On the Security of Recently Proposed RFID Protocols

Mete Akgün-[1,2], M. Ufuk Çağlayan[2]

[1]Tübitak UEKAE, 41470, Kocaeli, Turkey
makgun@uekae.tubitak.gov.tr
[2]Computer Engineering Department, Boğaziçi University, İstanbul, Turkey
caglayan@boun.edu.tr

Abstract. RFID authentication protocols should have a secret updating phase in order to protect the privacy of RFID tags against tag tracing attacks. In the literature, there are many lightweight RFID authentication protocols that try to provide key updating with lightweight cryptographic primitives. In this paper, we analyse the security of two recently proposed lightweight RFID authentication protocol against desynchronization attacks. We show that secret values shared between the back-end server and any given tag can be easily desynchronised. This weakness stems from the insufficient design of these protocols.

Key words: RFID, authentication protocols, de-synchronization attacks.

1 Introduction

Radio Frequency Identification (RFID) technology utilizes radio frequency in order to remotely identify people or objects. RFID systems typically consists of three elements: tags, readers and a back-end server. Many people in the world are aware of the benefits of this technology. However, these people have concerns about security and privacy problems of this technology. In the past, many authentication protocols have been proposed in order to provide adequate security and privacy level. However, many studies showed that authentication protocols that are suitable for low-cost RFID tags have serious security and privacy vulnerabilities.

Recently, two different authentication protocols have been proposed by Gao et al. [1] and Pang et al. [4]. It is claimed that these protocols provides almost all security properties in the literature. Nevertheless we show that their proposal have security weaknesses against de-synchronization attacks. These two protocols are vulnerable to the same attack type. The weak point of these protocols is that they do not use any random value that are created by the tag in their key-updating mechanism. The success probability of the proposed attack is significant and the attack complexity is polynomial.

2 Gao et al.'s Protocol

Gao et al. [1] proposed an ultralightweight RFID authentication protocol that utilizes CRC-16 and permutation (LPCP) functions. The authors formally verify the security of their protocol by using Simple Promela Interpreter (SPIN). They claim that their protocol provides resistance to the following attacks: desynchronization attacks, tracing attacks, replay attack and secret disclosure attack.

2.1 Protocol Description

In Gao et al.'s protocol, each tag $\mathcal{T}_i$ is assigned with four parameters (TID_i, $KeyH_i$, $KeyL_i$, $KeyM_i$). The server stores two entry for the tag $\mathcal{T}_i$: (TID_i^{old}, $KeyH_i^{old}$, $KeyL_i^{old}$, $KeyM_i^{old}$) and (TID_i^{new}, $KeyH_i^{old}$, $KeyL_i^{old}$, $KeyM_i^{old}$). Parameters represented with $_{new}$ substring are the current secrets of $\mathcal{T}_i$ and parameters represented with $_{old}$ substring are the last successfully verified secrets of $\mathcal{T}_i$. At the system initialization, these two entry equals each other. Table 6 gives the notations used in describing the proposed protocol. The details of the authentication process are composed of the following six steps:

Table 1. Notations for Gao et al.'s Protocol

Notation	Description
TID_i	The secure identity of the tag $\mathcal{T}_i$
$KeyX_i$	The secret keys of the tag $\mathcal{T}_i$
x_{new}	The current value of x
x_{old}	The previous value of x
CRC	The cyclic redundancy check operation
Per	The permutation operation
$\oplus$	The bit-wise XOR operation
$\in$	Random choice operator
$\leftarrow$	The substitution operation

1. The reader $\mathcal{R}$ sends a $Hello$ message to the tag $\mathcal{T}_i$.
2. Upon receiving the $Hello$ message, $\mathcal{T}_i$ sends TID_i to $\mathcal{R}$.
3. $\mathcal{R}$ gets the entry in the index $\mathcal{T}_i$ and generates a random number R_1. $\mathcal{R}$ computes α and β and sends them to $\mathcal{T}_i$.
   $\alpha \leftarrow CRC(Per(KeyM_i, KeyH_i)) \oplus R_1$
   $\beta \leftarrow CRC(Per(KeyM_i \oplus KeyH_i, CRC(KeyM_i \oplus R_1)) \oplus Per(KeyL_i, CRC(KeyM_i \oplus R_1)))$
4. Upon receiving α and β, $\mathcal{T}_i$ extracts R_1 from α and checks the validity of β in order to authenticate $\mathcal{R}$. If it is valid, $\mathcal{T}_i$ computes γ and sends it to $\mathcal{R}$.

