Vulnerability of an RFID Authentication Protocol Proposed in at SecUbiq 2005

Daesung Kwon, Daewan Han, Jooyoung Lee and Yongjin Yeom

National Security Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, Korea
{ds_kwon,dwh,jlee05,yjyeom}@etri.re.kr

Abstract. In this paper, we analyze the security of the RFID authentication protocol proposed by Choi *et al.* at SecUbiq 2005. They claimed that their protocol is secure against all possible threats considered in RFID systems. However, we show that the protocol is vulnerable to an impersonation attack. Moreover, an attacker is able to trace a tag by querying it twice, given the initial information from $2^{\lceil \log_2(\ell+1) \rceil} + 1 (\approx \ell + 2)$ consecutive sessions and $2 \cdot 2^{\lceil \log_2(\ell+1) \rceil} (\approx 2(\ell + 1))$ consecutive queries, where ℓ is the length of secret values (in binary).

1 Introduction

Recently a Radio-Frequency IDentification(RFID) system attracts much attention in various industries. Regarded as one of the leading technologies realizing so-called ubiquitous(or pervasive) computing societies, RFID is replacing the current bar-code system.

An RFID system consists of three components; tag, reader and back-end database(BED). Since the communications between a tag and a reader are executed on public RF channels, the system faces many problems in security and privacy. However, traditional cryptographic techniques are not suitable for RFID systems since tags are much constrained in terms of computational power and memory. The constraints make it a challenging problem to design RFID protocols which are both *secure* and *efficient*. We refer to [11] for security and constraint issues in RFID systems.

Initially, physical protections such as blocker tags[7], active jamming and Faraday cages are suggested. Since those methods have limits for broad usage, cryptographic solutions have been studied. Certain cryptographic or arithmetic primitives are proposed to use for RFID protocols, while the majority of them are still based on secure hash functions[2, 4, 5, 8–10]. Refer to [1, 6] for detailed surveys of this approach.

In SecUbiq 2005 Choi *et al.* proposed another hash-based RFID authentication protocol[3], named OHLAP protocol. OHLAP seems to be

an improved version of the protocol suggested in [8]. The authors emphasize that OHLAP is more suitable to ubiquitous computing environment than the protocol in [8] since OHLAP uses static IDs. They also claim that OHLAP is more efficient than those in [10] and [8] in terms of tag's computation cost and security against various attacks.

However, in this paper, we show that OHLAP protocol is not secure practically. First, we show that an attacker can easily impersonate a tag by eavesdropping only one session. We also show that one can recover two candidates for `ID` by eavesdropping consecutive $2^{\lceil \log_2(\ell+1) \rceil}+1$ sessions and by sending consecutive queries less than or equal to $2 \cdot 2^{\lceil \log_2(\ell+1) \rceil}$, where ℓ is the length of secret values. (Two phases - eavesdropping and querying - are not necessarily consecutive.) Once a tag's `ID` is revealed, then any other tag's `ID` in the same group can be easily computed just by eavesdropping a session and sending two consecutive queries. In particular, if a tag's `ID` is known, then one can trace a tag by two consecutive queries.

The paper is organized as follows. In Section 2, we briefly describe OHLAP protocol. We discuss the vulnerability of OHLAP in Section 3. In Appendix, we present the proofs for the lemmas used in Section 3.

2 OHLAP Protocol

2.1 System Set-Up

In the set-up phase, a tag and a back-end database(BED) store certain secret values. Data fields of a tag and a reader are initialized to the following values:

1. BED: First, BED divides identities of tags into several groups. Each group is associated with a group index `GI`. Data fields of a BED are initialized to ℓ-bit strings `GI`, `ID`, `K`, `S` and DATA where `GI` is a group index of tags, `ID` is a string used for identifying, `K` is a secret value which is stored in all tags, `S` is a tag's secret value and DATA is a storage of an accessible information about each tag. BED needs a one-way hash function $\mathtt{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ to execute hash operations.
2. Reader: A reader picks uniformly random value $\mathtt{r} \in \{0,1\}^\ell$. A reader does not need to execute any operation. It merely forwards a tag(BED)'s message to a BED(tag).
3. Tag: A tag stores its own `ID`, `GI`, `K`, `S` and a counter `c`. The counter `c` is initialized by an arbitrary value, which is ℓ-bit string. Whenever a tag receives a query from a nearby reader, the tag increase `c` by 1. To execute hash operations, the tag needs a one-way hash function `H`.

