Selective Collision Based Medium Access Control Protocol for Proactive Protection of Privacy for RFID

JuSung Park, Jeonil Kang, and DaeHun Nyang

Information Security Research Laboratory,
INHA University, Korea*
{security, dreamx}@seclab.inha.ac.kr, nyang@inha.ac.kr
http://seclab.inha.ac.kr

Abstract. RFID is rapidly being deployed because of its versatility. However, the privacy problem cannot be handled effectively because of limited capability of RFID tags. We propose a secure medium access control(MAC) protocol to solve the privacy problem. Our secure MAC protocol defines a special kind of tag called an "ownership tag". The singulation procedure is well defined only if the ownership tag is involved in the singulation process. Instead of scrambling the information of ordinary tag using the ownership tag's key information and just sending ownership tag's information in clear form, we use forced collision generated by the ownership tag to let a reader know garbage bits that are inserted in ordinary tags. Security and performance analysis for the protocol are provided.

1 Introduction

RFID(Radio Frequency IDentification) is a very useful tool for speeding up supply chain management. RFID tag can be attached to every product if the product is required to be identified in a computerized way and it will make the bar-code system obsolete sooner or later.

The advantage of RFID is not only identification without contact, but also huge amount of information that the tag can hold. The size of memory RFID tag is much larger that that of bar-code and thus, RFID tag can identify every instance of products as well as every kind of products. These advantages of RIFD works like double-edged sword, and they might be very harmful if abused. Thus, privacy problem such as stolen information of person and industry espionage have been raised continuously. The privacy problem cannot be handled effectively because of limited capability of RFID tags: the size of memory of an RFID tag is too small to store some useful information such as long keys and also a tag has very small number of gates to perform cryptographic operation such as one-way hash function, exponentiation, etc. Beside that, because passive tag does not have power supply in itself, external device such as a tag reader must supply power for proper operation of tags. All the limitations make the privacy problem hard to be solved.

* This work was supported by the Korea Research Grant funded by Korean Government(R03-2004-000-10023-0).

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 383–392, 2005.

In this paper, we propose a secure medium access control(MAC) protocol to solve the privacy problem. Our secure MAC protocol defines a special kind of tag called an "ownership tag". Our protocol is defined assuming the binary tree walking, but it can be extended to ALOHA-based MAC protocol. In section 2, we propose and discuss our secure MAC protocol in detail. In section 3, we consider the performance of the RFID system that adopts our scheme. In section 4, we analysis estimated security strength of our scheme. Finally, in section5, we summarize our whole paper, and discuss some possibilities of our scheme.

2 Ownership Tag

2.1 Overview

The goal of our scheme is to prevent the leakage of tag's information at any time except when the consumers want to offer tag's information, which is a kind of "proactive privacy protection". To do that, a tag's information should be encoded by some cryptographic operation, and the decoding key has to be provided. In our secure MAC protocol, the key is provided as a special kind of tag called an "ownership tag", and the ordinary tags are protected by inserting some bogus bits that are not known except when the ownership tag is not located nearby the ordinary tags. Due to these features, the ownership tag must be present in nearby, when a reader wants to read some tag's information correctly. Instead of scrambling the information of ordinary tags using the ownership tag's key information and just sending ownership tag's information in clear form, we use forced collision generated by the ownership tag to let a reader know garbage bits that are inserted in ordinary tags. This approach has more security against passive eavesdropper than that because an attacker must eavesdrop a whole session to recover the key information. If key information is sent in clear form by a special tag, an eavesdropper can obtain easily the key information by listening only the special tag's communication. On the contrary, key information of our secure MAC protocol is dispersed through the whole session. The ownership tag is a kind of RFID tag that has two antennas for the forced collision like the blocker tag, and thus, can operate as a blocker tag if proper logics are implemented. Note that forced collision in our proposal works for revealing the key information, whereas forced collision in the blocker tag is used for hiding tag's ID.

Ownership tag wrapped by a foil-receipt can be provided to customers. When a customer wants to make ownership tag enable, he can unfold the receipt to prevent RFID from being read.

