Robust RFID Authentication Protocol with Formal Proof and Its Feasibility

Miyako Ohkubo[1], Shin'ichiro Matsuo[1], Yoshikazu Hanatani[2,3], Kazuo Sakiyama[2] and Kazuo Ohta[2]

[1] National Institute of Information and Communication Technology
[2] The University of Electro-Communications
[3] Corporate Reserch & Development Center, Toshiba Corporation.

Abstract. The proloferation of RFID tags enhances everyday activities, such as by letting us reference the price, origin and circulation route of specific goods. On the other hand, this lecel of traceability gives rise to new privacy issues and the topic of developing cryptographic protocols for RFID- tags is garnering much attention. A large amount of research has been conducted in this area. In this paper, we reconsider the security model of RFID- authentication with a man-in-the-middle adversary and communication fault. We define model and security proofs via a game-based approach makes our security models compatible with formal security analysis tools. We show that an RFID authentication protocol is robust against the above attacks, and then provide game-based (hand-written) proofs and their verification by using CryptoVerif.
Keywords: RFID authentication, security model, formal proofs

1 Introduction

1.1 Background

In recent years, a huge number of the low-power devices called RFID tags, which communicate over wireless channels, have entered into use in our daily lives. In most cases, RFID tags are used for identifying goods, authenticating parties' legitimacy, detecting fakes, and billing for services. For such applications, secure authentication of each RFID tag is fundamental. Also if the output of a tag is fixed or related adifferent authentications, privacy issues arise in which an adversary can trace the tag and activity of the owner. Thus, most research on RFID authentication protocol realizes the importance of tag-unforgeability and forward-privacy. Here, we consider the -technical- feasibility of an RFID authentication protocol in terms of security.

Technical feasibility regarding security: Though a large number of secure protocols are proposed assuming wired networks, the next consideration is how to deal with issues caused via wireless networks. In a wireless network environment in which RFID is used, the adversary has chances to conduct for instance, man-in-the-middle or relay attack. The connection is less stable than in a wired

setting, and thus we must consider robustness against communication errors. We must also construct a security model and definition, secure protocol, and security proofs for such situation to clarify the security strength in actual usage.

1.2 Our Contribution

In this paper, we focus on solving the above issues in terms of feasibility. That is, we study adding robustness to existing tag-unforgeable and forward-private RFID authentication schemes as well as showing efficiency when we implement our scheme for actual RFID tags. Contributions of this paper are mainly on the following four points. (1) We provide a formal security model and definitions that deal with man-in-the-middle adversaries and communication faults. This model is like security models of the key exchange protocol and suitable for rigorously estimating the success probability of attacks. (2) We propose a robust RFID authentication protocol that satisfies the above modeland define security requirements. We choose a hash-based scheme because the cost of computation in a public key-based scheme is very high. Recently distance-bounding protocols have been proposed as a solution that cover relay attacks, but this type of protocol needs many rounds of communications and is not suitable for our setting. Therefore our protocol is mainly based on the OSK protocol, which can provide forward-privacy, and we combine a mechanism that synchronizes the internal status of the tag and the reader. (3) We prove the security and privacy of our proposed scheme. a game-based approach. This approach works favorably as a computationally rigorous proof and also as a formal verification tool. We first divide the security notions into several games, and show that the relation between games preserves them. We also show that the (handwritten) proof is correct by using the CryptoVerif formal verification tool, which helps us to understand the security of cryptographic protocols. As far as we know, this is the first work in the RFID world that defines the security notion and shows the security by a formal verification tool.

2 Related Work

Many schemes exist for secure RFID- authentication that protets privacy; these are summarized in [1]. For security models for RFID- authentication, Juels and Weis first proposed the privacy model [9]; then Vaudenay proposed a classification of security concepts for privacy regarding tag-authentication [13]. Paise and Vaudenay presented a classification of security concepts for mutual-authentication with privacy [12]. One type of RFID- authentication scheme robust against replay attacks and wireless settings is using the distance-bounding protocol [8] by Hancke and Kuhn. Because this protocol needs many rounds of communication we chose another construction.

A major contribution of this paper is proving the security of our scheme using a formal verification tool. Security verification using formal methods has a long history dating to the 1980s. Recently, combining "computational difficulty,"

a major concept in cryptography, and "automated verification," a prime benefit of the formal method, has become main-stream researches in this area. In 2000, Adabi and Rogaway pioneered work on the gap [2], many following works have been proposed. Some practical tools such as "CryptoVerif" [5] [6], which we use in this paper, were also proposed.

3 Security Model and Definitions

3.1 Model

Communication : Communication between servers and clients is provided via a wireless network, upon which third parties can easily eavesdrop and which is easily cut or disturbed.

Client : In this paper, we suppose small devices like passive RFID tag as clients. The clients only have poor electronic power provided by servers and can only perform light calculations. The memory in the client is not resilient against tamper attacks.

Server : We imagine PCs and devices readers as servers. Generally, an RFID system tag communicates with readers through wireless channels, and then the readers communicate with servers through secure channels. We assume that the communication between reader and server is secure by using ordinal cryptographic techniques such as SSL and VPN. Therefore, we describe the communications in an RFID system using two players - client and server.

Functions : Let these be indexes of the client ID, number of updating times of secret key $i \geq 0$, and secret key $sk_{\mathsf{ID},i}$. Intuitively, each function means the following. $\mathsf{KeyGen}()$ is a key generation processes. $\mathsf{FR}()$ is responses from server to client (i.e., tag). $\mathsf{SR}()$ is the returning responses from client (i.e., tag) to server. $\mathsf{Check}_Y()$ means the verification check of the client's output by the server. $\mathsf{TR}()$ means the returning response from server to client (i.e., tag). $\mathsf{Check}_Z()$ is the result of verification check of the server's output by the client. $\mathsf{SK}()$ is the key updating processeses. More formally, each function is defined as follows.

- $\mathsf{KeyGen}(\mathsf{ID}, i)$:
  [Input] ID, $i = 0$ [Output] $\mathrm{p}sk_{\mathsf{ID},0}$ uniquely.
- $\mathsf{FR}(sk_{\mathsf{ID},i}, \mathsf{ID}, i)$:
  [Input] ID, i, and $sk_{\mathsf{ID},i}$. [Output] X.
- $\mathsf{SR}(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X)$:
  [Input] ID, i, $sk_{\mathsf{ID},i}$, and X. [Output] Y.
- $\mathsf{Check}_Y(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y)$:
  [Input] ID, i, $sk_{\mathsf{ID},i}$, X, and Y. [Output] d_Y.
- $\mathsf{TR}(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y)$:
  [Input] ID, i, $sk_{\mathsf{ID},i}$, X, and Y. [Output] Z.
- $\mathsf{Check}_Z(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y, Z)$:
  [Input] ID, i, $sk_{\mathsf{ID},i}$, X, Y, and Z. [Output] d_Z.
- $\mathsf{SK}(sk_{\mathsf{ID},i}, \mathsf{ID}, i)$:
  [Input] ID, $i \geq 0$, and $sk_{\mathsf{ID},i}$. [Output] $sk_{\mathsf{ID},i+1}$ deterministically.

3.2 Definitions

Security notions for (robust) mutual authentication protocols are defined by the success probability of the adversary, which is allowed to access the oracles that follow functions as above. We first show oracles that the adversary can access.

Oracles : $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$ are oracles as server' output and client' output. Others are oracles as functions used in server or client. In the following definitions, ξ_i, η_i and μ_i are any elements.

$\mathcal{S}_f$: [Input] (ID, i) [Output] X
$\mathcal{C}$: [Input] (ID, i, X) [Output] Y
$\mathcal{S}_t$: [Input] (ID, i, X, Y) [Output] Z
$\mathcal{SK}$: [Input] (ID, i) [Output] $sk_{\mathsf{ID},i}$
SR^O : [Input] $(\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$ [Output] Y
TR^O : [Input] $(\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)$ [Output] Z
SK^O : [Input] (μ_1, μ_2, μ_3) [Output] $sk_{\mathsf{ID},i+1}$

Security notions : essentially we are concened with privacy and authenticity as well as identification of property. Moreover, as extended notions we take concerned with forward-security and (self)-synchronization. The security requirements for these notions are defined as follows.

