We study hash-based identification protocols in RFID systems which obtain privacy against active adversaries who can perform compromised attacks. Here, an active adversary can track a tag via successful or unsuccessful identifications with legal or illegal readers. In IEEE Transactions on Parallel and Distributed Systems, Alomair, Clark, Cuellar and Poovendran propose a novel hash-based RFID identification protocol called constant-time-identification protocol (CTI). Their protocol is provably against active adversaries who can perform compromised attack. However, in order to obtain high privacy in their protocol, the database in CTI is required to have quite large memory space.

In this work, we try to reduce the memory complexity of the database in CTI while preserving the same level of privacy. We discover that there is a large gap between tag's identification time and tag’s response time in the protocol CTI. We trade-off tag identification time for the sake of reducing the memory space. By using the well-known time-memory-trade-off techniques, we can obtain this goal. With a performance analysis, we show that the memory space of our modified protocol can be reduced by a factor \(t\) compared to that of CTI where the number of hash operations for one tag’s identification is \(O(t^2)\). Via a case study, we show that our modified protocol is able to save about 89% fraction of memory space of CTI in some practical setting of RFID systems while increasing the tag identification time.

Keywords: RFID identification, hash-based protocols, time-memory-trade-off, constant-time-identification, RFID privacy

1. INTRODUCTION AND RELATED WORKS

RFID (Radio-Frequency Identification) is a technology in which one can identify objects or people by embedding tags, a small microchip capable of wireless data transmission. RFID has many applications in the real life. In a shopping store, goods can be tagged in order to speed up the process of registration with wireless scanning instead of optical scanning. There are several characteristics for RFID tags. First, there is a unique identifier in each tag which can represent the tag itself. Second, tags can be identified at a distance. These characteristics introduce some important privacy issues. Objects embedded with tags that do not reveal any sensitive information may also be tracked. This is because the tag’s responses to the requesting readers are possible to help locate the tagged objects by analyzing information from the protocol view between the embedded
tag and the reader. This may cause objects to reveal their private data such as their identifications in the future. We refer the readers to two excellent survey papers on the privacy issue: one is [13] by Juels and the other is [18] by Najera and Lopez. We also refer the readers to the paper of Zhang et al. [23] in which RFID location issues are introduced.

There are many privacy definitions proposed in the past decade, e.g. [1, 5, 14, 16]. In this paper, we consider the C-existentaxial privacy model of Alomair et al. [5]. Basically, in their model, a system obtaining C-existentaxial privacy is defined with the following game. First, the challenger chooses two tags $T_0$ and $T_1$. Second, the adversary interacts with each tag for at most $C - 1$ number of times where $C$ is a pre-specified system security parameter. Third, the challenger chooses a random bit $b$, and sets $T = T_b$. Finally, the adversary interacts with $T$ once and predicts whether $b = 0$ or 1. An RFID system obtains C-existentaxial privacy if no efficient adversary can guess $b$ correctly with a non-negligible probability greater than one half. If a system obtains $\lambda^{(0)}$-existential privacy where $\lambda$ is the security parameter, then we say that it has optimal existential privacy.

Recently, there are many protocols which are designed based on non-symmetric-key cryptography such as protocols based on elliptic curve cryptography [9-11] or protocols based on the use of physically unclonable functions [3, 4]. However, the cost of protocols based on the non-symmetric-key cryptography is higher than the cost of those protocols based on the symmetric-key cryptography. Thus, for practical reasons [2, 5], we only consider RFID systems which use the symmetric-key cryptography to make the communication between tags and readers secure. Weis et al. propose a hash-based protocol called randomized hash lock which obtains optimal privacy [21]. However, the optimal privacy is obtained by trade-offing tag identification efficiency. In fact, the time complexity to identify one tag is linear in the number of legitimate tags. So the randomized hash-lock protocol is not scalable. For the sake of scalability, Molnar and Wagner propose a tree-based protocol in which the keys are stored in a tree structure [17]. In this tree, each edge corresponds to a unique secret key, each leaf corresponds to a unique tag, and each tag owns the secret keys corresponding to the edges in the path from the root to its corresponding leaf. In their protocol, the time to identify one tag is the logarithm of the number of legitimate tags. However, the communication cost is also the same. In addition, it is easy to choose few tags such that there are many secret keys which are shared by these tags. This fact threatens privacy. To realize this attack, the so-called compromise attack against the tree-based protocol is proposed by Avoine et al. [6]. Since an adversary may be a user who owns some legitimate tags, he can knows all secret keys of his tags. From this, he is able to choose two tags such that one is located in the left side of his tag and the other is located in the right side of his tag in the tree. In [6], Avoine et al. show that, in a binary tree structure, compromising 20 tags in a system of $2^{20}$ tags is sufficient to identify some uncompromised tags with an average probability close to 1.

