

# A Gen2-Based RFID Authentication Protocol for Security and Privacy

Hung-Min Sun and Wei-Chih Ting

**Abstract**—EPCglobal Class-1 Generation-2 specification (Gen2 in brief) has been approved as ISO18000-6C for global use, but the identity of tag (TID) is transmitted in plaintext which makes the tag traceable and clonable. Several solutions have been proposed based on traditional encryption methods, such as symmetric or asymmetric ciphers, but they are not suitable for low-cost RFID tags. Recently, some lightweight authentication protocols conforming to Gen2 have been proposed. However, the message flow of these protocols is different from Gen2. Existing readers may fail to read new tags. In this paper, we propose a novel authentication protocol based on Gen2, called *Gen2+*, for low-cost RFID tags. Our protocol follows every message flow in Gen2 to provide backward compatibility. *Gen2+* is a multiple round protocol using shared pseudonyms and Cyclic Redundancy Check (CRC) to achieve reader-to-tag authentication. Conversely, *Gen2+* uses the memory read command defined in Gen2 to achieve tag-to-reader authentication. We show that *Gen2+* is more secure under tracing and cloning attacks.

**Index Terms**—Protocol design and analysis, security, privacy.

## 1 INTRODUCTION

RADIO Frequency Identification (RFID) tags for the function of next-generation electronic product code (EPC) will become one of the most widely used devices in the near future [1]. An RFID application contains three basic roles:

1. tag,
2. reader, and
3. back-end database.

Each tag contains a unique identification, often called the tag identification (TID). The reader is used to query the tag's TID and forward it to the back-end database. Once the tag is found valid, the back-end database will look up its product information for further processing. RFID tags are classified into three types: active, semipassive, and passive. Active tags contain batteries so that they can actively communicate with the reader. Semipassive tags also contain batteries but they wait for the reader's query. As for passive tags, the power comes from the reader. The class of a tag represents the effective reading range. The reading range of a class-0 tag is 5-10 cm, and that of a class-1 tag is up to several meters.

EPCglobal class-1 generation-2 (Gen2 in brief) [2] was approved as ISO18000-6C in July 2006. It is widely believed that Gen2 tags will be the mainstream when developing RFID applications because the effective reading range is larger. However, the Gen2 specification has the vulnerability that the TID is transmitted without any guard. Thus,

- The authors are with the Department of Computer Science, National Tsing Hua University, 101, Kuang Fu Road, Sec. 2, Hsinchu, Taiwan 300, ROC. E-mail: hmsun@cs.nthu.edu.tw, sd@is.cs.nthu.edu.tw.

Manuscript received 20 Nov. 2007; revised 2 Aug. 2008; accepted 9 Dec. 2008; published online 18 Dec. 2008.

For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2007-11-0348. Digital Object Identifier no. 10.1109/TMC.2008.175.

in this paper, we focus on the protection of class-1 passive tags from being accessed by malicious readers.

### 1.1 Security Threats

Previous studies [1], [3], [4], [5], [6], [7] addressed several threats to RFID applications:

- *Eavesdropping*: An adversary listens to all the communications through Radio Frequency (RF) and dumps them for later cracking.
- *Hotlisting*: This attack, similar to eavesdropping, focuses on matching specific values with his chosen keywords to understand one's personal interests, like his favorite books or his reading behavior.
- *Replay attack*: An attacker repeats the same messages logged from eavesdropping to spoof a reader or tag.
- *Cloning*: Since RFID tags are usually placed in open environments such as hospitals, schools, and offices, they may be exposed under all kinds of malicious tools. An adversary can read the tag and then clone the tag by writing all the obtained data into a blank tag.
- *Tag tracing*: Readers and tags have different abilities of sending data. Attackers can either identify the same tag from passively logged messages or interact actively with the tag to understand its location.
- *Invoading privacy*: Customers may not want eavesdroppers to know what items they have bought from the store, which may indicate their interests.
- *Data forging*: Once RFID technology becomes the mainstream of retailing, a tag may store extra data such as date, price, and the place of production for convenience. However, this also attracts attackers to modify the price and cause great loss.
- *Denial of service*: It is an effective attack against some RFID systems, which utilize locking or killing approach as their protection. Even though this is the weakest test when designing secure protocols,

there exist many other simpler ways toward denial of service [25].

Attackers impeding the deployment of RFID technology are usually of two types: active and passive. An active attacker broadcasts previously designed messages to a reader and pretends to be a valid tag. He can broadcast messages to a tag as well, to spoof it and obtain its TID or its secret key. Once the TID is unveiled, the tag can be easily cloned if this TID is the only secret shared between the tag and the database. In contrast, a passive attacker only listens to the communication seeking information leakage. Any repeated message can be used for tracing.

## 1.2 End User's Concern

Currently, RFID tags are used for tracking cargos. Tags are mostly attached to the wrapping boxes instead of the product inside. Products are still being sold to customers at retail via traditional bar codes. Customers cannot enjoy the benefits of RFID technology. In the bright side, they do not have to worry about being tracked. But for the next generation, when many items are tagged with RFID tags, each person needs a reader to manage all his items, e.g., a mobile RFID reader [30]. It may become a disaster if anyone with a reader is able to access another person's possessions.

The following question is: Is it really necessary to tag every item? We note that RF technology can provide three functions: item awareness, information searching, and quality control. End users request convenience, location privacy, and anonymous purchase. Hence, it is crucial to evaluate security threats from different aspects:

- *The price of an item:* Ohkubo et al. [8] indicated the information leakage of user belongings. But many of them do not need the protection of privacy. We classify them into luxurious items, handful items, and cheap consumable items. Luxurious items can always adopt the latest technology to provide the three functions mentioned above. They can have tags with built-in cryptographic functions, which can protect tags from being cloned or tracked. In this way, however, information searching becomes less convenient because it needs the cooperation between the manufacturer and the seller. It may also need a key management system to keep product information. Handful items are those that people use every day, small but important, such as wallets, keys, e-cards, books, document folders, etc. They are massively manufactured items, and therefore, they are not expensive. Once we use class-1 tags to label them, users can easily find these items from a long distance, and so can the attacker. This points out the need for different modes, i.e., private mode and public mode. The tag should be switched to the private mode before it is carried outdoors. Cheap consumable items like tissue paper may have no need to be labeled. Currently, users may not have the urge to tag these items.
- *The effective reading range:* The effective range of a class-0 13.56 MHz tag is about 3.52 meters at most. A class-1 915 MHz passive tag may have an operating range around 3 meters, while the forward channel

may extend up to 100 meters [1]. Previous literature assumes that tracking is feasible because the reader's power is much greater than the tag's power. In fact, the effective reading range is smaller than expected even using UHF signals. One can tell if a stranger is stalking him with a large reader in his hand. Hence, the simplest way to avoid this is to leave the reading range. However, in the scenario of active scanning attack, an attacker may attach a piggy RF logger to the legal reader and try to scan from a distance far away. For example, anyone can attach such a logger under a cashier in a supermarket and regain it afterward. This attack is unavoidable no matter what tag (class-0 or class-1) we are using unless the manager examines the cashier all the time.

- *The popularity of RFID tags:* Lots of companies and airports are applying RFID for better flow control and quality control and try to benefit from the high speed. RFID technology requires the support from government and large organizations. The cost of tags and readers, especially readers, is dropping rapidly. We can see the increase of potential malicious readers in the future. To boost the usage of RFIDs, we have to overcome privacy issues in a reasonable way. When people begin to accept RFIDs, manufacturers are more willing to develop secure tags so that the cost can drop.

By far, studies on RFID protocols can be classified into two main categories. The first seeks for provable secure protocols using encryption functions or hash functions. The second focuses on lightweight and practical solutions that take costs and prices as their first consideration.