2.2 Authentication Process

When a reader queries to a tag, the tag and the reader authenticate each other as shown in Fig. 1.

Step 1. A reader picks a random value $\mathtt{r}$, and sends Query and $\mathtt{r}$ to a tag.

Step 2. If $\mathtt{r}$ is all zero, the tag sends "stop" message to the reader and halts the protocol. Otherwise, it performs processes as follows:

1. The tag computes $\mathtt{A}^1 = \mathtt{K} \oplus \mathtt{c}, \mathtt{A}^2 = \mathtt{ID} + (\mathtt{GI}_i \oplus \mathtt{r} \oplus \mathtt{c}) \mod (2^l - 1)$. using $\mathtt{r}$,$\mathtt{c}$ and its own $\mathtt{ID}$,$\mathtt{GI}_i$ and $\mathtt{K}$.
2. Also, the tag computes $\mathtt{B} = \mathtt{H}(\mathtt{ID} \| (\mathtt{S} \oplus \mathtt{GI}_i) \| (\mathtt{r} \oplus \mathtt{c}))$, and sends $\mathtt{A}^1, \mathtt{A}^2$ and $\mathtt{B}_\mathtt{R}$ to the reader, where $\mathtt{B}_\mathtt{R}$ is a right half of $\mathtt{B}$.
3. Then the tag increase the counter $\mathtt{c}$ which should not exceed $2^\ell - 1$. If the counter $\mathtt{c}$ exceeds $2^\ell - 1$, it is initialized by initial $\mathtt{c}$.

Step 3. Upon receiving $\mathtt{A}^1, \mathtt{A}^2$ and $\mathtt{B}_\mathtt{R}$ from the tag,

1. The reader forwards $\mathtt{A}^1, \mathtt{A}^2, \mathtt{B}_\mathtt{R}$ and $\mathtt{r}$ to the BED.
2. The BED computes $\mathtt{c}' = \mathtt{A}^1 \oplus \mathtt{K}$ and $\mathtt{ID}'_j = \mathtt{A}^2 - (\mathtt{GI}_j \oplus \mathtt{r} \oplus \mathtt{c}') \mod (2^\ell - 1)$ using all group indices $\mathtt{GI}_j$.
3. The BED checks if one of computed $(\mathtt{ID}'_j, \mathtt{GI}_j)$ is matching to one of the stored $(\mathtt{ID}, \mathtt{GI})$s. If this succeeds, the BED computes $\mathtt{H}(\mathtt{ID} \| (\mathtt{S} \oplus \mathtt{GI}_i) \| (\mathtt{r} \oplus \mathtt{c}))$. Otherwise, the BED halts this process.
4. Then, the BED authenticates the tag by checking if the right half of $\mathtt{H}(\mathtt{ID} \| (\mathtt{S} \oplus \mathtt{GI}_i) \| (\mathtt{r} \oplus \mathtt{c}))$ is equal to the received value $\mathtt{B}_\mathtt{R}$.
5. The BED sends $\mathtt{B}_\mathtt{L}$ to the reader, where $\mathtt{B}_\mathtt{L}$ is a left half of $\mathtt{B}$. The reader forwards $\mathtt{B}_\mathtt{L}$ to the tag.

Step 4. The tag authenticates the reader by checking if the received value $\mathtt{B}_\mathtt{L}$ is equal to the left half of $\mathtt{B}$ of step 2.

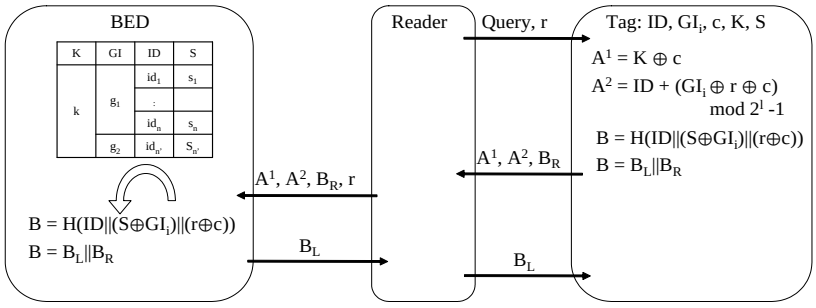

Fig. 1. Authentication process in OHLAP Protocol.