Another approach for solving scalability and keeping privacy simultaneously is to use hash chains to update the internal state of the tags. In [20], Ohkubo, Suzuki, and Kinoshita propose a protocol (called OSK) in which the identifier of the tag is modified each time it is queried by a reader such that the identifier can be recognized by legal readers. In the identification phase, the tag uses its identifier as an input of a hash function and sends the output to the reader. Then a legal reader can easily find out the tag from the received hash value. After completing identification, the tag and the database
both update the identifier by using another hash function. The problem in OSK is that
the tag can be interrogated by other illegal readers, its identifier is updated by the tag
each time it is queried, but the legal database does not update it. This gives rise to a syn-
chronizing issue. So, in order to identify a tag, the database computes a sequence of hash
values from the current identifiers stored in the database and checks if the received hash
value is equal to one of these hash values. This increases the time complexity of identi-
fication. In [8], Avoine and Oechslin use a time-memory trade-off technique to reduce
the identification complexity of OSK. By using this technique, the modified protocol
increases the storage of the database and decreases the identification time.

However, OSK is vulnerable against attacks of denial of service. To improve it,
Alomair Clark, Cuellar, and Poovendran propose a novel privacy-preserving protocol in
which each tag owns a reusable pseudonym and a counter with maximum value C. For
convenience, we call their protocol CTI. The main idea of CTI is that each tag uses a
pseudonym and a counter value to compute a hash value and the database can identify
the tag easily from this received hash value if these hash values are pre-stored in the da-
tabase. As a result, tag identification can be obtained with constant time [5]. However,
the level of privacy which CTI obtains is bounded by the maximum counter value C. In
order to obtain higher privacy, one must increase the space for storing the hash values in
CTI. Moreover, in CTI, the ability to be against compromise attacks depends on the
number of available pseudonyms. In order to have higher privacy against compromise
attacks, the back-end database should increases the set of pseudonyms. However, this
also increases the memory cost of the database.

1.1 Our Contributions

In this work, we use time-memory-trade-off (TMTO) techniques to reduce the me-
memory complexity of the database in CTI. We discover that there is a large gap between
tag’s identification time and tag’s response time in the protocol CTI. So, our idea is to
trade-off tag identification time for the sake of reducing the memory space. By using the
time-memory-trade-off technique [12, 19], we can obtain this goal. We show that the
memory space of our modified protocol is about $O(1/t)$ of that of CTI where the number
of hash operations for one tag’s identification is $O(t^2)$. Via a case study, we show that the
modified protocol is able to save about 89% fraction of memory space of CTI in some
practical setting of RFID systems.

Compared to [8], our protocol increases tag identification time and decreases the
memory space of CTI while the protocol of Avoine and Oechslin in [8] increases the
memory space and decreases time for tag identification of OSK.

1.2 Organization of This Paper

The remaining part of the paper is organized as follows. In Section 2, we give some
necessary definitions. Next, in Section 3, we introduce the constant-time identification
protocol of Alomair, Clark, Cuellar, and Poovendran. In Section 4, we propose our mod-
ified version of the constant-time identification protocol of Alomair, Clark, Cuellar, and
Poovendran. In Section 5, we give a case study of our proposed protocol. Finally we
conclude in Section 6.
2. PRELIMINARIES

2.1 RFID System Model

An RFID system $S$ consists of three parts: tags, readers, and a back-end database. We assume that the communication channel between the reader and the back-end database is secure while communication channel between the reader and each tag is insecure. The information about tags is assumed to be stored in the back-end database. Formally, we denote an RFID system $S$ as $(R, \{T_i\})$ where $\{T_i\}$ contains $N_T$ tags and $R$ is a reader. Each tag $T_i$ has an internal secret $k_i$ and a pseudonym $\psi_i$. For convenience, we use the following notations in the rest of the paper.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Corresponding Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_T$</td>
<td>Number of tags participat-\nging in the RFID system</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of pseudonyms</td>
</tr>
<tr>
<td>$k_i$</td>
<td>Secret key of the $i$th tag</td>
</tr>
<tr>
<td>$\psi_i$</td>
<td>Pseudonym of the $i$th tag</td>
</tr>
<tr>
<td>$c$</td>
<td>Internal counter of the tag</td>
</tr>
<tr>
<td>$h, g$</td>
<td>Hash functions</td>
</tr>
<tr>
<td>$c(k_i)$</td>
<td>The counter value generated from the key $k_i$</td>
</tr>
<tr>
<td>$\eta$</td>
<td>The running time of one hash operation (seconds)</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>The upper bound of tag’s response time (seconds)</td>
</tr>
</tbody>
</table>

2.2 Adversarial Model

We model an adversary $A$ as a polynomial-time algorithm. Given a tag $T$ and a reader $R$, $A$ is assumed to have access to the following oracles within polynomial number of times.

- **Query($T, a_1, t_2, a_3$):** First, $A$ sends message $a_1$ to $T$. Then, $A$ receives the response $t_2 = t_2(a_1)$. Finally, $A$ sends a message $a_3 = a_3(a_1, t_2)$ to $T$.
- **Send($R, r_1, a_2, r_3$):** First, $A$ receives message $r_1$ from the reader $R$. Then, $A$ replies $a_2 = a_2(r_1)$. Finally, $A$ receives the response $r_3 = r_3(r_1, a_2)$ from $R$.
- **Execute($T, R$):** When the tag $T$ and the reader $R$ execute a round of the authentication protocol, $A$ eavesdrops on their communication and can tamper with the messages exchanged between $T$ and $R$.
- **Block($\cdot$):** $A$ is allowed to block any part of the protocol.
- **Reveal($T$):** $A$ can know the secret information of the tag $T$.