$\gamma \leftarrow CRC(Per(CRC(KeyH_i \oplus R_1), CRC(R_1 \oplus KeyM_i)) \oplus Per(CRC(KeyM_i \oplus KeyL_i), CRC(R_1 \oplus KeyL_i)))$

5. When receiving γ, $\mathcal{R}$ checks the validity of γ in order to authenticate $\mathcal{T}_i$. If $\mathcal{T}_i$ is authenticated, $\mathcal{R}$ generates a random number R_2 and computes δ and ζ. Then δ and ζ are sent to $\mathcal{T}_i$. If $TID_i = TID_i^{new}$, $\mathcal{R}$ updates the old secrets of $\mathcal{T}_i$ with the new secrets of $\mathcal{T}_i$ otherwise old secrets of $\mathcal{T}_i$ remains unchanged. Then $\mathcal{R}$ updates the new secrets of $\mathcal{T}_i$.

   $\delta \leftarrow CRC(Per(KeyM_i, KeyL_i)) \oplus R_2$
   $\zeta \leftarrow CRC(Per(CRC(KeyH_i \oplus R_1), CRC(R_2 \oplus KeyM_i)) \oplus Per(CRC(KeyM_i \oplus R_2), CRC(R_1 \oplus KeyL_i)))$
   if $TID_i = TID_i^{new}$ **then**
      $TID_i^{old} \leftarrow TID_i^{new}$
      $KeyM_i^{old} \leftarrow KeyM_i^{new}$
      $KeyH_i^{old} \leftarrow KeyH_i^{new}$
      $KeyL_i^{old} \leftarrow KeyL_i^{new}$
   end if
   $TID_i^{new} \leftarrow CRC(Per(TID_i^{new}, R1 \oplus R2) \oplus KeyH_i^{old} \oplus KeyM_i^{old} \oplus KeyL_i^{old})$
   $KeyH_i^{new} \leftarrow CRC(Per(KeyH_i^{old}, R_1) \oplus KeyM_i^{old})$
   $KeyM_i^{new} \leftarrow CRC(Per(KeyM_i^{old}, R_2) \oplus KeyH_i^{old})$
   $KeyL_i^{new} \leftarrow CRC(Per(KeyL_i^{old}, R_1 \oplus R_2) \oplus TID_i^{old})$

6. Upon receiving δ and ζ, $\mathcal{T}_i$ extracts R_2 from β and checks the validity of ζ. If it is valid, $\mathcal{T}_i$ updates it secrets.

   $TID_i \leftarrow CRC(Per(TID_i, R1 \oplus R2) \oplus KeyH_i \oplus KeyM_i \oplus KeyL_i)$
   $KeyH_i \leftarrow CRC(Per(KeyH_i, R_1) \oplus KeyM_i)$
   $KeyM_i \leftarrow CRC(Per(KeyM_i, R_2) \oplus KeyH_i)$
   $KeyL_i \leftarrow CRC(Per(KeyL_i, R_1 \oplus R_2) \oplus TID_i)$

2.2 De-synchronization Attack

An adversary $\mathcal{A}$ performs the following attack in order to de-synchronize the secrets shared between $\mathcal{R}$ and $\mathcal{T}_i$. For simplicity, we assume that the attack begins after the last successful authentication session $s-1$. At the beginning of the attack, the states of the reader $\mathcal{R}$ and the tag $\mathcal{T}_i$ are shown in Table 2.