3 Vulnerability of OHLAP Protocol

We now discuss the vulnerability of the OHLAP protocol. For convenience, we introduce the following notations for ℓ-bit number $\mathtt{A}$.

$\mathtt{A}_i$	$(i+1)$-th bit of $\mathtt{A} = \mathtt{A}_{\ell-1}\|\mathtt{A}_{\ell-2}\|\cdots\|\mathtt{A}_1\|\mathtt{A}_0$.
$\mathtt{A}_{i_1,i_2,\cdots,i_k}$	$\mathtt{A}_{i_1}\|\mathtt{A}_{i_2}\|\cdots\|\mathtt{A}_{i_k}$
$\mathtt{A}_{\geq i_0}$ $(\mathtt{A}_{\leq i_0})$	$\mathtt{A}_{\ell-1}\|\cdots\|\mathtt{A}_{i_0}$ $(\mathtt{A}_{i_0}\|\cdots\|\mathtt{A}_0)$

3.1 Impersonation

In this subsection, we give a simple attack of spoofing a reader. By eavesdropping a session between a tag and a reader, an attacker can generate valid responses of the tag for any query from a reader.

Let $\mathtt{ID}, \mathtt{GI}, \mathtt{S}$ be the secret information of a tag. By eavesdropping a session, an attacker obtains the following:

- $\mathtt{r}$: query from a reader.
- $\mathtt{A}^1(=\mathtt{K}\oplus\mathtt{c}), \mathtt{A}^2(=\mathtt{ID}+(\mathtt{GI}\oplus\mathtt{r}\oplus\mathtt{c}) \mod 2^\ell-1), \mathtt{B}_R$: response from a tag where $\mathtt{K}$ is a common secret key in valid tags, $\mathtt{c}$ is a counter and $\mathtt{B}_R$ is a right half of $\mathtt{H}(\mathtt{ID}\|(\mathtt{S}\oplus\mathtt{GI})\|(\mathtt{r}\oplus\mathtt{c}))$.
- $\mathtt{B}_L$: response from a reader which is a left half of $\mathtt{H}(\mathtt{ID}\|(\mathtt{S}\oplus\mathtt{GI})\|(\mathtt{r}\oplus\mathtt{c}))$.

Now the attacker can generate a valid response $(\tilde{\mathtt{A}}^1, \tilde{\mathtt{A}}^2, \tilde{\mathtt{B}}_R)$ of the tag to a random query $\tilde{\mathtt{r}}$ from a reader as follows:

$$\tilde{\mathtt{A}}^1 = \mathtt{A}^1 \oplus \mathtt{r} \oplus \tilde{\mathtt{r}}, \quad \tilde{\mathtt{A}}^2 = \mathtt{A}^2, \quad \tilde{\mathtt{B}}_R = \mathtt{B}_R$$

After receiving $(\tilde{\mathtt{A}}^1, \tilde{\mathtt{A}}^2, \tilde{\mathtt{B}}_R)$, BED will validate the information as follows:

1. From $\tilde{\mathtt{A}}^1$, BED obtain the counter $\mathtt{c}'$ by xoring K:
$$\begin{aligned}\mathtt{c}' &= \mathtt{K}\oplus\tilde{\mathtt{A}}^1 = \mathtt{K}\oplus\mathtt{A}^1\oplus\mathtt{r}\oplus\tilde{\mathtt{r}}\\ &= \mathtt{c}\oplus\mathtt{r}\oplus\tilde{\mathtt{r}}\end{aligned}$$
2. BED recovers the same $\mathtt{ID}$ and B since
$$\begin{aligned}\mathtt{ID}' &= \tilde{\mathtt{A}}^2 - (\mathtt{GI}\oplus\tilde{\mathtt{r}}\oplus\mathtt{c}') = \mathtt{A}^2 - (\mathtt{GI}\oplus\tilde{\mathtt{r}}\oplus\mathtt{c}\oplus\mathtt{r}\oplus\tilde{\mathtt{r}}) = \mathtt{A}^2 - (\mathtt{GI}\oplus\mathtt{r}\oplus\mathtt{c})\\ &= \mathtt{ID}\\ \tilde{\mathtt{B}}' &= \mathtt{H}(\mathtt{ID}\|(\mathtt{S}\oplus\mathtt{GI})\|(\tilde{\mathtt{r}}\oplus\mathtt{c}')) = \mathtt{H}(\mathtt{ID}\|(\mathtt{S}\oplus\mathtt{GI})\|(\mathtt{r}\oplus\mathtt{c}))\\ &= \mathtt{B}(=\tilde{\mathtt{B}}),\end{aligned}$$
BED would accept $\mathtt{B}_R$ as a valid value.