In particular, the adversary can only call the **Reveal** oracle once on a tag. Such a tag can be viewed as the adversary’s own tag. In this sense, such a tag is called a compromised tag. We allow the adversary $A$ to call **Query, Send, Execute, Block** to the tag $T$ in order to track some other tag $T'$ after the **Reveal** oracle is applied to the tag $T'$. 
2.3 Security and Privacy Definitions

Before giving security and privacy definitions, we need the definition of negligible functions. A function \( g: \mathbb{N} \rightarrow \mathbb{R} \) is negligible if for any nonzero polynomial \( p \), there exists an integer \( n_0 \) such that for all integer \( n > n_0 \), \( |g(n)| < 1/p(n) \).

Honest protocol runs are defined as follows: a mutual authentication protocol run is said to be honest if the parties involved in the protocol run use their shared key to exchange messages, and the messages exchanged in the protocol run have been relayed without modification.

The formal definition of secure RFID mutual authentication is defined by the following game between the RFID system \( C \) and a polynomial-time adversary \( A \).

1. \( C \) chooses a tag \( T \) at random and a reader \( R \), and gives them to \( A \).
2. \( A \) calls the oracles \( \text{Query} \), \( \text{Send} \), \( \text{Execute} \), and \( \text{Block} \) using \( T \) and \( R \) for learning.
3. \( A \) stops learning and notifies \( C \).
4. \( A \) uses the oracle \( \text{Send} \) and \( \text{Query} \) to impersonate a tag and a reader, respectively.
5. If \( A \) is authenticated as a valid tag or a legal reader, then \( A \) wins the game.

We define the adversary’s probability of authentication \( \text{ADV}^\text{auth}_A \) as the probability that \( A \) wins the above game.

**Definition 1:** An RFID mutual authentication protocol is said to be secure if it satisfies the following conditions:

1. No information about the secret keys of an RFID tag is revealed by observing messages exchanged in protocol runs.
2. If the protocol run is honest, the tag-reader pair must authenticate each other with probability one.
3. If the protocol is not honest, then the probability of authentication \( \text{ADV}^\text{auth}_A \) is negligible in the security parameter for any polynomial adversary \( A \).

Next, we give formal definitions for the RFID privacy. The privacy definitions are based on the models proposed by Avoine [1]. As in [5], we define three privacy notions: universal privacy, forward privacy, and C-existential privacy.

Universal privacy is modeled by the following game between an RFID system \( C \) and a polynomial-time adversary \( A \).

**Game of Universal Privacy:**

1. \( C \) chooses two tags, \( T_0 \) and \( T_1 \), and a valid reader \( R \).
2. \( A \) makes queries on \( T_0 \), \( T_1 \), and \( R \) by using oracles \( \text{Query} \), \( \text{Send} \), \( \text{Execute} \), \( \text{Block} \) for a number of times of its choice.
3. After \( A \) stops querying, \( C \) carries out an instance of the protocol with \( T_0 \) and \( T_1 \), which is achieved during mutual authentication of both tags with \( R \).
4. \( C \) selects a random bit \( b \), sets \( T = T_b \), and sends \( T \) to \( A \).
5. \( A \) makes queries of \( T \) and \( R \) by using the oracles \( \text{Query} \), \( \text{Send} \), \( \text{Execute} \), \( \text{Block} \).
6. $A$ decides which of $T_0$ or $T_1$ is $T$, then outputs a bit $b'$, and wins the game if $b' = b$.

We define the adversary’s advantage of identifying the tag in the above game as $\text{ADV}_{d}^{\text{auth}} = |\Pr[b' = b] - 1/2|$.

**Definition 2:** An RFID system $S$ obtains universal privacy if $\text{ADV}_{d}^{\text{auth}}$ is negligible in the security parameter for any polynomial-time adversary $A$.

Next we define forward privacy. Forward privacy is modeled by the following game between an RFID system $C$ and a polynomial-time adversary $A$.

**Game of Forward Privacy:**

1. $C$ chooses two tags, $T_0$ and $T_1$, and a valid reader $R$.
2. $A$ makes queries on $T_0$, $T_1$, and $R$ by using oracles Query, Send, Execute, Block for a number of times of its choice.
3. After $A$ stops querying, $C$ carries out an instance of the protocol with $T_0$ and $T_1$, which is achieved during mutual authentication of both tags with $R$.
4. $C$ selects a random bit $b$, sets $T = T_b$, and sends $T$ to $A$.
5. $A$ calls the oracle Reveal on $T$.
6. $A$ decides which of $T_0$ or $T_1$ is $T$, then outputs a bit $b'$, and wins the game if $b' = b$.

