Designing secure RFID authentication protocols is (still) a non-trivial task

Panagiotis Rizomiliotis, Evangelos Rekleitis, Stefanos Gritzalis
Dep. of Information and Communication Systems Engineering,
University of the Aegean,
Karlovassi, Samos, GR 83200, Greece.
Email: http://www.icsd.aegean.gr/info-sec-lab

***Abstract*—In the last few years, a plethora of RFID authentication protocols have been proposed and several security analyses have been published creating the impression that designing such a protocol must be, more or less, a straightforward task. In this paper, we investigate the security of two recently proposed schemes, showing that designing a secure RFID authentication protocol is still a demanding process. One is a mature work; in the sense that it has predecessors that have been extensively analyzed, while the other is a fresh proposal. Our security analysis demonstrates that both are weak, as they suffer from a similar desychronization attack. In addition we prove the existence of a fatal tag impersonation attack against the second one.**

I. Introduction

Radio Frequency Identification (*RFID*) is a sensor-based technology, used, primarily, to identify and track products or living organisms [1]. This is achieved by using reader devices to query embedded integrated circuits, called tags. RFID tags may either be self-powered (active) or passive, requiring power from an external source (e.g. the reader) or a hybrid; using both internal and external power sources. RFID tags are expected to revolutionize our daily life, becoming the most pervasive device ever. Especially since low cost, passive tags are destined to replace and enhance the now ubiquitous barcode, as well as, allow new tracking, access management and security services. Supply-chain management, inventory monitoring, payment systems, passports being only few of the applications where RFID tags are currently employed.

However, the introduction of RFID-enabled systems is not without security and privacy woes and worries [2], [3]. Hence there is an active interest in deploying cryptographic mechanisms for tag authentication. Even though there exist many studies on how to design authentication protocols based on standard cryptographic primitives, the need to make RFID systems economically viable and maintain the tag cost as low as possible, enforces certain limitations on the tags in terms of computational power, memory and circuit space. Taking into consideration the implementation cost of a public key algorithm, it becomes clear that the application of one of the standard authentication schemes can be too expensive for low cost Electronic Product Code (EPC) tags that cannot devote several thousands of gates for security [4]–[7]. For comparison, one of the 'lightest' implementations, provided by Oren and Feldhofer in [8], fits a 1024-bit public key scheme into 4682 gate equivalents. A limitation clearly not present in the reader/server side that should be able to handle increasing amounts of work, as the tag population grows.

In this context, the need for efficient and secure authentication protocols is imperative and many proposals have been made – an extended list can be found in [9].

In [10] we defined five important security requirements that a security protocol should satisfy; namely:

- **Resistance to Tag impersonation:** an adversary should not be able to impersonate a legitimate tag to the reader.
- **Resistance to Reader impersonation:** an adversary should not be able to impersonate a legitimate reader/server to the tag.
- **Resistance to Denial of Service (DoS)/desychronization attacks:** manipulating or blocking communication during a given number of sessions, between the tag and the reader, should not prevent any future normal interaction between the legitimate reader and tag.
- **Indistinguishability** (**tag anonymity**): tag output must be indistinguishable from truly random values. Moreover, they should be unlinkable to the static ID of the tag. To achieve a stricter notion of tag anonymity, we further define:
  - **Forward security/untraceability:** even if an adversary acquires all the internal states of a target tag at time t, she should not be able to ascribe past interactions, that occurred at time $t' < t$, to the said tag.
  - **Backward security/untraceability:**[1] similarly to forward security, it requires that even if an adversary gains knowledge of a tag's internal state at time t, she should not be able to ascribe future/subsequent interactions, that occur at time $t' > t$, to the said tag.

As well as, a set of desirable tag management operations that a protocol should provide; namely:

- **Tag authentication:** the reader/back-end system should be able to authenticate the tag.
- **Revocable access delegation:** (aka tag delegation), the capability to allow a third party, tag authentication and read access to an owned tag, while maintaining the

[1]Some research works interchange the two terms, i.e. backward security is called forward security

978-1-4577-0460-4/11/$26.00 ©2011 IEEE

right to revoke this privilege, under some predefined conditions.

- **Ownership transfer:** the capability to pass ownership of a tag to a third party, without compromising backward untraceability for the said party, or forward untraceability for the previous owner.
- **Permanent and temporal tag invalidation:** more commonly known as kill and sleep operations; where initially proposed to offer a minimal degree of command over the tag. A legitimate tag owner can issue a command to disallow the tag from emitting any signals; in the case of the sleep operation this ban of communication can easily be revoked by the owner. Implementing them is trivial and it is obvious that these operations can also be achieved by physical means, e.g. breaking the tag or placing it in a faraday cage.

Despite the significant attention that this research area has gained the last decade and the huge amount of scientific papers that have been published, it seems that designing an authentication protocol, that achieves (at least) all the security goals it was designed for, is not a trivial task.

