Regular Paper

Unlinkable Identification for Large-scale RFID Systems

Yasunobu Nohara,† Toru Nakamura,† Kensuke Baba,†
Sozo Inoue† and Hiroto Yasuura†

Unlinkability, the property that prevents an adversary recognizing whether outputs are from the same user, is an important concept in RFID. Although hash-based schemes can provide unlinkability by using a low-cost hash function, existing schemes are not scalable since the server needs $O(N)$ hash calculations for every ID matching, where N is the number of RFID devices. Our solution is the *K-steps ID matching scheme*, which can reduce the number of hash calculations on the server to $O(\log N)$. In this paper, we explain the protocol, describe a test implementation, and discuss the application of this scheme to practical RFID systems. We also compare the scheme with other hash-based schemes from various viewpoints.

1. Introduction

RFID (Radio Frequency IDentification) is a technology to identify humans and objects through *RFID devices*; that is, silicon chips with IDs and radio frequency functions. In RFID, the server identifies an RFID device by *ID matching*, where the server compares the output of each RFID device with the IDs stored in the server. As pervasive computing environments become more commonplace, RFID devices, such as contact-less smart cards and RFID tags, are becoming part of our daily life.

Privacy is a serious concern when RFID is used. For example, the *location privacy problem* – the possibility that an adversary can trace a user's behavior by reading and linking a device's ID – is a privacy issue.

Unlinkability is a property which means an adversary cannot recognize whether outputs are from the same user, and this property is important with respect to the privacy problem. The randomized hash lock scheme [1] and the hash-chain scheme [2),3)] provide unlinkability against an adversary by using a hash function. These schemes are suitable for RFID systems because the implementation cost of an RFID device must be low in these systems. However, these schemes are not scalable since the server needs $O(N)$ hash calculations for every ID matching, where N is the number of RFID devices. Our solution is the *K-steps ID matching scheme* [4], which can reduce the number of hash calculations on the server to $O(\log N)$.

In this paper, we explain the protocol of this scheme, describe a test implementation, and discuss application of the scheme to practical RFID systems. In addition, we compare the scheme with other hash-based schemes from various viewpoints.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 explains the K-steps ID matching scheme, and discusses the ID structure. Section 4 describes the implementation of the scheme and discusses its application to practical RFID systems. Section 5 compares this scheme with other hash-based schemes. Section 6 concludes this paper.

2. Related Work

The randomized hash lock scheme [1] and the hash-chain scheme [2),3)] use a hash function to provide unlinkability against an adversary. These schemes are suitable for RFID systems, where the implementation cost of an RFID device must be low.

Let N be the number of RFID devices in an RFID system where the ID id_i of an RFID device D_i is a string of length L over a finite alphabet Σ for $1 \leq i \leq N$. We assume that if $i \neq j$, then $id_i \neq id_j$ for $1 \leq i, j \leq N$, and $2^L \gg N$. For $s, t \in \Sigma^*$, we denote by $s\|t$ the concatenation of s and t.

2.1 Randomized Hash Lock Scheme [1]

In this scheme, a hash function H, a ROM, and a pseudo-random number generator are embedded within each RFID device.

RFID device D_i stores id_i in the ROM. The server stores the IDs id_i $(1 \leq i \leq N)$ of all devices. The ID matching protocol of this scheme is as follows.

STEP1: RFID device D_i generates a random number R, and sends $X = H(id_i\|R)$

† Kyushu University

and R to the server.

STEP2: The server finds id_i that corresponds to X by checking $X = H(id_i \| R)$ for $1 \leq i \leq N$.

$X = H(id_i \| R)$ is not fixed since R changes every time. Hence, 2^L hash calculations are necessary when an adversary tries to get id_i from X and R. It is computationally difficult to calculate the hash function 2^L times. Therefore, this scheme provides unlinkability against an adversary.

2.2 Hash-chain Scheme [2),3)]

In this scheme, two different hash functions H and G, a ROM, and a non-volatile memory are embedded within each RFID device.

RFID device D_i stores id_i in the ROM, and stores secret information $cs_i^1 \in \Sigma^{L'}$ in the non-volatile memory. The server stores the pair (id_i, cs_i^1) $(1 \leq i \leq N)$ of all devices. The ID matching protocol of this scheme is as follows.