We define the adversary’s advantage of identifying the tag in the above game as $\text{ADV}_{d}^{\text{for}} = |\Pr[b' = b] - 1/2|$.

**Definition 3:** An RFID system $S$ obtains forward privacy if $\text{ADV}_{d}^{\text{for}}$ is negligible in the security parameter for any polynomial-time adversary $A$.

Finally, we define C-existential privacy. C-existential privacy is modeled by the following game between an RFID system $C$ and a polynomial-time adversary $A$.

**Game of C-Existential Privacy:**

1. $C$ chooses two tags, $T_0$ and $T_1$.
2. $A$ makes queries on $T_0$ and $T_1$ by using the oracle Query for at most $C$-1 number of times for each tag, where $C$ is a pre-specified parameter.
3. After $A$ stops querying, $C$ selects a random bit $b$, sets $T = T_b$, and sends $T$ to $A$.
4. $A$ makes a query of $T$ by using the Query oracle.
5. $A$ decides which of $T_0$ or $T_1$ is $T$, then outputs a bit $b'$, and wins the game if $b' = b$.

We define the adversary’s advantage of identifying the tag in the above game as $\text{ADV}_{d}^{\text{exi}} = |\Pr[b' = b] - 1/2|$.

**Definition 4:** An RFID system $S$ obtains C-existential privacy if $\text{ADV}_{d}^{\text{exi}}$ is negligible in the security parameter for any polynomial-time adversary $A$. 
3. THE CONSTANT-TIME-IDENTIFICATION PROTOCOL OF ALOMAIR, CLARK, CUELLAR, AND POOVENDRAN

In [5], Alomair et al. propose a privacy-preserving protocol with constant-time identification. As in [2], we denote their protocol as CTI. The protocol CTI is described as follows.

The Protocol CTI:

Setup: Each tag, each reader, and the database have a random number generator and share a hash function $h$. There are $N_T$ tags and a set of $N$ pseudonyms $\mathcal{P} = \{\psi_1, \ldots, \psi_N\}$ where $N = 2N_T$. Each tag has a pseudonym $\psi$, a key $k$, and a counter whose value is less than $C$. The back-end database is preloaded with a table $H$ which records all $NC$ values $h(\psi, c)$ for $\psi \in \mathcal{P}$ and $c \in \{0, 1, \ldots, C-1\}$. The slot with the value $h(\psi, c)$ has a pointer which points to the tag with the pseudonym $\psi$.

Identification Process: The identification process goes as follows.

1. The reader requests the tag and sends a random string $r$ to it.
2. The tag uses its secret key $k$, pseudonym $\psi$, and counter value $c$ to compute $h(\psi, c)$ and $r' = h(0, \psi, c, k, r)$. Next the tag increments $c$ as $c+1$ and sends the two hash values to the reader which passes them to the database.
3. By the value $h(\psi, c)$, the reader can access the database to identify the tag labeled by the pseudonym $\psi$ within constant time and can obtain the information about the tag including its pseudonym $\psi$, its secret key $k$, and a new pseudonym $\psi'$ which is randomly selected from $\mathcal{P}$ and used to update the tag. The reader responds the tag with $h(1, \psi, k, r') \oplus \psi'$ and $h(2, \psi', k, r')$.
4. With $h(1, \psi, k, r') \oplus \psi'$, the tag can obtain the new pseudonym $\psi'$. With $h(2, \psi', k, r')$, the tag authenticates the reader and checks the integrity of the new pseudonym $\psi'$.
5. After authenticating the reader, the tag sets its counter to zero, updates its secret key as $k' = h(k)$, and changes the pseudonym as $\psi'$.

3.1 Privacy, Security, and Performance Analysis of CTI

The security and privacy of CTI is guaranteed in [5].

Lemma 1: [5] The protocol CTI has the following properties: 
1. CTI is a secure mutual authentication protocol based on the random oracle model.
2. CTI obtains universal privacy.
3. CTI obtains forward privacy.
4. CTI obtains C-existential privacy.

To show that CTI is against compromised attacks, we assume that an adversary can compromise a tag in the system. Let $q$ be the number of protocol runs an adversary has performed with the system using compromised tags. In [5], the authors observe that the advantage of distinguishing between two tags is at most the fraction of pseudonyms col-
lected by the adversary. In addition, they show that the expected number of different pseudonyms collected by the adversary is $N(1-(1-1/N)^q)$. From this, they show that the advantage to perform a successful compromised attack is at most $1-(1-1/N)^2q$. Note that if $N$ is larger then CTI obtains higher privacy against compromised attacks. As a concrete example, if $N=2\cdot10^6$ and $q=10^6$, then the probability of distinguishing between two tags is at most 0.001.

Recently, in [2], the authors refine the analysis of [5] and propose a compromised attack against CTI which uses the information of the tag counter value to distinguish two given tags. In their setting, if $N=2\cdot10^6$, $C=10^3$, and $q=10^3$, then the probability to track a given tag is at least 0.1% which cannot be underestimated. One way to prevent this attack is to lock the information of the tag counter by using a hash function and the tag’s private key. We will detail this approach latter.

