravi and Sarkar [3] and Doerner, Kondi, Lee and shelat [8] showed additional issues in the proof for UC-security in the case of corrupt receiver.

We remark that in our IACR ePrint work we have been the first to point out the incorrect claims in the UC-security proof of the CO's protocol.

2. Ideal Functionality for 1-out-of-2 OT

In this section, we describe an ideal functionality for OT, which we use in our proof in Section 4. This ideal functionality takes into account an observation made by Li and Micciancio [14] on the definition of ideal functionality for OT that the protocol of CO realizes.

The functionality defined by CO does not impose any restriction on the order in which the sender and the receiver send their inputs to the ideal functionality. Li and Micciancio [14] observe that this is a problem to prove secure the OT protocol of CO. In the OT protocol of CO, the receiver has to decide his input bit in order to compute the second message (B) of the protocol. The sender decides what messages he inputs in order to compute the third message (e_0, e_1).

Ideal $\binom{2}{1}$-OT Functionality

The functionality waits for some input c from the receiver R and before receiving any input from R will ignore any input coming from the sender S. When the input c is received from R, if $c \notin \{0,1\}$, the functionality sends an error message $\perp$ to the receiver, otherwise it notifies the sender by transmitting a special symbol OK. The functionality will ignore further inputs from R.
After receiving $c \in \{0,1\}$ from R, if the functionality did not abort, the functionality waits for either a pair of messages (M_0, M_1) or an error message $\perp$ from the sender S. If the input of S is $\perp$ or a non-valid pair of messages (in the message space) the functionality sends $\perp$ to the receiver, otherwise it sends M_c to the receiver. The functionality ignores further inputs from S.

Figure 3: Ideal functionality for 1-out-of-2 OT.

In the security proof for the case of sender corruption, the simulator needs to extract the messages from the adversary. The simulator cannot perform such extraction until receiving the third message (e_0, e_1) of the protocol from the adversary. However, to receive this third message (e_0, e_1), the simulator has to send before the second message (B) to the adversary. The problem here is that the simulator does not know whether the receiver has already input his bit to the functionality, because the functionality does not tell the sender that the receiver has sent his input. Consequently, the simulator does not know whether it can send the second message to the adversary, and so it cannot provide a correct simulation. In the UC-security proof of CO, this problem is overlooked. We sketch the problem in Figure 2.

In Figure 3, we show a functionality for 1-out-of-2 OT for static corruptions. As suggested by Li and Micciancio, this functionality informs the sender when the receiver sends his input bit. We note that this functionality, like the one of CO, skips many details, such as the communication with the simulator and many other elements that are necessary in the UC framework (session identifiers, ...).

We remark that the functionality in Figure 3 requires the receiver to send his input bit c first, and then allows the sender to send her input messages (M_0, M_1). It would be possible to define a functionality that does not impose an order to receive the inputs from sender and receiver. However, we prefer to define this functionality because it is the concrete functionality that the CO protocol realizes. I.e., in the CO protocol, the receiver has to choose his input bit c when he sends the second message (B) to the sender, whereas the sender has to choose the messages (M_0, M_1) when he sends the third message (e_0, e_1). In other OT protocols (e.g. [4]), the sender has to decide her input messages before the receiver decides his choice. We think that it is important to distinguish between both types of OT protocols for two reasons. First, this may need to be taken into account when sender and receiver run other protocols concurrently with the OT protocol. For example, in the case of the CO protocol, it could happen that, after the receiver chooses c and sends the second message (B) to the sender, the sender receives information (from other concurrent protocols) that influences what messages (M_0, M_1) are used to compute the third message (e_0, e_1), whereas the receiver is no longer able to modify (B), despite possibly receiving relevant information from the sender from other protocols that are run in parallel. The second reason is that the restriction on the order of inputs (sender first or receiver first) in the functionality might also impact how efficiently the functionality can be realized.

We would like to stress that the mistake we found in the simulator of the security proof of CO is independent of the one found by Li and Micciancio. In fact, Li and Micciancio [14] provide a simulator for the CO protocol for the case of sender corruption to realize their modified OT functionality and say "we leave the verification that the simulator is indeed correct to the reader." However, the simulator by Li and Micciancio has the same problem as the simulator of CO.

