UC-Secure Multi-Session OT Using Tamper-Proof Hardware Tokens

Kaoru Kurosawa[1] Ryo Nojima[2] Le Trieu Phong[2]

Ibaraki University[1], Japan,
kurosawa@mx.ibaraki.ac.jp
NICT, Japan[2],
{ryo-no,phong}@nict.go.jp

Abstract

In this paper, we show the first UC-secure *multi-session* OT protocol using tamper-proof hardware tokens. [1] The sender and the receiver exchange tokens only at the beginning. Then these tokens are reused in arbitrarily many sessions of OT. An instantiation of the proposed scheme is UC-secure against static adversaries under the DDH assumption and the RSA assumption in the random oracle model.

Keywords: tamper-proof hardware token, UC-security, multi-session OT

1 Introduction

The framework of Universal Composability (UC), introduced by Canetti [2], guarantees very strong security properties of cryptographic protocols. UC-secure protocols are secure even if they are arbitrarily composed with other instances of the same or other protocols.

In this framework, any ideal functionality can be securely realized if a majority of the participants are uncorrupted. However, this result does not hold when half or more of the parties are corrupted. In particular, it does not hold for the important case of two-party protocols, where each party wishes to maintain its security [3, 4].

On the other hand, in the common reference string (CRS) model, any functionality can be realized in a universally composable way, regardless of the number of corrupted parties [3, 5]. However, the CRS model requires a trusted party who generates the CRS.

Katz introduced an alternative setup assumption which uses tamper-proof hardware tokens (i.e., smartcards) in order to eliminate such a trusted party [15]. In this model, each party can send a hardware token T implementing any polynomial-time functionality F to the other parties, and an adversary can do no more than observe the input/output of this token. This physical setup assumption has been studied extensively recently.

[1] This problem was raised in [7]. But [7] was withdrawn from ePrint on February 5, 2013.

	single (or bounded) use OT	multi-session OT
stateful tokens	[14] [8]	This paper
stateless tokens	[14]	

Table 1: UC secure OT protocols using tamper-proof hardare tokens

1.1 Token Based Commitment

Katz [15] showed a commitment protocol under the DDH assumption using stateful tokens. Moran and Segev improved his protocol in several directions [17]. In particular, they showed an unconditionally secure commitment protocol.

Chandran et al. showed a commitment protocol using stateless tokens based on enhanced trapdoor permutations [6]. It uses a concurrent zero knowledge protocol of [20], and runs in $\tilde{\Theta}(\log n)$ rounds, where n is the security parameter.

Goyal et al. [13] showed that a single stateless token is sufficient to implement statistically secure commitments and statistical zero-knowledge. Furthermore, if stateless tokens can be encapsulated into other stateless tokens, general statistically secure composable multi-party computation is possible in this setting.

1.2 Our Contribution

To summarize, three UC-secure OT protocols using hardware tokens are known, a stateful token OT [14], a stateless token OT [14] and a stateful token OT [8]. However, in these protocols, the same tokens cannot be reused in arbitrarily many OT sessions.

In this paper, we show the first UC-secure multi-session OT protocol. The sender and the receiver exchange tokens only at the beginning, and these tokens are reused in arbitrarily many sessions of OT. An instantiation of the proposed scheme is UC-secure against static adversaries under the DDH assumption and the RSA assumption in the random oracle model. It uses 2 stateful tokens and $2n$ stateless tokens, where n is the security parameter. Further each session of OT runs in constant rounds. (As shown in Table 1, the previous OT's are only for single (or bounded) use.)

Theoretically speaking, we can construct a UC-secure multi-session OT protocol from a UC-secure commitment protocol [5]. However, the resulting protocol is very inefficient.

Our protocol is obtained by combining an OT protocol of Naor-Pinkas [18] and the token-based commitment protocol of Katz [15] in a nice way. At the same time, we present a technique which can prevent the selective randomness failure attack.

2 Preliminaries

2.1 Naor-Pinkas OT protocol

Naor and Pinkas [18] showed an OT protocol based on the DDH assumption such as follows. Suppose that the sender has two strings (m_0, m_1) of length ℓ and the receiver has a choice bit σ as their inputs. Let G be a cyclic group of prime order p, and let g be a generator. Let $H : G \to \{0,1\}^\ell$ be a hash function.

step 1. The receiver chooses $a, b, c \in Z_p$ randomly, and computes

$$x = g^a, y = g^b, z_\sigma = g^{ab}, z_{1-\sigma} = g^c. \tag{1}$$

He then sends (x, y, z_0, z_1) to the sender.

step 2. The sender verifies that $z_0 \neq z_1$. She then chooses (r, u, r', u') randomly and computes

$$E_0 = (g^r x^u, m_0 \oplus H(y^r z_0^u)) \tag{2}$$
$$E_1 = (g^{r'} x^{u'}, m_1 \oplus H(y^{r'} z_1^{u'})) \tag{3}$$

She sends (E_0, E_1) to the receiver.

step 3. From $E_\sigma = (v, w)$, the receiver computes m_σ as $m_\sigma = w \oplus H(v^b)$.