Therefore, BED would regard the values $\tilde{\mathtt{A}}^1, \tilde{\mathtt{A}}^2, \tilde{\mathtt{B}}_R$ from an attacker as generated by the valid tag.

3.2 Recovering a tag's ID

In this subsection, we present the method of recovering two candidates for a tag's ID by eavesdropping consecutive $2^{\lceil \log_2(\ell+1) \rceil} + 1$ sessions and by sending the tag consecutive (well-chosen) queries less than or equal to $2 \cdot 2^{\lceil \log_2(\ell+1) \rceil} + 1$. This attack is based on the following three lemmas. The first lemma shows that if we know $(\mathtt{A} \oplus \mathtt{B})$ for 2^n consecutive B, we get the less significant n bits of unknown A and B.

Lemma 1. *Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be ℓ bit unknown values. If we know*

$$\boldsymbol{x} \oplus \boldsymbol{y}, \quad \boldsymbol{x} \oplus (\boldsymbol{y} + 2^i),$$

for $i = 0, \cdots, n-1$ where $n < \ell - 1$, then we can recover less significant n bits of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Proof. By bitwise xoring $\mathtt{x} \oplus \mathtt{y}$ and $\mathtt{x} \oplus (\mathtt{y} + 2^i)$, we get $\mathtt{z}(i) = \mathtt{y} \oplus (\mathtt{y} + 2^i)$ for $i = 0, \cdots, n-1$. Then, $\mathtt{z}(i)_{i+1}$ is given by

$$\mathtt{z}(i)_{i+1} = \mathtt{y}_{i+1} \oplus (\mathtt{y} + 2^i)_{i+1} = (\mathtt{y}_{i+1}) \oplus (\mathtt{y}_{i+1} \oplus \mathtt{y}_i) = \mathtt{y}_i$$

for $i = 0, \cdots, n-1$. Therefore, we can recover $\mathtt{y}_{\leq n-1}$ and $\mathtt{x}_{\leq n-1}$.

This lemma will be used to learn the less significant n bits of K, c from $\mathtt{A}^1$s in 2^n consecutive sessions. The following two lemmas play a key role in recovering ID from $\mathtt{A}^2$s.

Lemma 2. *Given*

$$\boldsymbol{x} + \boldsymbol{y} \mod 2^\ell - 1, \quad \boldsymbol{x} + (\boldsymbol{y} \oplus 1) \mod 2^\ell - 1$$

for $\ell(> 2)$-bit unknown values $\boldsymbol{x}, \boldsymbol{y}$, we can obtain two candidates for $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, a pair of LSB's of $(\boldsymbol{x}, \boldsymbol{y})$.

On the contrary to LSB of x, $\mathtt{x}_i (i > 0)$ are determined uniquely by $(\mathtt{x}_0, \mathtt{y}_0)$ and $\mathtt{x}_{\leq i-1}$.

Lemma 3. *Given*

$$\boldsymbol{x} + \boldsymbol{y} \mod 2^\ell - 1, \quad \boldsymbol{x} + (\boldsymbol{y} \oplus 2^i) \mod 2^\ell - 1$$

for $\ell(> 2)$-bit unknown values $\boldsymbol{x}, \boldsymbol{y}$ such that $\boldsymbol{x}_{\leq i-1} = 0$ $(0 < i < \ell - 1)$ and $\boldsymbol{y}_0 = 0$, we can obtain $\boldsymbol{x}_i$.

The proofs of Lemma 2 and Lemma 3 are given in Appendix.

By combining Lemma 2 and Lemma 3, we get the following theorem which is directly applicable to recovering tag's $\mathtt{ID}$.

Theorem 1. *Given*

$$x + y \mod 2^\ell - 1, \quad x + \left(y \oplus (\oplus_{j=0}^{i} 2^j)\right) \mod 2^\ell - 1, \quad i = 0, ..., \ell - 1$$

for $\ell(> 2)$-bit unknown values x, y, we can obtain two candidates for x.