STEP1: RFID device D_i sends $X = H(id_i \| cs_i^l)$ to the server. RFID device D_i updates $cs_i^{l+1} \leftarrow G(cs_i^l)$.

STEP2: The server finds the id_i corresponding to X by checking $X = H(id_i \| cs_i^l)$ for all $1 \leq i \leq N$ and all $1 \leq l \leq M$ (where M is the maximum length of the hash chain).

$X = H(id_i \| cs_i^l)$ is not fixed since cs_i^l changes every time. Hence, $2^{L+L'}$ hash calculations are necessary if an adversary tries to get id_i from X. It is computationally difficult to calculate the hash function $2^{L+L'}$ times. Therefore, this scheme provides unlinkability against an adversary. Moreover, it is computationally hard to get $cs_i^{l'}(l' < l)$ even if id_i and cs_i^l are tampered with. Therefore, the scheme provides forward security, meaning that no RFID device can be traced from past ID information even if the secret information in the device is tampered with.

2.3 Problems of Existing Hash-based Schemes

In the randomized hash lock scheme and the hash-chain scheme, the server needs to calculate a hash function for every candidate (e.g., $id_1, id_2, \cdots, id_N$) in every ID matching. This means these schemes are not scalable since the server has to perform $O(N)$ hash calculations.

Avoine, et al. [5),6)] developed a specific time-memory trade-off that reduces the amount of computation in the hash-chain scheme [2),3)]. This time-memory trade-off reduces the hash calculations on the server with help of pre-computation results. However, heavy pre-calculation is needed with Avoine's scheme [5),6)].

3. K-steps ID Matching Scheme

In this section, we describe the K-steps ID matching scheme [4)]. First, we explain the basic ideas for reducing the time complexity, and then we show a method for generating IDs and a protocol for ID matching.

Molnar, et al. propose a similar approach of reducing the number of hash calculations by constructing a tree of IDs [7)]. Compared to their work, we consider the ID structure more generally and address the question of the optimal tree form.

3.1 Basic Idea

To reduce the time complexity, we use group IDs. First, all RFID devices are classified into groups of size α, and a group ID is assigned to each RFID device in a group. When ID matching is executed, the number of hash calculations is reduced to $\frac{N}{\alpha}$ by having each RFID device send its group ID.

However, this approach weakens unlinkability since the group IDs will be exposed to attackers. To deal with this problem, our approach is to hash the group IDs, as well as the device IDs, to prevent group IDs being exposed to an adversary. This preserves unlinkability. The time complexity of the server becomes $\frac{N}{\alpha} + \alpha$, which is greater than that of the previous approach by α.

Furthermore, we apply the above procedures recursively.

Based on these approaches, we improve the generation of IDs and reduce the time complexity on the server by utilizing the tree property.

3.2 ID Configuration

We use a labeled tree of depth K, such as the tree shown in **Fig. 1**. The tree has N leaves, and each leaf corresponds to an RFID device. Each node has a unique label. ID id_i of an RFID device corresponding to a leaf node is defined as the sequence of labels from the root node to the leaf node (e.g., a2bX@ in Fig. 1).

In the following, the k-th $(1 \leq k \leq S_i)$ label of D_i is denoted by $id_i[k]$, where S_i is the depth of leaf i, and $1 \leq S_i \leq K$.

3.3 Protocol

In the K-steps ID matching scheme, the server recognizes an ID from the output of an RFID device through the following protocol.

STEP1: RFID device D_i generates a random number R. D_i then sends $(R, X_1, X_2, \ldots, X_K)$ to the server, where X_k is

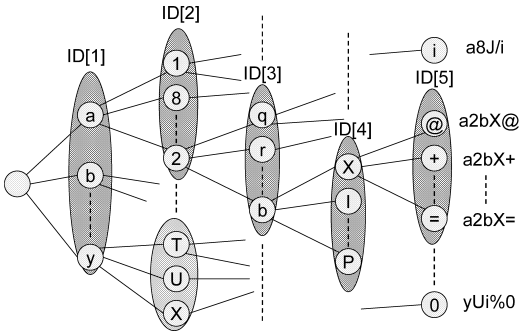

Fig. 1 An ID structure for the K-steps ID matching scheme.