### 1.2.1 Secure Solutions

Many researchers have suggested cryptographic primitives to encrypt TID in sessions. Hash-based protocols like [1], [9], [10], [11], [12], [13], [14] are taking the advantage of one-way function to prevent direct exposure of TID. They suggested using a hashed value, usually called *metaID* or secured *SID* for transmission instead of TID. To verify a tag, a verifier needs to search the back-end database and compute the same hashed value. Once found, the database looks up the corresponding TID and sends it back to the reader. *Hash-chain*-based solution [8] involves synchronized key update so that both the tag and the back-end database can communicate with each other. This method also provides *forward security* but often suffers from *desynchronization attack*. Blocker tag [15], *Guardian* [16], and *soft-blocking* [17] use blocker tags or equipments to disrupt illegal reading by either actively broadcasting RF signals or passively exploiting the singulation protocol. In this way, the adversary cannot normally access the protected tag. Other solutions are *signature-based* protocol [18] or *public-key-based* protocol [19], [20]. However, cryptographic primitives, such as DES or AES, are relatively expensive than hash functions. Asymmetric method is even more resource consuming and is mainly used for key management in RFID systems. See also [17] for indicated problems.

As mentioned in [21], most suggested protocols are theoretically plausible but demand complete new design in

practice. The cost of encryption functions remains high nowadays. Hash function with the uninvertible characteristic seems to be an excellent candidate for low-cost tag design. However, the cost of a hash function is still higher than basic operations such as AND, OR, XOR, and rotation.

### 1.2.2 Lightweight Solutions

A shared pseudonym between each tag and the back-end database is required in [17], [22], [23], [24], [25], [26], [27], [28], [29]. Exclusive-or (XOR) operation is the main functional component that is needed. They suggest key update after each session to guarantee forward security. Vajda and Buttyán [22] proposed a set of extremely lightweight tag authentication protocols based on XOR, subset, squaring, etc. Although they cannot prevent active tracking attacks, they present effective ideas for low-cost RFID tag design. Another utmost lightweight metric is based on the concept of one-time pad [24]. Each tag must store *ReaderID* for every reader. Although it provides sufficient security level, the tag usually needs to store more than one *ReaderID*. A more practical way is to have all tags of each person store the same *ReaderID*. However, this becomes a drawback because any compromised tag will reveal its *ReaderID* and will allow an attacker to access all other tags. Chabanne and Fumaroli [28] proposed a noisy protocol to prevent passive eavesdropping via taking advantage of a *Bit Pair Iteration Protocol* to correct transmission errors. Their work is particularly useful at initial setup and privacy amplification of RFID application scenarios.

*Privacy bit* suggested in [30], [31] prevents illegal reading via a minor modification in the tag. A privacy flag is asserted while privacy violation is seriously concerned. Then, the tag will not respond to normal interrogation of any reader. To access this tag again, they often require an extra command, called *private-read*, which needs the authentication process as well. *CRC-based protocol* [32] focuses on the enhancement of Gen2 protocol with the built-in CRC component to cut down cost.

In this paper, we propose a solution, called *Gen2+*, for current RFID application before safer tags with built-in cryptographic function come to market. We focus on the protection of UltraHigh Frequency (UHF) passive tags from malicious readers who will try to perform skimming, tracking, and cloning attacks after obtaining TID. Our scheme utilizes the built-in CRC function as a verification function to authenticate readers based on shared pseudonyms. A tag's ID is never revealed until the reader passes the challenge.

In RFID applications, readers are much more expensive than tags. We aim at following the Gen2 standard so that existing readers can read both Gen2 tags and *Gen2+* tags. Although the scheme makes a modification to the tag, the cost in tag is still cheaper than hash-based or encryption-based solutions.

Compared with other lightweight solutions, we provide better security level than [23], [26], [32], [27], [29]. The protocol only needs to update the key once in a while, instead of in every round as in [25]. The proposed *Gen2+* tag requires no cryptographic operations but only a Hamming distance calculator and some memory space. We evaluate the number of rounds required for authentication and give an algorithm to boost the search of the back-end database.

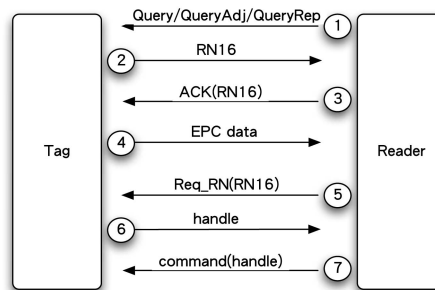


Fig. 1. EPCglobal class-1 generation-2 protocol.

The rest of the paper is organized as follows: We give a review of Gen2 protocol along with previous work in Section 2. The proposed protocol is given in Section 3, followed by simulation results and security analyses in Section 4. Comparisons with other works are shown in Section 5.

## 2 RELATED WORK

### 2.1 Generation-2 Protocol

First, we address the limitations of a Gen2 passive tag, and then we review the procedure of the Gen2 protocol. A Gen2 tag contains a Pseudorandom Number Generator (PRNG) and protects message integrity via Cyclic Redundancy Code (CRC-16). The memory space is separated into four banks: Reserved memory, EPC memory, TID memory, and User memory. It harvests power from readers through the antenna, and hence, cannot perform complex computations.

The Gen2 protocol has a main drawback pointed out in [32]. The exposition of tag identification in this protocol may violate user's privacy. Any revealed TID can be easily tracked and cloned. Our goal is to enhance the security of the Gen2 protocol and remain lightweight at the same time.

A tag can generate a 16-bit random string, denoted as RN16, and temporarily stores it in the memory. A read operation consists of three stages: *select*, *inventory*, and *access*. *Select* is used to select a specific population of tags. Fig. 1 shows the steps of the original Gen2 protocol (in Annex E [2]). Steps 1-4 in Fig. 1 are called the *inventory* stage. After the reader successfully selects a tag through singulation protocol as defined in the Gen2 specification, the tag enters its *ready* state and backscatters RN16, which is Step 2 in Fig. 1. At Step 3, note that any reader echoing RN16 will make the tag enter its *acknowledged* state and reply its EPC data. Hence, any malicious reader is able to obtain TID. This is due to the lack of the reader-to-tag authentication.

Steps 5-7 represent the access stage. Message 5 is a reader command, which asks for another random number. The tag will check if RN16 is correct and replies message 6, called *handle*. Step 7 involves memory access commands, such as *Read*, *Write*, and *BlockWrite*. The reader must provide the same *handle*.

From the specification of Gen2, one can see that optimal security is protected by the access password. In other words, the time complexity of exhaustive search is  $O(2^{32})$ . The scheme is safer when the password is longer.

Therefore, many researchers suggest a longer password like 64-bit or 128-bit.

To overcome the problem and provide lightweight solutions for Gen2 specification at the same time, Juels [23] and Duc et al. [32] proposed two challenge-response protocols.

## 2.2 Juels' PINSet

Juels' protocol [23] generates a PINSet to test a tag by randomly placing the correct PIN value in a set of size  $q_J$ , where PIN refers to the access password stored in the tag. Obviously, the probability of a correct guess is  $1/q_J$ . This method prevents tag cloning and can be extended to provide reader-to-tag authentication. At this part, PINSet is generated by a trusted (back-end) server and forwarded to the reader. In this way, the reader is not aware of the correct position of PIN in the set. This protocol, although very lightweight, only enhances the security level to  $O(q_J)$ .