Memory complexity of CTI is at least $NC\log_2(NC)$ since the database stores all hash values $h(\psi, c)$ for $1 \leq i \leq N$ and $0 \leq c \leq C-1$. The tag identification is carried out by executing a special hash table lookup. For tag identification, CTI only uses three times of table lookups. For detailed description, we refer the reader to [5].

4. THE SPACE-EFFICIENT SCHEME VIA TIME-MEMORY TRADE-OFF

In this section, we propose a memory-saving version of the constant-time identification protocol of Alomair et al. [5]. By previous discussion, the larger the maximum counter value $C$ is the more privacy the protocol CTI obtains. However, the storage demand is too large to be practical. To solve this problem, we suggest using a time-memory-trade-off approach to reduce the memory complexity. Methods of time-memory trade-off can reduce the space complexity $M$ which is needed to invert any given value in a set of $2^k$ outputs of a one-way function $F: \{0, 1\}^k \rightarrow \{0, 1\}^s$ with help of more computing time $T$. Generally, the time-memory-trade-off approach to invert the given function $F: \{0, 1\}^s \rightarrow \{0, 1\}^t$ for $s < t$ works as follows. First, $t$ reduction functions $R_i: \{0, 1\}^s \rightarrow \{0, 1\}^s$ are defined. Given an output value $C \in \{0, 1\}^t$, the goal is to search for an input value $z \in \{0, 1\}^s$ such that $F(z) = C$. A general time-memory trade-off requires an off-line preprocessing which tries to go through most input values of the function $F$ by repeatedly applying $F$ and these reduction functions $R_i: \{0, 1\}^s \rightarrow \{0, 1\}^s$. By repeatedly applying the function $F$ and the above reduction functions $R_i$, one can obtain the following chain.

$z_0 \xrightarrow{r} C_1 \xrightarrow{C} z_2 \xrightarrow{r} z_3 \xrightarrow{C} z_4 \xrightarrow{C} \ldots \xrightarrow{C} z_r$.

Let us define the function $f_i: \{0, 1\}^k \rightarrow \{0, 1\}^k$ a by letting $f_i(z) = R_i(F(z))$ for $1 \leq i \leq t$. This leads to the following chain of keys:

$z_0 \xrightarrow{\beta} z_1 \xrightarrow{\beta} z_2 \xrightarrow{\beta} \ldots \xrightarrow{\beta} z_r$.

The preprocessing task is completed as follows.

Preprocessing Phase: First of all, the method of time-memory trade-off chooses $mt$ starting points $\{SP_i; i \in \{1, \ldots, mt\}\}$ randomly from the key space. Then, for each start-
ing point $SP_i = X_{i,0}$ where $i \in \{1, \ldots, mt\}$, it computes $X_{i,j}$ by iteratively applying the function $f_i$ and only stores the starting point $SP_i = X_{i,0}$ and the value $X_{i,t}$ as the corresponding end point $EP_i$. Thus, after doing that, we obtain the following virtual table. In addition, the method sorts the stored values according to the end points. Clearly, only $2mt$ memory slots are required since the method only stores the starting and the end points ($SP_i = X_{i,0}, EP_i = X_{i,t}$). The resulting table is the output of the off-line preprocessing phase. Now let us take a look at the online searching phase.

Online Searching Phase: Assume that the database receives the function value $C = F(z)$. The goal of the database is to find an input $z'$ such that $F(z') = C$. The database takes the following procedure:

1. It applies the reduction function $R_i$ to get $Y_0 = R(C)$ and see whether $Y_0 = EP_i$ for some $i$. If $Y_0 = EP_i$, then he computes $X_{i,t-1}$ from the starting point $X_{i,0}$ by sequentially applying function $f_j$ from $j = 1$ to $t-1$. Either $F(X_{i,t-1}) = C$ or $F(X_{i,t-1}) \neq C$. For the first case, he outputs $X_{i,t-1}$. Again, we refer to the latter case as a false alarm.

2. For the $j$th iteration, let $Y_j = (f_{j+1} \circ \ldots \circ f_t)(Y_0)$. If $Y_j = EP_i$ for some $i$, then the database computes $X_{i,j}$ and see whether $F(X_{i,j}) = C$. Once $F(X_{i,j}) = C$, the algorithm stops and outputs $X_{i,j}$. Otherwise $Y_j$ is not one of the end points or a false alarm occurred. In this case, the cryptanalyst proceeds next iteration if $j < t-1$ and output “NONE” if $j = t-1$.

Probability of success of the table for the function $F: \{0,1\}^k \rightarrow \{0,1\}^s$ is defined as follows:

$$PS = 2^{-k} \text{(number of different inputs in the table)}$$

In [19], Oechslin shows that the success probability of the table is

$$PS = 1 - \prod_{i=1}^{t} (1 - m_i/2^s)$$

where $m_0 = mt$ and $m_i = 2^k \text{(}1 - e^{-m_i/2^i} \text{)}$.

To solve the recurrence, we use the following result due to [7].

Lemma 2: [7] $m_j \approx 2mt2^{k}(mtj+2^{k+1})$ for sufficiently large $j$ where $m_0 = mt$ and $2^k = mt^2$.