2.2 Ideal Functionality $\mathcal{F}_{\mathsf{multi-OT}}$

In a multi-session OT protocol, the sender and the receiver can run arbitrarily many sessions of OT. In the UC framework, the ideal functionality $\mathcal{F}_{\mathsf{multi-OT}}$ interacts with sender S, receiver R, and the adversary as follows [7].

From sender. Upon receiving an input $(\mathtt{send}, \mathtt{sid}, S, R, \mathtt{ssid}, (m_0, m_1))$ from S, where $m_0, m_1 \in \{0,1\}^\ell$, do:

1. Send $(\mathtt{send}, \mathtt{sid}, S, R, \mathtt{ssid})$ to R and the adversary.
2. Store $(\mathtt{ssid}, m_0, m_1)$.

From receiver. Upon receiving an input $(\mathtt{receive}, \mathtt{sid}, S, R, \mathtt{ssid}, \sigma)$ from R, where $\sigma \in \{0,1\}$, do:

1. Send $(\mathtt{receive}, \mathtt{sid}, S, R, \mathtt{ssid})$ to S and the adversary.
2. Store $(\mathtt{ssid}, \sigma)$.

Go. Upon receiving an input $(\mathtt{Go}, \mathtt{sid}, S, R, \mathtt{ssid})$ from the adversary, do:

1. Send $(\mathtt{sid}, S, R, \mathtt{ssid}, m_\sigma)$ to R.
2. Send $(\mathtt{received}, \mathtt{sid}, S, R, \mathtt{ssid})$ to S.

2.3 Ideal Functionality $\mathcal{F}_{wrap}$

The ideal functionality of stateless hardware token $\mathcal{F}_{wrap}^{stateless}$, parameterized by a polynomial $p(\cdot)$ and an implicit security parameter n, is described as follows [15, 14].

Create. Upon receiving $(\mathtt{create}, \mathtt{sid}, P_i, P_j, \mathtt{mid}, M)$ from P_i, where M is a Turing machine and $\mathtt{mid}$ is machine id, do:

1. Send $(\texttt{create}, \texttt{sid}, P_i, P_j, \texttt{mid})$ to P_j.
2. Store $(P_i, P_j, \texttt{mid}, M)$.

Execute. Upon receiving $(\texttt{run}, \texttt{sid}, P_i, \texttt{mid}, \texttt{msg})$ from P_j, find the unique stored tuple $(P_i, P_j, \texttt{mid})$. If no such tuple exists, do nothing. Run $M(\texttt{msg})$ for at most $p(n)$ steps, and let $\texttt{out}$ be the response ($\texttt{out} = \perp$ if M does not halt in $p(n)$ steps). Send $(\texttt{sid}, P_i, \texttt{mid}, \texttt{out})$ to P_j.

The ideal functionality of stateful hardware token $\mathcal{F}_{wrap}^{stateful}$ is defined similarly [15].

2.4 Randomness Extractor [10, 11, 9]

Let $\mathcal{H}_\infty(A)$ denote the min-entropy of a random variable A. That is, $\mathcal{H}_\infty(A) = -\log(\max_a(\Pr[A = a])$. If $\mathcal{H}_\infty(A) \geq m$, then the random variable A is called an m-source.

The conditional min-entropy of A given B is defined by

$$\mathcal{H}_\infty(A \mid B) = -\log E_b \left[\max_a \Pr[A = a \mid B = b]\right]$$

The statistical distance between two random variables A, B is defined as

$$\mathsf{SD}(A, B) = \frac{1}{2} \sum_v |\Pr(A = v) - \Pr(B = v)|.$$

We write $A \approx_\epsilon B$ if $\mathsf{SD}(A, B) \leq \epsilon$. Let U_ℓ denote the uniform distribution over $\{0,1\}^\ell$.

Definition 2.1 *A randomized function* $\mathsf{Ext} : \mathcal{M} \times \{0,1\}^s \to \{0,1\}^\ell$ *with randomness of length* s *is called an* (m, ℓ, ϵ)*-strong extractor if for any* m*-source* W *on* $\mathcal{M}$*.*

$$(\mathsf{Ext}(W; U_s), U_s) \approx_\epsilon (U_\ell, U_s),$$

where a value of U_s *is called a seed.*

The leftover hash lemma states that if $\mathsf{Ext} : \mathcal{M} \times \{0,1\}^s \to \{0,1\}^\ell$ is a universal function with

$$\ell = m + 2 - 2\log(\frac{1}{\epsilon}),$$

then it is a (m, ℓ, ϵ)-strong extractor. Here Ext is called universal if for any w_1, w_2,

$$\Pr_{\texttt{seed}}[\mathsf{Ext}(w_1; \texttt{seed}) = \mathsf{Ext}(w_2; \texttt{seed})] = 2^{-\ell}.$$

It is generalized to conditional min-entropy without any loss [9, Lemma 2.4]: for any X (possibly dependent on W), if $\mathcal{H}_\infty(W \mid X) \geq m$ and $\ell = m + 2 - 2\log(\frac{1}{\epsilon})$, then

$$(\mathsf{Ext}(W; U_s), U_s, X) \approx_\epsilon (U_\ell, U_s, X).$$

This is called the generalized leftover hash lemma.