$H(id_i[k]\|R)$ if $1 \le k \le S_i$ and a random number R_k if $S_i + 1 \le k \le K$.

STEP2: The server operates as follows:

STEP2-1: let Z be the root of the labeled tree and let $k \leftarrow 1$;

STEP2-2: find L_i s.t. $H(L_i\|R) = X_k$ by computing $H(L_i\|R)$ for each child L_i of Z, and update $Z \leftarrow L_i$;

STEP2-3: output the label corresponding to Z as the ID of the RFID device if Z is a leaf; otherwise, let $k \leftarrow k + 1$ and return to STEP2-2.

In Step 1, RFID device D_i sends a random number as X_k for $S_i + 1 \le k \le K$, which hides the depth of the leaf S_i to prevent weakening the unlinkability against an adversary.

When $K = 1$, the proposed protocol and the ID structure of the protocol correspond to those of the randomized hash lock scheme [1]. If some procedures of the protocol are changed, it becomes a protocol corresponding to the hash-chain scheme [2),3)].

3.4 Number of Hash Calculations

We next analyze the time complexity of the K-steps ID matching scheme. Specifically, we consider the expected number of hash calculations on the server and the RFID devices.

Although the time complexity depends on several factors (for example, the time for string matching), in practice, the hash calculation is the most important. The number of hash calculations depends on (at least)

- the number of leaves N,
- the depth K, and
- the number of edges α_n for each node n of the labeled ID tree.

We first find the number of edges α_n with respect to each node that minimizes the number of hash calculations on the server. In Section 4, we discuss the optimized K to minimize the total execution time.

We assume that the number of hash calculations necessary for ID matching with m candidates is m. The ID matching with the labeled tree is solved by K times ID matchings with α_n candidates. We also assume that any leaf is of the same depth K and $N^{\frac{1}{K}}$ is an integer.

Lemma 1 *The expected number of hash calculations for an ID matching with a labeled tree of depth K with N leaves is at least $KN^{\frac{1}{K}}$ for any $N > 0$ and any $1 \le K \le N$.*

Proof 1 *By induction for K. If $K = 1$, the number of hash calculations is N. Let ν be the root of the tree and $\nu_1, \nu_2, \ldots, \nu_{\alpha_\nu}$ be the children of ν. We denote by $g_K(N)$ the expected number of hash calculations for an ID matching with a tree of depth K with N leaves. Then,*

$$g_K(N) = \alpha_\nu + \sum_{i=1}^{\alpha_\nu} \frac{n_i}{N} g_{K-1}(n_i),$$

where n_i is the number of leaves in the sub-tree whose root is ν_i for $1 \le i \le \alpha_\nu$ and $\sum_{i=1}^{\alpha_\nu} n_i = N$. By induction hypothesis, $g_{K-1}(n_i)$ is at least $(K-1)n_i^{\frac{1}{K-1}}$ for each $1 \le i \le \alpha_\nu$. Hence, $g_K(N)$ is at least $\alpha_\nu + \frac{K-1}{N} \sum_{i=1}^{\alpha_\nu} n_i^{\frac{K}{K-1}}$. Since $\frac{K}{K-1}$ is larger than unity and $\sum_{i=1}^{\alpha_\nu} n_i = N$, the expected number is minimal if $n_i = \frac{N}{\alpha_\nu}$ for any $1 \le i \le \alpha_\nu$. Therefore, we have only to consider the case where $g_K(N)$ is of the form

$$\alpha_\nu + \frac{K-1}{N} \cdot \alpha_\nu \cdot \left(\frac{N}{\alpha_\nu}\right)^{\frac{K}{K-1}}$$
$$= \alpha_\nu + (K-1) \cdot N^{\frac{1}{K-1}} \cdot {\alpha_\nu}^{-\frac{1}{K-1}}.$$

This is minimal only if $1 - N^{\frac{1}{K-1}} {\alpha_\nu}^{-\frac{K}{K-1}} = 0$ and therefore $\alpha_\nu = N^{\frac{1}{K}}$. Thus, the minimal number of $g_K(N)$ is

$$N^{\frac{1}{K}} + (K-1) N^{\frac{1}{K-1}} N^{\frac{1}{K} - \frac{1}{K-1}} = KN^{\frac{1}{K}}.$$

By the previous proof, the number of hash calculations for a single matching is the minimal number $KN^{\frac{1}{K}}$ if the number of edges for each node is $\alpha_n = N^{\frac{1}{K}}$. Therefore, in the rest of this paper, we consider a labeled tree in which any node has the same number of children α. Then, $KN^{\frac{1}{K}} = \alpha \log_\alpha N$. Therefore, the previous lemma implies the following theorem.