## 2.3 Duc's CRC-Based Protocol

Duc et al. [32] used PIN to protect sessions directly. They require extra memory space to store a key value  $K_i$  shared between the back-end server and the tag. During each session  $i$ ,  $K_i$  is updated to provide forward secrecy. This easily suffers *desynchronization* attacks as claimed in their work. They suggest informing both the tag and the server to update at the same time. The following is the protocol, where  $S$  stands for the back-end database or server,  $R$  for the reader, and  $T$  for the tag, respectively. Note that  $\oplus$  denotes XOR. These notations are used throughout this paper. Here,  $f$  stands for one-way function:

1.  $R \rightarrow T$ : Query.
2.  $T$ : Compute  $M_1 = CRC(TID \oplus r) \oplus K_i$  and  $C = CRC(M_1 \oplus r)$ , where  $r$  is a nonce.
3.  $T \rightarrow R \rightarrow S$ :  $M_1$ ,  $C$ , and  $r$ .
4.  $S$ : Search all possible tuple  $(TID, K_i)$  such that  $M_1 \oplus K_i = CRC(TID \oplus r)$ .
5.  $R \rightarrow S, R \rightarrow T$ : Update  $K_{i+1} = f(K_i)$ .

Their scheme provides higher security level than Juels' and prevents forged message. However, we find that there exists a serious problem, i.e., replay attack. Because the random value nonce  $r$  is always generated by the tag, a passive attacker can record all the messages of Step 2 to replay afterward. As long as the tag has not been interrogated, which will trigger the key updating, replay attack is possible. Even after the key updating, the attacker can still query again and record this message. If the back-end server does not check whether this tag contains the correct session key, the above method becomes a cloning attack with complexity  $O(1)$ . This problem may be solved by having the reader generate  $r$  instead. But this also brings another issue, tracking. Any malicious reader can keep sending the same nonce and check if the tag responds with the same value. As for desynchronization attack, an active attacker, who never sends update message in Step 5, can break the protocol with time complexity of  $O(2^l)$ , where  $l$  is the length of  $K_i$ , and in this setting,  $l$  equals 16.

We can see CRC function as a mini hash function that exits collisions. The following is an example of a tracing attack against Duc's protocol with constant rounds. A

malicious reader sends query to the tag and obtains  $M_1$  without finishing the protocol. Whenever the attacker wants to trace the tag, he sends another query to obtain  $M_2$  from Step 2 and aborts. Now, the attacker has

$$M_1 = CRC(TID \parallel r_1) \oplus K_i, \quad (1)$$

$$M_2 = CRC(TID \parallel r_2) \oplus K_i, \quad (2)$$

where  $\parallel$  represents string concatenation and  $r_1, r_2$  are nonce values. In this way, the attacker can identify the tag by the following equation:

$$M_1 \oplus M_2 = CRC(r_1) \oplus CRC(r_2). \quad (3)$$

If the tag is ever queried by a valid reader which causes the key update, the attacker can restart the attack.

## 2.4 Li's Protocol with Substring Function

Li et al. [26] suggested using a shared secure ID (SID) between the tag and the database so that during each reading, the tag challenges the reader with two random numbers  $n_1, n_2$ , which mark a segment of SID and become the partial ID (PID). Their protocol runs as follows:

1.  $R \rightarrow T$ : Query  $\parallel r$ , where  $r$  is randomly generated.
2.  $T$ : Compute  $r' = r \oplus PID_{1L} \oplus PID_{2R}$ , where  $PID_{1L} = f(SID, n_1)$ ,  $PID_{2R} = f(SID, n_2)$ .
3.  $T \rightarrow R \rightarrow S$ :  $r', n_1, n_2, r$ .
4.  $S$ : Search all SIDs and verify the tag by checking whether  $r' \oplus r = f(SID, n_1) \oplus f(SID, n_2)$ .
5.  $S \rightarrow R \rightarrow T$ :  $PID''$ .
6.  $T$ : Check whether  $PID'' = f(SID, n_1, n_2)$ .
7.  $T \rightarrow R \rightarrow S$ : Send "OK."
8.  $S \rightarrow R$ :  $SID$ .

$f$  is a substring function for the calculation of  $PID_{1L}$  and  $PID_{2R}$  in Step 2.  $PID_{1L}$  is formed by the first  $n_1$  bits of  $SID$ , while  $PID_{2R}$  is formed by the last  $n_2$  bits of  $SID$ . However, this protocol has been broken by Chien and Huang in 2007 [33]. Li's protocol is simply based on shared pseudonym and XOR operation. However, since  $f$  is only a substring function, if  $n_1$  is greater than  $n_2$ , then the first  $(n_1 - n_2)$ -bits are revealed. In the same vein, if  $n_2$  is greater than  $n_1$ , then the last  $(n_2 - n_1)$ -bits are revealed. There exists a great chance that malicious reader can recover the whole  $SID$  with a few rounds of data.

## 2.5 M<sup>2</sup>AP

A minimalist mutual-authentication protocol (M<sup>2</sup>AP) for low-cost RFID tags was proposed in [27] based on simple operations such as XOR, OR, AND, and sum of modulo. A tag and a reader share a pseudonym  $SID$  and four keys  $K1, K2, K3$ , and  $K4$ . During each session, the reader generates two random numbers  $n1$  and  $n2$ . Let " $\vee$ " denote OR operation, " $\wedge$ " for AND, and " $+$ " for modular summation. Their protocol runs as follows:

1.  $R \rightarrow T$ :  $SID \oplus K1 \oplus n1 \parallel (SID \wedge K2) \vee n1 \parallel SID + K3 + n2$ .
2.  $T \rightarrow R$ :  $(SID \vee K4) \wedge n2 \parallel (SID + TID) \oplus n1$ .

The tag verifies the reader by checking the  $n1$  value extracted from the first two messages. The tag then

responds to the reader if it is correct. Both *SID* and four keys must be updated after each session to provide forward secrecy. Recently, an attack on  $M^2AP$  has been proposed in [35]. They could discover the tag's identity and some shared secrets from two eavesdropped rounds.

## 2.6 Juels' Minimalist Cryptography

Juels introduced the concept of multiple pseudonyms as one-time pads [25]. Each reading must search the whole database to rule out impossible tags and then mark the corresponding pseudonym invalid. The cut down version addresses that most tags only perform the first step of the protocol. So, the attacker cannot obtain useful information but only pseudonyms. This also brings a problem, i.e., running out of pseudonyms due to the fact that they were statically planted into the tag before being sold.

## 2.7 SASI

Chien's protocol, called SASI [29], used the same basic operations as in  $M^2AP$ . The proposed scheme is ultralight-weight, but active tracking is still possible among two valid reads because *IDS* in SASI is a static value. Sun et al. [34] showed a desynchronization attack on SASI with at most 96 trials.

To summarize previous literatures, as we have seen, encryption algorithms and hash functions are not suitable for low-cost RFID tags, at least in the near future. Jamming signals, although very effective, may be illegal and somehow interfere with normal reading. To minimize the cost in each tag, the design principle is to use built-in components.

## 3 PROPOSED SCHEME

Our main idea is to randomize each session and provide mutual authentication between legitimate readers and tags, like other pseudonym-based approaches [23], [25], [26], [27], [32], [29]. In order to be compatible with the original Gen2 specification, we proposed a new protocol called *Gen2+* using only PRNG and CRC-16 functions for authentication.

We assume that each tag shares an  $l$ -word-long random string, called *keypool*, with the back-end database. This string is randomly generated by the back-end server and is written into the tag's user-bank memory before deployment. The string is considered as a keypool where we can randomly draw keys. A threshold value is set in each tag to tolerate error bits of the received value and to boost the reading speed. Therefore, we need an extra circuitry element in the tag to calculate the Hamming distance of two 16-bit numbers. Hamming distance calculators are often seen in error-correcting hardwares. We believe the new tag can be implemented easily.