Table 2. State of $\mathcal{R}$ and $\mathcal{T}_i$ at the beginning of the attack

Reader $\mathcal{R}_{new}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^s$
Reader $\mathcal{R}_{old}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s-1}$
Tag $\mathcal{T}_i$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^s$

1. In a protocol session s between $\mathcal{R}$ and $\mathcal{T}_i$, $\mathcal{A}$ prevents $\mathcal{T}_i$ from taking last message flow and eavesdrops α^s, β^s, δ^s and ζ^s.

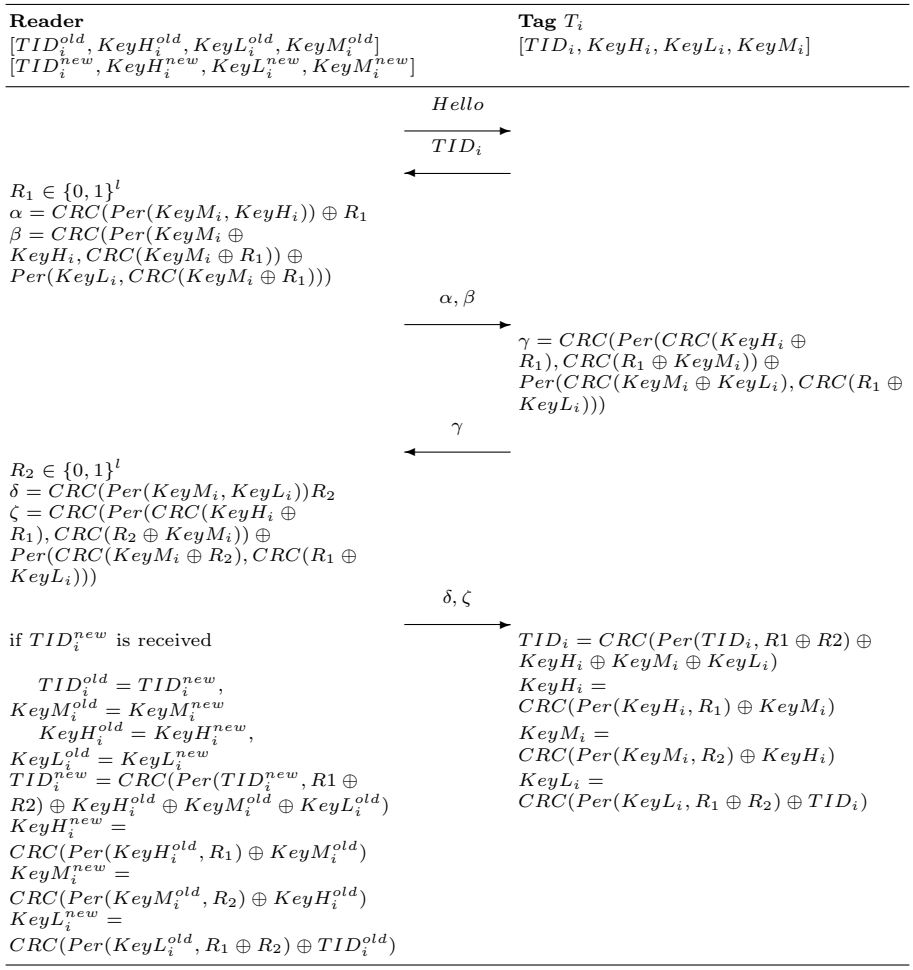

Fig. 1. Gao et al.'s Protocol

2. At the end of the protocol session s, $\mathcal{R}$ updated the secrets related with $\mathcal{T}_i$. However, $\mathcal{T}_i$ did not update its secrets because it did not receive the last message flow. The states of the reader $\mathcal{R}$ and the tag $\mathcal{T}_i$ are shown in Table 3.

Table 3. State of $\mathcal{R}$ and $\mathcal{T}_i$ at the end of the session s

Reader $\mathcal{R}_{new}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s+1}$
Reader $\mathcal{R}_{old}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s}$
Tag $\mathcal{T}_i$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s}$

3. In a protocol session $s+1$ between $\mathcal{R}$ and $\mathcal{T}_i$, $\mathcal{A}$ prevents $\mathcal{T}_i$ from taking last message flow.
4. At the end of the protocol session $s+1$, $\mathcal{R}$ updated the secrets related with $\mathcal{T}_i$. However, $\mathcal{T}_i$ did not update its secrets because it did not receive the last message flow. The states of the reader $\mathcal{R}$ and the tag $\mathcal{T}_i$ are shown in Table 4.