Theorem 1 *The K-steps ID matching protocol can find an ID in N candidates by $O(\log N)$ time.*

The number of hash calculations in the RFID device is K. Therefore, the time of total hash

Table 1 Execution time on the RFID device for various values of K.

K	1	2	3	4	5	6	7	8
t_d[s]	0.6333	0.9085	1.1950	1.4835	1.7638	2.0281	2.3097	2.5896

calculation T_{hash} is

$$T_{hash} = \beta_s K N^{\frac{1}{K}} + \beta_d K, \quad (1)$$

where β_s is the time of one hash calculation on the server and β_d is the time of one calculation on the RFID device.

4. Evaluation

In this section, we first show the implementation result of the proposed scheme. After that, we show the results from a simulation using the basic parameters obtained from the implemented scheme. Last, we discuss the application of the K-steps ID matching scheme to practical RFID systems.

4.1 Environments

Considering the typical environments for implementation, we adopted Java cards as RFID devices on which to implement our scheme. Moreover, we adopted SHA-1 [8] as the hash functions. We set the length of a random number to 4 bytes, and the length of an ID to 28 bytes. In the following, execution times were obtained as an average of 10 trials.

The following are the execution environments of the RFID devices and the server.

RFID device (Cyberflex Access e-gate 32 K)

- OS: Java Card OS
- ROM: 96 KB
- RAM: 4 KB
- EEPROM: 32 KB
- equipped with a cryptographic co-processor and random number generator hardware

Server

- OS: Linux Fedora Core 4 (kernel-2.6.13)
- Compiler: GCC 4.0.1 with O3 Option
- CPU: Pentium4, 1.7 GHz
- Memory: 1 GB

4.2 Implementation Results

For an RFID device, let t_d be *the execution time on the RFID device* from the time it receives a command from the server to the time it returns the hashed value of the ID to the server, where t_d corresponds to the time of STEP 1. Additionally, let t_s be *the execution time on the server* from the time it receives the hashed ID from the RFID device to the time it finishes searching the ID list, where t_s corresponds to STEP 2. We call $T = t_d + t_s$ the *total execution time*.

The execution time t_s on the server in the case of 1 million IDs and K=1 was 1.788 seconds, where the target ID was assumed to be found at the last entry in the IDs. To investigate the contents of the execution time, we examined 1 million SHA-1 hash calculations on the server, which took 1.785 seconds. Thus, the hash calculations clearly made up a major portion of t_s.

On the other hand, the execution time t_d on the RFID device for various K values was as shown in **Table 1**.

From the above, we can denote the execution time on the RFID device as

$$t_d = 0.2795\,K + 0.3561$$

with a correlation coefficient of 1.00, where the slope corresponds to the increased time of the hash calculations and the communication as K increases, and the constant time includes processes such as random number generation that are independent of K.

From the results shown in this section, the total execution time in this implementation is given as

$$T = 1.785 \times 10^{-6} \cdot K N^{\frac{1}{K}} + 0.2795\,K + 0.3561$$

4.3 Simulation

Using the formula obtained in the previous section, we simulated the total execution time varying N and K. **Table 2** and **Fig. 2** show the simulation results.

In the table, the values that are the minimum execution time for each N are shown in bold.

The simulation results show that the total execution time does not increase as N increases.

4.4 Discussion

From Eq. (1), the relationship between the optimized K and N, β_d/β_s, is as follows.

K	small	$\longleftrightarrow$	large
N	small	$\longleftrightarrow$	large
β_d/β_s	large	$\longleftrightarrow$	small

According to the implementation result, β_d/β_s is large (about 150,000). Therefore, the optimized K is much less than 10 even if N becomes 2^{100}. This means the proposed scheme's degree of unlinkability decreases only a little [4] compared with the other hash-based schemes [1)~3)].

Table 2 Simulation results for total execution time.

