A Vulnerability in the Song Authentication Protocol for Low-Cost RFID Tags

Sarah Abughazalah, Konstantinos Markantonakis, Keith Mayes

A Vulnerability in the Song Authentication Protocol for Low-Cost RFID Tags

Sarah Abughazalah, Konstantinos Markantonakis, and Keith Mayes

Smart Card Centre-Information Security Group (SCC-ISG)
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
Email: ⟨Sarah.AbuGhazalah.2012, K.Markantonakis, Keith.Mayes⟩@rhul.ac.uk

Abstract. In this paper, we describe a vulnerability against one of the most efficient authentication protocols for low-cost RFID tags proposed by Song. The protocol defines a weak attacker as an intruder which can manipulate the communication between a reader and tag without accessing the internal data of a tag. It has been claimed that the Song protocol is able to resist weak attacks, such as denial of service (DoS) attack; however, we found that a weak attacker is able to desynchronise a tag, which is one kind of DoS attack. Moreover, the database in the Song protocol must use a brute force search to retrieve the tag's records affecting the operational performance of the server. Finally, we propose an improved protocol which can prevent the security problems in Song protocol and enhance the server's scalability performance.

Keywords : RFID, mutual authentication, protocol, security, privacy.

1 Introduction

Radio frequency identification (RFID) technology is an identification technology that uses radio waves to identify objects such as products. An RFID system consists of three components, namely a tag, reader and server (database). An RFID tag is an identification device composed of an integrated circuit and antenna. It is designed to receive a radio signal and automatically transmit a reply to the reader. A passive RFID reader is a device that broadcasts a radio frequnecy (RF) signal through its antenna to power, communicate and receive data from tags. It is connected to the server to retrieve data associated with the connected tags. An RFID server is a database containing data related to the associated tags which it manages [1].

The major concerns of designing an RFID system are privacy and security [2]. Insecure communication between the reader and tag is inherently vulnerable to interception, modification, fabrication and replay attacks [2]. One of the problems that is encountered in designing an RFID system is a denial of service (DoS) attack. In a desynchronisation attack, which is one kind of DoS attack, the attacker tries to prevent both parties from receiving messages. For example, the attacker can block the exchanged message(s) from reaching the target causing the tag and the server to be unable to update their information synchronously. Thus, the tag and back-end server cannot recognise each other in subsequent transactions [3].

Song et al. [4] proposed an efficient RFID authentication protocol for low-cost tags. This protocol uses the hash functions, message authentication code (MAC) and PRNG functions for authentication and updating purposes. Each tag stores only the hash of a secret namely (t), and the server stores the old and new values of the secret (s_{new}, s_{old}), the hashed secret (t_{new}, t_{old}) and the tag's information (D). This scheme uses a challenge-response protocol, where the server and tag generate random numbers to

avoid replay attacks. However, Cai et al. [5] presented a paper showing that Song et al.'s protocol does not provide protection against a tag impersonation attack. Moreover, Rizomiliotis et al. [6] found that an attacker can impersonate the server even without accessing the internal data of a tag and launch a DoS attack.

As a result, a new version has been proposed in [7] (referred to here as the Song protocol). The Song protocol uses the same data and processes except that the construction of the exchanged message (M2 and M3) has been changed. In the new version of the Song protocol, Song claim that the proposed protocol resists DoS attack by storing the old and new values of the secret and the hashed secret, thus when the attacker blocks the transmitted message, the server still can use the recent old values to resynchronise with the tag.

In this paper, we focus on examining the new version of the Song protocol [7]. We discover that an attacker is able to desynchronise a tag without even compromising the internal data stored in the tag. Furthermore, this protocol is not scalable, as the server needs to perform a brute force search to retrieve the tag's records, which in turn affects the server performance, especially if it has to handle a large population of tags. After analysing the weaknesses of this protocol, we propose a revised protocol to eliminate these attacks with comparable computational requirements.

The rest of this paper is organised as follows: in Section 2, we present the Song protocol process in detail. In Section 3, the weaknesses of the Song protocol are illustrated. In Section 4, the revised protocol is presented. In Section 5, we analyse the proposed protocols with respect to informal analysis. In Section 6, we conclude and summarise the paper's contribution.