2.2 System Requirements

Our protocol assumes the followings:

- Ordinary tag's memory consists of OTP (One Time Programmable) memory and ROM (Read Only Memory). Directive to decide whether a bogus bit is to be inserted or not are saved in OTP, while tag's data are saved in ROM.

Table 1. Component of ownership tag scheme

Part	Components	Role
Reader	Reader	reads tag's information using binary tree walking algorithm
	Antenna	listens a query from a reader and send response
Ordinary Tag	ROM	stores tag's *data*
	OTP	stores directive *input_directive* to decide whether a bogus bit is to be inserted or not
	3 Pointers	*bogus_pointer* points to position value of bogus bits. *data_pointer* points at tag's data. *input_bogus_pointer* points at inserted bogus data.
Ownership Tag	2 Antennas	make a forced collision
	OTP	stores the same directive as that of ordinary tag
	Pointer	operates in the same way as the *input_bogus_pointer* of ordinary tag

- RFID tag is passive type. That is, it does not have any power supply in itself.
- Reader and tag use a binary tree walking for singulation.
- Ownership tag has OTP memory that is cheaper than EEPROM or flash memory.
- Ownership tag has two antennas in order to make a forced collision.
- Our scheme can adopt the silent tree walking for protection against trivial eavesdropping[3].

Forced collision is used both by blocker tag and ownership tag, but the purpose is quite different. Forced collision made by ownership tag works for revealing the key information, whereas forced collision in the blocker tag is used for hiding tag's ID.

2.3 Initialization

The ownership tag and the ordinary tags must be initialized as **figure 1** which is described as follows.

- After reader obtains ordinary tag's *data* using the binary tree walking scheme, it inserts randomly generated *input_directive* into the OTP of ordinary tag, where the *input_directive* decides whether a bogus bit is to be inserted between ordinary tag's information bits or not.
- Ordinary tags have 3 pointers to be initialized. One is called *data_pointer* that is initialized to point to the first bit of tag's data in ROM. The second one is called *input_bogus_pointer* that points to bogus bits to be inserted, which is initialized to point to the end of data bits and then decreased one by one whenever a bogus bit is inserted. The last one is called *bogus_pointer* that is initialized to point to the first bit of *input_directive* in OTP.
- An ownership tag has 1 pointer and *input_directive* like an ordinary tag, which are initialized in the same way as that of ordinary tags.

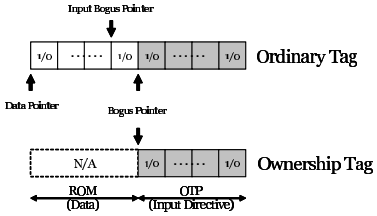


Fig. 1. Initialization of the ordinary tag and the ownership tag. The ordinary tag has *data* and *input_directive* with three pointers, and the ownership tag has only *input_directive* with one pointer.

2.4 The Protocol

We describe our protocol by showing each entity's role.

Ordinary Tag. Ordinary tag has its own data in ROM and *input_directive* in OTP memory, so its answer for the query looks scrambled if the ownership tag does not involve in the singulation process. To achieve this, we define operation of ordinary tag as following:

- If the number of reader's query is odd, ordinary tag responds with its data bit that *data_pointer* points to in ROM. Or else, it refers to *input_directive* in OTP memory.
- If the bit of *input_directive* is '0', ordinary tag regards it as 'null' and then, it responds with its data bit that *data_pointer* points to.
- Else if the bit value is '1', tag regards it as 'input bogus command' and then, it responds with bogus bit that *input_bogus_pointer* points to.