N	K							
	1	2	3	4	5	6	7	8
2^{10}	**0.6370**	0.9150	1.195	1.474	1.754	2.033	2.313	2.592
2^{20}	2.507	**0.9190**	1.195	1.474	1.754	2.033	2.313	2.592
2^{30}	1.917e3	**1.032**	1.200	1.475	1.754	2.033	2.313	2.592
2^{40}	1.963e6	4.659	**1.250**	1.481	1.756	2.034	2.313	2.593
2^{50}	2.010e9	120.7	1.752	**1.515**	1.763	2.037	2.314	2.593
2^{60}	2.058e12	3.834e3	6.810	**1.708**	1.790	2.044	2.317	2.595
2^{70}	2.107e15	1.227e5	57.79	2.798	**1.900**	2.068	2.325	2.598
2^{80}	2.158e18	3.925e6	571.7	8.961	2.339	**2.144**	2.347	2.607
2^{90}	2.210e21	1.256e8	5751	43.83	4.093	**2.384**	2.405	2.627
2^{100}	2.263e24	4.019e9	5.796e4	241.1	11.11	3.147	**2.562**	2.675

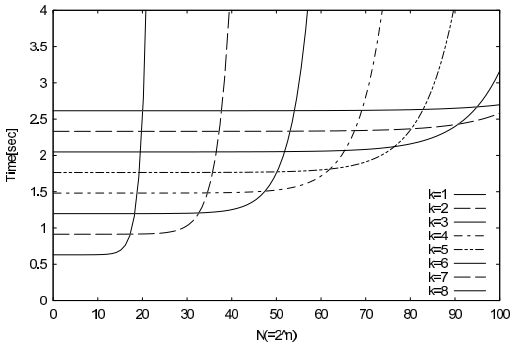

Fig. 2 Simulation results for total execution time.

Generally, the cost of an RFID tag must be low relative to the cost of a smart card. Therefore, we predict K in a practical RFID system will be smaller than the simulation result.

5. Comparison

In this section, we compare the proposed scheme with other hash-based schemes from the viewpoints of security, hash calculation time, the amount of memory needed, and the amount of communication.

5.1 Classification of Hash-Based Schemes

For our comparison, we classify hash-based schemes with regard to three characteristics:

- Base model (hash lock or hash chain)
- ID structure (normal or tree)
- Introduction of a time-memory trade-off technique [5] (yes or no)

The compared schemes are our K-steps ID matching scheme, the randomized hash lock scheme [1], the hash-chain scheme [2),3)], the tree-based private authentication scheme [7], Avoine's scheme [5),6)], and Yeo, et al.'s scheme [9].

We described the randomized hash lock scheme, the hash-chain scheme, and Avoine's scheme in Section 2, and introduced the tree-based private authentication scheme in Section 3. We describe Yeo's scheme in the following.

Yeo's scheme is a hash-chain scheme, and the same grouping technique as in our scheme is used to reduce the server complexity. Yeo proposes two types of scheme. One is a scheme without pre-computation which uses only a grouping technique. The other is a scheme with pre-computation which uses both a grouping technique and a time-memory trade-off technique [5].

Yeo, et al. described the schemes with an ID tree depth of 2 [9]. The tree structure of Yeo's schemes may be extended more generally as in our scheme.

Table 3 shows the classification results.

There are no proposed schemes with the combination (Hash Lock, Normal, Yes) or (Hash Lock, Tree, Yes) because the responses of RFID devices in a hash lock scheme are randomized, which means a large memory space is needed to apply a time-memory trade-off technique [6].

5.2 Security

We compare the security of the hash-based schemes with respect to three concerns:

- Unlinkability
- Forward security
- Prevention of replay attacks

5.2.1 Unlinkability

Unlinkability means the difficulty of finding relations between the outputs of RFID devices. Achieving unlinkability is important to prevent an adversary tracing a user's behavior.

We analyzed the unlinkability of the hash-based schemes by the *degree of unlinkability* [4]. The degree of unlinkability ranges from 0 to $\log_2 N$ [bit], and unlinkability becomes stronger as the degree of unlinkability increases.

When an adversary has no ID information, each degree of unlinkability for the hash-based

Table 3 Classification of Hash-Based Schemes.