The mistake we found in the simulator of CO is that the simulator does not send the correct messages to the functionality when the sender is corrupt. Therefore, it cannot be patched by using a different ideal functionality because any existing 1-out-of-2 OT functionality requires the sender to send the messages to the functionality.

3. Flaw in CO's Security Proof

In this section, we analyze the security proof provided by CO for the case of sender corruption. We show that the simulator described by CO for this case is incorrect. For simplicity, we analyze the instantiation of the protocol as a 1-out-of-2 OT scheme, but we remark that the mistake we found also holds for the case of m parallel executions of 1-out-of-n OT for other values of m and n.

The simulator needs to extract the messages from the corrupt sender in order to send them to the ideal functionality. To do this, when the corrupt sender makes a random oracle query, the simulator described by CO picks a random key, stores it and replies the query with this random key. After that, when the corrupt sender sends the ciphertexts, the simulator tries to decrypt the ciphertexts (e_0, e_1) by using all the stored keys until the result of one of the decryptions is not $\perp$. If the result of decryption is $\perp$ in all cases, then the message is set to $\perp$.

The problem in this simulator is the following. The corrupt sender can submit an oracle query on input $X \neq B^a$ (resp. $Y \neq (\frac{B}{A})^a$) and compute the ciphertexts e_0 (resp. e_1) using key $k'_0 = H(X)$ (resp. $k'_1 = H(Y)$). Then the simulator would decrypt using (k'_0, k'_1) and obtain messages different from $\perp$. However, the honest receiver in the real world would obtain $\perp$ because the oracle query made by the receiver is for the correct value $Z = A^b$, and so the key k_{R} that the honest receiver obtains is different from both k'_0 and k'_1.

CO argue that their simulator is correct thanks to the robustness of the encryption scheme. They claim that, because there is only one key that, for any ciphertext, decrypts the ciphertext to a message different from $\perp$, then the message decrypted by the simulator and the one obtained by the honest receiver have to be equal. However, this is untrue. The problem is that the corrupt sender can compute a ciphertext with a key different from the correct key used by the honest receiver. I.e., the corrupt sender can send a random oracle query for an incorrect value and then compute a ciphertext by using the key obtained for this query. In this case, the honest receiver obtains $\perp$, but the simulator decrypts the ciphertext to a message different from $\perp$ by using the key that was sent to the corrupt sender to answer the random oracle query for an incorrect value.

To fix the simulator, we would need a mechanism that allows the simulator to check whether a random oracle query from the corrupt sender is for a correct value, i.e., $X = B^a$ or $Y = (\frac{B}{A})^a$, or not. In Section 4, we show that the simulator cannot perform this check for both X and Y unless the simulator can solve a decisional problem with overwhelming probability.

4. On the Security Against a Corrupt Sender of CO's OT

In this section, we define a decisional problem in the group $\mathbb{G}$. We prove that, if a correct simulator for CO's OT protocol for the case of a corrupt sender exists, then this simulator can be used to solve this decisional problem with overwhelming probability. Therefore, if we assume that our decisional problem cannot be solved with overwhelming probability in $\mathbb{G}$, then a correct simulator cannot be provided. However, if our problem can be solved with overwhelming probability in $\mathbb{G}$, then a correct simulator can still be provided. Our decisional problem can be solved with overwhelming probability if the DDH problem is easy to solve in $\mathbb{G}$. Therefore, it seems likely that a correct simulator can be provided if the CO's OT protocol is instantiated with gap-DH groups. The gap-DH problem consists in solving an instance of the computational DH problem with the help of a decisional DH oracle. In gap-DH groups, it is believed that the gap-DH problem is hard.

As a consequence, security for a corrupt sender does not hold solely in the random oracle model under the assumption that the encryption scheme Enc and Dec is robust, as claimed by CO. An additional requirement is that, in the group $\mathbb{G}$, our decisional problem can be solved with overwhelming probability.