Requests of these extended notions come from the client's (i.e., tag's) original property and/or the communications between client and server. Since the client (i.e., tag) does not have tamper-resilient memory, the adversary can easily acquire the secret data stored in the tag. If the adversary acquires the secret data, it can break not only the tag's current privacy and/or authenticity but also the tag's previous history. Forward-secure property is required to protect the client's (i.e., tag's or tagged person's) privacy and/or authenticity. Moreover, communication through radio can easily to be cut and disturbed. Therefore we should be concerned about the fact that each transaction does not always succeed, and is frequently fails. (Self)-synchronization is required to recover the status in the case the communication between client and server fails.

Definition 1 (Forward-secure (Client-)Indistinguishability). *Adversary $\mathcal{A}_{FI}$ is allowed to access the oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$ SK^O . Adversary $\mathcal{A}_{FI}$ chooses $\hat{X}$ and j randomly, and sends a query with $(\hat{X}, j)$ to challenge oracle $\mathcal{CO}$, and receives Y. Where the challenge oracle $\mathcal{CO}$ flips a coin. In the case of $b = 1$, $\mathcal{CO}$ sends a random values as Y to $\mathcal{A}_{FI}$ In the case of $b = 0$, $\mathcal{CO}$ chooses α randomly, calculates $\mathsf{SR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X})$, and returns Y to $\mathcal{A}_{FI}$. After $\mathcal{A}_{FI}$ receives the challenge, he is allowed to access oracles. Finally, $\mathcal{A}_{FI}$ outputs b'. Let $\frac{1}{2} + \epsilon_{FI}$ as the probability of that b'= b, where each index i of all requests to oracle $\mathcal{SK}$ is $j < i$. (Note that if there are requests to oracle $\mathcal{SK}$ and the indices in these requests are $i \leq j$, it is not included in advantage of adversary $\mathcal{A}_{FI}$). If ϵ_{FI} is negligible, the scheme is Forward-secure Indistinguishable.*

$$
\begin{aligned}
& Pr\,[\mathcal{A}_{FI}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK}\mathsf{SK}^O}\ win] \\
&= Pr[b' = b| \\
&\qquad b' \leftarrow \mathcal{A}_{FI}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\ \mathsf{SK}^O}(Y), \\
&\qquad Y \leftarrow \mathcal{CO}(\hat{X},j), (\hat{X},j) \leftarrow \mathcal{A}_{FI}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\ \mathsf{SK}^O}|j<i] \\
&= \frac{1}{2} + \epsilon_{FI}, \\
&\quad where \epsilon_{FI} << \frac{1}{2^{\mathcal{K}}},\ \mathcal{K}\ is\ security\ parameter.
\end{aligned}
$$

Definition 2 (Forward-secure (Client)-Unforgeability). *Adversary $\mathcal{A}_{CU}$ is allowed to access the oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$, SK^O. Adversary $\mathcal{A}_{CU}$ chooses j, and sends the j to challenge oracle $\mathcal{CO}$, and receives X. After $\mathcal{A}_{CU}$ receives the challenge, it is allowed to access the oracles. Finally, $\mathcal{A}_{CU}$ outputs $\hat{Y}$. If the probability that the output of $\mathsf{Check}_Y(sk_{\mathsf{ID},j}, \mathsf{ID}, j, X, \hat{Y})$, d_Y is 1 (i.e., $\hat{Y}$ is equal to the output of $\mathsf{SR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, X)$), which is negligible, where each index i of all requests to oracle $\mathcal{SK}$ is $j < i$, the scheme is Forward-secure (Client)-Unforgeable. (Note that if there are requests to oracle $\mathcal{SK}$ and the indices in these requirements are $i \leq j$, it is not included in the advantage of adversary $\mathcal{A}_{CU}$.)*

$$
\begin{aligned}
& Pr\,[\mathcal{A}_{CU}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\mathsf{SK}^O}\ win] \\
&= Pr[d_Y = 1 \leftarrow \mathsf{Check}_Y(sk_{\mathsf{ID},j}, \mathsf{ID}, j, X, \hat{Y})| \\
&\qquad \hat{Y} \leftarrow \mathcal{A}_{CU}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\ \mathsf{SK}^O}(X), \\
&\qquad X \leftarrow \mathcal{CO}(j), j \leftarrow \mathcal{A}_{CU}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\ \mathsf{SK}^O}|j<i] \\
&= \epsilon_{CU} << \frac{1}{2^{\mathcal{K}}}, \\
&\quad where\ \mathcal{K}\ is\ the\ security\ parameter.
\end{aligned}
$$

Definition 3 (Forward-secure (Server)-Unforgeability). *Adversary $\mathcal{A}_{SU}$ is allowed to access the oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$, SK^O. Adversary $\mathcal{A}_{SU}$ chooses $\hat{X}$ and j, and sends $(\hat{X}, j)$ to challenge oracle $\mathcal{CO}$, and receives Y. After $\mathcal{A}_{SU}$ receives the challenge, it is allowed to access the oracles. Finally, $\mathcal{A}_{SU}$ outputs $\hat{Z}$. If the probability that the output $\mathsf{Check}_Z(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X}, Y, \hat{Z})$ is $d_Z = 1$ (i.e., $\hat{Z}$ is equal to the output of $\mathsf{TR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X}, Y)$) is negligible, where each index i of all requests to oracle $\mathcal{SK}$ is $j < i$, the scheme is Forward-secure (Server)-Unforgeable. (Note that if there are requests to oracle $\mathcal{SK}$ and the indices in these requirements are $i \leq j$, it is not included in the advantage of adversary $\mathcal{A}_{SU}$.)*

$$
\begin{aligned}
& Pr\,[\mathcal{A}_{SU}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\mathsf{SK}^O}\ win] \\
&= Pr[d_Z = 1 \leftarrow \mathsf{Check}_Z(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X}, Y, \hat{Z}) | \\
&\qquad \hat{Z} \leftarrow \mathcal{A}_{SU}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\mathsf{SK}^O}(Y), \\
&\qquad Y \leftarrow \mathcal{CO}(\hat{X}, j), (\hat{X}, j) \leftarrow \mathcal{A}_{SU}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\mathsf{SK}^O} | j < i] \\
&= \epsilon_{SU} << \frac{1}{2^{\mathcal{K}}}, \\
&\quad \textit{where } \mathcal{K} \textit{ is the security parameter.}
\end{aligned}
$$

Definition 4 (Forward-secure Secret Key Incrementability). *Adversary $\mathcal{A}_{FSK}$ is allowed to access oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$, SK^O. Adversary $\mathcal{A}_{FSK}$ chooses j, and sends j to challenge oracle $\mathcal{CO}$, and receives $sk_{\mathsf{ID},j}$. After $\mathcal{A}_{FSK}$ receives the challenge, it is allowed to access the oracles. Finally, $\mathcal{A}_{FSK}$ outputs $\hat{sk_{\mathsf{ID},t}}$. If the probability that the output $sk_{\mathsf{ID},j} = \mathsf{KeyGen}^{j-t}(\hat{sk_{\mathsf{ID},t}}, \mathsf{ID}, t)$, is negligible, where each index i in all requests to oracle $\mathcal{SK}$ is $j < i$, the scheme is Forward-secure Key Incrementable. Note that KeyGen^{j-t} means to calculate SK $j-t$ times.*

$$
\begin{aligned}
& Pr\,[\mathcal{A}_{FSK}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\mathsf{SK}^O}\ win] \\
&= Pr[sk_{\mathsf{ID},j} = \mathsf{KeyGen}^{j-t}(\hat{sk_{\mathsf{ID},t}}, \mathsf{ID}, t) | \\
&\qquad \hat{sk_{\mathsf{ID},t}} \leftarrow \mathcal{A}_{FSK}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\mathsf{SK}^O}(sk_{\mathsf{ID},j}), sk_{\mathsf{ID},j} \leftarrow \mathcal{CO}(j | j < \tilde{i}), \\
&\qquad (j | j < \tilde{i}) \leftarrow \mathcal{A}_{FSK}^{\mathcal{S}_f,\mathcal{C},\mathcal{S}_t,\mathsf{SR}^O,\mathsf{TR}^O,\mathcal{SK},\mathsf{SK}^O} | j < i] \\
&= \epsilon_{FSK} << \frac{1}{2^{\mathcal{K}}}, \\
&\quad \textit{where } \mathcal{K} \textit{ is security parameter.}
\end{aligned}
$$

4 Construction

4.1 Proposed Protocol and Its Design Concepts

Our proposed protocol meets the security requirements in Section 3. The details of the protocol are shown in Fig. 1. Here we explain the overview and concepts of our proposal.

The basic idea is combining the OSK protocol and key update mechanism from mutual authentication. Let H_0 and H_2 be hash functions (random oracle). H_0 and H_2 work in the same manner as the output function and key update function in OSK protocol, respectively.