	Base Model	ID Struct.	Time-memory
Hash Lock [1)]	Hash Lock	Normal	No
K-step [4)], Tree-based [7)]	Hash Lock	Tree	No
Hash-chain [2),3)]	Hash Chain	Normal	No
Avoine's scheme [5),6)]	Hash Chain	Normal	Yes
Yeo's without pre-comp. [9)]	Hash Chain	Tree (K=2)	No
Yeo's with pre-comp. [9)]	Hash Chain	Tree (K=2)	Yes

schemes is $\log_2 N$.

When an adversary obtains one ID, such as by tampering with an RFID device, the degree of unlinkability of each scheme differs depending on its ID structure. The degree of unlinkability for the normal ID structure and that for the tree ID structure are given as follows [4)].

$$U_{normal} = \frac{N-1}{N} \log_2 (N-1) \tag{2}$$

$$U_{tree} = \log_2 N + \frac{N-1}{N} \left\{ \log_2 \left(N^{\frac{1}{K}} - 1 \right) - \frac{N^{\frac{1}{K}}}{K \left(N^{\frac{1}{K}} - 1 \right)} \log_2 N \right\} \tag{3}$$

The normal ID structure schemes enable user unlinkability, except for the tampered user, but the tree ID structure schemes cannot enable user unlinkability since some users share part of the ID of the tampered user. From Eqs. (2) and (3), we can see that the degree of unlinkability with the tree ID structure is lower than that with the normal structure.

However, the tree ID structure schemes provide the same level of unlinkability as the normal ID structure if $\alpha = N^{\frac{1}{K}}$ is large enough. Since the optimized K is much less than 10 even if N becomes 2^{100}, as we discussed in Section 4, the tree ID structure decreases the degree of unlinkability only slightly.

Thus, the decrease in the degree of unlinkability with the proposed scheme is only small [4)] compared to that with the normal ID structure.

5.2.2 Forward security

Forward security is a property that means no RFID device can be traced from past ID information even if an adversary tampers with the secret information in the device.

Hash lock schemes, including our scheme, cannot provide forward-security because an adversary can easily get a random number R. On the other hand, hash-chain schemes can provide forward security since it is computationally difficult for an adversary to get $cs_i^{l'}$ $(l' < l)$ even if he has tampered with id_i and cs_i^l.

However, Juels, et al. pointed out that hash-chain schemes create a security risk in that an adversary can guess a device's count number [10)]. We discuss this problem in Section 5.3.

5.2.3 Prevention of Replay Attacks

A replay attack is one in which a valid data transmission is maliciously or fraudulently repeated. The attack is carried out by an adversary who masquerades as a legitimate user.

Replay attacks must be prevented when a server has to authenticate as well as identify an RFID device. One way to do this is to use a fresh challenge by the server. Hash lock schemes can prevent replay attacks if a step is added where the server sends a fresh challenge to the device and includes the challenge in the hash calculations. In our scheme, the protocol to prevent replay attacks is as follows.

STEP1: The server generates a random number R_s, and then sends R_s to RFID device D_i.

STEP2: RFID device D_i generates a random number R_d, and then sends $(R_d, X_1, X_2, \ldots, X_K)$ to the server, where X_k is $H(id_i[k] \| R_s \| R_d)$ if $1 \le k \le S_i$, and a random number R_k if $S_i + 1 \le k \le K$.

STEP3: The server operates as follows:

- **STEP3-1:** let Z be the root of the labeled tree and let $k \leftarrow 1$;
- **STEP3-2:** find L_i s.t. $H(L_i \| R_s \| R_d) = X_k$ by computing $H(L_i \| R_s \| R_d)$ for each child L_i of Z, and update $Z \leftarrow L_i$;
- **STEP3-3:** output the label corresponding to Z as the ID of the RFID device if Z is a leaf; otherwise, let $k \leftarrow k + 1$ and return to STEP 3-2.

Avoine, et al. propose a modified hash-chain scheme which prevents replay attacks using a challenge [6)]. This technique can be easily adopted in Yeo's scheme without pre-computation.

However, the technique of using a fresh challenge cannot be applied directly to Avoine's scheme or Yeo's scheme with pre-computation

Table 4 Comparison of required memory and time.

