Security Analysis of the Song-Mitchell Authentication Protocol for Low-cost RFID Tags

Panagiotis Rizomiliotis, *Member, IEEE,* Evangelos Rekleitis, *Student Member, IEEE,* and Stefanos Gritzalis, *Member, IEEE*

***Abstract*—In this paper, we describe an attack against one of the most efficient authentication protocols for low-cost RFID tags recently proposed by Song and Mitchell. A weak attacker, i.e. an attacker that has no access to the internal data of a tag, is able to impersonate a legitimate reader/server, and to desynchronize a tag. The attack is very efficient and has minimal computational complexity. Finally, we propose a simple solution to fix the flaw.**

***Index Terms*—Cryptography, security, protocol.**

I. INTRODUCTION

LOW-COST Radio Frequency Identification (RFID) tags will probably constitute the most pervasive device in history. Supply-chain management, inventory monitoring, payment systems, are only a few of the applications where RFID tags are used.

In most applications, many security and privacy threats arise from the use of wireless communications, thus there is much interest in deploying cryptographic mechanisms for tag authentication. In general, it is well studied how to design authentication protocols based on standard cryptographic algorithms. On the other hand, maintaining the cost of tags as low as possible, institutes space, as well as, peak and average power consumption limitations. These limitations force the tag designers to exclude almost all known authentication protocols based on standard cryptographic algorithms, making the need for new lightweight authentication protocols imperative.

Several lightweight authentication protocols have been proposed in the last few years for low-cost RFID tags ([1], [2], [3]). One such protocol, that seems to outperform previous proposals, in terms of security and performance, was introduced by Song and Mitchell in [4]. The authors in [4], provide a detailed analysis of the protocol in terms of security and privacy. Security threats are classified into weak and strong attacks. Weak attacks can be realized just by monitoring and manipulating the communication between a tag and a reader. In strong attacks the malicious user manages to compromise a legitimate tag, obtain access to its internal data and clone the tag.

In this paper, we present an attack against the Song-Mitchell protocol. More precisely, we show that, in antithesis to the security analysis presented in [4], a weak attacker can mount a reader/server impersonation attack against the tag. We also show that this attack can be combined with a denial of service attack, leading to the permanent desynchronization of the tag. Another security analysis of the same protocol appears in [5].

Manuscript received December 15, 2008. The associate editor coordinating the review of this letter and approving it for publication was C.-K. Wu.

The authors are with the Department of Information and Communication Systems Engineering, University of the Aegean, Karlovassi, Samos, GR-83200, Greece (e-mail: {prizomil, erecl, sgritz}@aegean.gr).

Digital Object Identifier 10.1109/LCOMM.2009.082117

II. THE SONG-MITCHELL AUTHENTICATION PROTOCOL

The protocol makes use of a hash function, a keyed hash function, used as a message authentication code (MAC), and a pseudorandom number generator. The following notation is used throughout the paper:

h	A hash function
f_k	A keyed hash function
N	The number of managed tags
l	The bit-length of a tag identifier
T_i	The i-th tag, where $1 \leq i \leq N$
D_i	Information about the tag stored in the database
u_i	A string of l bits assigned to T_i
t_i	T_i's l-bit identifier; $t_i = h(u_i)$
x_{new}	The new (refreshed) value of x
x_{old}	The most recent value of x
r	A random string of l bits
$\oplus$	XOR operator
$\leftarrow$	Substitution operator
$x \gg y$	Right circular shift operator, rotates all bits of x to the right by y bits
$x \ll y$	Left circular shift operator, rotates all bits of x to the right by y bits
$\in_R$	The random choice operator, randomly selects an element from a finite set using a uniform probability distribution

Every tag has a unique initiator, which creates and maintains security parameters for the tag as follows. For each new tag, the initiator, e.g. the tag owner, has to assign an l-bit string u_i to each tag T_i it manages. He then computes the hash value $t_i = h(u_i)$ and stores it on tag T_i's internal memory; l should be large enough to assure that an exhaustive search to find u_i and t_i is computationally infeasible. The initiator manages a repository that stores for each tag the entry $[(u_i, t_i)_{new}, (u_i, t_i)_{old}, D_i]$, that is, the current and the immediately previous tag identity pair (u_i, t_i) and the tag's information D_i. When first initialized $(u_i, t_i)_{new}$ holds the initial values and $(u_i, t_i)_{old}$ is set to null.

The authentication process consists of 8 steps (Fig. 1). We assume that the reader communicates with the server over a secure channel, while the tag and the reader communicate over an insecure channel.