Decisional problem in $\mathbb{G}$. Our decisional problem is parameterized by a group sampling algorithm $\mathcal{G}$ that on input a security parameter λ outputs a group description $(\mathbb{G}, p, g)$. We define the following game between a challenger and an adversary $\mathcal{A}$. The challenger runs $\mathcal{G}$ on input the security parameter λ to get a group description $(\mathbb{G}, p, g)$ for a group $\mathbb{G}$ of prime order p with generator g. The challenger picks a random value a from $\mathbb{Z}_p$. The adversary $\mathcal{A}$ receives as input the group description $(\mathbb{G}, p, g)$ and g^a. The adversary returns to the challenger a group element B. The challenger draws a bit b at random and proceeds as follows:

- If $b = 0$, set $Z_0 = B^a$ and $Z_1 = B^a/g^{a^2}$.
- If $b = 1$, draw randomly another bit d and proceed as follows.
  - If $d = 0$, set Z_0 as a random element in $\mathbb{G}$ and $Z_1 = B^a/g^{a^2}$.
  - If $d = 1$, set $Z_0 = B^a$ and set Z_1 as a random element in $\mathbb{G}$.

  The challenger sends the pair (Z_0, Z_1) to the adversary. The adversary outputs its guess b'. The adversary wins the game if $b' = b$.

The hardness of our decisional problem is based on the difficulty of deciding whether a value given by the challenger equals g^{a^2} or random. We conjecture that our decisional problem cannot be solved with overwhelming probability in groups $\mathbb{G}$ in which the DDH assumption holds. Concretely, we conjecture that, in such groups, the probability of an adversary in winning the game described above is non-negligibly greater than $3/4 + \nu(\lambda)$. On the other hand, it is easy to see that, if the DDH assumption does not hold in $\mathbb{G}$, then our decisional problem can be solved with overwhelming probability.

Theorem 1. *Assuming that our decisional problem cannot be solved with overwhelming probability in the group $\mathbb{G}$, the CO's OT protocol cannot be proven UC-secure in the random oracle model when the sender is corrupt.*

Proof 1. We prove Theorem 1 by contradiction. We show that, if a correct simulator for the case of a corrupt sender exists, then we can use that simulator to solve our decisional problem with overwhelming probability.

First, we make the following observation. Consider an environment that sends a random bit c as input to the honest receiver. Given such an environment, any correct simulator must be able to extract correctly the messages M_0 and M_1 from the corrupt sender in order to send them to the ideal functionality. As can be seen, if the message $M_{c'}$ $(c' \in \{0, 1\})$ sent by the simulator is not correct, i.e., if it does not equal the message that the honest receiver outputs in the real world, then the simulation fails whenever the environment sends $c = c'$ to the honest receiver. We omit a formal proof of this observation.

Second, we show that, given a simulator that is able to extract both M_0 and M_1 correctly for the CO protocol, we can build a reduction R to solve our decisional problem with overwhelming probability. R interacts with the challenger and runs a copy of the simulator. R plays the role of the environment, the corrupt sender and the ideal functionality towards the simulator. The reduction R works as follows:

- R receives the instance $(\mathbb{G}, p, g, g^a)$ from the challenger.
- R, acting as the corrupt sender, sends the message $A = g^a$ to the simulator.

- R, acting as the ideal functionality, informs the simulator that the receiver has input his bit c.
- R receives a message B from the simulator. We observe that, after being informed by the ideal functionality that the receiver has input his bit c, a correct simulator must always send a message B indistinguishable from the message B produced by the honest receiver in the real world. Otherwise the simulation fails.
- R sends B to the challenger.
- The challenger sends (Z_0, Z_1) to R.
- R, acting as the corrupt sender, sends (Z_0, Z_1) as a random oracle query to the simulator.
- R receives the reply (k_0, k_1) from the simulator.
- R picks two random messages M_0 and M_1, computes $e_0 \leftarrow \mathsf{Enc}(k_0, M_0)$ and $e_1 \leftarrow \mathsf{Enc}(k_1, M_1)$, and, acting as the corrupt sender, sends e_0 and e_1 to the simulator.
- R receives two messages M_0' and M_1' from the simulator. If $M_0 = M_0'$ and $M_1 = M_1'$, R sends $b' = 0$ to the challenger, else R sends $b' = 1$ to the challenger.