In this paper, we advocate in favour of the previous claim by presenting the security analysis of two characteristic protocols; i.e. the analysis of a mature work and of a fresh proposal. More specifically, we study the security of two recently published RFID authentication protocols. The first protocol is a mature work proposed by Song and Mitchell (*SM-2*) [11]. This protocol is the evolution of previous proposals by the same authors, after correcting all identified flaws. The second protocol is a fresh attempt by Cho, Yeo and Kim (*CYK*) [12]. As our analysis demonstrates, both protocols have severe weaknesses.

The *SM-2* protocol was (re)designed to resist desynchronization attacks. However, we show that an adversary, who is able to alter messages transmitted from the server to the tag, can mount an attack that causes a permanent lose of synchronization between the two communicating entities, as the server and tag update their shared secret to different values. As a result, they are not able to authenticate one another afterwards. The *CYK* protocol is much weaker, as expected, since it is a less mature work. Not only it is vulnerable to a similar desynchronization attack, but we also present an efficient tag impersonation attack, in which an adversary is able to authenticate as a legitimate tag to the server, without knowing the tag's internal secrets. The attacker only needs to eavesdrop the communication between the server and the target tag. To the best of our knowledge this is the first cryptanalysis attempt against the CYK protocol.

The paper is organised as follows. In Section II, we describe shortly the *SM-2* protocol, while in subsection II-B, we introduce the desychronization attack. In Section III, we present the *CYK* protocol, while in Section III-B, we introduce an efficient tag impersonation attack and discuss how the desychronization attack of subsection II-B can be applied. Section IV concludes our paper. There we propose a small amendment to the *SM-2* protocol that resists the attack presented in this paper. We do not believe that this is possible for *CYK*, without considerable redesigning from scratch.

Server/Reader		Tag T
$[k, r, x, M_1]$		$[k, x, c = 0, M_1]$
$s' \in_R \{0,1\}^l$		
$m' \in_R \mathcal{Z}^+$		
$k' = h(s')$		
$M_{3a} = g_k(r\|\|M_1)$		
$M_{3b} = (s\|\|k'\|\|m')$		
$M_3 = M_{3a} \oplus M_{3b}$	$\overset{M_3, r}{--\rightarrow}$	$(s\|\|k'\|\|m') = M_3 \oplus g_k(r\|\|M_1)$
		? $k == h(s)$
		update $k = k'$ and $c = m'$

TABLE I
THE SECRET UPDATING OPERATION OF *SM-2*.

II. THE *SM-2* PROTOCOL SECURITY ANALYSIS

A. *The* SM-2 *protocol*

In [11], Song and Mitchell presented a new RFID authentication protocol, *SM-2*. This protocol is based on a previous proposal by the same authors, the *SM* protocol [13]. The *SM* protocol was an RFID mutual authentication protocol designed to satisfy important privacy and security requirements, in an efficient manner. Unfortunately, the *SM* protocol was found vulnerable against several attacks [3], [14]–[16]. The first published attack appeared in the 2008 version of [3], where van Deursen & Radomirovi identified a tag authentication attack. They showed that, by misusing the associative and communicative properties of the XOR operation, an attacker could mount a replay attack against the verifying server and authenticate as a valid tag. In [14] we showed the protocol was also vulnerable to server impersonation attacks that could further lead to permanent desynchronization of the tag and proposed an efficient correction. An analysis of both attacks appeared in [16]. Cai et. al. [15], also identified the above mentioned flaws and proposed a revised protocol (SM-revised) to correct them. In fact, the 2009 updated version of [3] includes all three known attacks against the SM protocol.

The *SM-2* protocol is based on the *SM*-revised design, in order to avoid all the previously mentioned flaws, aiming to constitute a secure and highly scalable solution that covers both the security and functional requirements mentioned earlier.

SM-2 is designed as a five-part protocol, including tag initialization. The first part deals with tag authentication, the second provides secret updating, the third allows a soft resynchronization to recover from irregular runs of the protocol – not to be confused with our permanent desynchronization attack – and the fourth is an optional, slightly more efficient secret updating process, for use when one is certain of the tag's identity. The protocol makes use of simple operations such as XOR and concatenation, as well as, a pseudorandom number generator, a hash function (h) and 3 keyed hash functions (e, f, g); all of which are assumed to be one-way and collision-resistant. Each tag stores an l-bit secret key k, an l-bit pseudonym x and a l_m-bit counter c. For each managed tag the server has knowledge of all stored

values.