In the OSK protocol, the tag excecutes as follows. At first, client (i.e., tag) is requested from server, and then a secret key is input, which is recorded in the tag's memory to H_0, and then inputs the output to H_2. At the end, the client outputs the calculated results of H_2 to the server. The server receives the tag's

output and then searches its database for the relevant secret key, which is shared with each tag, to H_0 and then inputs the output to H_2 using the same processes.

To accomplish key updates in both the RFID tag and the server, we must cope with the problem of desynchronization. If only one side of the party updates its secret key, the protocol fails for further authentication attempts. Such desynchronization causes not only failure of authentication, but also risks of breaching privacy. We prevent desynchronization by using a key update in the mutual authentication. The mutual authentication consists of two challenge-response protocols via hash functions. First, the server sends a random challenge, and then the tag calculates a response with H_0. The second challenge-response is initiated by the tag. The tag sends the challenge with the calculated response in Y'. The server calculates the response by using the hash function H_1 with the current secret or previous (old) secret. Note that the server stores both the current and previous secret key and which secret key is used depends on which secret key the server detects to calculate the received Y'. The server only calculates the value with the previous secret key if detects that the received Y' is calculated using that previous secret key, and calculates the value with the current updated secret key if detects that the received Y' is calculated using that current updated one. This mechanism deals with desynchronization by communication error. After that the server sends the (second) response to the tag. Only when the tag confirms the response will it update the secret key.

The basic security requirements are fulfilled from OSK-like construction, and desynchronization is solved by mutual authentication by holding two secret keys the current and the previous on the server side.

i means times of key updating events, i.e., counts of key updating. $sk_{\mathsf{ID},i}$ is the i'th secret key of a client (i.e., tag). ID is the client's ID.

4.2 Discussions

Synchronization : The secret key is updated by both servers and clients, and if desynchronization occures the server can distinguish and follow to the client's current state in the next session. Therefore this protocol can solve the desynchronization problem. Note that if the adversary cuts all responses from server to tag, the secret key in the tag cannot be updated. So if the adversary cuts all response from server to tag through several sessions, and then tampers with the tag, the adversary can trace the history of the tag during these several sessions.

Resiliency against DOS-like attack : Ari and Weis discussed the requirements of RFID protocols in [10]. They introduced the attack against a hash-chain-based scheme like a DOS- attack against a server via the Internet. In the proposed scheme, the event in which the secret key updated in the tag proceeds only when the verification check is OK. Therefore a DOS-like attack cannot be applied to the proposed scheme.

Saving computational cost of server : Generally, hash-based identification schemes like [11] require many server calculation to identify a tag(i.e., a client), and server

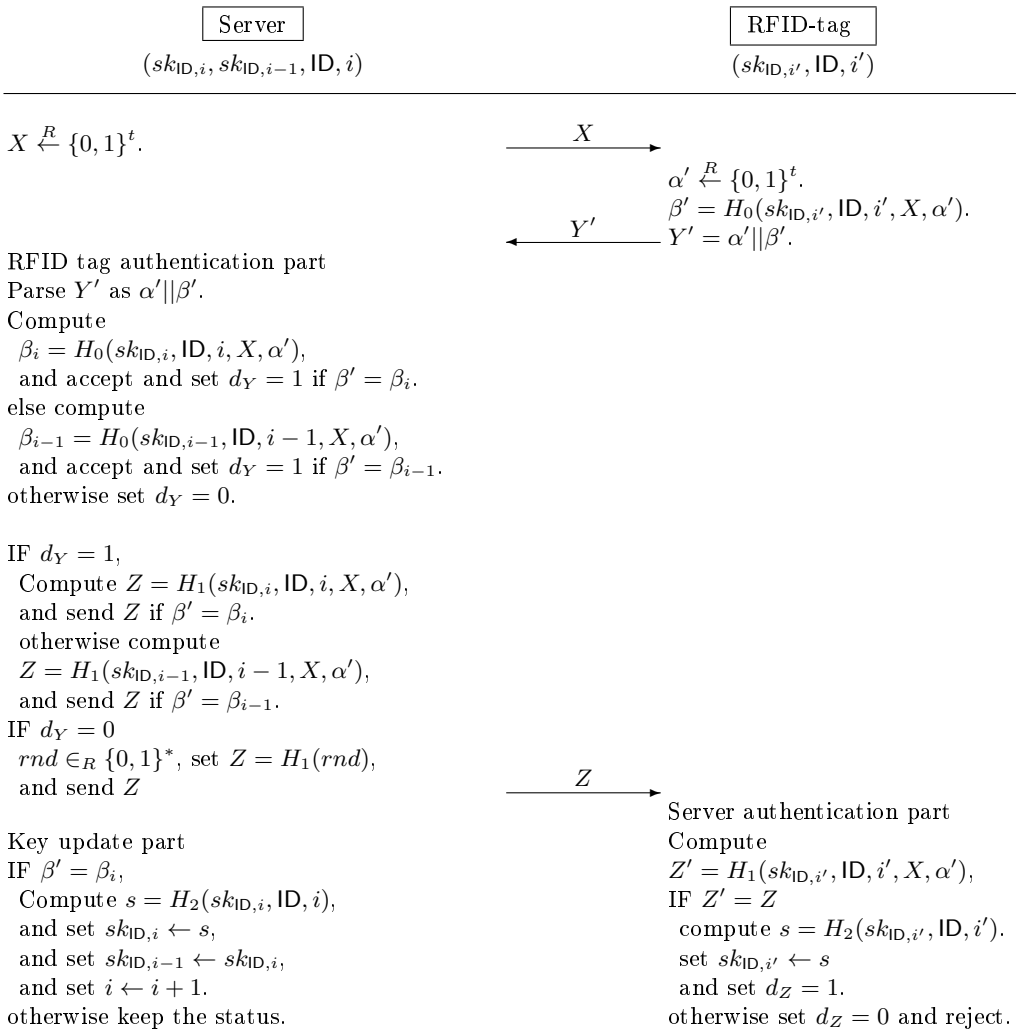

Fig. 1. Proposed protocol

must compute $2m$ hash calculations for each tag, where m is a maximum number of updates of the secret key. While in the proposed scheme, a server and a client can share the current state of the common secret key; therefore a server only needs to compute two hash calculations for each tag. The huge computational cost of a server can thereby be saved.

Resiliency against replay attack : In proposed scheme, fresh randomnesses chosen by both a server and a client are used; therefore a replay attack cannot be applied to the proposed scheme.

5 Security

The proposed scheme satisfies some requirements. Firstly, the basic properties are identification, authentication, and privacy. Then, forward-security and synchronization are extended properties.

Regarding privacy issues, there are two points of view. One is if a party identifies the ID, there are risks that the privacy of products or people, which are attached to tags, can be breached. Another is that if the output of a tag can be identified, the tag can be used as a tracing tool; for instance a tagged person (or something such as books, glasses, or bags) can be traced by tracing outputs from the tag. From the two points of views, indistinguishability is required; i.e., a tag's output is indistinguishable from random values. For authentication requirements, there are two sides - client- authentication and server authentication. Mutual authentication is achieved by satisfying both requirements. Since low cost is rewuirement of small devices such as RFIDs many are not able to satisfy the further requirement of tamper resistance. Therefore an adversary has a chance at a acquiring the secret key in these devices by tampering. This poses the risk of the tag's past output being traced, identified, and/or forged (i.e., client privacy and/or authenticity are breached). To protect the history in the tampered devices, the property of forward-security is required. Synchronization is another important property requirement since small devices such as RFIDs communicate wirelessly and wireless communication is often easily lost. Therefore when desynchronization occurs, a (state-ful) protocol requires the property of self-synchronization.

In this paper, we defined security notions to achieve not only basic properties, i.e., indistinguishability and authenticity, but also extended properties, i.e., forward-security and synchronization. We show the security proofs for these security notions in this section.

5.1 Game-based Theoretical Proof

The goal of our scheme is to achieve mutual authentication that preserves privacy. In [12], eight variations of security level are presented. Our scheme achieves *narrow-destructive privacy* , as defined in [12]. The defined security notions in

this paper are concrete for security notions to achieving *narrow-destructive privacy* for our scheme.