The simulator must extract the messages M_0' and M_1' from e_0 and e_1 correctly with overwhelming probability. Therefore, $b = b'$ with overwhelming probability, i.e. R solves our decisional problem with overwhelming probability. As can be seen, if $b = 0$, then both Z_0 and Z_1 are correctly computed by the challenger, and thus the keys (k_0, k_1) used to compute e_0 and e_1 equal the correct key used by the honest receiver in the real world. Because the simulator must send correct messages M_0' and M_1' to the functionality, if $M_0 = M_0'$ and $M_1 = M_1'$ we are in the case in which Z_0 and Z_1 are correctly computed by the challenger. If $b = 1$, either Z_0 or Z_1 is computed randomly by the challenger. In this case, either k_0 or k_1 differs from the key used by the honest receiver in the real world. Namely, if $Z_{c'}$ ($c' \in \{0, 1\}$) is random, then $k_{c'}$ differs from the key k_c used by the honest receiver in the real world whenever $c = c'$. Because of the robustness of the encryption scheme, the honest receiver in the real world outputs $\perp$ whenever $c = c'$. Because the simulator must send correct messages to the functionality, in this case the simulator must send $M_{c'}' = \perp$ to the functionality and never $M_{c'}' = M_{c'}$.

5. Conclusion

We have shown that the OT protocol of CO cannot be instantiated securely with every group $\mathbb{G}$ in which the CDH assumption holds, as originally claimed by CO. We have defined a decisional problem and we have shown that, for the protocol to be secure, this decisional problem should be solvable in $\mathbb{G}$ with overwhelming probability for the protocol to be secure. Our decisional problem can be conjectured to be hard in groups $\mathbb{G}$ in which the DDH assumption is hard. If the DDH assumption does not hold in $\mathbb{G}$, our decisional problem can be solved correctly with overwhelming probability. Therefore, it is likely that the CO protocol can be securely instantitated with gap-DH groups, as showed by subsequent works (see Section 1.1.1) that also pointed out patches to the CO protocol as well as additional problems with composable security.

Acknowledgments. We thank Claudio Orlandi for his insightful comments. Vincenzo Iovino is supported by an AIM grant (no. 1838734-2) of the Italian Ministry of the University and Research (MIUR) and he made part of this work while at the University of Luxembourg supported by a CORE Junior grant (no. 11299247) of the Luxembourg National Research Fund. Alfredo Rial is supported by the Luxembourg National Research Fund (FNR) CORE project "Stateful Zero-Knowledge" (Project code: C17/11650748).

References

[1] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, Aug. 2001.

[2] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-nothing disclosure of secrets. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 234–238. Springer, Aug. 1987.

[3] M. Byali, A. Patra, D. Ravi, and P. Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. Cryptology ePrint Archive, Report 2017/1165, 2017. `https://eprint.iacr.org/2017/1165`.

[4] J. Camenisch, G. Neven, and a. shelat. Simulatable adaptive oblivious transfer. In M. Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 573–590. Springer, May 2007.

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

[6] T. Chou and C. Orlandi. The simplest protocol for oblivious transfer. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 40–58, 2015.

[7] T. Chou and C. Orlandi. The simplest protocol for oblivious transfer. Cryptology ePrint Archive, Report 2015/267, 2015. https://eprint.iacr.org/2015/267.

[8] J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 980–997, 2018.

[9] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 205–210. Plenum Press, New York, USA, 1982.

[10] Z. A. Genç, V. Iovino, and A. Rial. "the simplest protocol for oblivious transfer" revisited. IACR Cryptology ePrint Archive, Report 2017/370, 2017. https://eprint.iacr.org/2017/370.

[11] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In A. Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987.

[12] E. Hauck and J. Loss. Efficient and universally composable protocols for oblivious transfer from the cdh assumption. Cryptology ePrint Archive, Report 2017/1011, 2017. https://eprint.iacr.org/2017/1011.

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

[14] B. Li and D. Micciancio. Equational security proofs of oblivious transfer protocols. *IACR Cryptology ePrint Archive*, 2016:624, 2016.

[15] M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981. Available at http://eprint.iacr.org/2005/187.

[16] S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, Jan. 1983.

[17] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, Oct. 1986.