Part 0: Initialization

For each tag T managed by the server S, S builds a look-up table, as follows:

- S chooses a random l-bit string s, and computes the l-bit key $k = h(s)$. The random string s is used for server authentication and k is T's secret key used in the keyed hash functions.
- S chooses a random l-bit string x_0, and computes the hash-chain values $x_i = e_k(x_{i-1})$ for $1 \leq i \leq m$. Thus the length of the hash-chain is m.
- For each managed tag T, S stores, in its look-up table, the corresponding s, k and the identifiers $x_0, x_1, \ldots, x_m$. In subsequent runs of the protocol the set of stored data for T will also contain the most recent previous values for s and k.
- Each tag T initially sets $k = h(s)$, $x = x_0$ and the counter $c = m$

Part 1: Authentication

When a tag approaches the vicinity of the server (actually the server's RFID reader device)

- S generates a suitably long ($\leq l$) random binary string r and sends it to T.
- When T receives r, it checks its counter c. If $c \neq 0$ it computes $M_1 = f_k(r \parallel x)$, updates x to $e_k(x)$ and subtracts 1 from c (c−−). The values r, M_1 and $M_2 = x$ are sent to S. If the counter reaches 0, the tag waits for a server response to perform a secret update, keeping r, M_1, M_2 in short term memory.
- When S receives r, M_1, M_2, it performs a search in its look-up table for a value x_i equal to the received M_2. If such a value is found, T has been successfully identified. Otherwise, the server assumes T is an alien tag. To authenticate T, the server computes $f_k(r \parallel x_{i-1})$ and compares it to the received M_1. If the values differ, T is not authenticated. Next the server checks if $x == x_m$, if true it needs to perform a secret update, otherwise the session is terminated successfully.

Part 2: Secret updating

If a secret update is in place; i.e. $x == x_m$ (see Table I):

- S generates a random l-bit binary string s' and an integer m'. It then computes the new tag key $k' = h(s')$ and the m'-long sequence of corresponding identifiers (x'_i) as in Part 0.
- Next S computes $M_3 = M_{3a} \oplus M_{3b}$; where $M_{3a} = g_k(r \parallel M_1)$ and $M_{3b} = (s \parallel k' \parallel m')$. M_3 it sent to T, along with r. The final action of S is to update the set of stored values for T, to $s, k, s', k', x, x'_1, \ldots x'_m$
- When T receives r and M_3, it computes $(s \parallel k' \parallel m') = M_3 \oplus g_k(r \parallel M_1)$. If $h(s)$ is equal to k, S is authenticated and T updates its key and counter to k' and m', respectively.

Part 3: Resynchronization

Server/Reader	Attacker	Tag T
$[k, r, M_1]$		$[k, x, c = 0, M_1]$
$s' \in_R \{0,1\}^l$		
$m' \in_R \mathcal{Z}^+$		
$k' = h(s')$		
$M_{3a} = g_k(r \| M_1)$		
$M_{3b} = (s \| k' \| m')$		
$M_3 = M_{3a} \oplus M_{3b}$	$\overset{\hat{M}_3 = M_3 \oplus (0 \| k'' \| m''), r}{--\rightarrow}$	$(s \| k' \oplus k'' \| m' \oplus m'') = M_3$ $\oplus (0 \| k'' \| m'') \oplus g_k(r \| M_1)$ $? \; k == h(s)$ update $k = k' \oplus k''$ and $c = m' \oplus m''$

TABLE II
THE DESYCHRONIZATION ATTACK AGAINST THE *SM-2* PROTOCOL.

The authors of *SM-2* provide an alternative secret updating process, in case any irregularities happen, e.g. a malicious attacker continuously queries the tag to zero its counter, or if the last secret update was unsuccessful. The logic is similar to Part 1 and 2, so we will only briefly describe the exchanged messages.

- T generates a random number r_T to use as a secret session key. It then computes $M_1 = f_k(r \parallel r_T)$ and $M_2 = r_T \oplus x$ and sends both to S, with a request for an update of the shared secrets
- When S receives M_1 and M_2, it begins a search in its look-up table for a matching value $x == x_0$ or $x == x_m$. If such an x is found T is identified and authenticated
- If $x == x_m$ then Part 2 is performed, only this time S needs to extract $r_T = M_2 \oplus x$ and compute $M_3 = M_{3a} \oplus M_{3b}$, where $M_{3a} = g_k(r \parallel r_T)$ and $M_{3b} = (s \parallel k' \parallel m')$.

The authors of *SM-2* describe two more secret updating processes. One is applicable when the reader's owner has prior knowledge of the tag's identity; in which case she may use an optional secret updating process that requires fewer operations. The second one is the same as Part 3, but instead of using a new key, it restores a previous key. Both are unrelated to the proposed attack and for that we omit their description.

B. A desynchronization attack against the SM-2 protocol

This attack can render a tag completely unmanageable by permanently desynchronizing it. It is an active attack, since the adversary must have the ability to manipulate messages exchanged between the tag and the server (reader actually) and it is mounted during the key update; i.e. Part 2. The attacker needs only to intervene on the last message sent to the tag; viz. the secret update message M_3.