Note that the bogus bits are not randomly generated bit sequence but reverse sequence of tag's data bits. According to the ordinary tag's behavior, even though tags have the same *input_directive*, they will produce different bogus bits respectively. Pseudo-code for operation of ordinary tag is followed:

```
loop
    QUERY := receive()
    if QUERY == odd-th //Data transmission
        if *data_pointer == EOF
            break loop
        else
            respond(*data_pointer);
            data_pointer++;
    else //Bogus bit transmission
        if input_directive == 0
            bogus_pointer++;
            respond(*data_pointer);
```

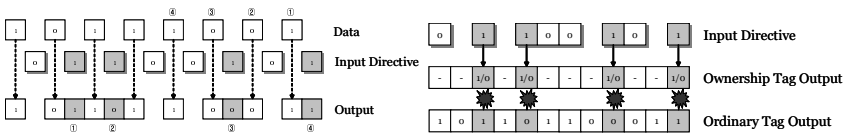
```
        data_pointer++;
    else
        bogus_pointer++;
        respond(*input_bogus_pointer);
        input_bogus_pointer--;
```

Ownership Tag. Ownership tag is in one of three states: state to make a forced collision, state to listen reader's query, and state to keep silence. It makes a transition to one of three states according to reader's query. Following pseudo-code shows how ownership tag works.

```
loop
    initialize bogus_pointer
    loop until one singulation completes
        QUERY := listen() /* from reader */
        if QUERY != re query && QUERY == even-th
            if *input_directive == 1
                send(collision)
            bogus_pointer++
```

Because an ordinary tag responds with a data bit that *data_pointer* points to for the odd numbered query, ownership tag does not respond for the ordinary tag to send its data to a reader. For the even numbered query, ownership tag makes collision according to the *input_directive* selectively. Also, if the query is received again after the collision, it must keep silent for the reader to know the existence of bogus bit.

Reader. Reader recognizes the bogus bit position of all involving tags during collision resolution process by sending re-query and checking whether the response comes or not. If there is at least one 'no response' for all re-queries in the same level where collision has occurred, reader can find that the level with collisions contains a bogus bit. However, even though ownership tag keeps silence, if one of ordinary tags responds with some data, reader will regard this position as a data bit. Actually, the position is for bogus bit, so we must be careful when a reader inserts *input_directive* into ordinary tags.



(a) (b)

Fig. 2. (a). How an ordinary tag works in our scheme. The ordinary tag's *data* are scrambled by its *input_directive*. (b). An ownership tag helps the reader be able to read the ordinary tags' *data* correctly.

The *input_directive* must satisfy some property to avoid the above malfunction. When reader generates *input_directive*, reader should avoid inserting bogus bit into the level h of the binary tree that has 2^h nodes (or 2^h collisions) fully. **Figure 3** shows binary tree constructed from the binary tree walking algorithm. Because the number of tags that involve in singulation process usually much less than the total number of possible data, we do not need to perform collision resolution procedure after resolving h1 bits. Thus, the tree looks like **figure 3** when the probability distribution of data is uniform. h_1 is approximately $\lceil log_2 n \rceil (n =$ number of tags). Assuming that n is 128, $h_1 \approx 7$ bits. Namely, a reader must avoid inserting bogus bits into the first 7 bits of tag.

Bit positions that bogus bits must not be inserted into are easily decided by a reader during its initialization phase. After reading original data of all tags during initialization, a reader can find bit positions where collisions occur. Except the level h of the binary tree that has 2^h nodes fully, bogus bits can be freely inserted into any level that has less than 2^h nodes. If we insert a bogus bit into the level h that has less than 2^h nodes, ownership tag can keep silence without being disturbed by other ordinary tags in any empty branch. Thus, the i-th bit of *input_directive* must be '0' if level i has 2^i nodes. Otherwise, reader fills the remaining bit positions randomly.

Reader operates as follows.

```
// begin binary tree walking
// collision_position_list : a stack for depth first search

input_directive[] := 0
send(next-bit query)
loop
    BIT := read()
    if BIT == EOF
        save(tag's data)
        if collision_position_list != empty
            pop(collision_position_list)
            send(wake-up query)
        else
            break loop
    else if BIT == collision
        push(collision_position_list, current bit position)
        send(re query)
    else
        send(next-bit query)
        input_directive[current depth] := 1

// begin process of inserting bogus bits
while i < bit length of input_directive
    if input_directive[i] == 1
        input_directive[i] := randomly generated bit;
```

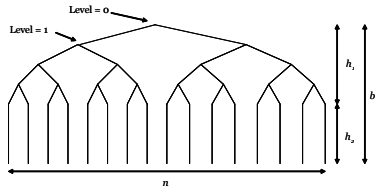


Fig. 3. Binary Tree Walking Scheme working Model. After h_1, the reader communicate with one tag by one-bit.