3 Our Idea

In this section, we show the idea of our UC-secure multi-session OT protocol which uses tamper-proof hardware tokens. Let n be the security parameter.

3.1 Basic Scheme

Katz [15] showed a UC-secure commitment scheme using stateful hardware tokens. We notice that his commit phase is closely related to the OT protocol of Naor-Pinkas [18] in the following sense.

- In his commit phase, a party P sends a statistically-binding commitment $\texttt{Scom}(\sigma)$ to another party P' to commit to a bit σ.
- In the OT protocol of Naor-Pinkas, the receiver sends (x, y, z_0, z_1) of eq.(1) to the sender at step 1. Here (x, y, z_0, z_1) can be considered as a statistically-binding commitment of the choice bit σ.

Our basic idea is to combine these two schemes in such a way that $\texttt{Scom}(\sigma) = (x, y, z_0, z_1)$. Namely, let $\mathsf{Katz\text{-}Commit}$ denote the commit phase of the Katz protocol such that $\texttt{Scom}(\sigma) = (x, y, z_0, z_1)$. Then our basic scheme is as follows.

(Our Basic Scheme)

1. The receiver commits a choice bit σ by running $\mathsf{Katz\text{-}Commit}$.
2. The sender computes (E_0, E_1) as shown in eq.(2) and eq.(3), and sends them to the receiver.

To prove the UC-security, we must show a simulator Sim which can extract σ from a corrupted receiver, and extract (m_0, m_1) from a corrupted sender. To extract σ from a corrupted receiver, we can use the simulator given by Katz in [15] because we use his commit phase to commit to σ.

However, we cannot construct a simulator which can extract (m_0, m_1) from a corrupted sender. Hence the above scheme is not UC-secure yet.

3.2 How to Extract (m_0, m_1)

We modify our basic scheme as follows. Let $\Sigma = (\texttt{Gen}, \texttt{Sign}, \texttt{Verify})$ be a secure signature scheme. In the setup phase, the receiver R generates public-key/secret-key $(\texttt{PK}_2, \texttt{SK}_2)$ of Σ, and sends to the sender S a stateless token T_R in which $\texttt{SK}_2$ is embedded.

(Our First Attempt)

Token	Sender	Receiver

1. R commits a choice bit σ by running $\mathsf{Katz\text{-}Commit}$.
2. S computes (E_0, E_1) and sends them to R.
3. S sends $X = (r, u, r', u', m_0, m_1)$ to T_R.
4. T_R computes (E_0, E_1) from X, and returns $\tau = \texttt{Sign}_{\texttt{SK}_2}(E_0, E_1)$.

5. S sends τ to R as the evidence that S sent X to T_R.

Now a simulator can extract (m_0, m_1) from a corrupted sender just by observing the sender's query $X = (r, u, r', u', m_0, m_1)$ to T_R.

However, this protocol is broken by the so called selective input failure attack: A malicious receiver sends a malicious token T_R to the sender which returns $\perp$ for some subset of (m_0, m_1). Then the receiver can learn some information on (m_0, m_1) (hence on $m_{1-\sigma}$) by observing if the sender aborts or not.

(Our Second Attempt)

Token Sender Receiver

1. R commits a choice bit σ by running Katz-Commit.

2. S computes (E_0, E_1) and sends them to R.

3. S sends $Y = (r, u, r', u')$ to T_R.

4. T_R returns $\tau = \mathtt{Sign}_{\mathtt{SK}_2}(g^r x^u, g^{r'} x^{u'})$.

5. S sends τ to R as the evidence that S sent Y to T_R.

Now this protocol is secure against the selective input failure attack because (m_0, m_1) is not the input to the token T_R. Also a simulator can compute (m_0, m_1) from (E_0, E_1) and $Y = (r, u, r', u')$. However, it is vulnerable to the selective randomness failure attack such as follows. A malicious receiver sends a malicious token T_R to the sender which returns $\perp$ for some subset of (r, u, r', u'). In other words, the receiver can control the distribution of the randomness (r, u, r', u'). Then the receiver would learn some information on $m_{1-\sigma}$.

We solve this problem as follows. In the setup phase, R sends $2n$ stateless tokens $T_R^1, \cdots, T_R^{2n}$ to S such that $\mathtt{SK}_2$ is embedded in each T_R^i.