From the protocol, we know that M_3 is calculated, at the server, as the XOR of M_{3a} and M_{3b}. While the actual construction of M_{3a} depends on the protocol phase in execution, M_3b is always derived from the concatenation of the shared secret s, the new key k' and the new length of the pseudonym hash-chain m'; i.e. $M_{3b} = (s \parallel k' \parallel m')$. The tag extracts $(s \parallel k' \parallel m')$ by simply computing $M_3 \oplus M_{3a}$. The tag authenticates the reader using only the value s, and then it updates the secret key with k' and the counter's value with $c = m'$. However, it does not protect the integrity of the whole (M_3) message, which can be fatal.

Server/Reader		Tag T
$[ID_T, s_j, s_{j-1}]$		$[ID_T, s_j]$
$R_r \in_R \{0,1\}^{96}$		
	$\overset{R_r}{- - - \rightarrow}$	
		$R_t \in_R \{0,1\}^{96}$
		$M_1 = R_t \oplus \beta$
		$\alpha = h(ID_T \oplus R_t \oplus R_r \oplus RID_i)$
	$\overset{\alpha, M_1}{\leftarrow - - -}$	
$\beta = s_j(0:47) \parallel ID_T(48:95)$		
$R_t = M_1 \oplus \beta$		
$RID_i = (R_t - R_t mods_j + 1)(0:47) \parallel (R_t + s_j - R_t mods_j)(48:95))$		
$\alpha' = h(ID_T \oplus R_t \oplus R_r \oplus RID_i)$		
if $\alpha' \overset{?}{=} \alpha$, then $M_2 = h(\beta \oplus RID_i)$		
$s_{j+1} \in_R \{0,1\}^{96}$, $M_3 = s_{j+1} \oplus R_t$	$\overset{M_2, M_3}{- - - \rightarrow}$	if $M_2 \overset{?}{=} h(\beta \oplus RID_i)$, then
store $[ID_T, s_{j+1}, s_j]$		$s_{j+1} = M_3 \oplus R_t$
		store $[ID_T, s_{j+1}]$

TABLE III
THE CYK PROTOCOL.

The attacker needs to modify message M_3 by $d = (0 \parallel k'' \parallel m'')$; so that the tag receives the message $M_3' = M_3 \oplus d$. The length of $(k'' \parallel m'')$ is $l' = \lceil log_2(m') \rceil$; i.e. the length of m'.

The tag then computes:
$(s \parallel k' \oplus k'' \parallel m' \oplus m'') = M_3' \oplus M_{3a} = M_3 \oplus d \oplus M_{3a}$.
The reader is authenticated successfully, as the value of s is computed correctly. However, both the new key and counter values are different than the ones stored by the reader. More precisely, the tag stores the values $k = k' \oplus k''$ and $c = m' \oplus m''$, while the reader has stored k' and m', for the key and the counter respectively. Thus, the tag has stored different values than those of the server/reader, rendering the authentication, key update and resychronization functions inoperative.

Note that there is no need to eavesdrop a full or more rounds of the protocol or to tamper with the tag, nor any assumptions are made on the security of the cryptographic primitives in use (hash functions, pseudorandom number generators). It is the linearity of the XOR and concatenation functions that allows arbitrary data to be authenticated as valid.

III. THE *CYK* PROTOCOL SECURITY ANALYSIS

A. *The* CYK *protocol*

In [12], Cho, Yeo & Kim (CYK) presented a hash-based RFID mutual authentication protocol aiming to solve the privacy and forgery problems with RFID system (sic). The authors define three security requirements; namely confidentialityof transmitted data, indistinguishability & forward security and mutual authentication, that need to be considered when designing an authentication protocol and should be used as evaluation criteria. Mutual authentication corresponds to the notion of preventing tag and/or server impersonation and our definition of 'indistinguishability' covers both confidentiality, indistinguishability and forward security. Regarding tag management operations, the protocol provides only authentication and doesn't inherently support tag delegation. Secure ownership transfer could be realized by updating the tag to a random temporary secret, which is revealed to the new owner and altered by him as soon as possible.

CYK is a 9-phases protocol, using Phase 0 for the tag initialisation. The next five phases (i.e. Phases $1-5$) support Tag identification, authentication and secret key/value update on the Server side, while the last three phases (i.e. Phases $6-8$) lead to Server authentication and secret value update on the Tag side.

The protocol requires the implementation of a secure hash function $h(\cdot)$ and a random number generator, while it makes use of simple operations, like XOR ($\oplus$) and concatenation ($\parallel$). Each tag T stores an 96-bit identifier ID_T and the current 96-bit secret value s_j, shared with the server. The secret value is updated with every protocol execution. For each managed tag the server S stores the ID_T, the current secret value s_j and the previous secret value s_{j-1}. For brevity, we treat the back-end server and the reader as one entity (Table III). The nine phases of CYK are as follows.

Phase 0: Initialisation
For each managed tag (T) the server (S) stores the triplet $\{ID_T, s_j, s_{j-1}\}$; i.e. the tag identifier, the current secret and previous secret. The tag stores $\{ID_T, s_j\}$.