### 3.1 Secured Gen2 Protocol

In the following, we show the basic version of the secured Gen2 protocol. Step 1 in Fig. 2 is the same as the singulation protocol described in the specification. The reader sends commands Query/QueryAdjust/QueryRep to select a specific tag. In Step 2, the tag whose *Q*-slot counted down to zero responds to the reader's request and backscatters RN16 to proceed. We now consider this 16-bit random number as two 8-bit addresses,  $a$  and  $b$ . These two numbers

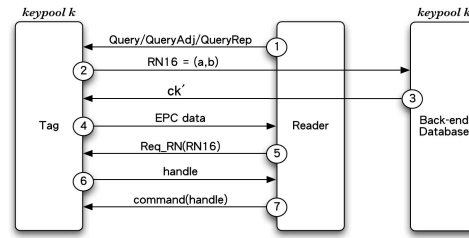


Fig. 2. *Gen2+* protocol: basic version.

mark a segment of the *keypool* stored in this tag. Let  $k[a : b]$  denote the segment (substring) from the  $a$ th word to the  $b$ th word of *keypool*  $k$ . Note that this marking is made circular, which follows the following equations. At the same time, the tag calculates the CRC residue of  $k[a : b]$  and keeps the residue, which is denoted by  $ck$

$$k[a : b] = \begin{cases} k[a : b], & \text{for } a \leq b, \\ k[a : l - 1] || k[0 : b], & \text{for } a > b. \end{cases} \quad (4a)$$

$$(4b)$$

In Step 2 of Fig. 2, the reader simply forwards this number to the back-end database. The database then searches through records and generates a 16-bit number  $ck'$ , called the *centralkey*, in Step 3. This value is sent back to the reader and the tag as well. In Step 4, the tag compares  $ck'$  with  $ck$ . If their *Hamming* distance is smaller than a threshold value  $t$ , the tag believes the reader is legal and replies with its EPC data; otherwise, it remains silent and returns to *arbitrate* state.

Most of the time, only Steps 1-4 are required. We suggest updating *keypool* through *MemWrite* command once in a while. Since the steps of requesting PIN number and the steps of *MemWrite* command are exactly the same as in the standard (Fig. 1), they are omitted here. During the update, the back-end server randomly generates another  $l$ -word-long string as the new *keypool* and overwrites the old value through memory write. Steps 1-4 only provide reader-to-tag authentication. If the back-end server wants to check whether a tag is valid or not, it can read out the whole *keypool* and check if it is the same in database.

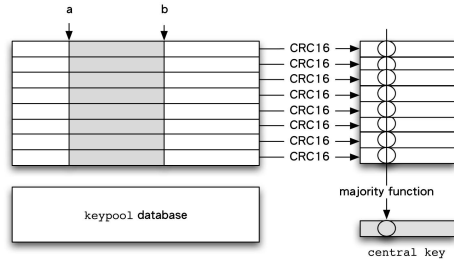
The threshold value  $t$  is also configurable by using *MemWrite* command. This value represents that the tag can tolerate up to  $t$ -bit Hamming distance between  $ck'$  and  $ck$ . This value is previously set to 16 before deployment, which means the tag always accepts and acts like a Gen2 tag. The threshold value is designed to boost the reading speed and provide a trade-off between security and efficiency. After the tagged item is sold at the retail,  $t$  is set to zero to prevent further active scanning. This means the tag only accepts if  $ck'$  exactly matches  $ck$ .

For example:

Let  $l = 32$ ,  $t = 1$ , and

$keypool = 1234567890FFF \dots FFF_h$ .

1.  $R \rightarrow T$ : Query.
2.  $T \rightarrow R$ : Choose  $(a, b) = (0, 4)$ . Hence, the segment is  $k[a : b] = 1234567890_h$ .
3.  $S \rightarrow R$ : Assume the back-end database calculates the central key from the current database and obtains  $ck' = 53D8_h$ .
4.  $R \rightarrow T$ :  $ck'$ .

Fig. 3. The computation of *central key*.

Recall the target CRC-16 residue  $ck$  of the tag within this round should be  $57D8h$ . At Step 3, the back-end database is able to produce a  $ck'$ , say  $53D8h$ . The hamming distance is 1, which is less equal to the specified threshold value  $t$ . In this way, the computed  $ck'$  is acceptable by this tag.

### 3.2 Central Key

Fig. 3 shows an example of the construction of *centralkey*. First, perform CRC-16 computations on each *keypool* marked by  $(a, b)$  within the database. After that, all drawn pseudonyms (16-bit strings) are passed through a *majority function* in bitwise. A *majority function* outputs "1" if its input contains more 1's than 0's; otherwise, it outputs "0." This can be realized by a counter. The output of each bit position forms the final 16-bit  $ck'$ .

The value of  $ck'$  has the minimalist Hamming distance in average with all these values passed through CRC-16. Hence, the tag can check if  $ck'$  is close enough to its own computation. If there is only one record left in the database,  $ck'$  should equal  $ck$ . If  $ck'$  is not accepted, this means  $ck'$  is not close enough to the desired value, which means the size of the candidate set in the database is still large. In other words, the server cannot identify this tag after this round. Therefore, we need a better way to identify the tag.

### 3.3 Candidate Set Finding Algorithm

Brute-force searching through database helps nothing if  $a$  and  $b$  change every round. Here, we present a candidate set finding algorithm that will produce a  $ck'$  close enough to be accepted. After singulation processes, the back-end database generates  $ck'$  and lets the reader forward it to the tag. Consider the target CRC value  $ck$  as a codeword in the code space of  $2^{16}$ , where  $t$  is the acceptable range of the tag. We want to find a suitable codeword that falls into this range. If the tag does not respond, which means  $ck'$  is outside the range, we then eliminate some entries in the database. Note that  $a$  and  $b$  changes every round and  $ck'$  must be recomputed. The following is a pseudocode of the candidate set finding algorithm for the back-end database:

#### Algorithm 1. Candidate Set Finding Algorithm

CandidateSet  $S = \{k_1, k_2, k_3, \dots, k_N\}$

//  $N$  is the size of database

While( true )

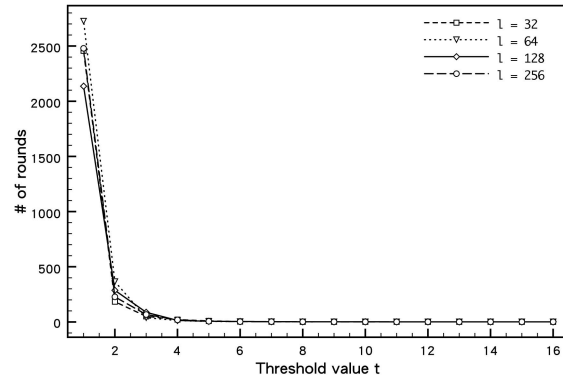
Do

Obtain  $(a, b)$

$ck' = \text{GenCentralKey}(S, a, b)$

send  $ck'$

If tag responds Then Exit

Fig. 4. Simulation result: relations between threshold value  $t$  and number of rounds with different length  $l$ .