The proposed scheme satisfies of *forward-secure indistinguishable*, *forward-secure (client)-unforgeable*, *forward-secure (server)-unforgeable*, and *forward-secure key-incrementable*. We prove that the proposed scheme satisfies the four security notions by using a game style proof technique. Note that *forward-secure indistinguishable* is focused on the output of client. By the same token, the definition focused on the output of the server can be described and the proposed scheme satisfies the notion. Due to the space limitations, we omit the discussion about the privacy related to the output from server. This section coversthe intuition of the proof technique and the theorems as results. See the details in the appendix.

The construction of the proofs is as followings. The proofs are constructed following game-based techniques. We make four steps as games as follows.

- Game 0: Simulator $\mathcal{SIM}$ executes simulations following protocols.
- Game 1: Simulator $\mathcal{SIM}$ executes simulations setting the outputs of oracles random values, instead of the results of functions.
- Game 2: Excluding the case in which adversary accesses to oracles with the information of the secret key directly from the adversary's win.
- Game 3: Replying changed from challenge oracle $\mathcal{CO}$ to adversary and set the replying random values set regardless of coin-flipping results.

Through these games we show that the adversary in the protocol (i.e., Game 0) is in the same situations in that it is given no information related to the secret key, and there are no means other than random guessing.

As a results the following theorems are shown.

Theorem 1. Forward-secure Indistinguishablity
The proposed scheme is Forward-secure Indistinguishable, if hash functions H_0, H_1, H_2 *are random oracles.*

Theorem 2. Forward-secure (Client)-Unforgeability
The proposed scheme is Forward-secure (Client)-Unforgeable if hash functions H_0, H_1, H_2 *are random oracles.*

Theorem 3. Forward-secure (Client)-Unforgeability
The proposed scheme is Forward-secure (Server)-Unforgeable if hash functions H_0, H_1, H_2 *are random oracles.*

Theorem 4. Forward-secure Key-incrementability
The proposed scheme is Forward-secure Key-incrementable if hash functions H_0, H_1, H_2 *are random oracles.*

5.2 Verification Using CryptoVerif

We show verification results of the proposed protocol using CryptoVerif ver. 1.10pl1 [3,4,7]. CryptoVerif is a software that verifies security of cryptographic protocols by the game- based approach. For the verification we describe attack games and rewriting rules by using CryptoVerif's syntactic rules, and input them to CryptoVerif. The attack games represent a cryptographic protocol and its security requirements. The rewriting rules represents computational assumptions, etc. CryptoVerif then repeatedly modifies the attack game by using the rewriting rules. If the modified game satisfies the security requirements, the cryptographic protocol is judged as secure.

In this paper, we cannot describe the key update mechanism by hash chain because of restrictions of the syntactic rules. Therefore we describe a simple protocol that omits the key update mechanism, and verify the indistinguishability and the unforgeabilities by CryptoVerif. Oracle SK is also omitted from the attack model because the simple protocol clearly has no forward securities.

We succeeded in the security proof of the securities of the simple protocol by using CryptoVerif. The indistinguishability is proven by 18 game modifications, and the unforgeabilities are proven by 14 game modifications. The time required for each proof is five seconds. These results mean that the proposed scheme satisfies *indistinguishability*, *(Client)-Unforgeability*, and *(Server)-Unforgeability* as long as a tag is not tampered. Moreover, these results contribute to understandings of the proofs of the proposed protocol's securities by using a hybrid argument technique.

6 Conclusion

In this paper, we proposed a formal security model and definitions for RFID authentication protocol that is robust against a man-in-the-middle adversary. Then we proposed an RFID authentication protocol from the OSK protocol and synchronization mechanism. We also prove the security of the protocol, (1) by using handwritten proof in the game-based scheme, and (2) by verifying the correctness of the handwritten proof by using the CryptoVerif formal verification tool.

References

1. RFID security & privacy lounge. http://www.avoine.net/rfid/.
2. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP TCS*, pages 3–22, 2000.
3. Bruno Blanchet. CryptoVerif computationally sound, automatic cryptographic protocol verifier user manual. http://www.cryptoverif.ens.fr/.
4. Bruno Blanchet. A computationally sound mechanized prover for security protocols. Cryptology ePrint Archive, Report 2005/401, 2005. http://eprint.iacr.org/.
5. Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *IEEE Symposium on Security and Privacy*, pages 140–154, Oakland, California, May 2006.

6. Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In *CRYPTO*, pages 537–554, 2006.
7. Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In *CRYPTO*, pages 537–554, 2006.
8. Gerhard Hancke and Markus Kuhn. An RFID Distance Bounding Protocol. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, pages 67–73, Athens, Greece, September 2005. IEEE, IEEE Computer Society.
9. Ari Juels and Stephen Weis. Defining strong privacy for RFID. Cryptology ePrint Archive, Report 2006/137, 2006.
10. Ari Juels and Stephen Weis. Defining Strong Privacy for RFID. In *International Conference on Pervasive Computing and Communications – PerCom 2007*, pages 342–347, New York City, New York, USA, March 2007. IEEE, IEEE Computer Society Press.
11. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, MIT, MA, USA, November 2003.
12. Paise Radu-Ioan and Serge Vaudenay. Mutual Authentication in RFID: Security and Privacy. In *Proceedings of the 3rd ACM Symposium on Information, Computer and Communications Security – ASIACCS'08*, pages 292–299, Tokyo, Japan, 2008. ACM Press.
13. Serge Vaudenay. On Privacy Models for RFID. In *Advances in Cryptology - Asiacrypt 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87, Kuching, Malaysia, December 2007. Springer-Verlag.

Appendix

The proof of Theorem 1. (sketch)

Game 0 Adversary $\mathcal{A}_{FI}$ is allowed to access the oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$, SK^O. Simulator $\mathcal{SIM}$ sets $sk_{\mathsf{ID},0}$ randomly, and replies to queries from the adversary using $sk_{\mathsf{ID},0}$ following the protocol. Adversary $\mathcal{A}_{FI}$ chooses $X \leftarrow \{0,1\}^{n_x}$ and sends $(\hat{X}, j)$ to challenge oracle $\mathcal{CO}$ and receives Y. When the challenge oracle $\mathcal{CO}$ flips a coin in the case of $b = 1$, $\mathcal{CO}$ sends a random values as Y to $\mathcal{A}_{FI}$ In the case of $b = 0$, $\mathcal{CO}$ chooses α randomly, calculates $\mathsf{SR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X})$, and returns the result Y to $\mathcal{A}_{FI}$. After $\mathcal{A}_{FI}$ receives the challenge, it is allowed to access the oracles. Finally, $\mathcal{A}_{FI}$ outputs b'. Let the probability of the case $b' = b$, where each index i in requests to oracle $\mathcal{SK}$ is $j < i$,

$$Pr[\mathcal{A}_{FI} \; in \; Game0] = \frac{1}{2} + \epsilon_{FI}$$

Game 1 Adversary $\mathcal{A}_{FI}$ is allowed to access the oracles as same as Game 0. Simulator $\mathcal{SIM}$ simulates as follows. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_f$ oracle, it replies X randomly as in Game 0. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_t$ oracle with (ID, i, X, Y), at first, it searches its database and checks if there is a pair of setting secret key $sk_{\mathsf{ID},i}$ and requested query (ID, i, X, Y) asked to $\mathcal{S}_t$ or TR^O

previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in the database as Z. Otherwise, it chooses Z randomly and replies to it, then records the pair of received values as input and Z in its database. If $\mathcal{SIM}$ receives a query to TR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y)$, he first searches its database and checks if there is a pair of the requested query and output as Z. If there is a pair, $\mathcal{SIM}$ replies wih the same output value recorded in its database as Z. Otherwise, it chooses Z randomly and replies to it, then records the pair of received values as input and Z, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y),\ Z)$ in his database. If $\mathcal{SIM}$ receives a query to $\mathcal{SK}$ oracle with (ID, i), it first searches its database and checks if there is a pair of the setting secret key $sk_{\mathsf{ID},i}$ and the requested query (ID, i), asked to $\mathcal{SK}$ or SK^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as $sk_{\mathsf{ID},i}$. Otherwise, it chooses $sk_{\mathsf{ID},i}$ randomly and replies to it, then records the value in its database. If $\mathcal{SIM}$ receives a query to SK^O oracle, with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i)$, it first searches its database and checks if there is a pair of the requested query and output as $sk_{\mathsf{ID},i+1}$. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as $sk_{\mathsf{ID},i+1}$. Otherwise, it chooses $sk_{\mathsf{ID},i+1}$ randomly and replies to it, then records the value in its database. If $\mathcal{SIM}$ receives a query to $\mathcal{C}$ oracle with (ID, i, X), at first, it searches its database and checks if there is a pair of the setting secret $sk_{\mathsf{ID},i}$ and the requested query (ID, i, X), requested from $\mathcal{C}$ or SR^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Y. Otherwise, if chooses Y randomly and replies to it, then records the pair of received values as input and Y in its database. If $\mathcal{SIM}$ receives a query to SR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X)$, at first, it searches its database and checks if there is a pair of the requested query and output as Y. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Y. Otherwise, it chooses Y randomly and replies to it, then records the pair of received values as input and Y, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X),\ Y)$ in its database. After then $\mathcal{A}_{FI}$ outputs b'. Let the probability of the case $b' = b$, where each index i in all requests to oracle $\mathcal{SK}$ is $j < i$, (same conditions as Game 0) $Pr[\mathcal{A}_{FI}\ in\ Game1]$. $Pr[\mathcal{A}_{FI}\ in\ Game0] - Pr[\mathcal{A}_{FI}\ in\ Game1]$ means the advantage that adversary can acquire in Game 0 against in Game 1, where in Game 0 $\mathcal{SIM}$ simulates using the setting secret key and following the protocol, while in Game 1 $\mathcal{SIM}$ simulates setting the oracle's output randomly.

