Chapter 15

Enhancing Security of Class I Generation 2 RFID against Traceability and Cloning

Dang Nguyen Duc[1], Hyunrok Lee[1], and Kwangjo Kim[1]

[1]CAIS Lab (R504), Information and Communication University (ICU). 119 Munjiro, Yuseong-gu, Daejeon, 305−732, Republic of Korea. {nguyenduc, tank, kkj}@icu.ac.kr

Abstract. In this whitepaper, we present a synchronization-based communication protocol for EPCglobal Class-1 Gen-2 RFID devices. The Class-1 Gen-2 RFID tag supports only simple cryptographic primitives like Pseudo-random Number Generator (PRNG) and Cyclic Redundancy Code (CRC). Our protocol is secure in a sense that it prevents the cloned tags and malicious readers from impersonating and abusing legitimate tags, respectively. In addition, our protocol provides that each RFID tag emits a different bit string (pseudonym) or meta-ID when receiving each and every reader's query. Therefore, it makes tracking activities and personal preferences of tag's owner impractical to provide the user's privacy.

Keywords: RFID, authentication, anti-counterfeiting, CRC.

1 Introduction

Radio Frequency Identification (RFID) technology is envisioned as a replacement for Barcode counterpart and expected to be massively deployed in the coming years. The advantages of RFID system over barcode system include many-to-many communication (i.e., one tag can be read by many readers and one reader can read many tags at once), wireless data transmission (versus optical communication, thus requiring light-of-sight, in case of Barcode) and its computing nature. Those major benefits enable much wider range of applications including: supply chain management, library management, anti-counterfeiting banknotes, smart home appliances, etc.

Despite many prospective applications, RFID technology also poses several security and privacy threats which could harm its global adoption. Ironically, the security weakness of RFID technology comes from the most basic operation of a RFID tag, that is to wirelessly release a unique and static bit string known as Electronic Product Code (EPC for short) identifying the object associated with the

tag upon receiving a query request from a reader. Using the unique EPC as a reference, one (equipped with a compatible reader) can track the moving history, the personal preferences and the belongings of a tag's holder. Even worse, absence of secure authentication results in revealing EPC to malicious readers (referred to as skimming attack). Once capturing EPC, an attacker can duplicate genuine tags and use the cloned tags for its malicious purposes. A natural solution to the aforementioned security problems is to employ cryptographic protocol in the RFID system. Unfortunately, the cost of manufacturing a tag has to be extremely low, *e.g.*, less than 30 cents (according to RFID journal, one RFID tag is expected to cost 5 cents by 2007). Therefore, the computationally intensive security protocols widely known in cryptographic literature cannot be incorporated into a small chip with tightly constrained computational power (at least in the foreseeable future).

To foster and publicize RFID technology, standardization is certainly important in order to allow interoperability at large scale. And the most viable standard is proposed by EPCglobal Inc. The latest RFID standard ratified by EPCglobal is named EPCglobal Class-1 Gen-2 RFID specification version 1.09 (Gen-2 RFID for short). We briefly summarize properties of a Gen-2 RFID tag as follows:

- Gen-2 RFID tag is passive, meaning that it receives power supply from readers.
- Gen-2 RFID tag communicates with RFID readers in UHF band (800–960 MHz) and its communication range can be up to 2 ~ 10 m.
- Gen-2 RFID tag supports on-chip *Pseudo-Random Number Generator* (PRNG) and *Cyclic Redundancy Code* (CRC) computation.
- Gen-2 RFID's privacy protection mechanism is to make the tag permanently unusable once it receives the kill command with a valid 32-bit kill PIN (*e.g.*, tag can be *killed* at the point-of-sale).
- Read/Write to Gen-2 RFID tag's memory is allowed only after it is in secure mode (*i.e.*, after receiving access command with a valid 32-bit access PIN).

We would like to note that that privacy protection mechanism suggested in the specification is arguably over-killed. In many scenarios like tracking animal, smart home appliances, *etc.*, the tag should never be killed. Furthermore, in case of supply chain management, the tag is likely to be helpful in many ways after items being purchased (*e.g.*, for warranty purpose). Therefore, in designing a new protocol, we should avoid this kind of mechanism and, hopefully make a new use for the *kill* PIN. For the *access* PIN, we want to stress that it is useless from security point of view since the 16 bits of PIN is XORed with a 16-bit pseudo-random number sent by the tag in a session. Just by eavesdropping the 16-bit pseudo-random number and the XORed PIN, an attacker can easily recover the access PIN. Losing the access PIN is very dangerous because it allows a malicious reader to read/write the entire memory of a tag.

