Employing a Secure Cipher Does not Guarantee the Security of RFID Protocols

Masoumeh Safkhani, Nasour Bagheri, Pedro Peris-Lopez, Juan M. E. Tapiador

***Abstract*—It is a common practice to employ encryption functions to provide security for an application. On the other hand, it is well known that a bad usage of even a very strong cryptographic primitive can destroy the security of an application/scheme. Several RFID protocols have recently been proposed whose security reside on encryption primitives. Among these works we pay attention on the following ones: 1) Khan and Moessner proposed a new RFID authentication protocol [5]; 2) Kapoor and Piramuthu presented a single-tag and single-owner ownership transfer protocol [3] ; 3) Kapoor et al. proposed a multi-tag and multi-owner ownership transfer scheme in supply chains [4]; and 4) Zhou et al. presented multi-level RFID tag ownership and transfer in health-care environments [8]. In these four proposals the authors state that their schemes are secure against de-synchronization attacks. Nevertheless, in this article we present efficient de-synchronization attacks against the above mentioned protocols. Our attacks are based on the inappropriate use of the encryption function. The success probability of all the proposed attacks is almost one while the complexity is just one execution of the protocol.**

***Index Terms*—IEEEtran, journal, LaTeX, paper, template. RFID, Cryptanalysis, Authentication, Protocol, De-synchronization Attacks, Man-In-The-Middle.**

I. Introduction

A common RFID system includes three parties: tags, readers and a back-end database. The tag, which can be accessed by authorized readers, is attached to an item (i.e., person, animal, or object) and includes some information linked to the tag holder. The back-end database may be used to provide some extra storage or computational capabilities to the reader and also keeps a high level information related to the tag holder.

In an RFID system authentication protocols are used by readers and tags in order to authenticate each other. It commonly is a game-playing argument between both entities. In such game, one of the principals in the protocol sends a(some) challenge(s) to the other principal and then it verifies the response(s) received. The majority of RFID authentication protocols require the authentication process on both sides. This sort of protocols are named as mutual authentication protocols. However, in certain applications only one party is authenticated [5].

On the other hand, nowadays, RFID tags are increasingly being used to identify, track and sense ambient conditions of tagged items [3]. Most of these applications impose that at various moments during the tag life, the owner of a tag can change. Therefore we need a mechanism to transfer the ownership of tagged objects from the current owner to a new owner. This kind of mechanisms are known as ownership transfer schemes. Attempting to provide the required security in a RFID system, strong (or regrettably weak) mutual authentication and ownership transfer protocols, which rely on encryption or cryptographic hash functions, have appeared in the literature. In this research direction, Khan and Moessner [5] recently proposed a lightweight mutual authentication protocol, Kapoor and Piramuthu [3] presented a single-tag and single-owner ownership transfer scheme, Kapoor *et al.* [4] generalized the above proposal for multi-tag and multi-owner ownership transfer scheme and Zhou *et al.* [8] proposed a multi-level RFID tag ownership and transfer in health-care environments. The just mentioned schemes use encryption functions to offer an appropriate security level. Furthermore the protocol designers claim optimal security for their proposed protocols –including protection against de-synchronization states. Nevertheless, in this paper we present efficient de-synchronization attacks against these four protocols. The success probability of the proposed attacks is almost one and only requires one execution of the protocol.

The proposed attacks mainly exploit that an encrypted message is accepted by an entity without any validation of its origin and/or without any checking of its integrity. Hence, in a man-in-the-middle attack whether the adversary changes either the key K used in the encryption function or the cipher-text C (in the RFID context this second strategy is the simplest one since messages passed through the insecure radio channel), then the receiver will decrypt that altered message and thus the extracted plain-text P' would be incorrect. It must be noted that any change on K or C will not affect the proper working of the decryption function, but the output P' value would be a random value (in comparison to the correct P). In Table I we provide some encryption examples of same/related Ps with same/related Ks for the AES block cipher. As shown, any modification on any of the inputs (K or P) makes the output (C) –and consequently the decrypted plain-text (P')– random and unpredictable.

Paper Organization: We introduce Khan and Moessner protocol in Section II, explain its security analysis concerning the mutual authentication protocol, and finally show a de-synchronization attack. In Section III we present our security analysis of Kapoor and Piramuthu ownership transfer protocol and then the security of the enhancement scheme proposed by

Masoumeh Safkhani is with the Department of Electrical Engineering, Iran University of Science and Technology (IUST), Tehran, Iran.

Nasour Bagheri is with the Department of Electrical Engineering, Shahid Rajaee Teachers Training University, Tehran, Iran.

Pedro Peris-Lopez and Juan M. E. Tapiador are with the Computer Security Lab (COSEC), Carlos III University of Madrid, Leganes, Madrid.

TABLE I
SAMPLE ENCRYPTIONS WITH AES [1].