Let us look at the output of each oracle. The output of oracle $\mathcal{S}_f$ is always a random value and independent of the secret key, therefore the output gives no advantage to adversary. The output of oracles $\mathcal{S}_t$ and TR^O are outputs of hash function H_1. H_1 assumes that the output of H_1 is random distribution. Therefore he adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly. Therefore the outputs give no advantage to adversary. The outputs of oracles $\mathcal{SK}$ and SK^O are related to the output of hash function H_2. As long as $j < i$, i.e., the secret key is not invoked, the adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly, since H_2 also assumes that the output of H_2 is random distribution. The outputs of oracles $\mathcal{C}$ and SR^O are outputs of hash

function H_0. H_0 also assumes that the output of H_0 is random distribution. Additionally, the adversary is not allowed to ask a query with a value that is same as the received value from $\mathcal{CO}$. Therefore the outputs give no advantage to adversary. At the end,

$$Pr[\mathcal{A}_{FI} \; in \; Game0] - Pr[\mathcal{A}_{FI} \; in \; Game1] = 0$$

So,

$$Pr[\mathcal{A}_{FI} \; in \; Game0] = Pr[\mathcal{A}_{FI} \; in \; Game1]$$

Game2 Adversary $\mathcal{A}_{FI}$ is allowed to access the oracles as in Game 0. However, when the adversary outputs b', if it has accessed oracles SR^O or TR^O, it is not included in adversary's win. If the same conditions as Game 0 are satisfied and the adversary has not accessed SR^O and TR^O, the adversary wins. Let the probability by $Pr[\mathcal{A}_{FI} \; in \; Game2]$.

Let us consider $Pr[\mathcal{A}_{FI} \; in \; Game1] - Pr[\mathcal{A}_{FI} \; in \; Game2]$. The difference comes in the following cases. In the first case, it happens that the adversary asks oracle SR^O with $(sk_{\mathsf{ID},j'}, \mathsf{ID}, j', \hat{X}, \hat{\alpha})$, where $(sk_{\mathsf{ID},j'}$ is identical with a secret key set by the simulator. In this case, if the query to $\mathcal{CO}$ is $(\hat{X}, j')$ and coin is $b = 0$, and then $\mathcal{CO}$ happens to choose $\hat{\alpha}$ as α, the adversary can distinguish the challenge from $\mathcal{CO}$. The second case is as follows, if it happens that the two values are the same in that one is a received value from oracle SR^O and the other is a received value from oracle $\mathcal{C}$, and the same value (ID, j', X) are included in both queries. In this case, the adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the adversary can acquire information that contributes to its advantage. The third case is as follows, similarly to the second case, if it happens that the two values are same in that one is a received value from oracle TR^O and another is a received value from oracle $\mathcal{S}_t$, and the same value (ID, j', X, Y) are included in both queries. In this case, the adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the adversary can acquire information that contributes to its advantage. Let the maximum number of queries be q times and the size of secret key be n_s bits. The probability of the above events occuring is at most $\frac{q}{2^{n_s}}$. Therefore,

$$Pr[\mathcal{A}_{FI} \; in \; Game1] - Pr[\mathcal{A}_{FI} \; in \; Game2] \leq \frac{q}{2^{n_s}}$$

Game3 Adversary $\mathcal{A}_{FI}$ is allowed to access the oracles as in Game 0. Let Game 3 changed as follows. When $\mathcal{CO}$ is requested from the adversary, he replies a random value and the value is independent from whether b is 0 or 1. Finally, $\mathcal{A}_{FI}$ outputs b'. If the same conditions as Game 0 are satisfied, the adversary wins. Let the advantage of the adversary in this game be $Pr[\mathcal{A}_{FI} \; in \; Game3]$. The probability is that $Pr[\mathcal{A}_{FI} \; in \; Game3] = \frac{1}{2}$. Assuming the hash function H_0 is a random oracle,

$$\begin{aligned} Pr[\mathcal{A}_{FI} \; in \; Game3] &= Pr[\mathcal{A}_{FI} \; in \; Game2] \\ &= \frac{1}{2} \end{aligned}$$

From Game 0, 1, 2, 3, the following is shown.

$$\begin{aligned} Pr[\mathcal{A}_{FI}\ in\ Game0] &= Pr[\mathcal{A}_{FI}\ in\ Game1] \\ &\leq Pr[\mathcal{A}_{FI}\ in\ Game2] + \frac{q}{2^{n_s}} \\ &= Pr[\mathcal{A}_{FI}\ in\ Game3] + \frac{q}{2^{n_s}} \end{aligned}$$

Therefore,

$$Pr[\mathcal{A}_{FI}\ in\ Game0] = \frac{1}{2} + \epsilon_{FI} \leq \frac{1}{2} + \frac{q}{2^{n_s}}.$$

Where we can say that

$$\epsilon_{FI} \leq \frac{q}{2^{n_s}}.$$

As a result, it can be shown that the proposed scheme is *forward-secure indistinguishable*, if $q << 2^{n_s}$ and H_0 is random oracle. ¶

Proof of Theorem 2. (sketch)

Game0 Adversary $\mathcal{A}_{CU}$ is allowed to access the oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$, SK^O. Simulator $\mathcal{SIM}$ sets $sk_{\mathsf{ID},0}$ randomly, and replies to queries from adversary using $sk_{\mathsf{ID},0}$ following the protocol. Adversary $\mathcal{A}_{CU}$ chooses j randomly, and sends it to challenge oracle $\mathcal{CO}$ and receives X. After $\mathcal{A}_{CU}$ receives the challenge, it is allowed to access the oracles. Finally, $\mathcal{A}_{CU}$ outputs $\hat{Y}$. Let the probability of the case be that the output of $\mathsf{Check}_Y(sk_{\mathsf{ID},j}, \mathsf{ID}, j, X, \hat{Y})$ is $d_Y = 1$ (i.e., $\hat{Y}$ is equal to the output of $\mathsf{SR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, X)$) , where each index i in all requests to oracle $\mathcal{SK}$ is $j < i$, $Pr[\mathcal{A}_{CU}\ in\ Game0]$.

Game1 Adversary $\mathcal{A}_{CU}$ is allowed to access the oracles as in Game 0. Simulator $\mathcal{SIM}$ simulates as follows. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_f$ oracle, it replies X randomly as in Game 0. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_t$ oracle with (ID, i, X, Y), at first, it searches its database and checks if there is a pair of the setting secret key $sk_{\mathsf{ID},i}$ and requested query (ID, i, X, Y) requested to $\mathcal{S}_t$ or TR^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Z. Otherwise, it chooses Z randomly and replies to it, then records the pair of receives values as input and Z in its database. If $\mathcal{SIM}$ receives a query to TR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y)$, it first searches its database and checks if there is a pair of the requested query and output as Z. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Z. Otherwise, it chooses Z randomly and replies to it, then records the pair of received values as input and