Phase 1: Read request
When a tag T approaches the vicinity of the server (actually the server's RFID reader device), S generates a 96-bit random number R_r and transmits it to T.

Phases 2-4: Tag response

- T generates a 96-bit random number R_t and the corresponding identifier for R_t's number group

$$RID_i = (R_t - R_t mods_j + 1)(0:47) \parallel (R_t + s_j - R_t mods_j)(48:95)). \quad (1)$$

- It then computes the authentication message

$$\alpha = h(ID_T \oplus R_t \oplus R_r \oplus RID_i), \quad (2)$$

Server/Reader		Attacker
$[ID_T, s_j, s_{j-1}]$		$[R_r, \alpha, M_1]$
$\hat{R}_r \in_R \{0,1\}^{96}$		
	$\overset{\hat{R}_r}{- - - \rightarrow}$	$d = \hat{R}_r \oplus R_r$
		$\hat{M}_1 = M_1 \oplus d$
	$\overset{\alpha, \hat{M}_1}{\leftarrow - - -}$	
$\beta = s_j(0:47) \parallel ID_T(48:95)$		
$\hat{R}_t = \hat{M}_1 \oplus \beta$		
$R\hat{I}D_i = (\hat{R}_t - \hat{R}_t mod s_j + 1)(0:47) \parallel (\hat{R}_t + s_j - \hat{R}_t mod s_j)(48:95)$		
$\hat{\alpha} = h(ID_T \oplus \hat{R}_t \oplus \hat{R}_r \oplus R\hat{I}D_i)$		
if $\hat{\alpha} \overset{?}{=} \alpha$, then $\hat{M}_2 = h(\beta \oplus R\hat{I}D_i)$		
$s_{j+1} \in_R \{0,1\}^{96}$, $\hat{M}_3 = s_{j+1} \oplus \hat{R}_t$	$\overset{\hat{M}_2, \hat{M}_3}{- - - \rightarrow}$	X
store $[ID_T, s_{j+1}, s_j]$		

TABLE IV
EXECUTION PHASE OF THE TAG IMPERSONATION ATTACK.

a blinding factor

$$\beta = s(0:47) \parallel ID_T(48:95) \tag{3}$$

and calculates message

$$M_1 = R_t \oplus \beta. \tag{4}$$

- T sends to the server messages α and M_1.

Phase 5: Tag authentication & secret update

For each managed tag, the server S:

- Computes the corresponding β
- Extracts $R_t = M_1 \oplus \beta$.
- Calculates group identifier RID'_i, using the extracted s_j and R_t .
- Generates α'. If α' is equal to the received α, then T has been successfully identified and authenticated.

After the tag has been authenticated the server updates the tag's data as follows:

- Generates a new secret value s_{j+1} using the random number generator and modifies the stored tag triplet $\{ID_T, s_{j+1}, s_j\}$.
- Computes the authentication message $M_2 = h(\beta \oplus RID_i)$ and the secret update message $M_3 = (R_t \oplus s_{j+1})$.

If no matching tag was found, S repeats the search using the previous secret values s_{j-1}. This might happen, if a synchronisation problem occurred in the previous execution of the protocol with T. After successfully authenticating the tag, the server S produces messages M_2 and M_3 and stores the new tag's triplet $\{ID_T, s_{j+1}, s_{j-1}\}$. If the authentication process fails again, then the tag is considered alien.

Phases 6-7: Server response

The server S sends to T messages M_2 and M_3.

Phase 8: Secret update

- T authenticates the server by calculating $h(\beta \oplus RID_i)$ and comparing it to M_2.
- T extracts the new secret from M_3, $s_{j+1} = M_3 \oplus R_t$ and replaces the previous secret value $s_j = s_{j+1}$.

B. A tag impersonation attack against the CYK *protocol*

We will now describe a tag impersonation attack that a malicious user can mount against the protocol with overwhelming probability. This is a passive attack, meaning that the attacker needs only to eavesdrop a protocol execution between the server and a legitimate tag to impersonate the tag; without extracting any of the secret values stored in it. The attack exploits the way group identifiers RID and the corresponding authentication message α are computed. Let T be a legitimate tag, with current secret value s_j and identity number ID_T. The attack is divided in two phases as follows:

1) **Preparation phase:** The attacker passively eavesdrops a protocol execution (Table III) between the server and the legitimate tag T and she stores the messages exchanged during phases 1 and 3; i.e. R_r, α and M_1. At the end, the tag and reader share a new secret value s_{j+1}, while the old secret s_j is stored by the server for protection from desynchronisation.
2) **Execution phase:** The server initiates a protocol run and the attacker attempts to impersonate the legitimate tag T. The Execution phase appears in Table IV.
   a) **Phase 1**: The attacker receives the new random value $\hat{R}_r$ from the reader.
   b) **Phase 2**: The attacker computes the difference $d = R_r \oplus \hat{R}_r$ and calculates a new message $\hat{M}_1 = M_1 \oplus d$.
   c) **Phase 3**: The attacker sends messages $\alpha, \hat{M}_1$; i.e. the stored value α from the Preparation phase and the new message $\hat{M}_1$.
   d) **Phases 4-7**: If the reader authenticates the attacker as the legitimate tag, it computes and transmits the messages $\hat{M}_2$ and $\hat{M}_3$ and the attacker ignores both messages. If the authentication fails, the attacker repeats the attack.