Plain text (in hex)	**Key** (in hex)	**Cipher text** (in hex)
00000000000000000000000000000000	00000000000000000000000000000000	66e94bd4ef8a2c3b884cfa59ca342b2e
00000000000000000000000000000001	00000000000000000000000000000000	58e2fccefa7e3061367f1d57a4e7455a
00000000000000000000000000000000	00000000000000000000000000000001	0545aad56da2a97c3663d1432a3d1c84
140f0f1011b5223d79587717ffd9ec3a	00000000000000000000000000000000	00000000000000000000000000000000
6c5a75d8a41802ad5a35818dd5abfddc	00000000000000000000000000000000	00000000000000000000000000000001
140f0f1011b5223d79587717ffd9ec3a	00000000000000000000000000000001	9e909d68c4c95f750feaf3f47934a69a

Kapoor et al. is also scrutinized in Section IV. In Section V we present the security analysis of Zhou *et al.* ownership transfer protocol. After this, in Section VI we present a discussion about the principles for designing secure cryptographic protocols and its non-conformity in the four protocols examined through this paper. Finally, Section VII concludes the paper.

II. DESYNCHRONIZATION ATTACK AGAINTS KHAN AND MOESSNER'S AUTHENTICATION PROTOCOL

Recently Khan and Moessner [5] have proposed a mutual authentication protocol for RFID systems, hence forth denoted by *KM*-protocol. A description of *KM*-protocol is depicted in Fig. 1. In this protocol (Key_{c1}, Key_{c2}) is a pair of keys for tags belonging to the same class and K_c represents the class key. $ID = ID_l \| ID_h$ is the unique identifier of a tag (lower and higher half respectively) and '$\|$' denotes concatenation. T_c symbolizes the authentication counter in the tag side and $T_{c_{stored}}$ denotes its corresponding record in the server. The authors state that $h(a, b)$ is a cipher or alternatively a hash function [5, Sec. III]. In the protocol calculation, they use $h^{-1}(a, b)$ but it is not possible to compute the inverse of a hash function and thus $h(a, b)$ must be an encryption function.

We present a roughly description of the protocol but details can be checked in the original paper. In *KM*-protocol, the tag is authenticated by the following exchanged of messages. First, the reader generates a random number R_r and sends $m_1 = R_r$ to the tag. Upon receiving m_1, the tag generates a random number R_t, increments T_c, computes $h_1 = h((R_r \| R_t) \oplus ID,\ Key_{c1})$ and $h_2 = h((R_r \| 0) \oplus (R_t \| R_t) \oplus ID \oplus T_c,\ Key_{c2})$, and finally sends $m_2 = k_c \| h_1 \| h_2$ to the reader. Then, the reader forwards $R_r \| m_2$ to the server. The server follows the following procedure: 1) it fetches the key pair that belongs to k_c; 2) it decrypts h_1 and h_2 with keys Key_{c1} and Key_{c2} respectively; 3) it reveals the upper part of ID as $ID_h = h^{-1}(h_1,\ Key_{c1}) \oplus (R_r \| 0)$; 4) it looks up ID_l and $T_{c_{stored}}$ from its database and reveals R_t as $R_t = h^{-1}(h_1,\ Key_{c1}) \oplus (R_r \| 0) \oplus (ID_h \| ID_l)$; 5) it reveals the tag counter as $T_c = h^{-1}(h_2,\ Key_{c2}) \oplus (R_t \| R_t) \oplus (R_r \| 0) \oplus (ID_h \| ID_l)$; 6) If $(T_c > T_{c_{stored}})$, it sets $m_3 = Authentic$. Otherwise $m_3 = Not\ \ Authentic$. Finally, the server sends m_3 to the reader.

A. De-synchronization Attack

Generally, in a de-synchronization attack the adversary forces the tag and the reader to update their shared values such that both entities will not be able to authenticate each other hence forth.

In *KM*-protocol each query sent to a target tag, either by a legitimate reader or an adversary, increments its counter T_c. Khan and Moessner [5] state that it would be possible to de-synchronize the tag and the reader if the adversary swamps the tag with many queries until its counter starts from zero again. To overcome this drawback, the authors assume that the authentication counter has a length l greater or equal to 32 bits (i.e., $|T_c| > 32$). Nevertheless, in this section, we present a different de-synchronization attack, which does not need to restart the counter to a zero value but forces the reader to set its counter to a value far ahead from the counter value in the tag side. The main observation is that the server only verifies the correctness of h_1 and does not verify the correctness of the received h_2. Hence the adversary could replace h_2 by any arbitrary value and it would be accepted by the server.