2 Review of the Song Protocol

This section reviews the Song protocol as shown in the original protocol [7]. Notation used in this paper are defined as follows:

- h: A hash function, $h : \{0, 1\}^* \rightarrow \{0, 1\}^l$
- f_k: A keyed hash function, $f_k : \{0, 1\}^* \times \{0, 1\}^l \rightarrow \{0, 1\}^l$ (a MAC algorithm)
- N: The number of tags
- l: The bit-length of a tag identifier
- T_i: The i^{th} tag ($1 \leq i \leq N$)
- D_i: The detailed information associated with tag T_i
- s_i: A string of l bits assigned to i^{th} tag T_i
- t_i: T_i's identifier of l bits, which equals $h(s_i)$
- x_{new}: The new (refreshed) value of x
- x_{old}: The most recent value of x
- r: A random string of l bits
- ε: Error message
- $\oplus$: XOR operator
- $\|$: Concatenation operator
- $\leftarrow$: Substitution operator
- $x \gg k$: Right circular shift operator, which rotates all bits of x to the right by k bits, as if the left and right ends of x were joined.
- $x \ll k$: Left circular shift operator, which rotates all bits of x to the left by k bits, as if the left and right ends of x were joined.

– $\in_R$: The random choice operator, which randomly selects an element from a finite set using a uniform probability distribution

The Song protocol consists of two processes: the initialisation process, and the authentication process, which are summerised below:

2.1 Initialisation Process

This stage only occurs during manufacturing when the manufacturer assigns the initial values in the server and tag. The initialisation process is summarised below:

– An initiator (e.g. the tag manufacturer) assigns a string s_i of l bits to each tag T_i, computes $t_i = h(s_i)$, and stores t_i in the tag, where l should be large enough so that an exhaustive search to find the l-bit values t_i and s_i is computationally infeasible.
– The initiator stores the entries $[(s_i, t_i)_{new}, (s_i, t_i)_{old}, D_i]$ for every tag that it manages in the server. D_i is for the tag information (e.g., price, date, etc.). Initially $(s_i, t_i)_{new}$ is assigned the initial values of s_i and t_i, and $(s_i, t_i)_{old}$ is set to null.

2.2 Authentication Process

The authentication process is shown in Table 1 as presented in the new version of the protocol [7]:

Table 1: The authentication process of the Song protocol

1. Reader $\rightarrow$ Tag: r1 $\in_R \{0, 1\}^l$
2. Tag $\rightarrow$ Reader: r2 $\in_R \{0, 1\}^l$, M1 = $t_i \oplus$ r2 and M2 = f_{ti} (r1 $\|$ r2)
3. Reader $\rightarrow$ Server: r1, M1 = $t_i \oplus$ r2 and M2 = f_{ti} (r1 $\|$ r2)
4. Server$\rightarrow$ Reader: M3 = $s_i \oplus f_{ti}$ (r2 $\|$ r1) and D_i
5. Reader $\rightarrow$ Tag: M3 = $s_i \oplus f_{ti}$ (r2 $\|$ r1)

1. Reader: A reader generates a random bit-string r1 $\in_R \{0, 1\}^l$ and sends it to the tag T_i.
2. Tag: The tag T_i generates a random bit-string r2 $\in_R \{0, 1\}^l$ as a temporary secret for the session, and computes M1 = $t_i \oplus$ r2 and M2 = f_{ti}(r1 $\|$ r2), then sends M1 and M2 to the reader.
3. Reader: The reader transmits M1, M2 and r1 to the server.
4. Server:
   (a) The server searches its database using M1, M2 and r1 as follows.
      i. It chooses t_i from amongst the values $t_i(_{new})$ or $t_i(_{old})$ stored in the database.
      ii. It computes M'2 =f_{ti}(r1 $\|$ (M1 $\oplus$ t_i)).
      iii. If M'2 = M2, then it has identified and authenticated T_i. It then goes to step (b). Otherwise, it returns to step (i). If no match is found, the server sends ε to the reader and stops the session.
   (b) The server computes M3 = $s_i \oplus f_{ti}$ (r2 $\|$ r1) and sends it with D_i to the reader.