Proof. By Lemma 2, we can recover two candidates for $(\mathtt{x}_0, \mathtt{y}_0)$. Inductively, we assume that two candidates for $(\mathtt{x}_{\leq i-1}, \mathtt{y}_0)$. Let

$$\mathtt{x}' = \mathtt{x} - \mathtt{x}_{\leq i-1}, \quad \mathtt{y}' = \left(\mathtt{y} \oplus (\oplus_{j=1}^{i-1} 2^i)\right) - \mathtt{y}_0.$$

Since $\mathtt{x}'_{\leq i-1} = 0, \mathtt{y}'_0 = 0$ and

$$\begin{aligned} \mathtt{x}' + \mathtt{y}' &= \mathtt{x} + \left(\mathtt{y} \oplus (\oplus_{j=0}^{i-1} 2^i)\right) - \mathtt{x}_{\leq i-1} - (\mathtt{y}_0 \oplus 1) \mod 2^\ell - 1 \\ \mathtt{x}' + (\mathtt{y}' \oplus 2^i) &= \mathtt{x} + \left(\mathtt{y} \oplus (\oplus_{j=0}^{i} 2^i)\right) - \mathtt{x}_{\leq i-1} - (\mathtt{y}_0 \oplus 1) \mod 2^\ell - 1 \end{aligned}$$

are known, $\mathtt{x}', \mathtt{y}'$ satisfy the condition of Lemma 3. Hence we get $\mathtt{x}'_i$ which is equal to $\mathtt{x}_i$ for each candidates. This finishes the proof.

Now, we describe the algorithm to find two candidates for the $\mathtt{ID}$ of a tag. Let $n = \lceil \log_2 (\ell + 1) \rceil$. We denote $\mathtt{c}_0$ the initial counter and $\mathtt{c}_p, \mathtt{c}_{p'}$ the counters of the tag before Step 1, Step 2 respectively. We assume that $\mathtt{c}_p, \mathtt{c}'_p$ are less than $2^\ell - 2^n - 2$ and $2^\ell - 2^{n+1} - 2$ respectively. Since the probability that $\mathtt{c}_p, \mathtt{c}'_p$ don't satisfy the assumption is very low, we will not consider those cases.

Step 1. Recover less significant n bits of $\mathtt{K}$ and $\mathtt{c}$.

- By eavesdropping consecutive $(2^n + 1)$ sessions, we could get

$$\mathtt{A}^1 = \mathtt{K} \oplus (\mathtt{c}_p + i), \quad i = 0, ..., 2^n.$$

- By Lemma 1, we can extract less significant n bits of $\mathtt{K}$ and $\mathtt{c}_p$.

Step 2. Set the less significant n bits of the counter $\mathtt{c}$ to 0.

- Since the less significant n bits of $\mathtt{K}$ are recovered, we can obtain the less significant n bits of the counter, say $\mathtt{c}_{p'}$, when we want.
- By sending Queries and non-zero random values to the tag, we can set the less significant n bits of the counter to 0 and denote the counter by $\bar{\mathtt{c}}$.

– Since we assumed that $\mathtt{c}_{p'} < 2^\ell - 2^{n+1} - 2$, then the number of queries is at most 2^n.

Step 3. Recover two candidates for $\mathtt{ID}$.

– Knowing that the less significant n bits of counter are 0, we send Queries and non-zero values $\mathtt{r}(i) = i \oplus (\oplus_{j=0}^{i} 2^j), (i = 0, \cdots, \ell - 1)$ and $\mathtt{r}(\ell) = \ell$ to the tag consecutively. Then, corresponding $A^1(i)$'s from the tag are given as follows:

$$\begin{aligned} A^1(i) &= \mathtt{ID} + \mathtt{GI} \oplus (\bar{\mathtt{c}} + i) \oplus i \oplus (\oplus_{j=0}^{i} 2^j) \mod 2^\ell - 1 \\ &= \mathtt{ID} + \mathtt{GI} \oplus \bar{\mathtt{c}} \oplus i \oplus i \oplus (\oplus_{j=0}^{i} 2^j) \mod 2^\ell - 1 \\ &= \mathtt{ID} + \mathtt{C}' \oplus (\oplus_{j=0}^{i} 2^j) \mod 2^\ell - 1 \end{aligned}$$

for $0 \le i \le \ell - 1$, and

$$\begin{aligned} A^1(\ell) &= \mathtt{ID} + \mathtt{GI} \oplus (\bar{\mathtt{c}} + \ell) \oplus \ell \mod 2^\ell - 1 \\ &= \mathtt{ID} + \mathtt{C}' \mod 2^\ell - 1. \end{aligned}$$

– By applying Theorem 1, we obtain two candidates for $\mathtt{ID}$.