(Final Scheme)

Tokens Sender Receiver

1. R commits a choice bit σ by running Katz-Commit.

2. S choose $r_1, \cdots, r_{2n}, u_1, \cdots, u_{2n}$ such that

$$r = r_1 + \cdots + r_n \bmod p,\ r' = r_{n+1} + \cdots + r_{2n} \bmod p,$$

$$u = u_1 + \cdots + u_n \bmod p,\ u' = u_{n+1} + \cdots + u_{2n} \bmod p$$

randomly, and queries (r_i, u_i) to T_R^i for all i.

3. T_R^i returns $\tau_i = \mathtt{Sign}_{\mathtt{SK}_2}(g^{r_i} x^{u_i})$ for $i = 1, \cdots, 2n$.

4. S sends $(g^{r_i} x^{u_i}, \tau_i)$ to R for $i = 1, \cdots, 2n$.

S also sends to R

$$w_0 = m_0 \oplus \mathsf{Ext}(y^r z_0^u),\ w_1 = m_1 \oplus \mathsf{Ext}(y^{r'} z_1^{u'})$$

Now suppose that $\sigma = 1$ and $m_{1-\sigma} = m_0$. Then from a view point of the receiver, we can prove that the min-entropy of $y^r z_0^u$ is greater than $\log(p) - 1$ no matter how malicious tokens $T_R^1, \cdots, T_R^{2n}$ behave. Hence $\mathsf{Ext}(y^r z_0^u)$ is a random string due to the (generalized) leftover hash lemma. Hence S learns nothing on $m_{1-\sigma} = m_0$.

Also our simulator can obtain all (r_i, u_i) by observing the sender's queries to T_R^i, and then compute (m_0, m_1).

3.3 On Signature Scheme

The last problem is that a malicious token T_R^i may output a signature τ_i which includes the information about (r_i, u_i) if the signing algorithm is probabilistic. Then the receiver would learn some information on $m_{1-\sigma}$. We can prevent this covert channel attack if the signing algorithm is deterministic.

4 UC-Secure Multi-Session OT Using Tokens

In this section, we show the first UC-secure multi-session OT protocol using hardware tokens. The sender and the receiver exchange tokens only at the setup phase. These tokens are reused in arbitrarily many sessions of OT.

Let p and $q = 2p + 1$ be two primes. Let G be the subgroup of Z_q^* such that $|\mathsf{G}| = p$.

4.1 Setup Phase

The first half of our setup phase is the same as the setup phase of Katz [15]. At a high level,

1. S and R exchange stateful tokens $\mathtt{mid}_S$ and $\mathtt{mid}_R$.
2. R and $\mathtt{mid}_S$ jointly generate $\mathtt{tuple}_S = (g, h, \hat{g}, \hat{h}) \in \mathsf{G}^4$, where g and h are generators of G. R then sends $\mathtt{tuple}_S$ to S.
3. S acts symmetrically, and sends $\mathtt{tuple}_R$ to R.

In the second half of our setup phase, R sends $2n$ stateless tokens $\mathtt{mid}_{R_1}, \cdots, \mathtt{mid}_{R_{2n}}$ to S as follows. Let $\Sigma = (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ be an unforgeable signature scheme against chosen message attack such that the signing algorithm $\mathtt{Sign}$ is deterministic.

1. R generates a public-key/secret-key $(\mathtt{PK}_2, \mathtt{SK}_2)$ of Σ.
2. For $i = 1, \cdots, 2n$, R sends $(\mathtt{create}, \mathtt{sid}, \mathit{receiver}, \mathit{sender}, \mathtt{mid}_{R_i}, M_{ot})$ to $\mathcal{F}_{wrap}^{stateless}$, where M_{ot} implements the following functionality: On input $(ssid, g, x, r, u)$, output $\tau = \mathtt{Sign}_{\mathtt{SK}_2}(ssid, g, x, g^r x^u)$, where $g, x \in \mathsf{G}$ and $r, u \in Z_p$.

4.2 Oblivious Transfer Phase

In each session of OT, the sender is given $(ssid, m_0, m_1)$ and the receiver is given $(ssid, \sigma)$ by an environment $\mathcal{Z}$, where $m_0, m_1 \in \{0,1\}^\ell$ and $\sigma \in \{0,1\}$. Let $\mathsf{Ext} : G \times \{0,1\}^s \to \{0,1\}^\ell$ be a universal hash function which is an $(\log(p) - 1, \ell, \epsilon)$-strong extractor (see Sec.2.4).

At step B-1, R commits the choice bit σ by using the commit phase of Katz [15] such that $\texttt{Scom}(\sigma) = (x, y, z_0, z_1)$. Given $\texttt{tuple} = (g, h, \hat{g}, \hat{h})$, let $\mathsf{Com}_{\texttt{tuple}}(\sigma) = (g^{s_1}h^{s_2}, \hat{g}^{s_1}\hat{h}^{s_2}g^{\sigma})$, where s_1 and s_2 are randomly chosen from Z_p.