Our adversary, $\mathcal{A}$, is a man in the middle adversary which stays in the public radio channel between the tag and the reader and has the ability to eavesdrop, block or modify the messages exchanged between the reader and the tag, and vice versa. In this attack, when the tag sends $m_2 = k_c \| h_1 \| h_2$ to the reader, $\mathcal{A}$ intercepts m_2, generates a random number and assigns it to h_2, denoted by h_2'. Then $\mathcal{A}$ sends $m_2' = k_c \| h_1 \| h_2'$ to the reader, and finally the reader forwards $R_r \| m_2'$ to the server. Upon the reception of that message, the server: 1) fetches the key pair that belongs to k_c; 2) decrypts h_1 and h_2' with Key_{c1} and Key_{c2} respectively; 3) discloses the upper part of ID as $ID_h = h^{-1}(h_1,\ Key_{c1}) \oplus (R_r \| 0)$; 4) looks up ID_l and $T_{c_{stored}}$ from its database and reveals R_t as $R_t = h^{-1}(h_1,\ Key_{c1}) \oplus (R_r \| 0) \oplus (ID_h \| ID_l)$; 5) it discloses the tag counter as $T_c' = h^{-1}(h_2',\ Key_{c2}) \oplus (R_t \| R_t) \oplus (R_r \| 0) \oplus (ID_h \| ID_l)$; 6) If $(T_c' > T_{c_{stored}})$ the server sets $m_3 = Authentic$ and $T_{c_{stored}} = T_c'$. Finally the server sends m_3 to the reader.

After conducting the above described attack, the tag can not be authenticated by the reader in future sessions as explained below. Assuming that the tag has not been employed for a long time, T_c would be a small value (i.e., $T_c << 2^{l-1}$ where l is the bit length of T_c. As consequence of sending a random h_2', the expected value for the revealed value of T_c' (step 5 in the previous paragraph) is 2^{l-1}, where $|T_c'| = l$. Hence, the server authenticates the tag (i.e., $T_c' > T_{c_{stored}}$) and updates $T_{c_{stored}} = T_c' \approx 2^{l-1}$ while the T_c in the tag side is not so bigger value – if the adversary is not so lucky and $T_c' < T_{c_{stored}}$, she simply has to repeat the attack and try with another h_2' random value. Therefore, the adversary could force the server to set its $T_{c_{stored}}$ to a value far advance from the value of T_c in the tag side. Hence forth, the server would not authenticate the tag since $(T_c' < T_{c_{stored}})$. For instance, assuming $l = 64$, which satisfies the designer criteria regarding

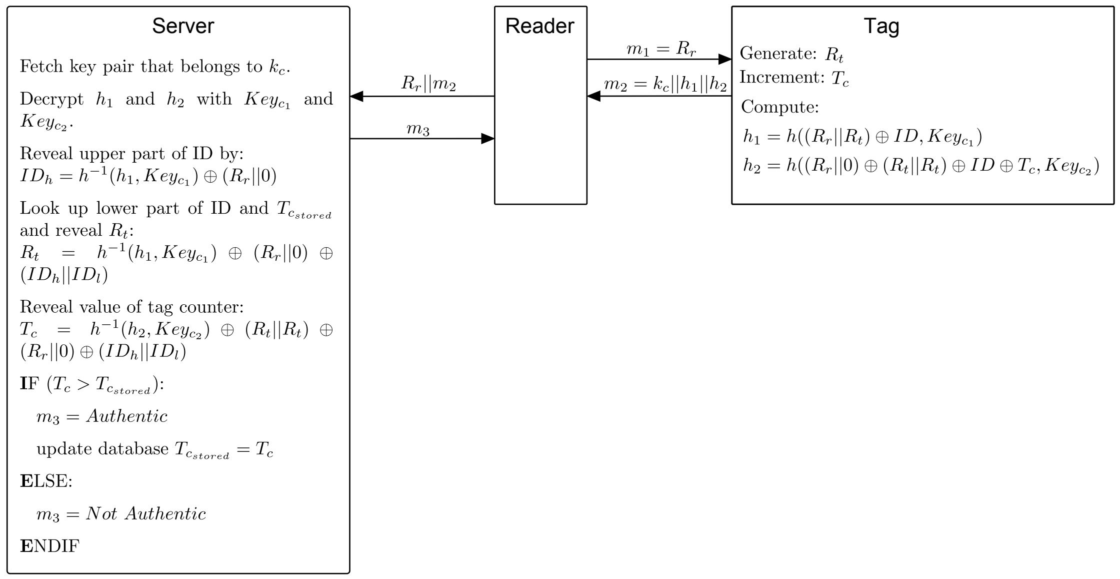
Server
Fetch key pair that belongs to k_c.
Decrypt h_1 and h_2 with Key_{c_1} and Key_{c_2}.
Reveal upper part of ID by:
$ID_h = h^{-1}(h_1, Key_{c_1}) \oplus (R_r||0)$
Look up lower part of ID and $T_{c_{stored}}$ and reveal R_t:
$R_t = h^{-1}(h_1, Key_{c_1}) \oplus (R_r||0) \oplus (ID_h||ID_l)$
Reveal value of tag counter:
$T_c = h^{-1}(h_2, Key_{c_2}) \oplus (R_t||R_t) \oplus (R_r||0) \oplus (ID_h||ID_l)$
IF ($T_c > T_{c_{stored}}$):
$m_3 = Authentic$
update database $T_{c_{stored}} = T_c$
ELSE:
$m_3 = Not\ Authentic$
ENDIF
Reader
$R_r||m_2$
m_3
$m_1 = R_r$
$m_2 = k_c||h_1||h_2$
Tag
Generate: R_t
Increment: T_c
Compute:
$h_1 = h((R_r||R_t) \oplus ID, Key_{c_1})$
$h_2 = h((R_r||0) \oplus (R_t||R_t) \oplus ID \oplus T_c, Key_{c_2})$