Lots of researchers have proposed several lightweight cryptographic protocols, designed specifically for low-cost RFID tags, to defend against security and privacy threats. Most of the proposed solutions make use of a hash function [7, 8, 9, 10, 13]. Even though the hash function can be efficiently implemented on low-power hardware, it is still beyond current capability of low-cost RFID tag. In particular, current EPCGlobal Class-1 Gen-2 RFID specification does not ratify

any cryptographic hash function like MD5 and SHA-1. Thus, we need to look for another solution which should use only the available functionalities of current RFID standards. In fact, Juels [3] suggested such a scheme to prevent the legitimate tags from being cloned. However, his protocol does not take eavesdropping and privacy issues into consideration, and thus provides no protection against privacy invasion and secret information leakage. In this paper, we present another scheme targeting most of security features for a RFID system including authentication, traffic encryption, and privacy protection as well. Last but not least, we think that a RFID reader should never be fully trusted (but a legitimate one acts honestly) because it is a portable device and would be used by many people. The only trusted party is a RFID system would be the backend server and all the secrets are kept only at tags and backend server's database. In addition, RFID reader should not be able to learn any secret information including PIN and EPC itself from data called *meta-ID* sent by a tag. The meta-ID should be forwarded to the backend server and backend server can retrieve detail object information keyed by that meta-ID. The advantage of this approach is as follows:

- *Accountability* and *Access Control*: The approach enables easy accountability and access control because the backend server is in charge of looking up object information so it can decide who can get which information as well as some statistics (*e.g.*, how many times of object is queried).
- *Reader-to-Tag Authentication*: It is obvious that tag querying will happen most frequent. And because reader needs to contact the backend server in order to learn useful information about an object, there is no need for Reader-to-Tag authentication in this case. Instead, we can require reader to authenticate to the backend server before sending a meta-ID.

2 Some Background

In this section, we will discuss briefly two primitives, namely *Pseudo-random Number Generator* (PRNG) and *CRC Checksum*, which we are going to use in our authentication protocol.

2.1 Pseudo-random Number Generator

When designing a security protocol, one often faces problems of the following forms:

- A value should not used more then twice, *e.g.*, a challenge in a challenge-response authentication protocol.
- A value should not be predictable, *e.g.*, a secret key.

In such cases, we need a randomly chosen number, more specifically a random number generator. Ideally, a random number is a number which is drawn from a pool of n numbers with probability exactly n^{-1}. In other words, each number

among n numbers has equal chance of being chosen. However, it is impossible to realize such truly random number generator. Instead, people come up with close approximation ones (in computational sense) and call them pseudo-random number generator. In a common setting, a PRNG is modelled as a deterministic function whose next output is computed from previous outputs (usually the last output). The output sequence starts from a (randomly chosen) seed number. The security strength of a PRNG depends on the period and probability distribution of the output sequence. A popular class of PRNG has the congruential form of $x_i = ax_{i-1} + b \bmod N$ where x_0 is the seed number and a, b and N are PRNG's parameters. Another popular class of PRNG is based on *Linear Feedback Shift Register* (LFSR) which could be efficiently implemented on a low-cost RFID tag. In this paper, we will use a PRNG to share a new session key between RFID tag and reader for each and every session. In the EPCglobal Class-1 Gen-2 specification, a Gen-2 RFID tag is capable of generating 16-bit pseudo-random number with the following properties (although the detail algorithm is not given):

- The probability that a single 16-bit number j is drawn shall be bounded by $0.8 \times 2^{-16} < \mathbf{Prob}[j] < 1.25 \times 2^{-16}$.
- Among a number of 10,000 tags, the chance that any two tags simultaneously generate the same 16-bit pseudo-random number is less than 0.1%.
- The probability of guessing the next pseudo-random number generated by a tag is less than 0.025% under the assumption that all previous outputs are known to an attacker.

Since Gen-2 standard requires only 16-bit pseudo-random number, the security margin, *i.e.*, success probability of adversary) of a security protocol using such PRNG is usually bounded by 2^{-16}. We suggest that Gen-2 standard should support 32-bit PRNG to take full advantage of 32-bit PIN currently supported by Gen-2 specification. Otherwise, XORing two halves of a 32-bit PIN with the same 16-bit nonce in one session provides no better security than using the full 16-bit PIN.