(B-1: Katz-Commit) R chooses a, b, c randomly from Z_p, and computes

$$x = g^a, y = g^b, z_\sigma = g^{ab}, z_{1-\sigma} = g^c$$

Let $\texttt{Scom}(\sigma) = (x, y, z_0, z_1)$. He also computes $\mathsf{Com}_{\texttt{tuple}_R}(\sigma)$ randomly, and sends $\texttt{Scom}(\sigma)$ and $\mathsf{Com}_{\texttt{tuple}_R}(\sigma)$ to S.

He then gives an interactive witness indistinguishable proof that either (i) both $\texttt{Scom}(\sigma)$ and $\mathsf{Com}_{\texttt{tuple}_R}(\sigma)$ are commitments to the same bit σ, or (ii) $\texttt{tuple}_S$ is a DH tuple. (We can construct a constant round protocol of this easily.)

(B-2) S aborts if $z_0 \neq z_1$. Otherwise for $i = 1, \cdots, 2n$, she chooses $r_i, u_i \in Z_p$ randomly, and queries $(ssid, g, x, r_i, u_i)$ to $\texttt{mid}_{R_i}$ to obtain

$$\tau_i = \texttt{Sign}_{\texttt{SK}_2}(ssid, g, x, g^{r_i}x^{u_i}).$$

(B-3) She aborts if $\texttt{mid}_{R_i}$ returns $\perp$ or an invalid signature τ_i for some i.

Otherwise for $i = 1, \cdots, 2n$, she sends τ_i and $t_i = g^{r_i}x^{u_i}$ to R.

Also she computes

$$r = r_1 + \cdots + r_n \bmod p,\ r' = r_{n+1} + \cdots + r_{2n} \bmod p,$$
$$u = u_1 + \cdots + u_n \bmod p,\ u' = u_{n+1} + \cdots + u_{2n} \bmod p.$$
$$v_0 = g^r x^u,\ v_1 = g^{r'} x^{u'}$$
$$w_0 = m_0 \oplus \mathsf{Ext}(y^r z_0^u),\ w_1 = m_1 \oplus \mathsf{Ext}(y^{r'} z_1^{u'}),$$

and sends (w_0, w_1) to R together with the seeds of Ext.

(B-4) R aborts if $\texttt{Verify}_{\texttt{PK}_2}((ssid, g, x, t_i), \tau_i) = reject$ for some i.

Otherwise he computes

$$v_0 = t_1 \times \cdots \times t_n,\ v_1 = t_{n+1} \times \cdots \times t_{2n}$$

and outputs $m_\sigma = w_\sigma \oplus \mathsf{Ext}(v_\sigma^b)$,

5 UC security

In this section, we prove the UC-security of our multi-session OT protocol in the $(\mathcal{F}_{wrap}^{stateful}, \mathcal{F}_{wrap}^{stateless})$-hybrid model against static adversaries.

In the real world, an environment $\mathcal{Z}$ runs our multi-session OT protocol, where the hardware tokens are idealized by $\mathcal{F}_{wrap}^{stateful}$ and $\mathcal{F}_{wrap}^{stateless}$. In the ideal world, $\mathcal{Z}$ interacts with the dummy sender and the dummy receiver, where the dummy players communicate with the ideal functionality $\mathcal{F}_{\mathsf{multi-OT}}$ which is given in Sec.2.2.

Let Sim_S denote the simulator for a corrupted sender, and Sim_R denote the simulator for a corrupted receiver given by Katz for his UC-secure commitment protocol [15]. Note that our setup phase is the same as that of Katz [15] (except step 5), and step B-1 of our oblivious transfer phase is the same as the commit phase of Katz [15].

5.1 Sender corruption

Suppose that the sender is corrupted by an adversary A in the real world. We show a simulator Sim in the ideal world which simulates A. Our Sim runs an internal copy of A playing a role of the honest receiver, forwarding all messages from $\mathcal{Z}$ to A and vice versa. (This means that Sim generates $(\mathtt{PK}_2, \mathtt{SK}_2)$ honestly.) Let L be a list which is empty first.

(S-1) In the setup phase and at step B-1, Sim behaves in the same way as Sim_S.

(S-2) At step B-2, if A queries $(ssid, g, x, r, u)$ to some $\mathtt{mid}_{R_i}$, then Sim computes $t = g^r x^u$ and $\tau = \mathtt{Sign}_{\mathtt{SK}_2}(ssid, g, x, t)$, and stores $[(ssid, x, r, u), (t, \tau)]$ in L.

(S-3) At step B-3, suppose that A sends $(t_1, \tau_1), \cdots, (t_{2n}, \tau_{2n})$, (w_0, w_1) and the seeds of Ext to the receiver such that each τ_i is a valid signature.

If there exists some (t_i, τ_i) which does not appear in L, then Sim aborts.