Z, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y),\ Z)$ in its database. If $\mathcal{SIM}$ receives a query to $\mathcal{SK}$ oracle with (ID, i), it first searches its database and checks if there is a pair of the setting secret key $sk_{\mathsf{ID},i}$ and the requested query (ID, i), requested to $\mathcal{SK}$ or SK^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as $sk_{\mathsf{ID},i}$. Otherwise, it chooses $sk_{\mathsf{ID},i}$ randomly and replies to it, then records the value in its database. If $\mathcal{SIM}$ receives a query to SK^O oracle, with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i)$, it first searches his database and checks if there is a pair of the requested query and output as $sk_{\mathsf{ID},i+1}$. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in his database as $sk_{\mathsf{ID},i+1}$. Otherwise, it chooses $sk_{\mathsf{ID},i+1}$ randomly and replies to it, then record the value in its database. If $\mathcal{SIM}$ receives a query to $\mathcal{C}$ oracle with (ID, i, X), it first searches its database and checks if there is a pair of the setting secret $sk_{\mathsf{ID},i}$ and the requested query (ID, i, X), requested to $\mathcal{C}$ or SR^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Y. Otherwise, it choose Y randomly and replies to it, then records the pair of received values as input and Y in its database. If $\mathcal{SIM}$ receives a query to SR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X)$, it first searches its database and checks if there is a pair of the requested query and output as Y. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Y. Otherwise, it chooses Y randomly and replies to it, then records the pair of received values as input and Y, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X),\ Y)$ in its database. Finally, adversary $\mathcal{A}_{CU}$ outputs $\hat{Y}$. Let the probability of the case be that the output of $\mathsf{Check}_Y(sk_{\mathsf{ID},j}, \mathsf{ID}, j, X, \hat{Y})$ is $d_Y = 1$ (i.e., $\hat{Y}$ is equal to the output of $\mathsf{SR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, X)$) , where each index i of all requests to oracle $\mathcal{SK}$ is $j < i$ (same conditions as Game 0), $Pr[\mathcal{A}_{CU}\ in\ Game1]$.

$Pr[\mathcal{A}_{CU}\ in\ Game0] - Pr[\mathcal{A}_{CU}\ in\ Game1]$ means the advantage that the adversary can acquire in Game 0 against in Game 1, where in Game 0 $\mathcal{SIM}$ simulates using the setting secret key and following the protocol, while in Game 1 $\mathcal{SIM}$ simulates setting the oracle's output randomly.

The output of oracle $\mathcal{S}_f$ is always a random value and independent of the secret key, therefore the output gives no advantage to the adversary. The output of oracles $\mathcal{S}_t$ and TR^O are outputs of hash function H_1. H_1 assumes that the output of H_1 is random distribution. Therefore the adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly. Therefore the outputs give no advantage to the adversary. The outputs of oracles $\mathcal{SK}$ and SK^O are related to the output of hash function H_2. As long as $j < i$, i.e., the secret key is not invoked, the adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly, since H_2 also assumes that the output of H_2 is random distribution. The outputs of oracles $\mathcal{C}$ and SR^O are outputs of hash function H_0. H_0 also assumes that the output of H_0 is random distribution. Additionally, the adversary is not allowed to ask a query with a value that is the same as the received value from $\mathcal{CO}$. Therefore the outputs give no advantage to the adversary. At the end,

$$Pr[\mathcal{A}_{CU}\ in\ Game0] - Pr[\mathcal{A}_{CU}\ in\ Game1] = 0$$

So,

$$Pr[\mathcal{A}_{CU}\ in\ Game0] = Pr[\mathcal{A}_{CU}\ in\ Game1]$$

Game 2 Adversary $\mathcal{A}_{CU}$ is allowed to access the oracles as in Game 0. However, When the adversary outputs $\hat{Y}$, if it has accessed oracles SR^O or TR^O, it is not included in the adversary's win. If the same conditions as Game 0 are satisfied and the adversary has not accessed to SR^O and TR^O, the adversary wins. Let the probability be $Pr[\mathcal{A}_{CU}\ in\ Game2]$.

Let us consider $Pr[\mathcal{A}_{CU}\ in\ Game1] - Pr[\mathcal{A}_{CU}\ in\ Game2]$. The difference comes in the following cases. In the first case, the two values are the same in that one is a received value from oracle SR^O and another is a received value from oracle $\mathcal{C}$, and the same value (ID, j', X) are included in both queries, the adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the adversary can acquire information that contributes to its advantage. Similarly, the two values are the same in that one is a received value from oracle TR^O and another is a received value from oracle $\mathcal{S}_t$, and the same value (ID, j', X, Y) are included in the both queries. In this case, the adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the adversary can acquire information that contributes adversary's advantage. Let the maximum number of queries be q times and the size of secret key is n_s bits. The probability of that the above events occuring is at most $\frac{q}{2^{n_s}}$. Therefore,

$$Pr[\mathcal{A}_{CU}\ in\ Game1] - Pr[\mathcal{A}_{CU}\ in\ Game2] \leq \frac{q}{2^{n_s}}$$

Game 3 Adversary $\mathcal{A}_{CU}$ is allowed to access the oracles as in Game 0. Let Game 3 changed as follows. When $\mathcal{CO}$ is requested from the adversary, it replies a random value and the value is independent from the requested queries. Finally, $\mathcal{A}_{CU}$ outputs $\hat{Y}$. If the same conditions as Game 0 are satisfied, the adversary wins. Let the advantage of the adversary in this game be $Pr[\mathcal{A}_{CU}\ in\ Game3]$. Since $\mathcal{A}_{CU}$ can receives only random values from the oracles, i.e., all received values are independent from the information of the secret key, $\mathcal{A}_{CU}$ wins, only when it happens to guess the correct $\hat{Y}$. Therefore, $Pr[\mathcal{A}_{CU}\ in\ Game3] = \frac{1}{2^{n_s}}$. Assuming hash function H_0 is a random oracle,

$$\begin{aligned} Pr[\mathcal{A}_{CU}\ in\ Game3] &= Pr[\mathcal{A}_{CU}\ in\ Game2] \\ &= \frac{1}{2^{n_s}}. \end{aligned}$$

From Game 0, 1, 2, 3, the following is shown.

$$
\begin{aligned}
Pr[\mathcal{A}_{CU}\ in\ Game0] &= Pr[\mathcal{A}_{CU}\ in\ Game1] \\
&\leq Pr[\mathcal{A}_{CU}\ in\ Game2] + \frac{q}{2^{n_s}} \\
&= Pr[\mathcal{A}_{CU}\ in\ Game3] + \frac{q}{2^{n_s}} \\
&= \frac{1}{2^{n_s}} + \frac{q}{2^{n_s}} = \frac{q+1}{2^{n_s}}
\end{aligned}
$$

As a result, it can be shown that the proposed scheme is *forward-secure (client)-unforgeable*, if $q << 2^{n_s}$ and H_0 is random oracle. ¶

Proof of Theorem 3. (sketch)

Game0 Adversary $\mathcal{A}_{SU}$ is allowed to access the oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$, SK^O. Simulator $\mathcal{SIM}$ sets $sk_{\mathsf{ID},0}$ randomly, and replies to queries from the adversary using $sk_{\mathsf{ID},0}$ following the protocol. The adversary $\mathcal{A}_{SU}$ chooses j and $\hat{X}$ randomly, and sends $(\hat{X}, j)$ to challenge oracle $\mathcal{CO}$ and receives Y. After $\mathcal{A}_{SU}$ receives the challenge, it is allowed to access the oracles. Finally, $\mathcal{A}_{SU}$ outputs $\hat{Z}$. Let the probability of the case in which the output of $\mathsf{Check}_Z(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X}, Y, \hat{Z})$. be $d_Z = 1$ (i.e., $\hat{Z}$ is equal to the output of $\mathsf{TR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X}, Y)$), where each index i in all requests to oracle $\mathcal{SK}$ is $j < i$, $Pr[\mathcal{A}_{SU}\ in\ Game0]$.