	Hash Calc. on Device	Hash Calc. on Server	Pre-comp. on Server	Memory on Server
Hash Lock	1	N	0	0
K-step, Tree-based	K	$KN^{\frac{1}{K}}$	0	0
Hash-chain	2	MN	0	N
Avoine's scheme	2 [+1]	$\frac{3^3}{2^3}\frac{M^3\gamma}{c^3\mu^2}$ [+1]	$\frac{NM^2}{2}$	cN
Yeo's without pre-comp.	4	$2M\sqrt{N}$	0	N
Yeo's with pre-comp.	4 [+1]	$\left(\frac{2^5M^6\gamma}{c^3\mu^2}\right)^{\frac{1}{4}}$ [+1]	$\left(\frac{2^3c^3N^4\mu^2}{3^4M^2\gamma}\right)^{\frac{1}{4}}\frac{M^2}{2}$	cN

Table 5 Communication cost.

	not preventing replay attacks	preventing replay attacks
Hash Lock	$r+h$	$2r+h$
K-step, Tree-based	$r+Kh$	$2r+Kh$
Hash-chain	h	$r+h$
Avoine's scheme	h	$r+2h$
Yeo's without pre-comp.	$2h$	$r+2h$
Yeo's with pre-comp.	$2h$	$r+3h$

since the randomization of the device's response prevents the server using a time-memory trade-off (see Section 5.1). Therefore, the RFID device must calculate a hash value (s) without a challenge and a hash value with the challenge [6]. The former value (s) enable (s) the server to identify the device, while the latter one prevents replay attacks.

All of the hash-based schemes proposed so far have a countermeasure against replay attacks, and preventing replay attacks increases both the calculation complexity and the communication amount. We discuss this problem in Sections 5.3 and 5.4.

5.3 Comparison of memory and time

In this section, we compare the different schemes regarding the number of hash calculations on the RFID device and on the server, the number of pre-computations on the server, and the memory required for the pre-computation results.

Table 4 compares the memory and the time needed for each scheme. In the table, M is the maximum length of the hash-chain, μ is the conversion factor, c is the memory size parameter for Avoine's scheme [5], and γ is the rate of successful search parameter in that scheme. For example, the success rate is 99.9% when $\gamma = 8$.

In the table, 'Memory on server' denotes the amount of secret information to be stored cs_i^l in hash-chain schemes. Note that the memory amount does not include the space for the ID list, which is required for every scheme.

The additional number of calculations for the scheme to prevent replay attacks is given in brackets.

As the table shows, the number of hash calculations on the server in a time-memory trade-off scheme includes M^3 or $M^{1.5}$, while that in the K-step ID matching scheme includes N. Therefore, our scheme might be disadvantageous in terms of the required time if M^3 or $M^{1.5}$ is sufficiently smaller than $N^{\frac{1}{K}}$.

The server cannot identify the RFID device when the device number is larger than M because it will be outside of the search range. In addition, there is a security risk in that an adversary can guess a device's count number if M is small [10]. Therefore, M must be sufficiently large.

Avoine, et al. pointed out that replacing cs_i^1 by cs_i^k in the database regularly expands the search range of the server [5]. However, the problem of M being too small remains, and heavy pre-computation (e.g., $M^2N/2$) is needed for every replacement.

5.4 Communication cost

Table 5 compares the communication cost for each scheme in terms of the amount of communication data. The costs are shown in each case of preventing or not preventing replay attacks. In the table, r is the length of the random value for the challenge, and h is the length of the hash output.

In tree ID structures, including those of the

K-step ID matching scheme, the communication cost increases in proportion to the tree depth. For the K-step ID matching scheme, we measured the practical time for the entire execution (including the communication time between the server and the RFID device), and found it is shorter than that of a naive scheme [1)] when N is sufficiently large.

Thus, we expect the communication cost of our scheme to be negligible in a practical situation. However, further evaluation is required since we used contact smart cards in our experiment. With contact-less smart cards or RFID tags, the communication cost might increase because of communication failures.

6. Conclusion

In this paper, we described the K-steps ID matching scheme and explained how it can reduce the number of hash calculations on the server to $O(\log N)$. We also showed results from an implementation of the proposed scheme, and discussed its application to practical RFID systems. In addition, we compared this scheme to other hash-based schemes from several viewpoints.

Acknowledgments This work has been partly supported by a Grant-in-Aid for Creative Scientific Research, No.14GS0218, and the 21st Century COE Program "Reconstruction of Social Infrastructure Related to Information Science and Electrical Engineering". We are grateful for this support.