1) The reader generates a random bit-string, $r_1 \in_R \{0,1\}^l$ and sends it to tag T_i.
2) The tag T_i generates a random bit string $r_2 \in_R \{0,1\}^l$, that it uses as a session secret and computes $M_1 =$

Server		Reader		Tag T_i
$[(u_i, t_i)_{new}, (u_i, t_i)_{old}, D_i]$				$[t_i]$
		$r_1 \in_R \{0,1\}^l$		
			$\xrightarrow{r_1}$	
				$r_2 \in_R \{0,1\}^l$ $M_1 = t_i \oplus r_2$ $M_2 = f_{t_i}(r_1 \oplus r_2)$
	$\xleftarrow{M_1, M_2, r_1}$		$\xleftarrow{M_1, M_2}$	
Search for a value t_i for which $r_2 \leftarrow M_1 \oplus t_i$ and $M_2 = f_{t_i}(r_1 \oplus r_2)$ $M_3 = u_i \oplus (r_2 >> l/2)$				
	$\xrightarrow{D_i, M_3}$		$\xrightarrow{M_3}$	
$u_{i(old)} \leftarrow u_i$ $t_{i(old)} \leftarrow t_i$ $u_{i(new)} \leftarrow (u_i << l/4) \oplus (t_i >> l/4) \oplus r_1 \oplus r_2$ $t_{i(new)} \leftarrow h(u_{i(new)})$				$u_i \leftarrow M_3 \oplus (r_2 >> l/2)$ $h(u_i) \stackrel{?}{=} t_i$ $t_i \leftarrow h((u_i << l/4) \oplus (t_i >> l/4) \oplus r_1 \oplus r_2)$

Fig. 1. The Song-Mitchell authentication protocol.

$t_i \oplus r_2$ and $M_2 = f_{t_i}(r_1 \oplus r_2)$, which it sends to the reader.

3) The reader forwards M_1, M_2 and r_1 to the backend server.
4) The server searches among its stored tag identity pairs — both new and old — for a t_i for which $r_2 \leftarrow M_1 \oplus t_i$ and $M_2 = f_{t_i}(r_1 \oplus r_2)$ both hold. If no suitable t_i is found the server sends an error message to the reader and the protocol stops.
   If t_i is found among the $(u_i, t_i)_{old}$ pairs, then the server assumes that the tag failed to complete properly the most recent authentication session and hence didn't update it's id. It thus sets $(u_i, t_i)_{new} \leftarrow (u_i, t_i)_{old}$ and continues with the protocol as normal.
5) Having found the corresponding t_i, the server computes $M_3 = u_i \oplus (r_2 \gg l/2)$ and sends it to the reader along with T_i's stored information D_i.
6) The server updates $(u_i, t_i)_{old}$ to $(u_i, t_i)_{new}$ and sets $u_{i(new)} \leftarrow (u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$ and $t_{i(new)} \leftarrow h(u_{i(new)})$.
7) The reader sends M_3 to the tag.
8) The tag sets $u_i \leftarrow M_3 \oplus (r_2 \gg l/2)$ and checks if $h(u_i) = t_i$ holds. If the check is successful it updates t_i to $h((u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2)$. Otherwise t_i remains the same.

III. Analysis of the Attack

In [4], the security analysis indicates that the proposed protocol is secure against denial of service (DoS) attacks from weak attackers and, against server impersonation attacks when the attacker is weak and, under certain conditions, when the attacker is strong. For the DoS attack, the protocol is designed in such a way that it can regain synchronization when it is lost. More precisely, a weak attacker can prevent the third flow — that is message M_3 — from reaching the tag and thus, causing the shared secrets of the server and the tag to get out of synchronization; inasmuch as the server will refresh the pair u_i and t_i, but the tag T_i will keep its current t_i. Nevertheless, due to the fact that the server maintains both the old and the new value of u_i and t_i, for each tag T_i in its database, it is possible to resynchronize with the tag in such a situation.

A server impersonation attack is successful when the attacker can create a valid M_3 message. In [4], it is claimed that a weak attacker, by only eavesdropping messages from legitimate executions of the protocol or interrupting such executions, can not create such an M_3 message. Even a strong attacker, who has compromised t_i, cannot create such a message, unless he also has access to all exchanged messages during a protocol run, in which case he can compute the next refreshed u_i value. That is, the protocol resists such an attack on the assumption that an adversary does not have access to at least one of the values r_1, M_1 and M_3 in an authentication session, that is performed between a legitimate server and a tag, for which the attacker knows the secret t_i.

In the rest of the section, we present an attack where a weak attacker can impersonate a server and desynchronize permanently the tag, without compromising the secret t_i.

A. Prerequisites

- The attacker must be able to eavesdrop on the communication between the tag and the reader. More precisely he must be able to intercept message M_1 sent from the tag to the reader and message M_3 sent from the reader to the tag.
- The attacker must be able to disrupt the communication between the reader and the tag. More precisely he must be able to prevent message M_3 from reaching the tag.

Both prerequisites are in accordance with the assumptions made for the weak attacker scenario ([4]).

B. Attack steps

a) Phase 1*:*

1) A normal authentication session takes place. The attacker is able to sniff the exchanged message M_1.
2) When the authentication protocol reaches step 5, the attacker eavesdrops message M_3 sent by the reader and prevents the tag from receiving it.