Next, we show that the success probability P_{succ} of the tag impersonation attack is greater or equal to $^1/_4$,

$$P_{succ} \geq \frac{1}{4} \ ,$$

i.e. the attacker has to repeat the two phases of the attack only 4 times on average in order to succeed. We are going to need Lemma 1.

Lemma 1: Two group numbers RID_i and $R\hat{I}D_i$ computed from (1) with random numbers R_t and $\hat{R}_t$, respectively, are equal with probability at least $\frac{1}{4}$,

$$P(RID_i = R\hat{I}D_i) \geq \frac{1}{4}$$

when the same random secret key s_j is used.

Proof: Let the two group numbers, computed from (1), given by

$$\begin{aligned} RID_i = &(R_t - R_t \bmod s_j + 1)(0:47) \\ &\| \ (R_t + s_j - R_t \bmod s_j)(48:95) \end{aligned}$$

and

$$\begin{aligned} R\hat{I}D_i = &(\hat{R}_t - \hat{R}_t \bmod s_j + 1)(0:47) \\ &\| \ (\hat{R}_t + s_j - \hat{R}_t \bmod s_j)(48:95). \end{aligned}$$

Let $R_t, \hat{R}_t < s_j$. In that case, it is straightforward to verify that

$$R_t - R_t \bmod s_j = \hat{R}_t - \hat{R}_t \bmod s_j = 0 \ ,$$

and that

$$RID_i = (1)(0:47) \ \| \ (s_j)(48:95) = R\hat{I}D_i \ .$$

Thus, the probability $P(RID_i = R\hat{I}D_i)$ is lower bounded by

$$P(RID_i = R\hat{I}D_i) \geq P(R_t, \hat{R}_t < s_j) \ .$$

Since, $R_t, \hat{R}_t, s_j$ are random 96-bit numbers, for a given secret value s_j, it holds that

$$P(R_t < s_j) = P(\hat{R}_t < s_j) = \frac{s_j}{N} \ ,$$

where $N = 2^{96}$. Given that values R_t and $\hat{R}_t$ are independent, it follows that

$$P(R_t, \hat{R}_t < s_j) = \frac{(s_j)^2}{N^2} \ .$$

Finally, since the average value of s_j is $\frac{N}{2}$, we have that

$$P(RID_i = R\hat{I}D_i) \geq P(R_t, R'_t < s_j) = \frac{1}{4} \ .$$

■

Theorem 1: The success probability P_{succ} of the tag impersonation attack is greater or equal to $^1/_4$,

$$P_{succ} \geq \frac{1}{4} \ .$$

Proof: After the Preparation phase, the server has stored the identity of the tag ID_T and the two secret values; viz. the current s_{j+1} and the old one s_j, and accepts both as valid. The attacker wants to impersonate the tag T and responds to the server in Phase 3 by producing a pair of messages $(\hat{\alpha}, \hat{M}_1)$, pretending that he knows the secret value s_j. In order for the pair to be valid, from (2), it must hold that

$$\hat{\alpha} = h(ID_T \oplus \hat{R}_r \oplus \hat{R}_t \oplus R\hat{I}D_i). \tag{5}$$

The attacker sends the pair of messages $(\alpha, M_1 \oplus d)$; i.e. the eavesdropped messages from the Preparation phase. The first one is left as it is and the second is modified by d, where $d = R_r \oplus \hat{R}_r$. From (3) and (4) we have that message $\hat{M}_1$ depends on $\hat{R}_t$, s_j and the tag identity ID_T. Since, the last two values remain the same during the Preparation phase and the Execution phase of the attack, the specific choice of the message $\hat{M}_1 = M_1 \oplus d$ implies that

$$\hat{R}_t = R_t \oplus d. \tag{6}$$

From (5) and (6) we have that,

$$\begin{aligned} \hat{\alpha} &= h(ID_T \oplus R_r \oplus d \oplus R_t \oplus d \oplus R\hat{I}D_i) \\ &= h(ID_T \oplus R_r \oplus R_t \oplus R\hat{I}D_i). \end{aligned} \tag{7}$$