2.2 CRC Checksum

In our proposed protocol, we also make use of checksum code to provide security and resolve possible collisions at backend server's database. A checksum code is often used to check the integrity of data being sent or received. The popular cryptographic checksum codes are cryptographic hash function, MAC and HMAC. In this paper, we will make use of a well-known, efficient (yet less cryptographically strong) checksum algorithm, namely CRC. This kind of checksum code is currently ratified in EPCglobal Class-1 Gen-2 RFID specification, version 1.09. CRC algorithm treats binary data as a polynomial whose coefficients are in GF(2) (*i.e.*, 1 or 0). For n-bit CRC, an irreducible and primitive polynomial of n degree (called CRC polynomial) over GF(2) should be chosen. The CRC checksum is then computed as a remainder of the division of the original data by the CRC polynomial. For example, the polynomial $x + 1$ is a CRC polynomial resulting in 1-bit CRC checksum equivalent to parity bit. In EPCglobal Class-2

Gen-2 specification, a 16-bit CRC checksum is used to detect error in transmitted data and the corresponding CRC polynomial of degree 16 is $x^{16}+x^{12}+x^5+1$. Even though calculating CRC checksum involves polynomial division, it actually can be implemented very efficiently by using a *shift register* in hardware and *look-up table* in software. Generally, if CRC is setup properly, we can expect that the probability of collision on n-bit CRC checksum is about 2^{-n}. Of course, we can always use cryptographically secure checksum algorithms as well.

Note that, CRC checksum of $0^*\|s$ where s is some bit string is the same as CRC checksum of s itself. To avoid this, we should specifically require a bit string s to start with a bit 1.

3 A New Authentication Protocol for Gen-2 RFID Specification

Main Idea. We first think of protecting data transmitted between the tag and reader against eavesdropping. The obvious way is to utilize encryption/decryption and the most simple encryption function that we are aware of is XORing which is popularly used in a stream cipher). The problem now turns to key management issue: that is to ensure that a new encryption key is used in every session. Solving this issue turns out to be a solution to privacy protection as well since RFID tag can XOR EPC with different key in every session, thus, prevent malicious readers from tracking the tag. And we suggest that the simplest, yet most efficient way of key sharing in this scenario is to use the same PRNG with the same seed at both RFID tag side and backend server side (see Figure 1). The session key can be computed by generating a new pseudo-random number from current session key after every session. This computation is required to be done at both RFID tag and reader/backend server in a synchronous way. Otherwise, subsequent traffic cannot be understood by both sides.

The next security problem that we need to solve is authentication. We argue that, in most cases, a reader just needs to know EPC stored in a tag and then eventually contact the backend server to get/update information about the object carrying the tag. Keeping this in mind, we propose that reader-to-tag authentication can be delegated to tag-to-backend server authentication. More specifically, reader can only receive EPC from RFID tag in an encrypted form. It needs to authenticate itself till backend server first, and then, depending on its privileges,

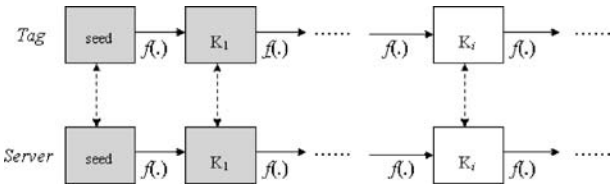

Fig. 1 Using PRNG $f(.)$ to generate session key.

backend server can decide what kind of information to send back to reader (for example, in case of a public reader, only information describing what the referenced object is; and in case of a manufacturer's reader, actual EPC and PIN associated with that tag can be sent). Actual reader-to-tag authentication needs to be carried out when reader wants to access (read/write) other sections of tag's memory bank. To do so, we can use PIN-based approach just like in the original Gen-2 RFID specification.

We also would like to note that, there exists another scheme that allows a reader to be able to decipher EPC without help from backend server for several sessions [10]. We have a different view in this respect. We believe that, in a ubiquitous environment, connectivity is abundant and exercising practical security and simplicity is a key factor to the successful adoption of new technology. In addition, we think that backend server's database can be partitioned in a hierarchical way, thereby reducing overhead at each backend server. This scenario naturally fits in both DNS-like hierarchical structures of EPCglobal Object Naming System (ONS) and real-life situations (for example, each department in a company manages its own inventories, thus, should have its own backend server). We want to stress that our proposed scheme is simple and provides reasonable security strength within the bound of the low-cost RFID tag's functionalities.