At the end of the first phase, the secret t_i stored in the tag remains unchanged, while the server has updated his own with $u_{i(new)} \leftarrow (u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$ and $t_{i(new)} \leftarrow h(u_{i(new)})$, where r_1 and r_2 are the random bit-strings used in the session. Of course, the old values

Attacker		Tag T_i
$[M_1, M_3]$		$[t_i]$
$r'_1 \in_R \{0,1\}^l$		
	$\xrightarrow{r'_1}$	
		$r'_2 \in_R \{0,1\}^l$
		$M'_1 = t_i \oplus r'_2$
		$M'_2 = f_{t_i}(r'_1 \oplus r'_2)$
	$\xleftarrow{M'_1, M'_2}$	
$M'_3 \leftarrow M_3 \oplus ((M_1 \oplus M'_1) >> 2)$		
	$\xrightarrow{M'_3}$	
		$u_i \leftarrow M'_3 \oplus (r'_2 >> l/2)$
		$h(u_i) = t_i$
		$t_i \leftarrow h((u_i << l/4) \oplus (t_i >> l/4) \oplus r'_1 \oplus r'_2)$

Fig. 2. Phase 2 of proposed attack.

$t_{i(old)} = t_i$ and $u_{i(old)} = u_i$ are retained to resist this loss of synchronization. In addition, now the attacker possesses messages $M_1 = t_i \oplus r_2$ and $M_3 = u_i \oplus (r_2 \gg l/2)$. Before a new legitimate authentication session takes place, the attacker initiates a new session, impersonating a legitimate reader/server.

b) Phase 2:

1) The attacker generates a bit string $r'_1 \in_R \{0,1\}^l$, and sends it to the tag.
2) The tag (T_i), generates a random bit-string $r'_2 \in_R \{0,1\}^l$, computes $M'_1 = t_i \oplus r'_2$ and $M'_2 = f_{t_i}(r_1 \oplus r'_2)$ and sends M'_1 and M'_2 to the attacker.
3) The attacker uses the previously eavesdropped message M_1 to compute $M_1 \oplus M'_1 = t_i \oplus r_2 \oplus t_i \oplus r'_2 = r_2 \oplus r'_2$. Subsequently he uses the previous message M_3 to compute $M'_3 = M_3 \oplus ((M_1 \oplus M'_1) \gg l/2)$. It is straightforward to verify that $M'_3 = u_i \oplus (r_2 \gg l/2) \oplus ((r_2 \oplus r'_2) \gg l/2) = u_i \oplus (r'_2 \gg l/2)$. The attacker transmits M'_3 to the tag.
4) The tag sets $u_i \leftarrow M'_3 \oplus (r'_2 \gg l/2)$ and checks that $h(u_i) = t_i$ holds. The check succeeds and the tag sets $t_i \leftarrow h((u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r'_1 \oplus r'_2)$.

After the completion of Phase 2 (see Fig. 2), the new secret of the tag is

$$t'_i = h((u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r'_1 \oplus r'_2).$$

For $(r'_1 \oplus r'_2) \neq (r_1 \oplus r_2)$, we have that $t'_i \neq t_{i(new)}$ and $t'_i \neq t_{i(old)}$. That is, the tag has lost synchronization. For a randomly chosen r'_1, we have that $(r'_1 \oplus r'_2) \neq (r_1 \oplus r_2)$ with probability $1 - 2^{-l}$, thus, the attack will be successful with almost certainty.

IV. Countermeasures and Conclusion

In order to mend the protocol, one needs to alter at least one of the messages M_1 and M_3 adding nonlinearity. Considering the hardware limitations, we will reuse one of the already implemented primitives, and more precisely the implemented keyed hash function. Further, scalability is one of the necessary properties that the reader-server site must possess; that is, it should be able to handle increasing amounts of work in, as the tag population grows.

From the description of the protocol, the server must extract the random bit-string r_2 from message M_1 approximately $N/2$ times, (for half the tags on average), in order to find the correct tag; therefore, we must keep this operation as simple as possible. On the other hand, the computation of M_3 is performed only once.

Taking the above into consideration, we propose the following modification. The server computes $M_3 = f_{r_2}(u_i)$ and the reader transmits M_3 to the tag. The tag computes $f_{r_2}(h(t_i))$, and compares it with the received message. If the two messages are identical the tag authenticates successfully the server.

Compared with the original authentication protocol by Song and Mitchell, our proposal requires one more execution of the keyed hash function from the tag and one from the server, while the storage requirements remain the same.

The resistance against denial of service attacks relies on the storage of older values of the secrets t_i and u_i in the server's database, as in the original protocol.

References

[1] D. Molnar and D. Wagner, "Privacy and security in library RFID: issues, practices, and architectures," in *Proc. 11th ACM Conference on Computer and Communications Security*, pp. 210–219, New York, NY, USA, 2004.

[2] D. Henrici and P. Müller, "Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers," in *Proc. 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops*, p. 149, Washington, DC, USA, 2004.

[3] S. Karthikeyan and M. Nesterenko, "RFID security without extensive cryptography," in *Proc. 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 63–67, New York, NY, USA, 2005

[4] B. Song and C. J. Mitchell, "RFID authentication protocol for low-cost tags," in *Proc. First ACM Conference on Wireless Network Security*," pp. 140–147, New York, NY, USA, 2008

[5] T. van Deursen and S. Radomirovic, "Attacks on RFID protocols," in *IACR eprint archive 2008/310*, July 2008.