Thus, the attack is successful when $\hat{\alpha} = \alpha$; i.e. the probability of success is given by

$$P_{succ} = P(\hat{\alpha} = \alpha). \tag{8}$$

From (2) and (7), it holds that $\hat{\alpha} = \alpha$, only when $RID_i = R\hat{I}D_i$. That is,

$$P(\alpha = \alpha') \geq P(RID_i = R\hat{I}D_i) \ .$$

Finally, from Lemm 1, it holds that

$$P(\alpha = \alpha') \geq P(RID_i = R\hat{I}D_i) \geq \frac{1}{4} \ ,$$

and from (8) we have that

$$P_{succ} \geq \frac{1}{4} \ .$$

■

Note 1: The impersonation attack causes also a desynchronisation between the server and the legitimate tag T. At the end of the Preparation phase, the server has stored the triplet $\{ID_T, s_{j+1}, s_j\}$, while the tag T has the secret value s_{j+1}. After, the successful completion of the Execution phase, the server has stored the triplet $\{ID_T, \hat{s}_{j+1}, s_j\}$, while the tag T still has the secret value s_{j+1}. Thus, the server will not be able to authenticate the legitimate tag T after the attack. This side effect is not usually desirable as it reveals the existence of the impersonation attack.

In order for the attacker to avoid the desychronisation, an active step must be added in the Preparation phase. More precisely, during phase 7, the attacker has to interfere and prevent messages M_2 and M_3 from reaching the tag; disallowing thus the secret value update. This means that after the Preparation phase tag T maintains the same secret value s_j. By design the server retains the triplet $\{ID_T, \hat{s}_{j+1}, s_j\}$, containing the old value; thus the tag remains synchronized and recognizable. The modified version of the Preparation phase appears in Table V.

Server/Reader	Attacker	Tag T
$[ID_T, s_j, s_{j-1}]$ $R_r \in_R \{0,1\}^{96}$		$[ID_T, s_j]$
	$\overset{R_r}{- - - \rightarrow}$	
		$R_t \in_R \{0,1\}^{96}$ $M_1 = R_t \oplus \beta$ $\alpha = h(ID_T \oplus R_t \oplus R_r \oplus RID_i)$
$\beta = s_j(0:47) \parallel ID_T(48:95)$ $R_t = M_1 \oplus \beta$ $RID_i = (R_t - R_t mods_j + 1)(0:47) \parallel (R_t + s_j - R_t mods_j)(48:95))$ $\alpha' = h(ID_T \oplus R_t \oplus R_r \oplus RID_i)$ if $\alpha' \overset{?}{=} \alpha$, then $M_2 = h(\beta \oplus RID_i)$	$\overset{\alpha, M_1}{\leftarrow - - -}$	
$s_{j+1} \in_R \{0,1\}^{96}$, $M_3 = s_{j+1} \oplus R_t$ store $[ID_T, s_{j+1}, s_j]$	$\overset{M_2, M_3}{-/\rightarrow}$	The tag does not update the secret. It stores $[ID_T, s_j]$

TABLE V
MODIFIED PREPARATION PHASE OF THE TAG IMPERSONATION ATTACK.

Note 2: The desynchronization attack devised against the SM-2 protocol (v.s. II-B) can also be used against CYK, rendering the tag completely unmanageable by permanently desynchronizing it. As stated before the adversary must have the ability to manipulate messages exchanged between the tag and the reader during the 'Server Response' phase; i.e. Phases 6-7. More specifically, the attacker needs only to tamper with the secret update message M_3.

From the protocol, we know that M_3 is calculated, at the server, as the XOR of R_t and the new secret s_{j+1}. The tag authenticates the reader using (only) message M_2, and then it updates the secret key to s_{j+1}, which it extracts from M_3. The protocol does not protect the integrity of both messages (M_2, M_3), which, as in the case of SM-2, proves to be fatal.

The attacker needs to modify message M_3 by a random value f; so that the tag receives message $M_3' = M_3 \oplus f$. The length of f needs to be equal or less than that of M_3. As long as message M_2 remains intact the tag will authenticate the reader successfully and update its secret value to $M_3' \oplus R_t = s_{j+1} \oplus f$. Thus, the tag is fooled to store a different value than the one in the server, rendering any further protocol runs inoperative.

As stated previously, this attack is based solely on the linearity of the XOR and function and the fact that the protocol fails to protect the integrity of all its messages.

IV. COMMENTS AND CONCLUSIONS

In this article, we have provided a security analysis of two new and fairly recent, lightweight RFID authentication protocols. The first one by Song and Mitchell, is a mature work as it has well studied predecessors. The second is a more fresh idea by Cho et al. We show that there are efficient attacks against both the protocols. More precisely, there is a desyncronization attack against both protocols, where the attacker must be able to manipulate the communication between the honest reader and tag, while a passive attacker can impersonate a legitimate tag in the Cho et al. scheme.