(c) The server updates:

$s_i(_{old}) \leftarrow s_i(_{new})$
$s_i(_{new}) \leftarrow (s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r1 \oplus r2$
$t_i(_{old}) \leftarrow t_i(_{new})$
$t_i(_{new}) \leftarrow h(s_i(_{new}))$

5. Reader: The reader forwards M3 to the tag T_i.
6. Tag: The tag T_i computes $s_i = M3 \oplus f_{ti}(r2 \parallel r1)$ and checks that $h(s_i) = t_i$. If the check fails, the tag keeps the current value of t_i unchanged. If the check succeeds, the tag has authenticated the server, and sets:

$t_i \leftarrow h((s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r1 \oplus r2)$

3 Weaknesses of the Song Protocol

This section shows that the Song protocol suffers from DoS attack and database overloading.

3.1 DoS Attack

The Song protocol aims to meet some of the main security and privacy features. Resistance to DoS attack is one of the main security features. This is achieved by keeping the old values of the tag's secret (s_{old}) and hashed secret (t_{old}) in the server database just once; they are then renewed continuously once authentication is achieved. However, the Song protocol does not provide resistance to DoS attacks. Without knowing the secret value (t_i) which is stored in the tag, an adversary can easily cause synchronisation failure by twice intercepting the communication between the reader and the tag.

The protocol will fail if the attacker intercepts the communication in this way; if the server's message (M3) is intercepted, tampered or blocked up to twice, the server database will have no matching data to complete the mutual authentication, causing the DoS attack. For example, in the first access of the tag, the server's values (s_{old}, t_{old}) are set to null, while (s_{new}, t_{new}) values are set to specific values where (t_{new}) is equal to the tag's value (t_i). If the authentication succeeds, then (t_{new}) and (t_i) will be updated to the same value and (s_{old}, t_{old}) will take the previous values of (s_{new}, t_{new}). However, if the attacker blocks M3 from reaching the tag, then the server will update the server's data and the tag will be unable to update (t_i). In this situation, the value (t_i in the tag will have to match the value (t_{old}) in the database and mutual authentication can still be achieved. Now we suppose that the attacker blocks M3 for the second time; then the tag will also not update (t_i), while at that moment, (s_{old}, t_{old}) in the database have been renewed. As a result, the tag's data will not match the server's data, causing an authentication failure.

3.2 Database Overloading

The Song protocol claims that the server should be able to handle a large tag population without exhausting the server in identifying the tags. However, as shown in [7], the server needs to perform [(k+2)*F] computations to authenticate the connected tag, where F is a relatively computationally complex function

(such as a MAC or hash function) and k is an integer satisfying $1 \leq k \leq 2n$, where n is the number of tags. Hence, in every tag access, the server database has to run [k*F] computations on all its records to find the matching record, thereby exhausting the server in the searching process and affecting operational performance.

4 Revised Protocol

We propose an improvement to the Song protocol by eliminating the two issues discussed in Section 3. In the Song protocol, if the authentication is achieved, the server's data will be updated even if the matching record is found in (s_{old}) and (t_{old}). In the revised protocol, we propose that the updating process should only take place when the authentication is achieved and the matching record is found in (s_{new}) and (t_{new}); otherwise, the data remains the same. The solution is based on Yeh et al.'s protocol [8] which was designed to avoid a DoS attack found in Chien et al.'s protocol [9].

In order to reduce the number of computations required by the server to authenticate the tag, we use the notion of indexing. This requires the server and tag to store another value to serve as an index. The server stores a new index (I_{new}) and an old index (I_{old}), where the tag stores an index value (I_i). The value of the index is assigned during manufacturing. In addition, the tag stores a flag value, which is kept as either 0 or 1 to show whether the tag has been authenticated by the server or not. Moreover, for calculating the index the server and tag need a new value (k) stored by both parties. We assume all the operations in the tag are atomic i.e. either all of the commands or none are processed.

In the revised protocol, we use the same notation as presented in the Song protocol. The initialisation and authentication processes are as follows:

4.1 Initialsation Process

This stage only occurs during manufacturing when the manufacturer assigns the initial values in the server and tag. The initialisation process is summarised below:

- The server assigns random values of L bits for each tag it manages to (s_{new}, t_{new}, k_{new}, I_{new}) in the server and (t_i, k_i, I_i) in the tag.
- Initially, (s_{old}, t_{old}, k_{old}, I_{old}) in the server is set to null.
- The Flag value in the tag is set to zero.