Else

For each entry  $k_i$  in  $S$

Delete if  $\text{distance}(ck', \text{CRC-16}(k_i[a:b])) \leq t$

Next

EndIf

Loop

The central key guarantees that on each position of the 16-bit string, there exists at least  $N/2$  records having the same bit, where  $N$  is the total number of records in the back-end database. If the distribution of CRC function is not uniform, perhaps only less than  $N/3$  of them have a mismatch. The characteristic allows legal readers to quickly narrow down the candidate set. We assume that  $p_i$  is the probability of a CRC residue matching  $ck$  in the  $i$ th bit, which is  $p_i = 1/2 + \epsilon_i$ . Let  $p$  be the average probability of  $p_i$ . The probability that the Hamming distance between  $ck'$  and  $ck$  is less than or equal to  $t$ , denoted as  $P''$ , is calculated by the following:

$$P'' = \sum_{j=0}^t \binom{16}{j} * (1-p)^j * (p)^{16-j}. \quad (5)$$

The main idea of multiple rounds is that a legitimate reader can prune some neighbor values of  $ck'$  during each round and quickly narrow down the space. At the first round, if  $ck'$  is not accepted, then  $N * P''$  tags are removed from the candidate set. Afterward, the size of the set shrinks to  $N * (1 - P'')$ , then  $N * (1 - P'')^2$ , and so forth. If the size of the candidate set is less than or equal to 1, the database identifies the tag successfully. Therefore, the number of rounds  $q$  is calculated by

$$N * (1 - P'')^q \leq 1, \quad (6)$$

$$q \approx \frac{-\log N}{\log(1 - P'')}. \quad (7)$$

In our simulation of the basic version, the length of *keypool*  $l$  is set to 256, which means the overall size of *keypool* is 512 bytes. Note that  $a, b$  are both 8-bit numbers. Simulation results in Fig. 4 show that when  $t = 1$ , it takes 1,876.8 rounds to read the tag. When  $t = 2$ , it takes 277.8 rounds. We eliminate the result of  $t = 0$ , which should take 65,536 rounds. This is a special case of the basic

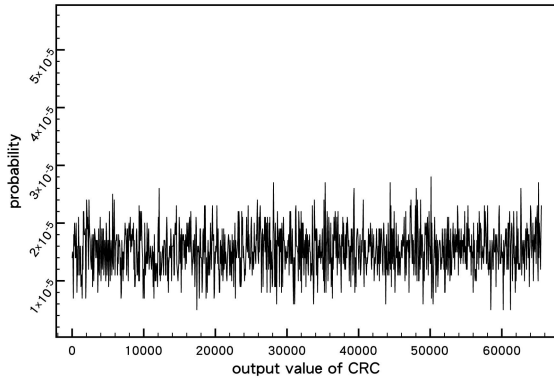


Fig. 5. The distribution of CRC-16 with different length of input.

version and should be implemented by linear search. If the candidate set is too small, sometimes it takes more rounds to identify the tag by Algorithm 1 than by linear search. The situation happens less in the complete version.

### 3.4 CRC-16 Distribution

Previously proposed CRC-based protocols tend to assume that CRC distribution is uniform. But this claim is not accurate since there are still relatively high probabilities of certain values. The following graph (Fig. 5) shows the distribution of the output of a CRC-16 function. The input length is arbitrary and the output range in decimal is from 0 to 65,535. Given an  $l$ -word-long string as the *keypool* and two random values  $(a, b)$ , the probability is calculated by the number of accumulated residues over the number of tests. The result shows that the highest probability may be twice as large as the lowest one. It explains the fact that if the bias in probability  $p$  is larger, the candidate set shrinks faster.

### 3.5 Complete Version: Gen2<sup>+</sup>

In order to boost the reading speed, it is obviously helpful if the tag can provide extra bits of information about itself to the back-end database. We hereby introduce a new approach by applying extra bits in Step 2 of Fig. 2. The last two bits are now considered as a check message, which is constructed by taking XOR operation on the two LSBs of the  $a$ th word and the  $b$ th word of the *keypool*. That is,

$$check = (keypool[a] \oplus keypool[b]) \wedge 0003_h. \quad (8)$$

Message 2 of Fig. 2 becomes  $(a, b, check)$ , which means  $a, b$  are now both 7-bit address only. The overall length of message 2 remains 16 bits. It is possible to use more bits to accelerate the reading speed furthermore. Assume that the tag provides  $x$  bits of extra information, then  $(1 - 1/2^x)$  candidates are pruned away every round. The number of rounds required is now calculated by the following equation:

$$q_{complete} \approx \frac{-\log N}{\log(\frac{1}{2^x} * (1 - P^x))}. \quad (9)$$

The complete version shows an improvement in reading speed. Due to the bit length of  $a$  and  $b$ , the largest size of the *keypool* can only be 128 words, which is different from Fig. 4. The following figure shows the number of rounds required for the complete version with different lengths of *keypool*. When  $l = 128$  and  $t = 0$ , it needs about nine rounds to finish

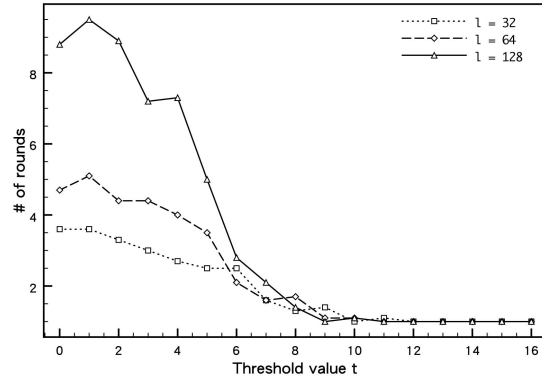


Fig. 6. Simulation result of complete version: relations between threshold value  $t$  and number of rounds with different length  $l$ .

the authentication process. This makes a great difference from the basic version. Note that if  $l = 64$ , where  $a, b$  are both 6-bit long, one dummy bit is added to the MSB of both  $a$  and  $b$ . In this way, the total length of message 2 is still 16 bits.

### 3.6 Tag-to-Reader Authenticity

After the reader has proved itself to the tag, the reader can check the validity of the tag by reading the whole *keypool* and compare it with the record in the database. If the attacker tries to recover the whole *keypool* from previous captured sessions, the reader can update the tag's *keypool* by memory write.

## 4 SECURITY ANALYSIS

We classify readers into four types based on their behavior:

- *Associate reader*: The reader is legal and works with a back-end database, which stores the information of the tag that it is going to read.
- *Semifriendly reader*: A reader that is legal and works with a back-end database, but has no information of the tag.
- *Malicious active reader*: A powerful device that has no information of the tag, but it can actively interact with the tag and tries to trace it.
- *Malicious passive logger*: A small device that logs all the RF signals and tries to obtain the information of the tag or identifies the tag from logged data.

In the following, we analyze how our protocol performs against these attacks.

### 4.1 Against Tracing Attack

A tracing attacker is the most powerful attacker who has both a "malicious active reader" and several "malicious passive" loggers. The goal of the attack is to discover the presence of a specific tag. The attacker actively scans the tag from a far distance and logs all the RF signals by the small device near the tag. If the tag ever replies the same message twice, such as the same TID, it can be traced. The associate reader with the knowledge of *keypool* database has the advantage of creating a possible value  $ck'$  for authentication. As for the attacker, there are two options. The attacker may choose a random value of  $ck'$  and interact with the tag every round, or he may choose a fixed value of  $ck'$  every round.

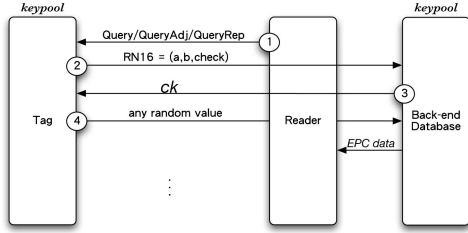


Fig. 7. A simple way to avoid eavesdropping near a legitimate reader.

Note that in some cases, guessing a fixed value will never succeed. We will leave the discussion to the Appendix.

In the basic version, if the tag does not respond, the attacker learns that this  $ck'$  is not acceptable by the tag. But we have both  $a$  and  $b$  changes every round, it is possible that the  $ck'$  that is not acceptable in this round will be accepted in the next round. Therefore, the most effective way for the attacker is random guessing. The number of rounds required for an attacker to succeed through random guessing can be estimated by the following equation:

$$q_{attack} \approx \frac{2^{16}}{\sum_{i \geq 0} \binom{16}{i}}. \quad (10)$$

The attacker requires around 3,855 rounds to succeed when the threshold value of  $t$  is set to 1, and it takes a legal reader 1,800 rounds to succeed.