(S-4) Otherwise Sim finds (r_i, u_i) such that $(ssid, x, r_i, u_i), (t_i, \tau_i)] \in L$ for $i = 1, \cdots, 2n$. By using these (r_i, u_i), Sim computes (v_0, v_1) in the same way as step B-3, and computes (m_0, m_1) as

$$m_0 = w_0 \oplus \mathsf{Ext}(v_0^b),\ m_1 = w_1 \oplus \mathsf{Ext}(v_1^b). \tag{4}$$

(S-5) Sim sends the above (m_0, m_1) to $\mathcal{F}_{\mathsf{multi-OT}}$.

Theorem 5.1 *Suppose that the underlying signature scheme Σ is unforgeable against chosen message attack and the signing algorithm is deterministic. Then no environment $\mathcal{Z}$ can distinguish between the ideal world and the real world under the DDH assumption.*

(Proof) We consider a sequence of games.

Game_0**:** In this game, $\mathcal{Z}$ interacts with a simulator Sim_0 only. That is, Sim_0 receives both (m_0, m_1) and σ from $\mathcal{Z}$, and Sim_0 plays both roles of the corrupted sender A and the honest receiver. It then internally simulates a real execution of the protocol between A and the receiver. Clearly Game_0 is identical to the real world.

Game_1**:** In this game, a simulator Sim_1 behaves in the same way as as Sim_0 except for that Sim_1 does (S-2). It is clear that Game_0 and Game_1 are identical from a view point of $\mathcal{Z}$.

Game_2**:** In this game, a simulator Sim_2 behaves in the same way as as Sim_1 except for that Sim_2 does (S-3). Since the signature scheme Σ is unforgeable, Game_1 and Game_2 are indistinguishable.

Game_3**:** In this game, a simulator Sim_3 behaves in the same way as as Sim_2 except for that Sim_3 does (S-4). Sim_3 then outputs m_σ of eq.(4) as the output of the receiver. It is easy to see that the receiver outputs the same m_σ as in Game_2. Hence Game_2 and Game_3 are identical.

Game_4**:** In this game, a simulator Sim_4 behaves in the same way as as Sim_3 except for that Sim_4 does (S-1). Then Game_3 and Game_4 are indistinguishable under the DDH assumption as shown in [15].

Game_5**:** This is the ideal world. In particular, Sim extracts (m_0, m_1) as shown in (S-4), and sends it to $\mathcal{F}_{\mathsf{multi-OT}}$. The output of the receiver in this case is exactly the output in Game_4. Thus this game is identical to Game_4.

Therefore no $\mathcal{Z}$ can distinguish between the real world and the ideal world under the DDH assumption. Q.E.D.

5.2 Receiver corruption

Suppose that the receiver is corrupted by an adversary A in the real world. We show a simulator Sim in the ideal world which simulates A. Our Sim runs an internal copy of A playing a role of the honest sender, forwarding all messages from $\mathcal{Z}$ to A and vice versa. Let L be a list which is empty first.

(R-1) In the setup phase and at step B-1, Sim behaves in the same way as Sim_R. Note that Sim_R can extracts σ from A as shown in [15].

(R-2) Sim sends the above σ to $\mathcal{F}_{\mathsf{multi-OT}}$, and receives m_σ.

(R-3) Let $m_{1-\sigma}$ be a random string. Sim uses this $(m_\sigma, m_{1-\sigma})$ at step B-3.

Theorem 5.2 *No environment $\mathcal{Z}$ can distinguish between the ideal world and the real world under the DDH assumption.*

(Proof) We consider a sequence of games.

Game_0**:** In this game, $\mathcal{Z}$ interacts with a simulator Sim_0 only. That is, Sim_0 receives both (m_0, m_1) and σ from $\mathcal{Z}$, and Sim_0 plays both roles of the corrupted receiver A and the honest sender. It then internally simulates a real execution of the protocol between A and the sender. Clearly Game_0 is identical to the real world.

Game_1**:** In this game, a simulator Sim_1 behaves in the same way as as Sim_0 except for that Sim_1 does (R-1) and extracts σ. Then as shown in [15], Game_0 and Game_1 are indistinguishable under the DDH assumption.

Game_2**:** In this game, a simulator Sim_2 behaves in the same way as Sim_1 except for that Sim_2 lets $m_{1-\sigma}$ be a random string. We will show that Game_1 and Game_2 are indistinguishable below.

Game_3**:** This is the ideal world. It is easy to see that Game_2 and Game_3 are indistinguishable.

We finally prove that Game_1 and Game_2 are indistinguishable. Let T_i be the set of (r_i, u_i) such that (a malicious) $\mathtt{mid}_{R_i}$ returns τ_i correctly on input $(ssid, g, x, r_i, u_i)$. [2]

Lemma 5.1 *If $|T_i| < p^2/2$ for $i = 1, \cdots, n$, then the sender aborts at step B-3 with probability more than $1 - 1/2^n$.*

[2] The definition of T_i is meaningful even for (a malicious) $\mathtt{mid}_{R_i}$ which is stateful or probabilistic. Just fix the current state and the current randomness.