3 Performance Evaluation

In accordance with **table 2** and **figure 2**, $h_1 \approx log_2 n$ and $h_2 \approx b - log_2 n$. Thus, amount of bit transferred in h_2 are

$$2n(b - log_2 n) \tag{1}$$

Amount of bits transferred in h_1 by 'wake-up' query is

$$m \sum_{k=1}^{log_2 n - 1} = m(n-2) \tag{2}$$

In h_1, sum of 'next-bit' query and responses is

$$2 \sum_{k=1}^{log_2 n} = n - 4 \tag{3}$$

Thus, total sum of bits transferred in binary tree is

$$(2b - 2log_2 n + m + 1)n - 2m - 4 \tag{4}$$

We can say that it take $(2b - 2log_2 n + m + 1)n - 2m - 4$ BT to singulate n number of tags. Because total sum of IBTs required is the same as that of (4) if we let $m = 1$, the sum of IBTs in binary tree is

$$2(b - log_2 n + 1)n - 6 \tag{5}$$

Table 2. Component of ownership tag scheme

Notation	Explanation
b	The number of bits of tag
m	Bit length of b, or $\lceil log_2 b \rceil$
n	The number of tags
BT(bit time)	Unit time required for transmitting a bit
IBT(inter bit time)	Unit time between BTs

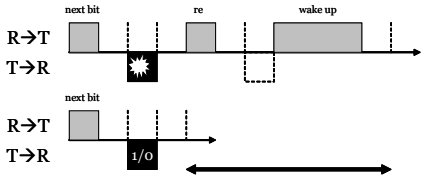


Fig. 4. When a collision occurs, it requires more BT and IBT to sigulate

Table 3. Comparison of singulation time, $n = 100$

Total Length	Un-scrambled Ordinary tag		Protected tags without ownership tag		Protected tags with ownership tag	
(+bogus bits)	BT	IBT	BT	IBT	BT	IBT
64(+32)	12157	11667	18655	18067	44255	28255
128(+64)	25055	24467	37953	37267	95553	57153
256(+128)	50573	50067	76451	75667	204451	114851
512(+256)	102051	101267	153349	152467	434949	230149

As shown in **figure 4**, when ownership tag is involved in singulation, amount of time required to singulate tags increase by $m + 1$ BTs and 3 IBTs for one forced collision. Remind that reader should not insert bogus bits into the h_1. If we let the number of inserted bogus bits c, which must be inserted bogus bits into h_2 only, then the number of added collision is in proportion to c and n. If we denote amount of time required to send all bits by BT_{off} and IBT_{off} when ownership tag is absent, or else BT_{on} and IBT_{on}, then

$$\mathrm{BT}_{on} = \mathrm{BT}_{off} + cn(m+1) \tag{6}$$

$$\mathrm{IBT}_{on} = \mathrm{IBT}_{off} + 3cn \tag{7}$$

BT_{off} and IBT_{off} are the same as (4) and (5) respectively.

Ownership tag increases BT and IBT by almost $50\% \sim 250\%$, but the degradation is not so significant. After reader detects a bit position of a bogus bit, it can ignore the same bit position of other tags, because all the tags have bogus bits in the same bit positions. In such a case, $n = 1$ or 2 in (4). Using the idea, only the added overhead is not $5\% \sim 250\%$, but the same as addition of one or two scrambled ordinary tags.