In summary, the recovery attack consists of two phases: eavesdropping consecutive (ℓ+2) sessions and sending consecutive queries at most 2(ℓ+1) times. As a result, the attacker can obtain two candidates for $\mathtt{ID}$.

3.3 Tracing

Since the information obtained by eavesdropping consecutive $(\ell + 2)$ sessions can be applied to tags anytime, we can trace tags by sending consecutive queries $2(\ell + 1)$ times whenever necessary. In this subsection, we present another property that allows an attacker to trace tags more easily.

Theorem 2. *If a tag's* $\mathtt{ID}$ *is compromised, then the* $\mathtt{ID}$*s of tags having the same group index* $\mathtt{GI}$ *can be revealed by two consecutive queries.*

Proof. We assume that we have $\mathtt{ID}(0)$ of a tag and the information of a session between the tag and a reader except for counter, group identity and secret key. In other words, we have following information of the tag.

– $\mathtt{ID}(0)$
– $\mathtt{r}(0), \mathtt{A}^1(0) = \mathtt{K} \oplus \mathtt{c}(0), \mathtt{A}^2(0) = \mathtt{ID}(0) + \big(\mathtt{GI} \oplus \mathtt{r}(0) \oplus \mathtt{c}(0)\big)$
– $\mathtt{GI} \oplus \mathtt{c}(0)$.

Let $\mathtt{ID}(i)$ be the identity of another tag, say tag_i, having the same group identity. Then we can get $\mathtt{ID}(i)$ by two consecutive queries.

In the first session, we send a Query and nonzero value $\mathtt{r}$ to tag_i. Then we get the information

- $\mathtt{A}^1(i) = \mathtt{K} \oplus \mathtt{c}(i)$.

In the next session, we send a Query and $\mathtt{A}(0)^1 \oplus \mathtt{A}^1(i) = \mathtt{c}(0) \oplus \mathtt{c}(i)$ to the tag. Then the tag responds with

- $\mathtt{A}^1(i)' = \mathtt{K} \oplus (\mathtt{c}(i) + 1)$
- $\mathtt{A}^2(i)' = \mathtt{ID}(i) + \big(\mathtt{GI} \oplus \mathtt{c}(0) \oplus \mathtt{c}(i) \oplus (\mathtt{c}(i) + 1)\big) \mod 2^\ell - 1$.

The identifier $\mathtt{ID}(i)$ of tag_i can be extracted as follows:

$$\mathtt{ID}(i) = \mathtt{A}^2(i)' - \big(\mathtt{GI} \oplus \mathtt{c}(0)\big) \oplus \mathtt{A}^1(i) \oplus \mathtt{A}^1(i)' \mod 2^\ell - 1$$

where $\mathtt{GI} \oplus \mathtt{c}(0)$ and $\mathtt{A}^1(i) \oplus \mathtt{A}^1(i)'$ are known values.

As a special case of Theorem 2, by two consecutive queries, we can determine whether there is a tag with a specified $\mathtt{ID}$. It means that tags in OHLAP protocol are traceable in practical sense.

4 Conclusion

We have shown that the RFID authentication protocol[3] proposed at SecUbiq 2005 is vulnerable to impersonation attack and tracing. First, an attacker can easily impersonate a tag by eavesdropping one session. Second, an attacker can trace a tag by sending two consecutive queries, once he keeps initial information from (at most) $2(\ell + 1)$ consecutive queries. Third, if a tag's $\mathtt{ID}$ is revealed, then any other tag's $\mathtt{ID}$ in the same group can be easily computed just by eavesdropping a session and sending two consecutive queries. In particular, if a tag's $\mathtt{ID}$ is known, then one can trace a tag by two consecutive queries.

Impersonation and tracing are due to the method of using a counter and random values in computing authentication values. Easy traceability is due to usage of $\mathtt{GI}$, which is one of the main characteristics for the protocol. Our work shows that efficient design might result in a serious weakness in security and privacy.