4.2 Authentication Process

The authentication process is summarised below:

- Reader: A reader generates a random bit-string $r1 \in_R \{0, 1\}^l$ and sends it to the tag T_i.
- Tag: A tag T_i generates a random bit-string $r2 \in_R \{0, 1\}^l$ as a temporary secret for the session, and computes $M1 = t_i \oplus r2$ and $M2 = f_{ti}(r1 \parallel r2)$. The tag then checks the value of the Flag:
  1. If Flag=0, which means the tag was authenticated successfully, the tag will use the new updated index which is equal to the server's value (I_{new}), and sends I_i, M1 and M2 to the reader. Finally, the tag sets Flag=1, and recomputes the value of an index $I_i = h(k_i \oplus r2)$.
  2. If Flag=1, which means the tag has not been authenticated, the tag will use the value of the index computed in the former transaction (after setting Flag=1) which is equal to the server's value (I_{old}), then the tag transfers I_i, M1, and M2 to the reader. Finally, the tag sets Flag=1, and recomputes the value of an index $I_i = h(k_i \oplus r2)$.

– Reader: The reader transmits M1, M2, I_i and r1 to the server.
– Server:
  1. The server searches the received value of (I_i) in (I_{new}) and (I_{old}) to find a match and retrieves the attached tag data. If there is a match in I_{new}, it retrieves (s_{new}, t_{new}, k_{new}) associated to (I_{new}). Then the server sets r2 $\leftarrow$ M1 $\oplus$ t_{new}, and computes M'2 =f_{tnew}(r1 $\|$ r2) to authenticate the tag. Then it marks x=new.
  2. If there is a match in I_{old}, the server retrieves the associated data (s_{old}, t_{old}, k_{old}), and computes M1 $\oplus$ t_{old} to obtain r2. The server computes M'2 =f_{told} (r1 $\|$ r2). If M'2 = M2, then it has identified and authenticated T_i. Then it marks x=old.
  3. The server computes M3 = $s_x \oplus f_{tx}$(r2 $\|$ r1) and sends it with D_i to the reader.
  4. In case the index is found in I_{new}, the server sets:

     $s_{old} \leftarrow s_{new}$
     $s_{new} \leftarrow (s_{new} \ll l/4) \oplus (t_{new} \gg l/4) \oplus r1 \oplus r2$
     $t_{old} \leftarrow t_{new}$
     $t_{new} \leftarrow h(s_{new})$
     $k_{old} \leftarrow k_{new}$
     $k_{new} \leftarrow h(t_{new})$
     $I_{old} \leftarrow h(k_{old} \oplus r2)$
     $I_{new} \leftarrow h(k_{new} \oplus r2)$

     Otherwise, if I_i is found in I_{old}, the server keeps the data the same without any update except for:

     $I_{old} \leftarrow h(k_{old} \oplus r2)$
     $I_{new} \leftarrow h(k_{new} \oplus r2)$

– Reader: The reader forwards M3 to the tag T_i.
– Tag: The tag T_i computes s_i = M3 $\oplus$ f_{ti}(r2 $\|$ r1) and checks that h(s_i) = t_i. If the check fails, the tag keeps the current values unchanged. If the check succeeds, the tag has authenticated the server, and sets:

  $t_i \leftarrow h((s_i \ll l/4) \oplus (t_i \gg l/4) \oplus r1 \oplus r2)$
  $k_i \leftarrow h(t_i)$
  $I_i \leftarrow h(k_i \oplus r2)$
  Flag $\leftarrow$ 0

5 Analysis

Due to the fact that the server updates its data after each successful authentication, the Song protocol cannot achieve resistance to a DoS attack. In this section, we analyse our revised protocol and show that it can provide immunity to several attacks including the DoS attack and at the same time improve the server performance. Although, the tag's storage, communication and computation costs will be higher than the Song protocol, but the revised protocol appears to meet stronger privacy and security requirements.