Notations. Before describing our protocol in detail, we give the definition of notations that we use in the description of our protocol.

- $f(.)$ – pseudo-random number generator
- CRC(.) – cyclic redundancy check function (produce checksum)
- Ki – secret key at the i-th session
- EPC – Electronic Product Code
- r – random nonce.
- PIN – "access" command password
- T – Tag
- R – Reader
- S – Backend Server

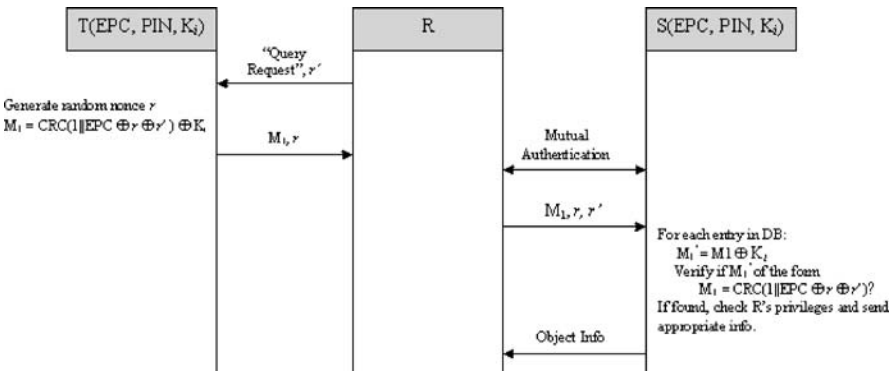

Fig. 2 Tag querying protocol.

The Protocol. There are three sub-protocols including: tag querying protocol, tag access protocol and key updating protocol. During the manufacturing time, a tag is initialized by assigning EPC and other parameters. Then, it chooses a random seed number *seed* and store $K_1 = f(seed)$ into tag's memory and backend server's database entry corresponding to matching EPC. A random PIN (say, *access* PIN defined in Gen-2 specification) is also stored in both tag's memory and backend server database in a similar way. The tag querying protocol is described in the Figure 2.

The tag access protocol is carried out when a reader needs to read/write to tag's memory. In this case, we need to perform actual Reader-to-Tag authentication. The approach is the backend server sends a one-time authentication token to the reader and it uses it to authenticate itself to the tag. The protocol is described as follows:

- $S \rightarrow R$: $M_2 = CRC(1 \| EPC \| PIN \| r) \oplus K_i$
- $R \rightarrow T$: forward authentication token M_2 to T.
- T: Verify $M_2 \oplus K_i = CRC(1 \| EPC \| PIN \| r)$?

The last step of a tag querying or accessing session is to update session key. This step can be optional for non-critical applications but for a security-sesitive application, this should be enforced to guarantee better security. The key updating protocol is described as follows:

- $R \rightarrow T$, S: 'End Session'
- T: $Ki + 1 = f(Ki)$
- S: $Ki + 1 = f(Ki)$

There is might be synchronization issue with the above protocol since a false 'End Session' message might be sent by a malicious reader. To prevent such interference, reader might announce 'End Session' with a token CRC(r' $\oplus$ PIN') where r' is a random nonce broadcasted to Tag with 'Query Request' messae and PIN' is another secret shared between T and legitimate R.

Protocol Analysis. In the querying protocol, the message M_1 is blinded with the session key K_i and therefore we can avoid weak one-way property of CRC checksum. However, CRC checksum is useful for the backend server to search through its database because of its collision-resistant property. If collision does occur due to short length of the checksum, backend server might instruct reader to request for another message M_1. With two different CRC checksums, it is likely that there will be no collision. On the other hand, without knowledge of EPC, PIN and K_i, it is very difficult to construct a message M_1 which can be recognized at backend server. Therefore, the protocol provides tag authentication. As we mentioned before, we require that every input to CRC function should starts with a bit 1. This is to avoid obvious collision attack on CRC.