Fig. 1. The *KM*-protocol description [5].

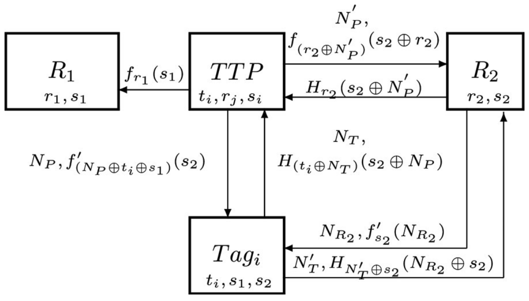
R_1
r_1, s_1
$f_{r_1}(s_1)$
TTP
t_i, r_j, s_i
$N'_P,$
$f_{(r_2 \oplus N'_P)}(s_2 \oplus r_2)$
$H_{r_2}(s_2 \oplus N'_P)$
R_2
r_2, s_2
$N_P, f'_{(N_P \oplus t_i \oplus s_1)}(s_2)$
$N_T,$
$H_{(t_i \oplus N_T)}(s_2 \oplus N_P)$
$N_{R_2}, f'_{s_2}(N_{R_2})$
$N'_T, H_{N'_T \oplus s_2}(N_{R_2} \oplus s_2)$
Tag_i
t_i, s_1, s_2

Fig. 2. The *KP*-protocol description [5].

the length of the counter ($l > 32$), we expect that after executing the proposed attack, $T_{c_{stored}}$ is updated to a value around 2^{63}. For this, we can fairly state that the tag would not be authenticated by the reader in the next-legitimate sessions as result of the unsynchronized counters (i.e., $T_{c_{stored}} >> T'_c$).

III. De-synchronization Attack agaínts Kapoor and Piramuthu's Ownership Transfer Protocol

Ownership transfer protocols are designed with the purpose of transferring ownership of an item that is RFID labelled. Kapoor and Piramuthu [3] have recently proposed an Ownership Transfer Protocol (OTP) with Trusted Third Party (TTP), hence forth denoted by *KP*-protocol. The protocol is depicted in Fig. 2, but for more details we urge the reader to consult the original paper [3, Sec. III]. Note that $f_k(.)$ and $f'_k(.)$ denote keyed (with key k) encryption functions.

In this protocol, upon receiving an ownership transfer (OT) request, the TTP generates a random nonce N_P and a secret key s_2 which will be shared between the tag and the new owner (R_2). It then computes $f'_{(N_P \oplus t_i \oplus s_1)}(s_2)$ and sends the tuple $(N_P, f'_{(N_P \oplus t_i \oplus s_1)}(s_2))$ to the tag, where s_1 is the secret shared between the tag and the current owner (R_1) and t_i is the secret shared between the tag and TTP. Upon receiving that message, the tag extracts s_2 from the encrypted message and replaces the current secret key s_1 by the new key s_2. In addition, the tag acknowledges by generating a random nonce N_T and sending the pair $(N_T, H_{(t_i \oplus N_T)}(s_2 \oplus N_P))$. Then, TTP informs R_1 that his privileges on this tag are being revoked. It simply sends a revoke message and a keyed encryption token $f_{r_1}(s_1)$. Next, TTP grants new owners (R_2) full permissions along with the privileges for the tag. The rest of the protocol can be followed from Fig. 2 but it is not relevant for our attack.

A. De-synchronization Attack

As described previously, once the tag updates its secret key from s_1 (old owner R_1) to s_2 (new owner R_2), it will not be accessible by R_1 any more, otherwise it would be vulnerable to a window attack in which for a fraction time both current and new owners could access to the tag [9]. On the other hand, the new secret s_2 is derived from the message $X = f'_{(N_P \oplus t_i \oplus s_1)}(s_2)$ sent by TTP (i.e., $s_2 = f'^{-1}_{(N_P \oplus t_i \oplus s_1)}(X)$). Nevertheless, the protocol does not include any checking mechanism to verify the origin (source) and correctness of the extracted s_2 value. That is, any value obtained from the tag calculations –decryption of X– is accepted without further checking and it is used as the new secret shared with the new owner. An adversary can exploit this fault to render a tag in a de-synchronized state and consequently the OT system. The procedure of the proposed attack is described below.