However, in the complete version, the attacker can learn extra bits from *check* of the second message in Fig. 2. Assume that the attacker records every failed session. In the following rounds, once the tag challenges the same  $(a, b, check)$ , the attacker knows the related  $ck'$  has a 3/4 chance not acceptable by the tag, and hence, he tends to guess another value. Due to the length of  $a$  and  $b$  in the complete version, which is 7 bits, the probability of  $(a_i, b_i) = (a_j, b_j)$  for  $i \neq j$  is  $1/2^{14}$ . If the attacker has recorded near  $2^{14} = 16,384$  failed sessions, he may be able to trace the tag.

To emphasize the difference, a legal reader requires less than 15 rounds to succeed in the complete version.

## 4.2 Against Skimming Attack

A skimming attacker has a “malicious passive logger.” The goal of the attack is to filter out information about the tag from logged RF signals. The attacker is an eavesdropper who does not want to expose himself. It is a trivial tracing case if there exists only one tag in the public area under the attacker’s surveillance. If there exists more than one tag, each *Gen2+* tag will reply a 16-bit message if it is accidentally queried by a “semifriendly” reader. Only the “associate reader” who wants to read the tag will continue the protocol; all “semifriendly” readers will ignore the 16-bit message as if the protocol fails. In this way, these *Gen2+* tags become blocker tags similar to passive-jamming tags in [15].

## 4.3 Against Tag Spoofing Attack

A spoofing attacker is similar to the tracing attacker, but has a different goal. The goal of the spoofing attack is to masquerade as a valid tag once.

In our protocol, there is no checking on the validity of the tag from Step 1 to Step 4. Hence, the attacker can always

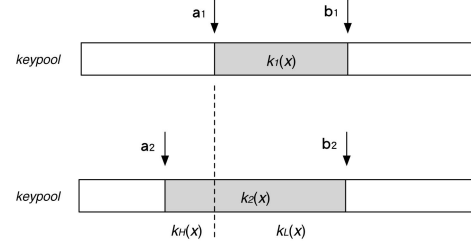


Fig. 8. Cryptanalysis based on two CRC residues.

accept  $ck'$  from the legal reader. Then, he sends the EPC data of the tag he wants to masquerade, or replays previously logged sessions so that the legal reader will take it as an approval.

Another weak point of the protocol is that after the tag approves, it will still transmit EPC data in plaintext. If the attacker happens to appear in the neighborhood of the associate reader and logs all the transmissions, then he is able to obtain its TID.

Here is a simple solution. We modify Step 4 in Fig. 2 like the following graph (Fig. 7). The back-end database, instead of the tag, informs the reader with the tag’s EPC data.

Another possible solution is to have the tag send a hashed value instead of TID, which is known as the hash-lock approach. The tag does not have to do the hash function by itself. The hashed value is computed by the server and written onto the tag before deployment. Still, the attacker can replay the hashed value to spoof the server. As a result, to achieve the best security, the server needs to update the tag every round. We do not include this approach in our protocol due to inconvenience. Instead, we provide a way to authenticate the tag.

In order to prevent the spoofing attack, a legal reader reads out the whole keypool. If the attacker wants to spoof the back-end database, he has to obtain the keypool first. Furthermore, there is a second defense line, which is the 32-bit access password (PIN). Getting the PIN cannot be done easily through interactions because it never appears in any protocol flow.

## 4.4 Against Tag Cloning Attack

A cloning attacker wants to make a “hard copy” on the tag. The goal is to spoof the server forever, which is different from a spoofing attacker. To make a complete clone, the attacker needs to know the whole *keypool* which is a  $(16 * l)$ -bit-long random string and also the 32-bit PIN.

It can be seen that our protocol mainly relies on the computation of CRC functions. Here, we give an example of cloning attack to show that the best way for an attacker to break the protocol is still random guessing. First, in the basic version, assume that the attacker has logged two sessions, denoted as  $(a_1, b_1, ck_1)$  and  $(a_2, b_2, ck_2)$ , such that  $b_1 = b_2$ . Let  $k_1(x)$  represent the polynomial of  $k[a_1 : b_1]$  and  $k_2(x)$  that of  $k[a_2 : b_2]$ , as shown in Fig. 8.

For simplicity, we assume the length of  $k_2(x)$  is longer than  $k_1(x)$ . If we align  $k_1(x)$  and  $k_2(x)$ , then  $k_2(x)$  is separated at  $a_1$  into two polynomials  $k_H(x)$  and  $k_L(x)$ . If both  $ck_1$  and  $ck_2$  contain no error bit in both sessions,

TABLE 1  
Complexity of Security

	<i>Gen2</i> <sup>+</sup>	Juels's[23]	Duc's[32]	Li's[26]	M <sup>2</sup> AP[27]	Juels's[25]	Chien's[29]
Tracing	$O(2^{14})$	†	†	$O(l^2)$	†	–	†
Skimming	–	†	$O(2^{16})$	†	†	–	$O(2^{2l})$
Spoofing	$O(2^{16})$	$O(q_J)$	†	†	†	$O(2^{2L})$	†
Cloning	$O(2^{32})$	$O(q_J)$	$O(2^{32})$	†	†	$O(2^{3kLm})$	$O(2^{2l})$

$q_J$  size of PINSET – not possible † constant time  
 $l$  bit length of pseudonym  $m$  security parameter  
 $L$  bit length of one-time pad  $k$  number of pseudonyms

which means the Hamming distance equals 0, the following equations hold:

$$ck_1 \oplus ck_2 = CRC(k_1(x)) \oplus CRC(k_2(x)) \quad (11)$$

$$= CRC(k_1(x) \oplus k_2(x)) \quad (12)$$

$$= CRC(k_H(x) * x^{|a_1-b_1|} \oplus k_L(x) \oplus k_L(x)) \quad (13)$$

$$= CRC(k_H(x) * x^{|a_1-b_1|}) \quad (14)$$

The attacker then generates a lookup table for all 65,536 cases of  $k_H(x)$  to see which one equals the value of (11). In this way, the attacker obtains  $k_H(x)$ , i.e., the keypool bit from  $a_2$  to  $a_1$ . After performing the same trick several times, the attacker may recover the whole keypool piece by piece.

To succeed, first the attacker must have at least two sessions such that the second index of message 2 is the same. The length of the keypool is  $l$  and the index value  $b$  is uniformly chosen from  $l$ . By the birthday paradox, the probability of finding a collision is the following, where  $x$  represents the number of logged rounds

$$p(x) = 1 - \frac{l!}{(l-x)!} \quad (15)$$

The probability reaches 53 percent when  $x \geq 20$  with  $l = 256$  in the basic version. The probability reaches 52 percent when  $x \geq 14$  with  $l = 128$  in the complete version. Note that the probability here is to recover a piece of the keypool, not the whole keypool.

Fortunately, this is the best case for the attacker. If  $ck'$  does not equal  $ck$  exactly, for example, the Hamming distance is 2. The attacker cannot locate which two bits in  $ck'$  are different from  $ck$ . Thus, the attacker cannot use this  $ck'$  to derive the keypool by (11). There is still a chance that the attacker is really lucky. As a result, we suggest to perform a key update every 14 successful rounds.