Game1 Adversary $\mathcal{A}_{SU}$ is allowed to access the oracles as in Game 0. Simulator $\mathcal{SIM}$ simulates as follows. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_f$ oracle, it replies X randomly as in Game 0. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_t$ oracle with (ID, i, X, Y), at first, it searches its database and checks if there is a pair of the setting secret key $sk_{\mathsf{ID},i}$ and the requested query (ID, i, X, Y) requested to $\mathcal{S}_t$ or TR^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Z. Otherwise, it chooses Z randomly and replies to it, then records the pair of received values as input and Z in its database. If $\mathcal{SIM}$ receives a query to TR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y)$, at first, it searches its database and checks if there is a pair of the requested query and output as Z. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Z. Otherwise, chooses Z randomly and replies to it, then records the pair of received values as input and Z, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y),\ Z)$ in its database. If $\mathcal{SIM}$ receives a query to $\mathcal{SK}$ oracle with (ID, i), it first searches its database and checks if there is a pair of the setting secret key $sk_{\mathsf{ID},i}$ and the requested query (ID, i), requested to $\mathcal{SK}$ or SK^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as $sk_{\mathsf{ID},i}$. Otherwise, it chooses $sk_{\mathsf{ID},i}$ randomly and replies to it, then records the value in its database. If $\mathcal{SIM}$ receives a query to SK^O oracle, with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i)$, it first searches its database and checks if there is a pair of the requested query and output as $sk_{\mathsf{ID},i+1}$. If there is a pair, $\mathcal{SIM}$ replies wih the same output value recorded in its database as $sk_{\mathsf{ID},i+1}$. Otherwise, it choose

$sk_{\mathsf{ID},i+1}$ randomly and replies to it, then records the value in its database. If $\mathcal{SIM}$ receives a query to $\mathcal{C}$ oracle with (ID, i, X), it first searches its database and checks if there is a pair of the setting secret $sk_{\mathsf{ID},i}$ and the requested query (ID, i, X), requested to $\mathcal{C}$ or SR^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Y. Otherwise, it chooses Y randomly and replies to it, then records the pair of received values as input and Y in its database. If $\mathcal{SIM}$ receives a query to SR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X)$, it first searches its database and checks if there is a pair of the requested query and output as Y. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Y. Otherwise, it chooses Y randomly and replies to it, then records the pair of received values as input and Y, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X), Y)$ in its database. Finally, adversary $\mathcal{A}_{SU}$ outputs $\hat{Z}$. Let the probability of the case where the output of $\mathsf{Check}_Z(sk_{\mathsf{ID},i}, \mathsf{ID}, j, \hat{X}, Y, \hat{Z})$: be $d_Z = 1$ (i.e., $\mathcal{Z}$ is equal to the output of $\mathsf{TR}(sk_{\mathsf{ID},j}, \mathsf{ID}, j, \hat{X}, Y)$), where each index i in all requests to oracle $\mathcal{SK}$ is $j < i$ (same conditions as Game 0), $Pr[\mathcal{A}_{SU}\ in\ Game1]$.

$Pr[\mathcal{A}_{SU}\ in\ Game0] - Pr[\mathcal{A}_{SU}\ in\ Game1]$ means the advantage that the adversary can get in Game 0 versus Game 1, where in Game 0 $\mathcal{SIM}$ simulates using the setting secret key and following the protocol, while in Game 1 $\mathcal{SIM}$ simulates setting the oracle's output randomly.

The output of oracle $\mathcal{S}_f$ is always a random value and independent of the secret key, therefore the output gives no advantage to the adversary. The output of oracles $\mathcal{C}$ and SR^O are outputs of hash function H_0. H_0 assumes that the output of H_0 is random distribution. Therefore the adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly. The outputs therefore give no advantage to the adversary. The outputs of oracles $\mathcal{SK}$ and SK^O are related to the output of hash function H_2. As long as $j < i$, i.e., the secret key is not invoked, the adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly, since H_2 also assumes that the output of H_2 is random distribution. The outputs of oracles $\mathcal{S}_t$ and TR^O are outputs of hash function H_1. H_1 also assumes that the output of H_1 is random distribution. Additionally, the adversary is not allowed to ask a query with a value that is same as the received value from $\mathcal{CO}$. Therefore the outputs give no advantage to the adversary. At the end,

$$Pr[\mathcal{A}_{SU}\ in\ Game0] - Pr[\mathcal{A}_{SU}\ in\ Game1] = 0$$

Game 2 Adversary $\mathcal{A}_{SU}$ is allowed to access the oracles as in Game 0. However, when the adversary outputs $\hat{Z}$, if it has accessed oracles SR^O or TR^O, it is not included in the adversary's win. If the same conditions as Game 0 are satisfied and the adversary has not accessed to SR^O and TR^O, the adversary wins. Let the probability be $Pr[\mathcal{A}_{SU}\ in\ Game2]$.

Let us consider $Pr[\mathcal{A}_{SU}\ in\ Game1] - Pr[\mathcal{A}_{SU}\ in\ Game2]$. The difference comes from the following cases. In the first case, the two values are the same in that one is a received value from oracle SR^O and another is a received value from oracle $\mathcal{C}$, and the same value (ID, j', X) are included in the both queries. In this case, adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the

adversary can acquire information that contributes to its advantage. Similarly, the two values are the same in that one is a received value from oracle TR^O and another is a received value from oracle $\mathcal{S}_t$, and the same value (ID, j', X, Y) are included in the both queries. In this case, the adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the adversary can acquire information that contributes to its advantage. Let the maximum number of queries be q times and the size of secret key be n_s bits. The probability of that the above events occuring is at most $\frac{q}{2^{n_s}}$. Therefore,

$$Pr[\mathcal{A}_{SU}\ in\ Game1] - Pr[\mathcal{A}_{SU}\ in\ Game2] \leq \frac{q}{2^{n_s}}$$

Game3 Adversary $\mathcal{A}_{SU}$ is allowed to access the oracles as in Game 0. Let Game 3 changed as follows. When $\mathcal{CO}$ is requested from the adversary, it replies with a random value and the value is independent from the requested queries. Finally, $\mathcal{A}_{SU}$ outputs $\hat{Z}$. If the same conditions as Game 0 are satisfied, the adversary wins. Let the advantage of adversary in this game be $Pr[\mathcal{A}_{SU}\ in\ Game3]$. Since $\mathcal{A}_{SU}$ can receives only random values from oracles, i.e., all received values are independent with the information of secret key, $\mathcal{A}_{SU}$ wins, only when it happens to guess the correct $\hat{Z}$. Therefore, $Pr[\mathcal{A}_{SU}\ in\ Game3] = \frac{1}{2^{n_s}}$. Assuming the hash function H_1 is a random oracle,

$$\begin{aligned} Pr[\mathcal{A}_{SU}\ in\ Game3] &= Pr[\mathcal{A}_{SU}\ in\ Game2] \\ &= \frac{1}{2^{n_s}}. \end{aligned}$$

From Game 0, 1, 2, 3, the following is shown.

$$\begin{aligned} Pr[\mathcal{A}_{SU}\ in\ Game0] &= Pr[\mathcal{A}_{SU}\ in\ Game1] \\ &\leq Pr[\mathcal{A}_{SU}\ in\ Game2] + \frac{q}{2^{n_s}} \\ &= Pr[\mathcal{A}_{SU}\ in\ Game3] + \frac{q}{2^{n_s}} \\ &= \frac{1}{2^{n_s}} + \frac{q}{2^{n_s}} = \frac{q+1}{2^{n_s}} \end{aligned}$$

As a result, it can be shown that the proposed scheme is forward-secure (server)-unforgeable, if $q << 2\ n_s$ and H_1 is a random oracle. ¶

Proof of Theorem 4. (sketch)

Game0 Adversary $\mathcal{A}_{FSK}$ is allowed to access the oracles, $\mathcal{S}_f$, $\mathcal{C}$, $\mathcal{S}_t$, SR^O, TR^O, $\mathcal{SK}$, SK^O. Simulator $\mathcal{SIM}$ sets $sk_{\mathsf{ID},0}$ randomly, and replies to queries from the adversary using $sk_{\mathsf{ID},0}$ following the protocol. The adversary $\mathcal{A}_{FSK}$ chooses j randomly, and sends it to challenge oracle $\mathcal{CO}$ and receives $sk_{\mathsf{ID},j}$.

After $\mathcal{A}_{FSK}$ receives the challenge, it is allowed to access the oracles. Finally, $\mathcal{A}_{FSK}$ outputs $\hat{sk_{\mathsf{ID},t}}$. Let the probability of the case be that the output of $sk_{\mathsf{ID},j} = \mathsf{KeyGen}^{j-t}(\hat{sk_{\mathsf{ID},t}}, \mathsf{ID}, t)$ (where, SK^{j-t} means executing SK $j-t$ times) , where each index i in all requests to oracle $\mathcal{SK}$ is $j < i$, $Pr[\mathcal{A}_{FSK}\ in\ Game0] = \frac{j-1}{2^{n_s}} + \epsilon_{FSK}$.