(Proof) Let RET_i be the event that $\mathtt{mid}_{R_i}$ returns τ_i correctly. Then

$$\Pr(\mathsf{RET}_i) = |T_i|/p^2 < 1/2$$

because the honest sender chooses (r_i, u_i) randomly from Z_p^2. Let RET be the event that all $\mathtt{mid}_{R_i}$ return τ_i correctly. Note that each RET_i is an independent event because the honest sender chooses (r_i, u_i) independently. Therefore

$$\Pr(\mathsf{RET}) = \Pr(\mathsf{RET}_1 \wedge \cdots \wedge \mathsf{RET}_n) < 1/2^n.$$

Hence the sender aborts at step B-3 with probability more than $1 - 1/2^n$.

Q.E.D.

Let $\mathcal{R}$ denote the random variable of (r, u), and V_0 denote the random variable of v_0.

Lemma 5.2 *If $|T_i| \geq p^2/2$ for some $i \in \{1, \cdots, n\}$, then*

$$\max_{(r,u)} \Pr(\mathcal{R} = (r, u)) \leq 2/p^2.$$

(Proof) Wlog, suppose that $|T_1| \geq p^2/2$. Let X denote the random variable of (r_1, u_1) and X' denote the random variable of $(r_2 + \cdots + r_n, u_2 + \cdots + u_n)$. Hence $\mathcal{R} = X + X' \bmod p$. Then for any $(r, u) \in Z_p^2$,

$$\begin{aligned}
\Pr(\mathcal{R} = (r, u)) &= \Pr(X + X' = (r, u) \bmod p) \\
&= \sum_{(\alpha,\beta)\in T_1} \Pr(X = (\alpha, \beta)) \Pr(X' = (r - \alpha, u - \beta) \bmod p) \\
&= 1/|T_1| \sum_{(\alpha,\beta)\in T_1} \Pr(X' = (r - \alpha, u - \beta) \bmod p) \\
&\leq 2/p^2 \sum_{(\alpha,\beta)\in T_1} \Pr(X' = (r - \alpha, u - \beta) \bmod p) \leq 2/p^2.
\end{aligned}$$

Q.E.D.

Lemma 5.3 *If $|T_i| \geq p^2/2$ for some $i \in \{1, \cdots, n\}$, then*

$$\mathcal{H}_\infty(\mathcal{R} \mid V_0) \geq \log(p) - 1.$$

(Proof) Note that

$$\mathcal{H}_\infty(\mathcal{R} \mid V_0) = -\log E_v \left[\max_{(r,u)} \Pr[\mathcal{R} = (r, u) \mid V_0 = v]\right]$$

Let $q_v = \max_{(r,u)} \Pr[\mathcal{R} = (r, u) \mid V_0 = v]$. Suppose that q_v is given by $(r, u) = (r_v, u_v)$. Then v is uniquely determined by (r_v, u_v) as $v = g^{r_v} x^{u_v}$. Therefore

$$q_v = \Pr[\mathcal{R} = (r_v, u_v) \mid V_0 = v] = \frac{\Pr[\mathcal{R} = (r_v, u_v), V_0 = v]}{\Pr(V_0 = v)} = \frac{\Pr[\mathcal{R} = (r_v, u_v)]}{\Pr(V_0 = v)}$$

Hence

$$\begin{aligned} E_v[q_v] &= \sum_{v \in Z_p} \frac{\Pr[\mathcal{R} = (r_v, u_v)]}{\Pr(V_0 = v)} \times \Pr(V_0 = v) \\ &= \sum_{v \in Z_p} \Pr[\mathcal{R} = (r_v, u_v)] \le p \times 2/p^2 = 2/p \end{aligned}$$

from Lemma 5.2. Therefore $\mathcal{H}_\infty(R \mid V_0) = -\log E_v[q_v] \ge \log(p) - 1$. Q.E.D.

Lemma 5.4 *Fix v_0 arbitrarily. If $z_0 \neq g^{ab}$, then $f(r,u) = y^r z_0^u$ is an injection from $\{(r,u) \mid v_0 = g^r x^u\}$ to G.*

(Proof) Let $g^r x^u = g^\alpha$ and $y^r z_0^u = g^\beta$. Then

$$\begin{pmatrix} 1 & a \\ b & c \end{pmatrix} \begin{pmatrix} r \\ u \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

where $x = g^a, y = g^b, z_0 = g^c$ and α is fixed. Since $z_0 \neq g^{ab}$, the 2×2 matrix of the left hand side is nonsingular. This means that $f(r,u) = y^r z_0^u$ is an injection from $\{(r,u) \mid v_0 = g^r x^u\}$ to G. Q.E.D.

Lemma 5.5 *Game_1 and Game_2 are indistinguishable.*

(Proof)

(Case 1) $|T_i| < p^2/2$ for $i = 1, \cdots, n$.

In this case, from Lemma 5.1, the receiver aborts with probability more than $1 - 1/2^n$ at step B-3. Hence Game_1 and Game_2 are indistinguishable.