Table 4. State of $\mathcal{R}$ and $\mathcal{T}_i$ at the end of the session $s+1$

Reader $\mathcal{R}_{new}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s+2}$
Reader $\mathcal{R}_{old}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s}$
Tag $\mathcal{T}_i$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s}$

5. $\mathcal{A}$ broadcasts a $Hello$ message and $\mathcal{T}_i$ sends its TID_i^s to $\mathcal{A}$.
6. $\mathcal{A}$ sends α^s and β^s to $\mathcal{T}_i$. $\mathcal{A}$ passes the check by the tag $\mathcal{T}_i$ because α^s and β^s are generated with the current secrets of $\mathcal{T}_i$ and they are not generated with random values created by $\mathcal{T}_i$. Therefore, $\mathcal{T}_i$ sends γ^s to $\mathcal{A}$.
7. $\mathcal{A}$ sends δ^s and ζ^s to $\mathcal{T}_i$. $\mathcal{A}$ passes the check by the tag $\mathcal{T}_i$ because δ^s and ζ^s are generated with the current secrets of $\mathcal{T}_i$ and they are not generated with random values created by $\mathcal{T}_i$. $\mathcal{T}_i$ updates its secrets by using TID_i^s, $KeyH_i^s$, $KeyL_i^s$, $KeyM_i^s$, R_1^s and R_2^s values from the session s. The states of the reader $\mathcal{R}$ and the tag $\mathcal{T}_i$ are shown in Table 5. In the next session, $\mathcal{R}$ will not be able to authenticate the tag.

Table 5. State of $\mathcal{R}$ and $\mathcal{T}_i$ at the end of the attack

Reader $\mathcal{R}_{new}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s+2}$
Reader $\mathcal{R}_{old}$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s}$
Tag $\mathcal{T}_i$	$[TID_i, KeyH_i, KeyL_i, KeyM_i]^{s+1}$

In the above, we show that the adversary $\mathcal{A}$ breaks synchronization between $\mathcal{R}$ and $\mathcal{T}_i$. $\mathcal{A}$ impersonates the valid reader by using the messages eavesdropped in the session s and makes the tag to update its keys. At the end of the attack, the secret values at server side and the tag side are shown in Table 5.

3 Pang et al.'s Protocol

Pang et al. [4] proposed a lightweight RFID authentication protocol. This protocol utilized cyclic redundancy check (CRC) and PRNG to create a new tag indexing method, called the two-layer tag indexing mechanism. The authors claim that their protocol provides the following security and privacy properties:

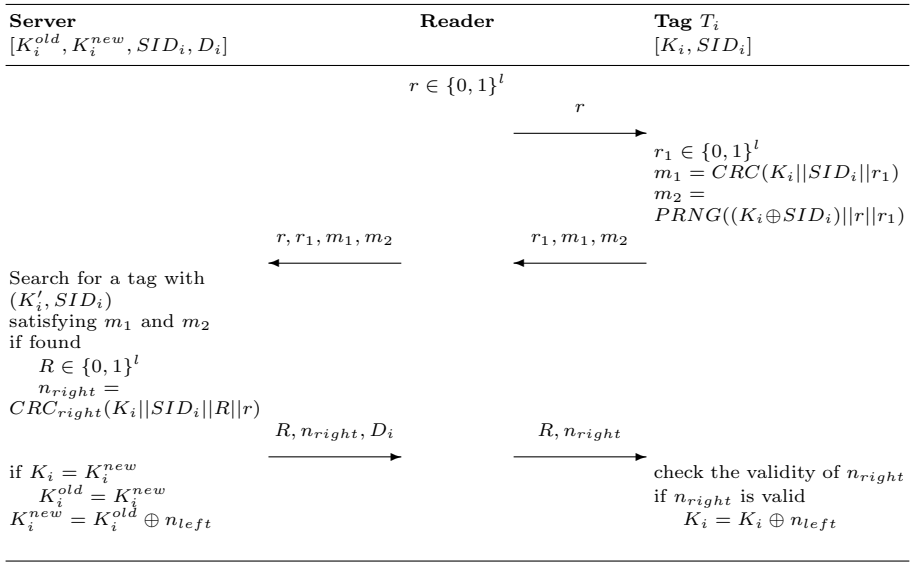

Fig. 2. Pang et al.'s Protocol

tag information privacy, tag location privacy, forgery resistance, replay attack, de-synchronization resistance, backward security and forward security. However, Safkhani and Bagheri [3] presented de-synchronization attack and traceability attack against this protocol by using the following linear property of CRC function [2, 5]:

$$CRC(A||B) = CRC(A \ll n) \oplus CRC(B) \tag{1}$$

Safkhani and Bagheri [3] also strengthened Pang et al.' protocol by using PRNG instead of CRC. The de-synchronization attack presented in Section 3.2 can be applied both Pang et al.' protocol and its revised version by Safkhani and Bagheri.

3.1 Protocol Description

In Pang et al.'s protocol, each tag $\mathcal{T}_i$ is assigned with two parameters (K_i, SID_i). The server stores an entry for the tag $\mathcal{T}_i$: (K_i^{old}, K_i^{new}, SID_i, D_i). K_i^{new} is the current secret of $\mathcal{T}_i$ and K_i^{old} is the last successfully verified secret of $\mathcal{T}_i$. At the system initialization, $K_i^{old} = K_i^{new}$. Table 6 gives the notations used in describing the proposed protocol. The details of the authentication process are composed of the following six steps:

1. The reader $\mathcal{R}$ generates a random number r and sends it to the tag $\mathcal{T}_i$.
2. Upon receiving r, $\mathcal{T}_i$ generates a random number r and computes $m_1 = CRC(K_i||SID_i||r_1)$ and $m_2 = PRNG((K_i \oplus SID_i)||r||r_1)$. $\mathcal{T}_i$ sends r_1, m_1 and m_2 to $\mathcal{R}$.

Table 6. Notations for Pang et al.'s Protocol

Notation	Description
SID_i	The secure identity of the tag $\mathcal{T}_i$
D_i	The detailed information of the tag $\mathcal{T}_i$
K_i	The secret key of the tag $\mathcal{T}_i$
x_{new}	The current value of x
x_{old}	The previous value of x
x_{left}	The left part of the massage x
x_{right}	The right part of the massage x
CRC	The cyclic redundancy check operation
$PRNG$	The pseudorandom number generator
$\oplus$	The bit-wise XOR operation
$\|\|$	The concatenation operator
$\in$	Random choice operator
$\leftarrow$	The substitution operation

3. $\mathcal{R}$ forwards r, r_1, m_1 and m_2 to the back-end server $\mathcal{S}$.
4. $\mathcal{S}$ searches its database in order to identify $\mathcal{T}_i$ by checking the validity of m_1 and m_2. If $\mathcal{S}$ identifies $\mathcal{T}_i$, it generates a random number R and computes $n_{right} = CRC_{right}(K_i||SID_i||R||r)$ and sends R, the detailed information D_i and n_{right} to $\mathcal{R}$. $\mathcal{S}$ updates K_i^{old} with K_i^{new} and K_i^{new} with $K_i^{new} \oplus n_{left}$.
5. $\mathcal{R}$ forwards R and n_{right} to $\mathcal{T}_i$.
6. $\mathcal{T}_i$ checks the validity of n_{right} in order to authenticate $\mathcal{R}$. If it is valid, $\mathcal{T}_i$ updates K_i with $K_i \oplus n_{left}$.

3.2 De-synchronization Attack

An adversary $\mathcal{A}$ performs the following attack in order to de-synchronize the secrets shared between $\mathcal{R}$ and $\mathcal{T}_i$. For simplicity, we assume that the attack begins after the last successful authentication session $s-1$. $\mathcal{R}$ stores $K_i^{old} = K_i^{s-1}$ and $K_i^{new} = K_i^s$. $\mathcal{T}_i$ stores $K_i = K_i^s$.