References

1. G. Avoine, Cryptography in Radio Frequency Identification and Fair Exchange Protocols, PhD thesis, EPFL, Lausanne, Switzerland, December 2005.
2. G. Avoine, P. Oechslin, A scalable and provably secure hash based RFID protocol, Workshop on Pervasive Computing and Communication Security(PerSec) 2005, March 2005.
3. E.Y. Choi, S.M. Lee and D.H. Lee, Efficient RFID Authentication Protocol for Ubiquitous Computing Environment, SecUbiq 2005, LNCS 3823, 945-954, Dec. 2005.
4. T. Dimitriou, A lightweight RFID protocol to protect against traceability and cloning attacks, Conference on Security and Privacy for Emerging Areas in Communication Networks(SecureComm) 2005, September 2005.
5. D. Henrici and P. Müller, Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers, Workshop on Pervasive Computing and Communication Security(PerSec) 2004, March 2004.
6. A. Juels, RFID Security and Privacy: A Research Survey, To be appeared in IEEE Journal on Selected Areas in Communications, 2006.
7. A. Juels, R. Rivest and M. Szydlo, The blocker tag: Selective blocking of RFID tags for consumer privacy, Conference on Computer and Communications Security - CCS'03, p 103-111 ACM Press, October 2003.
8. S.M. Lee, Y,J. Hwang, D.H. Lee and J.I. Lim, Efficient Authenticaiton for Low-Cost RFID Systems, International Conference on Computational Science and Its Applications(ICCSA) 2005, May 2005.
9. M. Ohkubo, K. Suzuki and S. Kinoshita, Efficient hash-chain based RFID privacy protection scheme, International Conference on Ubiquitous Computing - Ubicomp, Workshop Privacy: Current Status and Future Directions, September 2004.
10. K. Rhee, J. Kwak, S. Kim and D. Won, Challenge-response based RFID authentication protocol for distributed database environment, International Conference on Security in Pervasive Computing(SPC) 2005, April 2005.
11. S. Weis, Security and privacy in radio-fequency identification devices, Master thesis, Massachusetts Institute of Technology(MIT), Massachusetts, USA, May 2003.

Appendix: Proofs of lemmas in Section 3.2

We prove Lemma 2 and Lemma 3 in Section 3.2.

Proof of Lemma 2. First, we consider following two cases, which can be discriminated by the values of $\mathtt{x}+\mathtt{y} (\mathrm{mod}\, 2^\ell - 1)$ and $\mathtt{x}+(\mathtt{y}\oplus 1) (\mathrm{mod}\, 2^\ell - 1)$.

Case 1. $\mathtt{x}+\mathtt{y} = 0 (\mathrm{mod}\, 2^\ell - 1)$ and $\mathtt{x}+(\mathtt{y}\oplus 1) = 1 (\mathrm{mod}\, 2^\ell - 1)$.
Case 2. $\mathtt{x}+\mathtt{y} = 1 (\mathrm{mod}\, 2^\ell - 1)$ and $\mathtt{x}+(\mathtt{y}\oplus 1) = 0 (\mathrm{mod}\, 2^\ell - 1)$.

In Case 1, $\mathtt{x}+\mathtt{y}$ could be $0, 2^\ell - 1$ modulo 2^ℓ. Depending on $\mathtt{x}+\mathtt{y} (\mathrm{mod}\, 2^\ell)$, $\mathtt{x}_0, \mathtt{y}_0$ is determined as follows:

$$(\mathtt{x}_0, \mathtt{y}_0) = \begin{cases} (0,0) & \text{if } \mathtt{x}+\mathtt{y} = 0 \mod 2^\ell, \\ (1,0) & \text{if } \mathtt{x}+\mathtt{y} = 2^\ell - 1 \mod 2^\ell. \end{cases}$$

In Case 2, by setting $\mathtt{y}' = \mathtt{y} \oplus 1$, we can apply the above argument. Therefore, both in Case 1 and Case 2, we can obtain two candidates for a pair $(\mathtt{x}_0, \mathtt{y}_0)$.

Now, we consider Case 3 which is neither Case 1 nor Case 2. This case can be subdivided to two following cases again.