Table 2: Computational requirements

		The Song protocol [7]	Our improved protocol Section 4	
Tag	Sending	MAC	MAC	
	Authenticating	MAC + H	MAC+ H	
	Updating	H	3H	
	Total	2MAC + 2H	2MAC + 4H	
			If x=new	If x=old
Server	Sending	MAC	MAC	MAC
	Authenticating	k*MAC	MAC	MAC
	Updating	H	4H	2H
	Total	(k+1)*MAC + H	2MAC +4H	2MAC +2H

n : The number of tags
k: An integer satisfying $1 \leq k \leq 2n$
x: The value kept as either new or old to show whether the tag uses the old or new values of the tag's record
H: Hash function
MAC: Message authentication code

- DoS attack: We tend to use the old and new values of (s_{new}, s_{old}, t_{new}, t_{old}) , as pointed in the Song protocol, to avoid DoS attack caused by M3 being intercepted. Moreover, in the proposed improved protocol, the server can still use (s_{old}, t_{old}, I_{old}) to identify a tag, even when the attacker blocks the message (M3) more than once, and thus can reach synchronisation.
- Database overloading: Table 2 demonstrates that the Song protocol needs to perform MAC functions on all the stored hashed secrets (t_{new}, t_{old}) until it finds the matched tag's record and authenticates the connected tag; in the improved protocol, on the other hand, the server can retrieve the associated tag's record directly according to the received value of index (I_i) and apply the MAC function only on the retrieved data.
- Tag location tracking: To prevent tracking the location of the tag's holder, the server's and tag's responses should be anonymous. In the proposed protocol, the server and tag update their data after each successful communication, so the exchanged values are changing continuously. Moreover, in the case the authentication failed, the attacker will still not be able to track the location.
- Tag impersonation attack: To impersonate the tag, the attacker must be able to compute a valid response (I_i, M1, M2) to a server query. However, it is hard to compute such responses without the knowledge of (t_i, k_i, r2). Moreover, the current values of M1, M2 and I_i are independent from the values sent previously due to the existence of fresh random numbers.
- Replay attack: The proposed protocol resists replay attack because it utilises challenge-response scheme. In each session the protocol uses a new pair of fresh random numbers (r1, r2), thus the messages cannot be reused in other sessions.
- Server impersonation attack: To impersonate the server, the attacker must be able to compute a valid response (M3). However, it is hard to compute such responses without knowledge of s_i, ID_i and r2.
- Traceability: All the messages transmitted by the tag are not static, they change continuously due to the existence of random numbers and the stored data are updated after each successful authentication. In addition, after the unsuccessful authentication, the tag's data will not change, however, M1 and M2 values still will be different in every session due to the existence of random numbers (r2 and r2).

Furthermore, the index of the tag is changed in both cases (successful authentication and unsuccessful authentication).

6 Conclusion

This paper showed that the Song protocol has a security problem and a performance issue, specifically a DoS attack and database overloading. To improve the Song protocol, we presented a revised protocol which can prevent the desynchronisation issues without violating any other security properties. Moreover, the newly proposed protocol enhances the overall performance, since it is based on using index values for retrieving the data associated to the connected tags.

References

1. Weis, S.: Security and privacy in Radio Frequency Identification devices. PhD thesis, Massachusetts Institute of Technology (2003)
2. Avoine, G.: Cryptography in Radio Frequency Identification and fair exchange protocols. PhD thesis, Ecole Polytechnique Federale de Lausanne (EPFL) (2005)
3. Habibi, M., Gardeshi, M., Alaghband, M.: Practical attacks on a RFID authentication protocol conforming to EPC Class 1 Generation 2 standard. arXiv preprint arXiv:1102.0763 (2011)
4. Song, B., Mitchell, C.: RFID authentication protocol for low-cost tags. In: Proceedings of the first ACM conference on Wireless network security, ACM (2008) 140–147
5. Cai, S., Li, Y., Li, T., Deng, R.: Attacks and improvements to an RIFD mutual authentication protocol and its extensions. In: Proceedings of the second ACM conference on Wireless network security, ACM (2009) 51–58
6. Rizomiliotis, P., Rekleitis, E., Gritzalis, S.: Security analysis of the Song-Mitchell authentication protocol for low-cost RFID tags. Communications Letters, IEEE **13**(4) (2009) 274–276
7. Song, B.: RFID Authentication Protocols using Symmetric Cryptography. PhD thesis, Royal Holloway, University of London (2009)
8. Yeh, T., Wang, Y., Kuo, T., Wang, S.: Securing RFID systems conforming to EPC Class 1 Generation 2 Standard. Expert Systems with Applications **37**(12) (2010) 7678–7683
9. Chien, H., Chen, C.: Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 Standards. Computer Standards Interfaces **29**(2) (2007) 254–259