1. In a protocol session s between $\mathcal{R}$ and $\mathcal{T}_i$, $\mathcal{A}$ prevents $\mathcal{T}_i$ from taking last message flow and eavesdrops r^s, R^s and n_{right}^s.
2. At the end of the protocol session s, $\mathcal{R}$ updated the secrets related with $\mathcal{T}_i$. However, $\mathcal{T}_i$ did not update its secrets because it did not receive the last message flow. $\mathcal{R}$ stores K_i^s and K_i^{s+1}. $\mathcal{T}_i$ stores K_i^s.
3. In a protocol session $s+1$ between $\mathcal{R}$ and $\mathcal{T}_i$, $\mathcal{A}$ prevents $\mathcal{T}_i$ from taking last message flow.
4. At the end of the protocol session $s+1$, $\mathcal{R}$ updated the secrets related with $\mathcal{T}_i$. However, $\mathcal{T}_i$ did not update its secrets because it did not receive the last message flow. $\mathcal{R}$ stores K_i^s and K_i^{s+2}. $\mathcal{T}_i$ stores K_i^s.
5. $\mathcal{A}$ sends r^s to $\mathcal{T}_i$ and $\mathcal{T}_i$ sends r_1,m_1 and m_2 to $\mathcal{A}$.

6. $\mathcal{A}$ sends R^s and n^s_{right} to $\mathcal{T}_i$. $\mathcal{A}$ passes the check by the tag $\mathcal{T}_i$ because n^s_{right} was generated with the current secret K^s_i of $\mathcal{T}_i$ and random values r^s and R^s. It is not generated with the random value r_1 created by $\mathcal{T}_i$. Therefore, $\mathcal{T}_i$ updates its secret K^s_i by using n^s_{right}. $\mathcal{R}$ stores K^s_i and K^{s+2}_i. $\mathcal{T}_i$ stores K^{s+1}_i. In the next session, $\mathcal{R}$ will not be able to authenticate the tag.

In the above, we show that the adversary $\mathcal{A}$ breaks synchronization between $\mathcal{R}$ and $\mathcal{T}_i$. $\mathcal{A}$ impersonates the valid reader by using the messages eavesdropped in the session s and makes the tag to update its keys. At the end of the attack, the secret values at server side are $K^{new}_i = K^{s+2}_i$ and $K^{old}_i = K^s_i$, and the secret value at the tag side is $K_i = K^{s+1}_i$.

4 Conclusion

In this paper, we presented de-synchronization attacks on recently proposed lightweight authentication protocols. The weak point of these protocols is that they do not use any random value that are created by the tag in their key-updating mechanism. The success probability of the proposed attack is significant and the attack complexity is polynomial. Our attack needs two consecutive protocol sessions in which an adversary prevents the tag from taking the last message flow of the protocol. Right after the last interaction between the reader and the tag, the adversary starts a session with the tag by using the messages that are eavesdropped in the first blocked session and makes the tag to update its keys.

References

1. Gao, L., Ma, M., Shu, Y., Wei, Y.: An ultralightweight {RFID} authentication protocol with {CRC} and permutation. Journal of Network and Computer Applications (2013) –
2. Han, D., Kwon, D.: Vulnerability of an RFID Authentication Protocol Conforming to EPC Class 1 Generation 2 Standards. Comput. Stand. Interfaces **31** (2009) 648–652
3. Masoumeh Safkhani, N.B.: For an EPC-C1 G2 RFID compliant Protocol, CRC with Concatenation : No; PRNG with Concatenation : Yes. Cryptology ePrint Archive, Report 2013/490 (2013) http://eprint.iacr.org/.
4. Pang, L., Li, H., He, L., Alramadhan, A., Wang, Y.: Secure and efficient lightweight RFID authentication protocol based on fast tag indexing. International Journal of Communication Systems (2013)
5. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: Cryptanalysis of a Novel Authentication Protocol Conforming to EPC-C1G2 Standard. Comput. Stand. Interfaces **31** (2009) 372–380