Now, the fraction of values in $\{0, 1\}^k$ which do not appear in the matrix $M$ is
Thus, if $K = mt^2$ and $t$ is large enough, then we can obtain

$$\prod (1 - m_i / K) \approx \prod (mt(j - 2) + 2K) / (mtj + K) \approx 4/9$$

So, a single matrix can only cover 4/9 fraction of $2^k$ inputs. Thus, we generate $w$ multiple tables. This increases the total success of probability $PS_{\text{total}}$ to

$$PS_{\text{total}} = 1 - (1 - PS)^w \geq 1 - (4/9)^w \geq 1 - \varepsilon.$$

when $w = \log_{9/4}(1/\varepsilon)$. For success probability $1 - \varepsilon$, let $T_\varepsilon$ denote the number of hash operations used in searching in these $w$ tables and $M_\varepsilon$ be the total memory for all these tables. Clearly

$$T_\varepsilon = t(t - 1)\log_{9/4}(1/\varepsilon)/2,$$

and

$$M_\varepsilon = 2mtk\log_{9/4}(1/\varepsilon) = 2k2^t\log_{9/4}(1/\varepsilon)/t.$$

For a specific parameter $\varepsilon=0.001$, we have $w=9$, $T_{0.001}=9(t(t-1))/2$, and $M_{0.001}=(18k2^t)/t$.

### 4.1 Applying time-memory trade-off on Database Setup of CTI

We apply the above time-memory trade-off on the setup of the database of the protocol CTI. We follow a similar way of [8] to adapt the time-memory trade-off to our case. Let $\psi_0, \ldots, \psi_{N-1}$ be an enumeration of the pseudonyms in $\Psi$. We define the function $F: \mathbb{Z}_N \times \mathbb{Z}_C \to \{0,1\}^s$ by $F(i,j)=h(\psi_{ij})$. We also need many reduction functions $R_i: \{0,1\}^s \to \mathbb{Z}_N \times \mathbb{Z}_C$. These functions can be defined as follows. Let $\{\pi^i : i=1, \ldots, s\}$ be a family of permutations on the set $\{1, \ldots, s\}$ and $\{\Pi^i : i=1, \ldots, s\}$ be a family of functions from $\{0,1\}^s$ to $\{0,1\}^s$ defined by

$$\Pi^i(x_1, \ldots, x_s) = (x_{\pi^i(1)}, \ldots, x_{\pi^i(s)})$$

for each $i$. $R_i(x)$ defined by

$$R_i(x) = (\Pi^i(x) \mod N, \lfloor \Pi^i(x)/N \rfloor \mod C)$$

are appropriate reduction functions.

Let $m, t$ be integers such that $NC = mt^2$. To obtain $(1-\varepsilon)$ probability of success, the database precalculates $\log_{9/4}(1/\varepsilon)mt^2t$ tables and stores the start and end points for each tables. For the database, the number of hash operations used in its preprocessing time is $NC\log_{9/4}(1/\varepsilon)$ and it only needs to store at most $2NC \log_{9/4}(1/\varepsilon)(\log_2(NC))/t$ bits. In particular, if $\varepsilon=(4/9)^{30}$, the database precalculates 30 tables by using $30NC$ hash operations and stores at most $60NC(\log_2(NC))/t$ bits. In the case that $\varepsilon=0.001$, the database precalculates 9 tables by using $9NC$ hash operations and stores at most $18NC(\log_2(NC))/t$ bits.
4.2 CTI/TM(ε) Protocol

Now we describe the proposed protocol CTI/TM(ε).

The Protocol CTI/TM(ε):

Setup: The tag setup is the same as the setup of CTI except that the tag can use another hash function g such that g(·, k) is a permutation on {0, 1, ..., C − 1} with high probability by choosing k uniformly due to the hardness of finding collision of hash functions. The setup in the back-end database is implemented via the time-memory-trade-off method with success rate (1 − ε) described in the previous subsection.

Identification Process: The database can identify the tag by the following procedure.
1. The reader requests the tag and sends a random string r to it.
2. The tag uses its secret key k, pseudonym ψ and counter value c to compute ĉ = g(c, k), h(ψ, ĉ), and r′ = h(0, ψ, ĉ, k, r). Next the tag increments c as c + 1 and sends the two hash values to the reader which passes them to the database.
3. After receiving r′ = h(ψ, ĉ), the database uses the time-memory-trade-off method to search the corresponding pair (ψ′, c′), retrieves the secret k, and checks if r′ = h(0, ψ′, c′, k, r).
4. The remaining processing is the same as the procedure of CTI.

4.3 Security, Privacy and Performance of CTI/TM(ε)

CTI/TM(ε) inherits from the security and the privacy proofs of the protocol CTI shown in [5] since CTI/TM(ε) does not modify the information exchanged or the internal content of the tag. In fact, the protocol view of CTI/TM(ε) is the same as the one of CTI and CTI/TM(ε) only changes the way of key management in the back-end database. As a result, protocol CTI/TM(ε) obtains the same security and privacy level as CTI. So we have the following theorems.