We believe that the Cho et al. protocol is very weak and there is no simple fix that can make it secure, without redesigning the whole protocol. On the other hand, the scheme proposed by Song and Mitchell is much more well documented and we believe that it can be relatively easily protected against our desynchronization attack. Using well known techniques, like sending with M_3 the keyed hash digest $g_k(M_3)$, using the common secret key k, the tag can verify the integrity of M_3. However, we are still a little bit puzzled by the fact that the authors require the implementation of four different hash functions on the tag. This design choice is peculiar, especially if one takes into account a) the implementation costs, b) the difficulty of devising secure hash functions, suitable for lightweight tags (or secure hash functions in general) and c) the fact that two of the keyed hash functions have identical input and output requirements. We feel that a lightweight protocol should reuse as much of its hardware as possible, especially when it comes to costly crypto-IC.

REFERENCES

[1] OECD, "Radio-Frequency identification (RFID): drivers, challenges and public policy considerations," Organisation for Economic Co-operation and Development (OECD), Paris, Tech. Rep. DSTI/ICCP(2005)19/FINAL, Mar. 2006, last visited May 2011. [Online]. Available: http://www.oecd.org/dataoecd/57/43/36323191.pdf

[2] S. E. Sarma, S. A. Weis, and D. W. Engels, "Rfid systems and security and privacy implications," in *CHES*, ser. LNCS, B. S. K. Jr., Çetin Kaya Koç, and C. Paar, Eds., vol. 2523. Redwood Shores, CA, USA: Springer, Aug. 2003, pp. 1–19. [Online]. Available: http://dx.doi.org/10.1007/3-540-36400-5

[3] T. van Deursen and S. Radomirovi, "Attacks on RFID protocols," Tech. Rep. 310, 2008, last revised 6 Aug 2011, last visited May 2011. [Online]. Available: http://eprint.iacr.org/2008/310

[4] G. Avoine, E. Dysli, and P. Oechslin, "Reducing time complexity in RFID systems," ser. Lecture Notes in Computer Science, B. Preneel and S. Tavares, Eds., vol. 3897. Kingston, Canada: Springer-Verlag, Aug. 2005, pp. 291–306. [Online]. Available: http://dx.doi.org/10.1007/11693383_20

[5] S. Kumar and C. Paar, "Are standards compliant elliptic curve cryptosystems feasible on RFID?" Graz, Austria, Jul. 2006, workshop on RFID Security – RFIDSec 06.

[6] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol," in *Workshop on Information Security Applications*, ser. Lecture Notes in Computer Science. Jeju Island, Korea: Springer-Verlag, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00306-6_5

[7] P. Kitsos and Y. Zhang, Eds., *RFID Security: Techniques, Protocols and System-On-Chip Design*. Springer, 2009, isbn: 978-0-387-76480-1.

[8] Y. Oren and M. Feldhofer, "A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes," in *Proceedings of the 2nd ACM Conference on Wireless Network Security – WiSec'09*, D. A. Basin, S. Capkun, and W. Lee, Eds., ACM. Zurich, Switzerland: ACM Press, March 2009, pp. 59–68. [Online]. Available: http://dx.doi.org/10.1145/1514274.1514283

[9] G. Avoine and U. Information Security Group (GSI), "RFID security & privacy lounge," last visited May 2011. [Online]. Available: http://www.avoine.net/rfid/

[10] E. Rekleitis, P. Rizomiliotis, and S. Gritzalis, "A holistic approach to RFID security and privacy," in *1st International workshop on the security of the Internet of Things, SecIoT'10*, Tokyo, Japan, Nov. 2010.

[11] B. Song and C. J. Mitchell, "Scalable RFID security protocols supporting tag ownership transfer," *Computer Communications*, vol. 34, no. 4, pp. 556–566, Apr. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.02.027

[12] J. Cho, S. Yeo, and S. K. Kim, "Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value," *Computer Communications*, vol. 34, no. 3, pp. 391–397, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.02.029

[13] B. Song and C. J. Mitchell, "RFID authentication protocol for low-cost tags," in *ACM Conference on Wireless Network Security, WiSec'08*, V. D. Gligor, J. Hubaux, and R. Poovendran, Eds. Alexandria, Virginia, USA: ACM Press, Apr. 2008, p. 140147.

[14] P. Rizomiliotis, E. Rekleitis, and S. Gritzalis, "Security analysis of the song-mitchell authentication protocol for low-cost RFID tags," *Communications Letters, IEEE*, vol. 13, no. 4, pp. 274–276, 2009. [Online]. Available: http://dx.doi.org/10.1109/LCOMM.2009.082117

[15] S. Cai, Y. Li, T. Li, and R. H. Deng, "Attacks and improvements to an RIFD mutual authentication protocol and its extensions," in *Proceedings of the second ACM conference on Wireless network security*. Zurich, Switzerland: ACM, Mar. 2009, pp. 51–58. [Online]. Available: http://dx.doi.org/10.1145/1514274.1514282

[16] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Tapiador, T. Li, and Y. Li, "Vulnerability analysis of RFID protocols for tag ownership transfer," *Computer Networks*, vol. 54, no. 9, pp. 1502–1508, Jun. 2010.