An adversary intercepts the pair $(N_P, f'_{(N_P \oplus t_i \oplus s_1)}(s_2))$ sent from TTP to the tag and replaces it by any desired pair (x, y) such that $(x, y) \neq (N_P, f'_{(N_P \oplus t_i \oplus s_1)}(s_2))$. The tag is thus cheated to update its current secret to $s'_2 = f'^{-1}_{(x \oplus t_i \oplus s_1)}(y)$. This new key s'_2 does not match any of the TTP records (new s_2 and old s_1). The probability of success in the proposed attack is $1 - 2^{-n+2}$, where n is the bit length of s_1. Hence, following this attack, the tag updates its secret key to a value that neither new owners nor TTP has knowledge of it and prevents its access in future-legitimate communications –that is, the tag is render inaccessible. The success probability of the given attack is almost one and the complexity is negligible.

In the above attack, one may argue that the message sent from the tag to TTP ($(N_T, H_{(N_T \oplus t_i)}(s'_2 \oplus N_P))$) would not be accepted by TTP because the adversary previously altered the tuple sent from TTP to the tag – in consequence it provoked an incorrect key derived by the tag – and the acknowledge tuple generated by the tag would not be valid with high probability. Although it gives the TTP the ability to discover potential attacks, however, it cannot react. In fact, to contact with the tag, TTP needs the tag's secret, which was just updated to s'_2 due to the attack, and it does not know it. Hence, the tag is de-synchronized forever and neither TTP nor the owners can access it any more.

IV. Desynchronization Attack againts Kapoor *et al.*'s Ownership Transfer Protocol

While in the previous Section we have analysed the security of a protocol which transfers the ownership of a single tag to a new owner reader, it is not uncommon scenarios in which the tag ownership is shared among multiple owners, and its dual, the case of an object labelled with multiple tags. Motivating by this kind of scenarios, Kapoor *et al.* [4] proposed a shared ownership transfer protocol (multiple-owners) and a protocol for inclusion and exclusion of tags in a multiple-tagged object. The scheme, called *KZP*-protocol in short, is the object of our security analysis. We sketch the protocol in Fig. 3 and interested readers can consult the original paper for details [4, Sec. 3.1].

Similarly to the *KP*-protocol, in this protocol upon receiving an ownership transfer (OT) request, the TTP generates a random nonce N_P and a new key s_2. This new secret key will be shared between the tag and the new owner R_2 (or a group of owners $R_{21}, R_{22}, \ldots, R_{2M}$). Finally, TTP computes $f'_{(N_P \oplus t_i \oplus s_1)}(s_2)$ and sends the pair $(N_P, f'_{(N_P \oplus t_i \oplus s_1)}(s_2))$ to the tag, where s_1 is the secret shared between the tag and the current owner R_1 (or the group $R_{11}, R_{12}, \ldots, R_{1M}$) and t_i is the secret shared between the tag and the TTP. Once that message is received, the tag extracts s_2 from the encrypted message and updates its secret key by replacing the current key s_1 by the new secret key s_2. The tag then acknowledges the server by generating a random nonce N_T and sending the token $(N_T, H_{(N_T \oplus t_i)}(s_2 \oplus N_P))$. After that, TTP informs the current owners $R_{11}, R_{12}, \ldots, R_{1M}$ that their privileges on this tag were revoked. It sends both a revoke message and a cryptographic token $f_{r_{1i}}(s_1)$ to each of the owners in the group, where r_{1i} is a secret shared between R_{1i} and TTP. Next, TTP grants the new owners $(R_{21}, R_{22}, \ldots, R_{2M})$ full permissions along with any privileges for the tag. The rest of the protocol can be followed from Fig. 3, but this part has not influence on our attack.

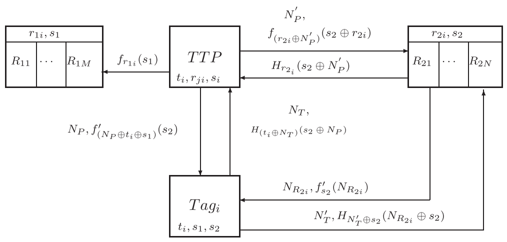
r_{1i}, s_1
R_{11}
$\cdots$
R_{1M}
$f_{r_{1i}}(s_1)$
TTP
t_i, r_{ji}, s_i
$N'_P,$
$f_{(r_{2i} \oplus N'_P)}(s_2 \oplus r_{2i})$
$H_{r_{2_i}}(s_2 \oplus N'_P)$
r_{2i}, s_2
R_{21}
$\cdots$
R_{2N}
$N_P, f'_{(N_P \oplus t_i \oplus s_1)}(s_2)$
$N_T,$
$H_{(t_i \oplus N_T)}(s_2 \oplus N_P)$
Tag_i
t_i, s_1, s_2
$N_{R_{2i}}, f'_{s_2}(N_{R_{2i}})$
$N'_T, H_{N'_T \oplus s_2}(N_{R_{2i}} \oplus s_2)$

Fig. 3. The *KZP*-protocol description [4].