## 5 DISCUSSION

### 5.1 Private Mode and Public Mode

End users demand their handful items to be switched to private mode and switched back whenever they want. Under such a scenario, our scheme provides flexibility between the private mode and the public mode. For those items, which need to be invisible from the outside world, users switch them into the private mode by setting the threshold value  $t$  to 0. Then, the tag can avoid tracing and skimming attacks. They can pass through attackers' effective reading range in a public domain. In addition, these tags in

their private mode react to the malicious reader one after another by sending nonsense just like passive-jamming tags. As for the public mode, we set  $t$  to a higher value. Then, the reading speed increases rapidly (shown in Fig. 6). However, this also exposes the tag to the environment. This is configurable according to user's preference.

### 5.2 Scalability

From (7) and (9), the number of rounds is proportional to  $\log N$ . Our scheme is scalable no matter how large the database is. The number of rounds increases very slightly for both versions when  $N$  grows. The computational cost to the back-end server is mainly CRC computations. The number of CRC operations required to read a tag is  $O(N * q_R)$ , where  $q_R$  is the number of rounds.

### 5.3 Comparisons

Table 1 shows comparisons between different proposals that are similar to ours. The security level is evaluated by the resistance to four major attacks. The size of challenge set in Juels' protocol is denoted as  $q_J$ . Parameter  $l$  in [26] represents the length of shared pseudonym,  $L$  is the length of each one-time pad in [25], and parameter  $M$  in [27] is the length of the key. It is suggested that  $M = \{8, 16, 32, 64, 96\}$ .  $m$  denotes the security parameter, which is a small value or zero in [25]. The symbol "–" means the attack is not possible or not likely to happen. "†" in Table 1 means there exists a known attack by which the attacker can reach his goal in constant time.

Both Duc's protocol and ours use the access PIN for tag-to-reader authentication, and hence, the time complexity to resist cloning attack is  $O(2^{32})$ .

We can see that [25] is also a good solution. Tracing and skimming attacks are not possible because the one-time pad is discarded every round. However, it requires key update every round. In our protocol, we suggest to update the keypool every 14 successful rounds, due to the attack introduced in Section 4.4.

Table 2 counts the number of operations required among these protocols on different entities.  $N$  is the number of tags in database.  $C, R, X$  stand for CRC operation, PRNG, and basic operations, respectively. Basic operations are XOR, AND, OR, and modular summation.  $q_R$  is to estimate the number of rounds required for legitimate readers. Though our scheme seems to create more overheads,  $q_R$  is proportional to  $\log N$ , and hence, the computational loads will not grow fast when supporting large-scale databases.

TABLE 2  
Comparison of Computational Loads

	Gen2 <sup>+</sup>	Juels's[23]	Duc's[32]	Li's[26]	M <sup>2</sup> AP [27]	Juels's[25]	Chien's[29]
Server/Verifier	$O(N)(q_R C)$	$O(N)$	$O(N)(C)$	$O(N)(2X)$	$O(N)(6X + 2R)$	$O(N)$	$O(N)$
Reader	$O(q_R)$	$O(q_J)$	$O(1)$	$O(1)$	—	—	—
Tag	$q_R(C + R)$	$O(q_J)$	$2C + 2X + R$	$2X + 2R$	$O(N)(25X)$	$2LmX$	$14X$

$N$  number of tags       $q_J$  size of PINSET      — not involved  
 $L$  bit length of one-time pad       $C$  CRC operation       $R$  PRNG operation  
 $X$  Basic operation       $q_R$  number of rounds

## 6 CONCLUSION

We proposed a lightweight authentication protocol based on Gen2 to resist various attacks. The proposed tag uses no cryptographic function, and hence, is suitable for low-cost RFIDs. Without changing the protocol flow of Gen2, the existing reader can read both Gen2 tags and Gen2<sup>+</sup> tags. Gen2<sup>+</sup> provides sufficient security level for real-world settings. We analyzed the number of rounds required and the period of key update for practical deployment.

## APPENDIX A

### SELECTION OF PARAMETERS

According to Gen2 standard, one word equals two bytes. When a Gen2<sup>+</sup> tag is queried, the input space of the CRC function is  $l^2$ , where  $l$  is the word length of *keypool*. The size of all acceptable values of  $ck'$  (with threshold value  $t$ ) is  $l^2 * T$ , where  $T = \sum_{i \geq 0} \binom{16}{i}$ . It tells us if  $t = 1$  and  $l = 32$ , then the size will be 17,408, which is less than 65,536. This means there are about 40,000 values among them that will never be accepted. If a naive attacker keeps guessing a fixed value of  $ck'$ , he might never succeed. When  $t = 2$ , this size extends to 65,536; hence, a naive attacker and a random guessing attacker have the same success rate. We denote this size over 65,536 as a *fraction*. The protocol is safer if the *fraction* is higher.

If  $l$  is larger, the distribution of CRC is more uniform and makes *fraction* = 1 no matter what the threshold value is. But this also consumes more memory space of the tag. We do not want to occupy too much space and make the tag impractical. When *fraction* is too small, an attacker can take the advantage of a failed  $ck'$  to exclude the region, where  $ck'$  is the center of it. To sum up, the suggestion is to have  $l$  smaller but each acceptable region shares at least several points so that the attacker cannot exclude this region.

The best way to achieve this criterion is to have *fraction* = 1 no matter what  $t$  value is, i.e., optimal  $l$  should be 256 to guarantee the quest.

Gen2 tag is based on a 16-bit word, and each address is represented in 8-bit exactly. This makes it natural when we let  $(a, b)$  denote two addresses of the memory bank. Note that *keypool* in our scheme is of  $l$  words, which are  $2l$  bytes, not just  $l$  bytes. Let us use optimal  $l = 256$ , which is the basic version of our protocol that needs 512 bytes of the tag from address  $00h$  to  $FFh$ ; in other words, all of the user's memory bank is occupied. On the other hand, the reading speed is slower than the complete version. We take both  $t$  and  $l = 32$

into consideration to see how they affect the security level of Gen2<sup>+</sup>.

When  $t = 0$ , *fraction* =  $l^2 * (1) / 65,536 = 256 / 65,536$ . This fraction is not very secure, but the  $t$  value requires an exact match. When  $t = 1$ , *fraction* =  $l^2 * (1 + 16) / 65,536 = 17,408 / 65,536$ . Still, the fraction is low. When  $t = 2$ , *fraction* = 1 because  $l^2 * (1 + 16 + 120) > 65,536$ . For those  $t \geq 2$ , we only need to consider the impact of the threshold value.

Our suggestion is  $l = 128$  and  $t = 1$ , because when  $t = 0$ , *fraction* =  $16,384 / 65,536$ . When  $t = 1$ , *fraction* =  $l^2 * (1 + 16) / 65,536 > 65,536$ .

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported in part by the National Science Council, Taiwan, under Contracts NSC 97-2745-P-001-001 and NSC 97-2221-E-007-055-MY3.