Theorem 1: The protocol CTI/TM(ε) is a secure mutual authentication protocol based on the random oracle model.

Theorem 2: The protocol CTI/TM(ε) obtains universal privacy, forward privacy, and C-existential privacy.

In CTI/TM(ε), the advantage to perform a successful compromised attack is also at most 1−(1−1/N)^2. Note that the compromised attack against CTI proposed in [2] cannot be applied since the proposed attack relies on the information of the tag counter and such information is now locked by using a hash function g and the tag’s private key.

In the protocol CTI/TM(ε), we assume that the running time of one hash operation is η seconds and the upper bound of tag’s response time is φ seconds. The memory complexity of CTI/TM(ε) is 2NC(log log 4(1/ε))(log(NC))/t and the identification time is η(log log 4(1/ε))(t−1)/2 seconds which is bounded by φ seconds. We set t as the maximum integer less than or equal to \( \lceil \varphi(\eta \log_{10}(1/\epsilon)) \rceil^{1/2} \) in order to obtain the minimum
memory cost in the protocol CTI/TM(\(\varepsilon\)).

Let us assume that \(\varepsilon = 0.001\). In the previous discussion, to guarantee 99.9% probability of success of searching, the memory complexity of CTI/TM(0.001) is \(18NC\log_2(NC)/t\) and the identification time is \(9(t-1)\eta/2\) seconds. Compared to CTI, the memory complexity of CTI is \(NC\log_2(NC)\) and the identification time is \(O(1)\) seconds. We conclude that if the tag identification time is allowed to be \(\eta\) (that is \(9(t-1)\eta/2 \leq \phi\)), then the memory complexity of CTI/TM can be \(18/t\) times less than the storage of CTI. We give a comparison between the randomized hash-lock protocol (denoted by RHL), CTI, and CTI/TM(\(\varepsilon\)) in Table 1.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Identification time (number of hash operations)</th>
<th>Database memory (bits)</th>
<th>Success Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHL [21]</td>
<td>(O(N))</td>
<td>(O(N\log_2 N))</td>
<td>1</td>
</tr>
<tr>
<td>CTI [5]</td>
<td>(O(1))</td>
<td>(NC\log_2(NC))</td>
<td>1</td>
</tr>
<tr>
<td>CTI/TM((\varepsilon)) This paper</td>
<td>((\log_{9/4}(1/\varepsilon))(t-1)/2)</td>
<td>(2(\log_{9/4}(1/\varepsilon))NC\log_2(NC)/t)</td>
<td>(1-\varepsilon)</td>
</tr>
<tr>
<td>CTI/TM(0.001) This paper</td>
<td>(9(t-1)/2)</td>
<td>(18NC\log_2(NC)/t)</td>
<td>0.999</td>
</tr>
<tr>
<td>CTI/TM((4/9)^{30}) This paper</td>
<td>(15(t-1))</td>
<td>(60NC\log_2(NC)/t)</td>
<td>(1-2^{-35})</td>
</tr>
</tbody>
</table>

5. CASE STUDY

As in the previous discussion, the memory complexity of CTI/TM(0.001) is reduced by a factor of \(t/18\) compared to that of CTI. We give some practical cases in this section. We have the following assumptions on the database.

- The database takes \(2^{-25}\) seconds to perform a hash operation. So \(\eta = 2^{-25}\).
- The database spends at the most \(2^{-8}\) seconds to identify one tag. So \(\phi = 2^{-8}\).

The first assumption is based on the approximation by a dual-core processor operating at 1.83 GHz released in 2007 and we refer the reader to [15] (Section 7). The second assumption is due to the fact that a passive EPC tag responds to request of a reader in the order of 4 milliseconds [5, 22].

From the above assumptions, when identifying one tag, the number of hash operations performed by a single computer is \(2^{17}\). Thus, in CTI/TM(0.001), the number of hash operations used for identifying one tag is required that \(T_{0.001} = 9(t-1)/2 \leq 2^{17}\). So if \(t = 2^{9}/3\) then this requirement can be fulfilled. As a result, the memory complexity of CTI/TM(0.001) is \(27/256\) times less memory space than that of CTI. So, CTI/TM(0.001) saves at least 89.5% storage space.

We give some concrete examples for comparison. The number of tags \(N_T\) is assumed to be \(10^9\) and the number of pseudonyms \(N\) is \(2\cdot10^9\). Suppose the maximum counter val-

---

Table 1. Comparison between the randomized hash-lock protocol (RHL), CTI, and CTI/TM(\(\varepsilon\)) where the identification time is measured by the number of hash operations used for one tag identification.
ue $C=10^6$. In this case, CTI has space complexity approximately 12.7 Petabytes while CTI/TM(0.001) has space complexity about 1.34 Petabytes.