4 Security Analysis

Without ownership tag, neither any reader nor attacker can extract tag's information because the information is scrambled with randomly generated *input_directive*. Even though every tag has the same *input_directive* with each other, inserted bogus bits are different. Thus, attacker who has collected many tag's information scrambled with the same *input_directive*, it is hard to find out

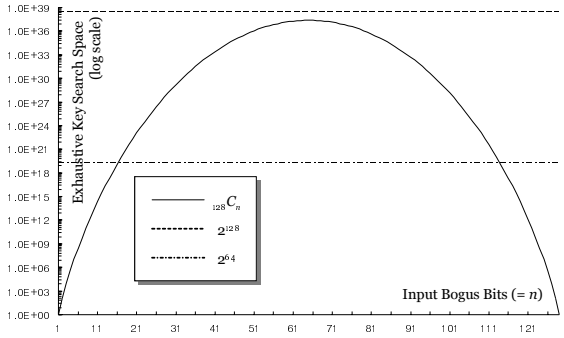


Fig. 5. Key Space for exhaustive search according to the number of inserted bogus bits

exact position of bogus bits because any same bit pattern does not appear in the scrambled data. Actually, this scrambling is a kind of simple block cipher using substitution with expansion and transposition. If higher level of security is required at the expense of cost-effectiveness, other strong block ciphers can be used instead of this scrambling. In that case, reader can recover (or decrypt) all tags' information only after it finds all bits of *input_directive*(or a key).

Figure 5 shows the size of exhaustive key search space for the number of inserted bogus bits. The size of key search space against ownership tag is MIN$\{_{128}C_n, 2^n\}$ if we use 128-bit *input_directive*, and $_{128}C_n > 2^n$ for $n > 0$, where C denotes combination operator. Thus, the size of key search space of brute force attack is $_{128}C_n$. As shown in Figure 6, we can obtain the highest level of security when the number of bogus bits inserted is 64 bits. The number of actually inserted bogus bits will be 64 bits on average, if we use random number generator for *input_directive* that has uniform probability distribution.

If reader detects exact position of a bogus bit during singulation process owing to ownership tag, it does not need to send any re-query for the bit position because he already knows that other tag's response for query of the bit position is bogus. This property makes our scheme more secure and more efficient. That is, if illegal reader (eavesdropper) wants to know all positions of bogus bit, it must eavesdrop whole communication session between tags and a legal reader. In that sense, our scheme can be seen as a MAC protocol that defines how to send key information secretly through open channel.

5 Conclusion

We propose a secure MAC protocol, which defines a special tag called ownership tag. Only one who holds the ownership tag can insist his ownership of products with RFID tags paired with the ownership tag. From a customer's point of view, ownership tag is more proactive than the blocker tag to protect privacy, because information of tags is scrambled at normal times and is in clear form only when ownership tag is involved in the protocol. Only when customer needs insist or

identify his tags, he will carry the ownership tag. If customer wants to hide ordinary tags while carrying the ownership tag, the ownership tag should be in a faraday cage, which can be provided as a form of receipt.

Actually, the concept of ownership tag can be easily implemented by making ownership tag send key information in clear form using normal binary tree walking. However, our scheme has more security against passive eavesdropper than this because an attacker against our scheme must eavesdrop a whole session to recover the key information. If key information is sent in clear form by a special tag, an eavesdropper can obtain easily the key information by listening only the special tag's communication. On the contrary, key information of our secure MAC protocol is dispersed through the whole session.

Though we show only binary tree walking version of ownership tag, it can be adopted to the ALOHA protocol.

References

1. Ari Juels, Ronald L. Rivest and Michael Szydlo : The Blocker tag: Selective Blocking of RFID Tag for Consumer Privacy, RSA Laboratory, MIT
2. Ari Juels and John Brainard : Soft Blocking: Flexible Blocker Tags on the Cheap, RSA Laboratory
3. Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest and Daniel W. Engels : Security and Privacy Aspect of Low-Cost Radio Frequency Identification Systems, In Security in Pervasive Computing, 2003
4. Stephen A. Weis : Security and Privacy in Radio-Frequency Identification Devices, MIT, 2003