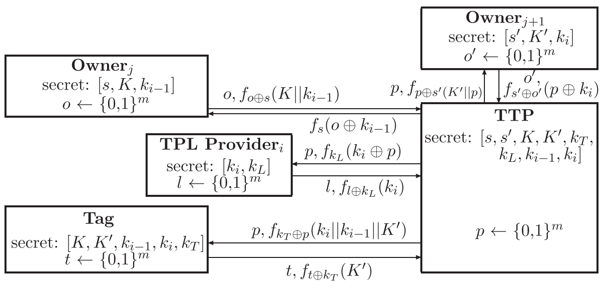
Owner$_j$
secret: $[s, K, k_{i-1}]$
$o \leftarrow \{0,1\}^m$
$o, f_{o \oplus s}(K || k_{i-1})$
$f_s(o \oplus k_{i-1})$
Owner$_{j+1}$
secret: $[s', K', k_i]$
$o' \leftarrow \{0,1\}^m$
$p, f_{p \oplus s'(K'||p)}$
$o',$
$f_{s' \oplus o'}(p \oplus k_i)$
TTP
secret: $[s, s', K, K', k_T, k_L, k_{i-1}, k_i]$
TPL Provider$_i$
secret: $[k_i, k_L]$
$l \leftarrow \{0,1\}^m$
$p, f_{k_L}(k_i \oplus p)$
$l, f_{l \oplus k_L}(k_i)$
Tag
secret: $[K, K', k_{i-1}, k_i, k_T]$
$t \leftarrow \{0,1\}^m$
$p, f_{k_T \oplus p}(k_i || k_{i-1} || K')$
$t, f_{t \oplus k_T}(K')$
$p \leftarrow \{0,1\}^m$

Fig. 4. The *ZYP*-protocol description [8].

A. De-synchronization Attack

In *KZP*-protocol, as occurred in *KP*-protocol, once the tag executes the updating of its secret key from s_1 to s_2, R_1 cannot gain access to the tag any more, otherwise it will be vulnerable to window attack [9]. As an important drawback in the design of the protocol, the new key s_2 is exclusively derived from the messages that was supposedly sent by the OTP. Let assume that OTP sends $X = f'_{(N_P \oplus t_i \oplus s_1)}(s_2)$, then the key would be obtained by decrypting the received message $s_2 = f'^{-1}_{(N_P \oplus t_i \oplus s_1)}(X)$. Nevertheless, the receiver (tag) can not check neither the correctness of the new key nor the source of this message – it incorrectly assumes that all messages come from a honest OTP. These bad properties can be exploited by an attacker against the protocol as explained below.

Simply, and due to the resemblance with the *KP*-protocol, the adversary could follow the same strategy. She intercepts the tuple $(N_P, f'_{(N_P \oplus t_i \oplus s_1)(s_2)})$ sent by a honest TTP and changes this by a random pair x, y completely different to the values intercepted (i.e., $(x, y) \neq (N_P, f'_{(N_P \oplus t_i \oplus s_1)(s_2)})$). As consequence of this, the tag would update its secret key ($s'_2 = f'^{-1}_{(x \oplus t_i \oplus s_1)(y)}$) with the key sent by the supplanted TTP – the attacker. So, there would not be a match between s'_2 and the records (s_2, s_1) kept by the TTP. The probability of success in this attack is almost one $(1 - 2^{-n+2})$, being n the bit length of s_i, and it renders the tag inaccessible to the honest TTP in future communications.

V. De-synchronization Attack againts Zhou *et al.*'s Ownership Transfer Protocol

Zhou, Yoon and Piramuthu [8] have considered a case that includes third party logistics providers and their role in temporary ownership transfer of items in a distributed health-care supply chain environment. Hence forth this protocol is denoted by *ZYP*-protocol. In this protocol, the authors consider the simultaneous RF access of an RFID-tagged object by the object's owner and a temporary owner which could be the Third Party Logistics (TPL) provider. The protocol counts with two keys to access the tag, a main key (K) for the owner and a sub-key (k_i) for the TPL provider. The protocol is depicted in Fig. 4, but for more details we urge the reader to consult the original paper [8, Sec. 2.4]. Note that $f_k(.)$ denotes keyed (with key k) encryption functions and TPL provider is assumed to be the one who transport the item between the two owners.