Game1 Adversary $\mathcal{A}_{FSK}$ is allowed to access the oracles as in Game 0. Simulator $\mathcal{SIM}$ simulates as follows. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_f$ oracle, it replies X randomly as in Game 0. If $\mathcal{SIM}$ receives a query to $\mathcal{S}_t$ oracle with (ID, i, X, Y), it first searches its database and checks if there is a pair of the setting secret key $sk_{\mathsf{ID},i}$ and the requested query (ID, i, X, Y) requested to $\mathcal{S}_t$ or TR^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Z. Otherwise, it chooses Z randomly and replies to it, then records the pair of received values as input and Z in its database. If $\mathcal{SIM}$ receives a query to TR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y)$, it first searches its database and checks if there is a pair of the requested query and output as Z. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Z. Otherwise, it chooses Z randomly and replies to it, then records the pair of received values as input and Z, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X, Y), Z)$ in its database. If $\mathcal{SIM}$ receives a query to $\mathcal{SK}$ oracle with (ID, i), it first searches its database and checks if there is a pair of the setting secret key $sk_{\mathsf{ID},i}$ and the requested query (ID, i), asked to $\mathcal{SK}$ or SK^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as $sk_{\mathsf{ID},i}$. Otherwise, it chooses $sk_{\mathsf{ID},i}$ randomly and replies to it, then records the value in its database. If $\mathcal{SIM}$ receives a query to SK^O oracle, with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i)$, it first searches its database and checks there is a pair of the requested query and output as $sk_{\mathsf{ID},i+1}$. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as $sk_{\mathsf{ID},i+1}$. Otherwise, it chooses $sk_{\mathsf{ID},i+1}$ randomly and replies to it, then records the value in its database. If $\mathcal{SIM}$ receives a query to $\mathcal{C}$ oracle with (ID, i, X), it first searches its database and checks if there is a pair of the setting secret $sk_{\mathsf{ID},i}$ and the requested query (ID, i, X), requested to $\mathcal{C}$ or SR^O previously. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in its database as Y. Otherwise, it chooses Y randomly and replies to it, then records the pair of received values as input and Y in its database. If $\mathcal{SIM}$ receives a query to SR^O oracle with $(sk_{\mathsf{ID},i}, \mathsf{ID}, i, X)$, it first searches its database and checks if there is a pair of the requested query and output as Y. If there is a pair, $\mathcal{SIM}$ replies with the same output value recorded in his database as Y. Otherwise, it chooses Y randomly and replies to it, then records the pair of received values as input and Y, i.e., $((sk_{\mathsf{ID},i}, \mathsf{ID}, i, X), Y)$ in its database. Finally, adversary $\mathcal{A}_{FSK}$ outputs $\hat{sk_{\mathsf{ID},t}}$. Let the probability of the case be that the output of $sk_{\mathsf{ID},j} = \mathsf{KeyGen}^{j-t}(\hat{sk_{\mathsf{ID},t}}, \mathsf{ID}, t)$ (where KeyGen^{j-t} means that executing SK $j-t$ times)„ where each index i in all requests to oracle $\mathcal{SK}$ is $j < i$ (same conditions as Game 0), $Pr[\mathcal{A}_{FSK}\ in\ Game1]$.

$Pr[\mathcal{A}_{FSK}\ in\ Game0] - Pr[\mathcal{A}_{FSK}\ in\ Game1]$ means the advantage that adversary can get in Game 0 against in Game 1, where in Game 0 $\mathcal{SIM}$ simulates

using the setting secret key and following the protocol, while in Game 1 $\mathcal{SIM}$ simulates setting the oracle's output randomly.

The output of oracle $\mathcal{S}_f$ is always a random value and independent from the secret key, therefore the output gives no advantage to the adversary. The output of oracles $\mathcal{C}$ and SR^O are outputs of hash function H_0. H_0 assumes that the output of H_0 is random distribution. Therefore the adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly. The outputs therefore give no advantage to the adversary. The output of oracles $\mathcal{S}_t$ and TR^O are outputs of hash function H_1. H_1 assumes that the output of H_1 is random distribution. Therefore the adversary cannot distinguish whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly. Therefore the outputs give no advantage to the adversary. The outputs of oracles $\mathcal{SK}$ and SK^O are related to the output of hash function H_2. As long as $j < i$, i.e., the secret key is not invoked, the adversary cannot distinguish between whether $\mathcal{SIM}$ simulates using the setting secret key or setting output randomly, since H_2 also assumes that the output of H_2 is random distribution. At the end,

$$Pr[\mathcal{A}_{FSK}\ in\ Game0] - Pr[\mathcal{A}_{FSK}\ in\ Game1] = 0$$

So,

$$Pr[\mathcal{A}_{FSK}\ in\ Game0] = Pr[\mathcal{A}_{FSK}\ in\ Game1]$$

Game2 Adversary $\mathcal{A}_{FSK}$ is allowed to access the oracles as same as Game 0. However, when adversary outputs $\hat{sk_{\mathsf{ID},t}}$, if it has accessed oracles SR^O or TR^O, it is not included in the adversary's win. If the same conditions as Game 0 are satisfied and adversary has not accessed to SR^O and TR^O, the adversary wins. Let the probability be $Pr[\mathcal{A}_{FSK}\ in\ Game2]$.

Let us consider $Pr[\mathcal{A}_{FSK}\ in\ Game1] - Pr[\mathcal{A}_{FSK}\ in\ Game2]$. The difference comes from the following cases. In the first case, the two values are the same in that one is a received value from oracle SR^O and the other is a received value from oracle $\mathcal{C}$, and the same value (ID, j', X) is included in both queries. In this case, the adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the adversary can acquire information that contributes to its advantage. Similarly, the two values are the same in that one is a received value from oracle TR^O and the other is a received value from oracle $\mathcal{S}_t$, and the same value (ID, j', X, Y) are included in both queries. In this case, the adversary can identify the value of the $j' - th$ secret key. If $j' \leq j$, the adversary can acquire information that contributes to its advantage. As another case, it occurs that the adversary asks oracle SK^O with $(sk_{\mathsf{ID},j-1}, \mathsf{ID}, j-1)$, where $(sk_{\mathsf{ID},j-1}$ identical to a secret key set by the simulator. In this case, $\mathcal{A}_{FSK}$ could acquire the correct secret key $(sk_{\mathsf{ID},j}$. If the query to $\mathcal{CO}$ is (ID, j), $\mathcal{A}_{FSK}$ receives $(sk_{\mathsf{ID},j}$. Therefore, in this case $\mathcal{A}_{FSK}$ identifies the correct secret key $(sk_{\mathsf{ID},j-1}$, and this case gives the adversary an advantage. Let the maximum number of queries be q times and the size of secret key be n_s bits. The probability of that the above events occuring is at most $\frac{q}{2^{n_s}}$. Therefore,

$$Pr[\mathcal{A}_{FSK}\ in\ Game1] - Pr[\mathcal{A}_{FSK}\ in\ Game2] \leq \frac{q}{2^{n_s}}$$

Game3 Adversary $\mathcal{A}_{FSK}$ is allowed to access the oracles as in Game 0. Let Game 3 changed as follows. When $\mathcal{CO}$ is requested from the adversary, it replies a random value and the value is independent from the requested queries. Finally, $\mathcal{A}_{CU}$ outputs $\hat{sk_{\mathsf{ID},t}}$. If the same conditions as Game 0 are satisfied, the adversary wins. Let the advantage of the adversary in this game be $Pr[\mathcal{A}_{FSK}\ in\ Game3]$. Since $\mathcal{A}_{FSK}$ can receives only random values from oracles, i.e., all received values are independent from the information of the secret key, $\mathcal{A}_{FSK}$ wins, only when it happens to guess the correct $\hat{sk_{\mathsf{ID},t}}$. Therefore, $Pr[\mathcal{A}_{FSK}\ in\ Game3] = \frac{1}{2^{n_s}}$. Assuming the hash function H_2 is a random oracle,

$$\begin{aligned} Pr[\mathcal{A}_{FSK}\ in\ Game3] &= Pr[\mathcal{A}_{FSK}\ in\ Game2] \\ &= \frac{j-1}{2^{n_s}}. \end{aligned}$$

From Game 0, 1, 2, 3, the following is shown.

$$\begin{aligned} Pr[\mathcal{A}_{FSK}\ in\ Game0] &= Pr[\mathcal{A}_{FSK}\ in\ Game1] \\ &\leq Pr[\mathcal{A}_{FSK}\ in\ Game2] + \frac{q}{2^{n_s}} \\ &= Pr[\mathcal{A}_{FSK}\ in\ Game3] + \frac{q}{2^{n_s}} \\ &= \frac{j-1}{2^{n_s}} + \frac{q}{2^{n_s}} = \frac{q+j-1}{2^{n_s}} \end{aligned}$$

As a result, it can be shown that the proposed scheme is *forward-secure key-incrementable*, if $q << 2^{n_s}$ and H_0 is random oracle. ¶