References

1) Weis, S.A., Sarma, S.E., Rivest, R.L. and Engels, D.W.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems, *1st International Conference on Security in Pervasive Computing* (*SPC2003*), LNCS, Vol.2802, pp.201–212, Springer (2004).
2) Ohkubo, M., Suzuki, K. and Kinoshita, S.: Cryptographic Approach to a Privacy Friendly Tag, *RFID Privacy Workshop@MIT* (2003).
3) Ohkubo, M., Suzuki, K. and Kinoshita, S.: Hash-Chain Based Forward-Secure Privacy Protection Scheme for Low-Cost RFID, *2004 Symposium on Cryptography and Information Security* (*SCIS2004*), Vol.1, pp.719–724 (2004).
4) Nohara, Y., Inoue, S., Baba, K. and Yasuura, H.: Quantitative Evaluation of Unlinkable ID Matching Schemes, *2005 ACM Workshop on Privacy in the Electronic Society* (*WPES2005*), pp.55–60, ACM Press (2005).
5) Avoine, G. and Oechslin, P.: A Scalable and Provably Secure Hash-Based RFID Protocol, *2nd International Workshop on Pervasive Computing and Communications Security* (*PerSec2005*), pp.110–114, IEEE Computer Society Press (2005).
6) Avoine, G., Dysli, E. and Oechslin, P.: Reducing Time Complexity in RFID Systems, *12th Annual Workshop on Selected Areas in Cryptography* (*SAC2005*), LNCS, Vol.3897, pp.291–306, Springer (2005).
7) Molnar, D. and Wagner, D.: Privacy and Security in Library: RFID Issues, Practices, and Architectures, *11th ACM Conference on Computer and Communications Security* (*CCS2004*), pp.210–219, ACM Press (2004).
8) National Institute of Standards and Technology: *SECURE HASH STANDARD*, Federal Information Processing Standards Publication 180-2 (2002).
9) Yeo, S.-S. and Kim, S.K.: Scalable and Flexible Privacy Protection Scheme for RFID Systems, *2nd European Workshop on Security in Ad-Hoc and Sensor Networks* (*ESAS2005*), LNCS, Vol.3813, pp.153–163, Springer (2005).
10) Juels, A. and Weis, S.A.: Defining Strong Privacy for RFID, *IACR Cryptology ePrint Archive Report*, No.2006-137 (2006).

(Received November 29, 2005)
(Accepted June 1, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.489–497.)

Yasunobu Nohara is a Ph.D. candidate at the Graduate School of Information Science and Electrical Engineering, Kyushu University. He received his B.E. and M.E. degrees in Computer Science from Kyushu University. His current research focuses on the privacy and security of RFID systems. He is a student member of IPSJ, IEICE, and IEEE.

Toru Nakamura is a master's student at the Graduate School of Information Science and Electrical Engineering, Kyushu University. He received his B.E. degree in Electrical Engineering and Computer Science from Kyushu University. His current research focuses on computer security and social infrastructure.

Kensuke Baba is a research associate of the Department of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University. He received his B.E. and M.E. degrees and his Ph.D. in Science from Kyushu University in 1996, 1998, and 2002, respectively. His current interests include algorithms for string processing and formal analysis of security. He is a member of IPSJ.

Sozo Inoue is a research associate in the Graduate School of Information Science and Electrical Engineering, Kyushu University, and in the System LSI Research Center, Kyushu University. His research interests include RFID information systems, particularly security, privacy, and reliability in RFID systems, system LSIs, and database systems. Inoue was born in 1974 and received his Doctorate of Engineering from Kyushu University. He is a member of IPSJ, the Database Society of Japan (DBSJ), ACM, and the IEEE Computer Society. He can be reached at sozo@acm.org.

Hiroto Yasuura is a professor of the Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University. He is also a director of the System LSI Research Center in Kyushu University. Prof. Yasuura received his B.E. and M.E. degrees and his Ph.D. in Computer Science from Kyoto University, Kyoto, Japan, in 1976, 1978, and 1983, respectively. His current interests include embedded system design, hardware/software codesign, system design methodology, and social infrastructure. He is a member of IPSJ, IEICE, ACM, and IEEE.