Next, we consider the protocol CTI/TM(\(\varepsilon\)) for \(\varepsilon=(4/9)^{30}\). In this protocol, \(T_\varepsilon=30t(\varepsilon-1)/2 \leq 2^{17}\). So if \(t=2^{7}(30)^{1/2}\) then this requirement can be fulfilled. In this case, the memory complexity of CTI/TM(\(\varepsilon\)) is 60 \((30)^{1/2}/512\) times less memory space than that of CTI. So, CTI/TM(\((4/9)^{30}\)) saves at least 35.8% storage space compared to the protocol CTI. We summarize the above discussion in Table 2. Moreover, we also show the memory complexities of CTI/TM(0.001) and CTI/TM((4/9)^{30}) in Table 2 when the identification time is allowed to be \(2^8\) and 1 second, respectively. In Table 2, we also show that the identification time of the randomized hash-lock protocol (RHL) is very large in the same setting.

Table 2. Practical comparison between the randomized hash-lock protocol (RHL), CTI, CTI/TM(0.001), and CTI/TM((4/9)^{30}) under the assumption that \(N_T=10^9\), \(C=10^6\), \(\eta=2^{-25}\), and \(\varphi\in\{2^{-8},2^{-4},1\}\).

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Identification time (seconds)</th>
<th>Database memory</th>
<th>Success Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHL [21]</td>
<td>29.8</td>
<td>3.74 GB</td>
<td>1</td>
</tr>
<tr>
<td>CTI [5]</td>
<td>(2^{-24})</td>
<td>12.7 PB</td>
<td>1</td>
</tr>
<tr>
<td>CTI/TM(0.001)</td>
<td>(2^{-8})</td>
<td>1.34 PB</td>
<td>0.999</td>
</tr>
<tr>
<td>CTI/TM(0.001)</td>
<td>(2^{-1})</td>
<td>335 TB</td>
<td>0.999</td>
</tr>
<tr>
<td>CTI/TM(0.001)</td>
<td>1</td>
<td>83.75 TB</td>
<td>0.999</td>
</tr>
<tr>
<td>CTI/TM((4/9)^{30})</td>
<td>(2^{-8})</td>
<td>8.15 PB</td>
<td>1–2^{-35}</td>
</tr>
<tr>
<td>CTI/TM((4/9)^{30})</td>
<td>(2^{-4})</td>
<td>2.04 PB</td>
<td>1–2^{-35}</td>
</tr>
<tr>
<td>CTI/TM((4/9)^{30})</td>
<td>1</td>
<td>509.4 TB</td>
<td>1–2^{-35}</td>
</tr>
</tbody>
</table>

On the other hand, if the number of pseudonyms and the memory space of CTI/TM(0.001) and CTI are all equal, then CTI/TM(0.001) obtains approximately 10C-existential privacy while CTI obtains C-existential privacy. Let us see the following example. Typically, the response time of EPC tags is about 4ms [22]. As explained in [5], when \(C=10^6\), the adversary must interrogate the tag for about 1.11 consecutive hours in order to find the correlation of its responses. If the number of pseudonyms and the storages of CTI/TM(0.001) and CTI are 2\(\cdot10^9\) and 13 Petabytes respectively, then the maximum counter value in CTI/TM(0.001) can be \(10^5\). Thus, the adversary must spend about 11.1 consecutive hours to trace a tag.

Based on the above discussion, we conclude that CTI/TM(0.001) is more practical than CTI in the same level of security and privacy by the above case study. Moreover, in the same memory requirement, CTI/TM(0.001) obtains higher privacy than CTI.

6. CONCLUSIONS

In this paper, we modify the hash-based RFID identification protocol of Alomair, Clark, Cuellar and Poovendran [5] which is called constant-time-identification protocol (CTI) to obtain a new protocol which is called CTI/TM(\(\varepsilon\)). CTI/TM(\(\varepsilon\)) is designed by using the well-known time-memory-trade-off technique. We use the time-memory-trade-off technique to improve the storage requirement of the protocol CTI while keeping the
same level of privacy and the number of available pseudonyms. After a performance analysis, we show that the storage of $\text{CTI/TM}(\epsilon)$ is about $O(1/t)$ of that of $\text{CTI}$ where the number of hash operations for one tag’s identification is $O(t^2)$ and the hidden constant is dependent on the parameter $\epsilon$. In a practical parameter setting, we give an example in which the memory space of $\text{CTI/TM}(0.001)$ is about 11% of that of $\text{CTI}$. According to this, we conclude that the protocol $\text{CTI/TM}(0.001)$ is more practical than the protocol $\text{CTI}$ while preserving the same level of privacy.

REFERENCES


Jen-Chun Chang (張仁俊) received the B.S. and M.S. degrees in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1989 and 1991, respectively. He received his Ph.D. degree in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2000. Now he is a Professor of the Department of Computer Science and Information Engineering in National Taipei University, Taipei, Taiwan. His research interests include cryptography, coding theory, reliability theory, and algorithms.

Hsin-Lung Wu (吳信龍) received his Ph.D. degree in Computer Science and Information Engineering from National Chiao Tung University, Taiwan in 2008. He is currently an Assistant Professor in the Department of Computer Science and Information Engineering at National Taipei University, Taiwan. His main research interests include design and analysis of algorithms, computational complexity, and information security.