This protocol, includes four loops. The first loop occurs between the current owner and the TTP . This is used by the current owner to inform the TTP of the status change in the RFID-tagged object with the main key K and the sub-key k_{i-1} (i.e., this tagged object either changes ownership or goes through a provider or both). The second loop includes the TTP and the next owner, where the TTP generates a new main key (K') and informs the latter of that new key. In response, the new owner generates a new sub-key k_i and sends it to TTP in an encrypted token (i.e., $f_{s\oplus o}(p \oplus k_i)$ where p is a random number generated by TTP, o is a random number generated by the new owner and s denotes the shared key between the TTP and the new owner). The third and fourth loops are used by the TTP to inform the TPL provider and tag respectively of the new key(s) for the tag. It must be noted, that the first loop closed at the end of protocol.

To access the tag in the above mentioned protocol, the i^{th} owner is enforced to have knowledge of both main-key and sub-key for communicating with the tag, which can be replaced by the composite key, which might be obtained by computing the bitwise XOR of both values: $K \oplus k_i$ [8, Lats Para. Sec. 2.4].

A. De-synchronization Attack

As described previously, once the tag updates its secret key from k_{i-1} (old owner R_{i-1}) to k_i (new owner R_i), it will not be accessible by R_i any more, otherwise it would be vulnerable to a window attack in which for a fraction time both current and new owners could access to the tag [9]. On the other hand, the new secret k_i is derived from the message $X = f_{s\oplus o}(p \oplus k_i)$ sent by R_i (i.e., $k_i = f^{-1}_{s\oplus o}(X)$). Nevertheless, the protocol does not include any checking mechanism to verify the origin and correctness of the extracted k_i value. That is, any value obtained from the TTP calculations –decryption of X– is accepted without further checking and used as the new secret, which is expected to have been shared with R_i. An adversary could exploit this fault to render a tag in a de-synchronized state and consequently the OT system. The procedure is described below.

An adversary intercepts the pair $(o, f_{s\oplus o}(p \oplus k_i))$ sent from the i^{th} owner to the TTP and replaces it by any chosen pair (x, y) such that $(x, y) \neq (o, f_{s\oplus o}(p \oplus k_i))$. For this, the TTP is deceived to force the tag to update its current sub-key into $k'_i = f^{-1}_{s\oplus x}(y)$. This new key k'_i does not match any of the owners' sub-keys related to this tag (new k_i and old k_{i-1}). The probability of success in this proposed attack is $1 - 2^{-n+2}$, where n represents the bit length of k. Therefore, following this attack, the tag would update its secret key to a value that neither new owners nor old owner would have knowledge of it and would render it inoperative for future-legitimate communications. The success probability of the given attack is almost one and the attack only requires the intervention in one protocol execution. We emphasize here that TTP has the knowledge of the new sub-key $k'_i = f^{-1}_{s\oplus x}(y)$ and it could update the tag sub-key. Nevertheless, it is not clear whether it is viable for some of the owners to convince the TTP to change the tag sub-key while she can not prove her privilege over the tag – the supposed owner does not have knowledge of the mentioned sub-key. Note that the main-key and the sub-key are mandatory for communicating with the tag. Even in the case that the TTP accepts to restart the whole protocol under the above assumption, it would be a serious pitfall for the protocol due to the lack of knowledge about one of the keys is disregarded, which contradicts the initial assumptions of the protocol and its claimed security properties.

VI. Discussion

RFID protocols are prone to security pitfalls of every kind. Among theses, de-synchronization attacks are the most extended and commonly the first reported against new proposals. Nevertheless, the attacks presented in this article could have been prevented whether well-known principals and guidelines for the design of cryptographic protocols would have been followed. For instance, in 1996 Abadi and Needham already introduced principles and guidelines for designing secure cryptographic protocols [2]. More precisely, they state eleven principles that we summarize bellow:

- **Principle 1:** Every message should say what it means.
- **Principle 2:** The conditions applicable to a message have to be clearly set out in order its acceptability or not can be evaluated by an external reviewer.
- **Principle 3:** If the identity of a principal is essential to the meaning of a message, it should be included in the message.
- **Principle 4:** Be clear about why encryption is being done. Encryption is not cheap and is not synonymous with security.
- **Principle 5:** When a principal signs a token that has already been encrypted, it should not be inferred that the principal knows the content of the message. Contrary to this, we can infer that the principal knows the content of the message when a principal signs a message and then encrypts it.
- **Principle 6:** Be clear about what properties you are assuming about nonces.
- **Principle 7:** If a predictable quantity (e.g., counter) is used, it should be protect to avoid attacks such as replay attacks.
- **Principle 8:** If timestamps are used, the differences between clocks at various entities must be much less than the allowable age of a message deemed to be valid.
- **Principle 9:** Recent use of a key does not guarantee to be enough fresh neither its security.