(Case 2) $|T_i| > p^2/2$ for some $i \in \{1, \cdots, n\}$.

Wlog, suppose that $\sigma = 1$. Then $\mathcal{H}_\infty((r,u) \mid V_0) \ge \log(p) - 1$ from Lemma 5.3. This means that $\mathcal{H}_\infty(y^r z_0^u \mid V_0) \ge \log(p) - 1$ from Lemma 5.4. Therefore $(\mathsf{Ext}(y^r z_0^u), seed, v_0)$ and $(\mathtt{rand}, seed, v_0)$ are indistinguishable from the generalized leftover hash lemma (see Sec.2.4), where $\mathtt{rand}$ is a random string of length ℓ and $seed$ is the seed of the extractor Ext.

Hence Game_1 and Game_2 are indistinguishable. Q.E.D.

Therefore no $\mathcal{Z}$ can distinguish between the real world and the ideal world. Q.E.D.

5.3 UC-Security

The full domain hash (FDH) RSA signature scheme [1] is unforgeable against chosen message attack under the RSA assumption in the random oracle model, and its signing algorithm is deterministic. Therefore we have the following corollary.

Corollary 5.1 *Suppose that Σ is the FDH RSA signature scheme. Then our protocol UC-realizes the ideal functionality $\mathcal{F}_{\mathsf{multi-OT}}$ in the $(\mathcal{F}_{wrap}^{stateful}, \mathcal{F}_{wrap}^{stateless})$-hybrid model against static adversaries under the DDH assumption and the RSA assumption in the random oracle model.*

References

[1] Mihir Bellare, Phillip Rogaway: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. EUROCRYPT 1996: pp399-416

[2] Ran Canetti, Universally Composable Security: A New Paradigm for Cryptographic Protocols. FOCS 2001. Full version available from Cryptology ePrint Archive, Report 2000/067 http://eprint.iacr.org/

[3] R. Canetti and M. Fischlin. Universally Composable Commitments. Crypto 2001.

[4] R. Canetti, E. Kushilevitz, and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. J. Cryptology, 19(2), pp.135-167 (2006)

[5] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, Amit Sahai: Universally composable two-party and multi-party secure computation. STOC 2002, pp.494-503 (2002)

[6] Nishanth Chandran, Vipul Goyal, Amit Sahai: New Constructions for UC Secure Computation Using Tamper-Proof Hardware. EUROCRYPT 2008, pp.545-562 (2008)

[7] Seung Geol Choi and Jonathan Katz and Dominique Schroder and Arkady Yerukhimovich and Hong-Sheng Zhou: (Efficient) Universally Composable Two-Party Computation Using a Minimal Number of Stateless Tokens. Cryptology ePrint Archive, Report 2011/689 (2011), **Withdrawn on February 5, 2013**

[8] Nico Dottling, Daniel Kraschewski, Jorn Muller-Quade: Unconditional and Composable Security Using a Single Stateful Tamper-Proof Hardware Token. TCC 2011, pp.164-181 (2011)

[9] Yevgeniy Dodis, Leonid Reyzin, Adam Smith: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. Cryptology ePrint Archive, Report 2003/235 (2008)

[10] Yevgeniy Dodis, R.Ostrovsky, Leonid Reyzin, Adam Smith: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. EUROCRYPT 2004, pp.523-540 (2004)

[11] Yevgeniy Dodis, Leonid Reyzin and Adam Smith: Fuzzy Extractors: A Brief Survey of Results from 2004 to 2006. http://www.cs.bu.edu/ reyzin/papers/fuzzysurvey.pdf

[12] S.Even, O.Goldreich, A.Lempel: A Randomized Protocol for Signing Contracts. Commun. ACM 28(6): 637-647 (1985)

[13] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, Amit Sahai: Interactive Locking, Zero-Knowledge PCPs, and Unconditional Cryptography. CRYPTO 2010, pp.173-190 (2010)

[14] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, Akshay Wadia: Founding Cryptography on Tamper-Proof Hardware Tokens. TCC 2010, pp.308-326 (2010)

[15] Jonathan Katz: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. EUROCRYPT 2007: 115-128 (2007)

[16] Vladimir Kolesnikov: Truly Efficient String Oblivious Transfer Using Resettable Tamper-Proof Tokens. TCC 2010: 327-342 (2010)

[17] Tal Moran, Gil Segev: David and Goliath Commitments: UC Computation for Asymmetric Parties Using Tamper-Proof Hardware. EUROCRYPT 2008, pp.527-544 (2008)

[18] Moni Naor, Benny Pinkas: Efficient oblivious transfer protocols. SODA 2001. pp.448-457 (2001)

[19] Torben P. Pedersen: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. CRYPTO 1991, pp.129-140

[20] Manoj Prabhakaran, Alon Rosen, Amit Sahai: Concurrent Zero Knowledge with Logarithmic Round-Complexity. FOCS 2002, pp.366-375 (2002)