## REFERENCES

- [1] S. Weis, "Security and Privacy in Radio-Frequency Identification Devices," master's thesis, Mass. Inst. of Technology (MIT), May 2003.
- [2] EPCglobal, Inc., <http://www.epcglobalinc.org/>, 2005.
- [3] A. Juels, "RFID Security and Privacy: A Research Survey," manuscript, RSA Laboratories, Sept. 2005.
- [4] S. Weis et al., "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," *Proc. First Int'l Conf. Security in Pervasive Computing (SPC '03)*, Mar. 2003.
- [5] H. Lee and J. Kim, "Privacy Threats and Issues in Mobile RFID," *Proc. First Int'l Conf. Availability, Reliability and Security (ARES '06)*, Apr. 2006.
- [6] D. Molnar and D. Wagner, "Privacy and Security in Library RFID: Issues, Practices, and Architectures," *Proc. 11th ACM Conf. Computer and Comm. Security (CCS '04)*, Oct. 2004.
- [7] S. Sarma, S. Weis, and D. Engels, "RFID Systems and Security and Privacy Implications," *Proc. Fourth Int'l Workshop Cryptographic Hardware and Embedded Systems (CHES '02)*, Aug. 2002.
- [8] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to 'Privacy-Friendly' Tags," *Proc. Radio Frequency Identification (RFID) Privacy Workshop*, Nov. 2003.
- [9] T. Dimitriou, "A Lightweight RFID Protocol to Protect against Traceability and Cloning Attacks," *Proc. First IEEE Conf. Security and Privacy for Emerging Areas in Comm. Networks (SecureComm '05)*, Sept. 2005.
- [10] X. Gao et al., "An Approach to Security and Privacy of RFID System for Supply Chain," *Proc. IEEE Int'l Conf. E-Commerce Technology for Dynamic E-Business (CEC-East '04)*, Sept. 2004.
- [11] C.C. Tan, B. Sheng, and Q. Li, "Severless Search and Authentication Protocols for RFID," *Proc. Fifth IEEE Int'l Conf. Pervasive Computing and Comm. (PerCom '07)*, Mar. 2007.
- [12] G. Tsudik, "YA-TRAP: Yet Another Trivial RFID Authentication Protocol," *Proc. Fourth IEEE Int'l Conf. Pervasive Computing and Comm. (PerCom '06)*, Mar. 2006.

- [13] G. Avoine and P. Oechslin, "A Scalable and Provably Secure Hash Based RFID Protocol," *Proc. Third IEEE Int'l Workshop Pervasive Computing and Comm. Security (PERCOMW '05)*, Mar. 2005.
- [14] D. Henrici and P. Müller, "Hash-Based Enhancement of Location Privacy for Radio-Frequency Identification Devices Using Varying Identifiers," *Proc. First IEEE Int'l Workshop Pervasive Computing and Comm. Security (PerSec '04)*, Mar. 2004.
- [15] A. Juels, R. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03)*, Oct. 2003.
- [16] M. Rieback, B. Crispo, and A. Tanenbaum, "RFID Guardian: A Battery-Powered Mobile Device for RFID Privacy Management," *Proc. Australasian Conf. Information Security and Privacy (ACISP '05)*, July 2005.
- [17] A. Juels and J. Brainard, "Soft Blocking: Flexible Blocker Tags on the Cheap," *Proc. Workshop Privacy in the Electronic Soc. (WPES '04)*, Oct. 2004.
- [18] L. Bolotnyy and G. Robins, "Physically Unclonable Function-Based Security and Privacy in RFID Systems," *Proc. Fifth IEEE Int'l Conf. Pervasive Computing and Comm. (PERCOM '07)*, Mar. 2007.
- [19] G. Ateniese, J. Camenisch, and B. De Medeiros, "Untraceable RFID Tags via Insubvertible Encryption," *Proc. 12th ACM Conf. Computer and Comm. Security (CCS '05)*, Nov. 2005.
- [20] I. Kim, B. Lee, and H. Kim, "Privacy-Friendly Mobile RFID Reader Protocol Design Based on Trusted Agent and PKI," *Proc. 10th IEEE Int'l Symp. Consumer Electronics (ISCE '06)*, June 2006.
- [21] I. Kim, B. Lee, and H. Kim, "Privacy Protection Based on User-Defined Preferences in RFID System," *Proc. Eighth Int'l Conf. Advanced Comm. Technology (ICACT '06)*, Feb. 2006.
- [22] I. Vajda and L. Buttyán, "Lightweight Authentication Protocols for Low-Cost RFID Tags," *Proc. Second Workshop Security in Ubiquitous Computing (Ubicomp '03)*, Oct. 2003.
- [23] A. Juels, "Strengthening EPC Tags against Cloning," manuscript, RSA Laboratories, Mar. 2005.
- [24] Y.C. Chen, W.L. Wang, and M.S. Hwang, "RFID Authentication Protocol for Anti-Counterfeiting and Privacy Protection," *Proc. Ninth IEEE Int'l Conf. Advanced Comm. Technology (ICACT '07)*, Feb. 2007.
- [25] A. Juels, "Minimalist Cryptography for Low-Cost RFID Tags," *Proc. Fourth Int'l Conf. Security in Comm. Networks (SCN '04)*, Sept. 2004.
- [26] Y.Z. Li et al., "Security and Privacy on Authentication Protocol for Low-Cost RFID," *Proc. Int'l Conf. Computational Intelligence and Security (CIS '06)*, Nov. 2006.
- [27] P. Peris-Lopez et al., "M2AP: A Minimalist Mutual-Authentication Protocol for Low-Cost RFID Tags," *Proc. Third Int'l Conf. Ubiquitous Intelligence and Computing (UIC-06)*, Sept. 2006.
- [28] H. Chabanne and G. Fumaroli, "Noisy Cryptographic Protocols for Low-Cost RFID Tags," *IEEE Trans. Information Theory*, vol. 52, no. 8, pp. 3562-3566, Aug. 2006.
- [29] H.Y. Chien, "SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity," *IEEE Trans. Dependable and Secure Computing*, vol. 4, no. 4, pp. 337-340, 2007.
- [30] D.H. Choi, T.S. Kim, and H.W. Kim, "Privacy Protection for Secure Mobile RFID Service," *Proc. First IEEE Int'l Symp. Wireless Pervasive Computing (ISWPC '06)*, Jan. 2006.
- [31] A. Juels, "A Bit of Privacy," <http://www.rfidjournal.com/article/view/1536/1/82>, May 2005.
- [32] D.N. Duc et al., "Enhancing Security of EPCglobal Gen2 RFID Tag against Traceability and Cloning," *Proc. Third Conf. Soft Computing and Intelligent Systems (SCIS '06)*, Jan. 2006.
- [33] H.Y. Chien and C.W. Huang, "A Lightweight RFID Protocol Using Substring," *Proc. IFIP Int'l Conf. Embedded and Ubiquitous Computing (EUC '07)*, Dec. 2007.
- [34] H.M. Sun, W.C. Ting, and K.H. Wang, "On the Security of Chien's Ultralightweight RFID Authentication Protocol," *Cryptology ePrint Archive*, <http://eprint.iacr.org/>, Feb. 2008.
- [35] M. Bátorás et al., "Passive Attack Against the M2AP Mutual Authentication Protocol for RFID Tags," *Proc. First Int'l Workshop RFID Technology (EURASIP)*, Sept. 2007.



**Hung-Min Sun** received the BS degree in applied mathematics from National Chung-Hsing University in 1988, the MS degree in applied mathematics from National Cheng-Kung University in 1990, and the PhD degree in computer science and information engineering from National Chiao-Tung University in 1995, respectively. He was an associate professor in the Department of Information Management, Chaoyang University of Technology from 1995 to 1999, and the Department of Computer Science and Information Engineering, National Cheng-Kung University from 1999 to 2002. Currently, he is a professor in the Department of Computer Science, National Tsing Hua University. He has published more than 100 international journal and conference papers. He was a program cochair of the 2001 National Information Security Conference and a program committee member of the 1997 and 2005 Information Security Conference, the 2000 Workshop on Internet and Distributed Systems, the 2001, 2002, and 2005 Workshop on the 21st Century Digital Life and Internet Technologies, the 1998-1999, 2002-2004, and 2006-2007 National Conference on Information Security, ACISP 2004, NCS 2001, ICS 2002, ITRE 2005, and NCS 2007. His research interests include information security, wireless network security, cryptography, and multimedia security.



**Wei-Chih Ting** received the BS degree in computer science from National Chiao Tung University in 2004. He is currently working toward the PhD degree in computer science at National Tsing Hua University. His research interests are in the areas of RFID security, system security, network security, and cryptography.

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).