For reader authentication, Reader-to-Tag authentication is delegated to Reader-to-Backend authentication (which we can use available cryptographic authentication protocol). Actual Reader-to-Tag authentication is achieved by using the token M_2. M_2 is a one-time token because of the session key K_i and the random nounce r.

Table 1 Comparision between Juels' and the proposed protocol

	Juels' Protocol	Our Protocol
Server's complexity	O(n)	O(n)O(CRC)
Reader's complexity	O(q)	O(1)
Tag's complexity	O(q)	2CRC+2PRNG
Tag authentication	YES	YES
Reader authentication	YES	YES
Eavesdropping protection	NO	YES
Privacy protection	NO	YES

Note: n – # of tags; O(CRC) – complexity of CRC; q – number of PIN-test round; Reader-to-Server authentication complexity not counted.

By a the same argument as above, it is difficult to duplicate this token for another session.

Lastly, for privacy concern, the proposed protocol provide privacy protection by randomizing tag's response to each and every query request. In addition, tag never gives out EPC in cleartext, therefore malicous readers have no reference to perform tracking.

A Comparison with Juels' Protocol. We compare our proposed protocol and the one by Juels in the Table 1.

4 Concluding Remarks

We have presented a simple communication protocol for RFID devices, especially EPCglobal Class-1 Gen-2 RFID devices. Our protocol achieves desirable security features of a RFID system including: implicit reader-to-tag authentication, explicit tag-to-reader authentication, traffic encryption and privacy protection (against tracking). Our scheme makes use of only PRNG and CRC which are all ratified in current Gen-2 RFID specification. Moreover, there should be little overhead to adapt our protocol into the Gen-2 RFID specification.

Comparing to Juels' protocol, our suggested protocol offers more security features and better performance at the tag and reader sides. While Juels' protocol requires a tag and a reader to invoke q rounds of communication and PIN testing, our protocol has only one round of communication. We think that reducing computational and communication burden on the RFID tag is very crucial for the sake of the low-cost RFID tag. From security point of view, our scheme is also a more viable solution to the security threats than Juels' scheme. It is because Juels' scheme does not solve the privacy invasion issue which is considered to be the most serious problem faced by RFID technology.

References

1 EPCglobal Inc. Available from: http://www.epcglobalinc.org/
2 EPCglobal Inc.: Class 1 Generation 2 UHF Air Interface Protocol Standard Version 1.09. Available from: http://www.epcglobalinc.org/standards_technology/specifications.html
3 Juels, A.: Strenthening EPC tag against cloning. In: Jakobsson, M., Poovendran, R. (eds.): ACM Workshop on Wireless Security (WiSe) (2005) 67–76
4 Juels, A.: RFID security and privacy: A research survey. In: IEEE Journal on Selected Areas in Communication (2006)
5 Weis, S.: Security and privacy in radio frequency identification devices. In: Master Thesis (2003) Available from: http://theory.lcs.mit.edu/~sweis/masters.pdf
6 Molnar, D., Soppera, A., Wagner, D.: A Scalable, delegatable pseudonym protocol enabling ownership transfer of a RFID tag. In: Preneel, R., Tavares, S. (eds.): Selected Areas in Cryptography (SAC), LNCS, Vol. 3897. Springer-Verlag, Berlin Heidelberg New York (2005) 276–290
7 Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient hash-chain based RFID privacy protection scheme. In: the Proceedings of International Conference on Ubiquitous Computing, Workshop Privacy (2004)
8 Avoine, G., Dysli, E., Oechslin, P.: Reducing time complexity in RFID system. In: Preneel, R., Tavares, S. (eds.): Selected Areas in Cryptography (SAC), LNCS, Vol. 3897. Springer-Verlag, Berlin Heidelberg New York (2005) 291–306
9 Yang, J.: Security and privacy on authentication protocol for low-cost radio frequency identification. In: Master Thesis (2005). Available from: http://caislab.icu.ac.kr/Paper/thesis_files/2005/thesis_jkyang.pdf
10 Avoine, G., Oechslin, P.: A scalable and provably secure hash-based RFID protocol. In: Proceedings of Workshop on Pervasive Computing and Communications Security (2005)
11 NIST: Random number generation and testing. Available from: http://csrc.nist.gov/rng/
12 RFID Journal: Available from: http://www.rfidjournal.com/
13 Dimitriou, T.: A lightweight RFID protocol to protect against traceability and cloning attacks. In: Proceedings of SecureComm'05 (2005)