Case 3-1. $\mathtt{x} + \mathtt{y} < 2^\ell$ and $\mathtt{x} + (\mathtt{y} \oplus 1) < 2^\ell$,
Case 3-2. $\mathtt{x} + \mathtt{y} \geq 2^\ell$ and $\mathtt{x} + (\mathtt{y} \oplus 1) \geq 2^\ell$.

Note that the case that $\mathtt{x} + \mathtt{y} < 2^\ell$ and $\mathtt{x} + (\mathtt{y} \oplus 1) \geq 2^\ell$ are contained in Case 1 and the case that $\mathtt{x} + \mathtt{y} \geq 2^\ell$ and $\mathtt{x} + (\mathtt{y} \oplus 1) < 2^\ell$ are contained in Case 2.

In the two subcases of Case 3, less significant two bits of $\mathtt{x}+\mathtt{y} (\mathrm{mod}\, 2^\ell - 1), \mathtt{x} + (\mathtt{y} \oplus 1) (\mathrm{mod}\, 2^\ell - 1)$ are given as follows:

$$(\mathtt{x} + \mathtt{y} \bmod 2^\ell - 1)_{1,0} = \begin{cases} (\mathtt{x} + \mathtt{y})_{1,0} & \text{in Case 3-1,} \\ (\mathtt{x} + \mathtt{y} + 1)_{1,0} & \text{in Case 3-2,} \end{cases}$$

$$(\mathtt{x} + (\mathtt{y} \oplus 1) \bmod 2^\ell - 1)_{1,0} = \begin{cases} (\mathtt{x} + (\mathtt{y} \oplus 1))_{1,0} & \text{in Case 3-1,} \\ (\mathtt{x} + (\mathtt{y} \oplus 1) + 1)_{1,0} & \text{in Case 3-2.} \end{cases}$$

Let $\mathtt{z} = (\mathtt{x} + \mathtt{y}) \oplus (\mathtt{x} + (\mathtt{y} \oplus 1)) (\mathrm{mod}\, 2^\ell - 1)$. Then $\mathtt{z}_1$ is calculated as follows.

$$\mathtt{z}_1 = \begin{cases} \mathtt{x}_0 & \text{in Case 3-1,} \\ \mathtt{x}_0 \oplus 1 & \text{in Case 3-2.} \end{cases}$$

Therefore, in Case 3, $(\mathtt{x}_0, \mathtt{y}_0)$ is given by

$$(\mathtt{x}_0, \mathtt{y}_0) = \begin{cases} (\mathtt{z}_1, \mathtt{z}_1 \oplus (\mathtt{x} + \mathtt{y} \bmod 2^\ell - 1)_0) & \text{in Case 3-1,} \\ (\mathtt{z}_1 \oplus 1, \mathtt{z}_1 \oplus (\mathtt{x} + \mathtt{y} \bmod 2^\ell - 1)_0) & \text{in Case 3-2.} \end{cases}$$

This implies that in Case 3, we can also obtain two candidates for $(\mathtt{x}_0, \mathtt{y}_0)$.

Proof of Lemma 3. Since $\mathtt{x}_0 = \mathtt{y}_0 = 0$,

$$(\mathtt{x} + \mathtt{y} (\mathrm{mod}\, 2^\ell - 1))_{j \bmod \ell} = (\mathtt{x} + \mathtt{y})_j$$
$$(\mathtt{x} + (\mathtt{y} \oplus 2^i) (\mathrm{mod}\, 2^\ell - 1))_{j \bmod \ell} = (\mathtt{x} + (\mathtt{y} \oplus 2^i))_j$$

hold irrespective of the carries for $0 < i < \ell$, $0 < j \leq \ell$.

Let $\mathtt{z} = (\mathtt{x} + \mathtt{y}) \oplus (\mathtt{x} + (\mathtt{y} \oplus 2^i))$. Since $x_{\leq i-1} = 0$, $\mathtt{z}_{i+1}$, $(i+2)$-th bit of $\mathtt{z}$, is equal to $(i+1)$-th bit of $\mathtt{x}$, because

$$\mathtt{z}_{i+1} = (\mathtt{x}_{i+1} \oplus \mathtt{y}_{i+1} \oplus \mathtt{x}_i \mathtt{y}_i) \oplus (\mathtt{x}_{i+1} \oplus \mathtt{y}_{i+1} \oplus \mathtt{x}_i (\mathtt{y}_i \oplus 1)) = \mathtt{x}_i.$$

This finishes the proof.