- **Principle 10:** The encoding mechanism has to be public and well-known.
- **Principle 11:** The protocol designer should know which trust relations her protocols depends on, and why the dependence is necessary.

Among them, appropriate consideration of Principle 3 can help to guarantee the integrity of the received message. On the other hand, it is well known that encryption by itself does not guarantee the integrity of the received message [7]. A good example of that could be One-Time-Pad (OTP), an encryption system with perfect security. In OTP if you change any bit of the output, a bit of the retrieved plain text will change, and the recipient has no way to detect this. As shown in previous sections, the four analyzed protocols miss the above mention considerations being susceptible to desynchronization attacks. Taking into consideration Principle 3, suggested by Abadi and Needham, could have avoided the problem. More precisely, describing the original protocols in general terms and assuming that two entities (A and B) share two keys (K_1 and K_{new}) the following message would be exchanged for the ownership transfer:

$$\begin{aligned} A \quad \rightarrow \quad & B: X = f_{K_1}(K_{new}) \\ & B: K_1 = K_{new} \end{aligned}$$

where f symbolizes an encryption function.

Nevertheless, the sending of X message neither guarantees its origin nor its integrity. Essentially, under this design assumption, the receiver would be forced to accept as valid all the messages received. This fault has been exploited in the proposed attacks.

Taking into account the considerations of Principle 3, the attacks could have been prevented by adding the sender signature at the end of the message in order to guarantee the source of the message and also provide the message integrity checking:

$$\begin{aligned} A \quad \rightarrow \quad & B: X = f_{K_1}(K_{new}) \| Y = HMAC(K_1, K_{new}) \\ & B: K_1 = K_{new} \end{aligned}$$

where $HMAC$ symbolizes a Hash-based Message Authentication Code and $\|$ represents the concatenation operation. When A sends this message, B can discover K_{new}, know that A signed the message and check the integrity of K_{new}. Assuming $HMAC$ complaints with the RFC 2104 (i.e. $HMAC(K, text) = H(K \oplus opad, H(K \oplus ipad, text)$, where H and K denote a cryptographic hash function and a secret key respectively (please review [6] for more details), it means that B is certain that A knows K_{new}.

Summarizing, the attacks presented in this article can be prevented with minor changes in the initial designs. The design of a new cryptographic protocol is quite challenging and the designers should check its conformance with well-known principles of design – formal methods are also useful tools. These principles are neither necessary nor sufficient but are very helpful to prevent straightforward errors.

Remark 1: It must be noted that just using the message signature to fix the vulnerability of the analysed protocols would not be enough. For instance, with only including signature of the message in the revised protocol, it would be possible to mount a desynchronization attack as described below:

1) When A sends to B message $X = f_{K_1}(K_{new}) \| HMAC(K_1, K_{new})$, the adversary logs the message and blocks it.
2) Since the ownership transfer has not been finished successfully, A sends to B a new message $X' = f_{K_1}(K'_{new}) \| HMAC(K_1, K'_{new})$.
3) Adversary intercepts the new message X' and replaces it by the logged message $X = f_{K_1}(K_{new}) \| HMAC(K_1, K_{new})$.
4) B receives X and updates K_1 to K_{new} while A expects K'_{new}.

The same attack is applicable to the original version of KP, KZP and ZYP protocols. The complexity of this attack over these schemes is two protocol executions and the success probability is also almost 1. A solution could be keeping the new key fixed but randomizing sessions by nonces contributed by both protocol parties.

VII. Conclusion

In this article we analyze the security of four protocols that have been recently proposed for RFID systems and whose authors claim to be secure against common RFID attacks – including expressly desynchronization attacks. Nevertheless, we show efficient desynchronization attacks against this range of schemes, having high success probability (~ 1) and negligible complexity (one protocol execution). As shown in Section VI, the authors missed well-know principles for designing cryptographic protocols. Therefore, new security protocols should be checked from an informal (using well-established principles and guidelines) and a formal perspective (e.g., BAN logic).

Acknowledgements

This work was supported by the MINECO grant TIN2013-46469-R (SPINY: Security and Privacy in the Internet of You).

References

[1] AES Calculator. http://testprotect.com/appendix/AEScalc.

[2] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Trans. Softw. Eng.*, 22(1):6–15, Jan. 1996.

[3] G. Kapoor and S. Piramuthu. Single RFID Tag Ownership Transfer Protocols. *IEEE Trans. on Sys., Man, and Cyber. PART C: Applications and Reviews*, 42(2):164–173, 2012.

[4] G. Kapoor, W. Zhou, and S. Piramuthu. Multi-tag and multi-owner RFID ownership transfer in supply chains. *Decision Support Systems*, 52(2):258–270, 2011.

[5] G. Khan and M. Moessner. Secure authentication protocol for rfid systems. In 20^{th} *International Conference on Computer Communications and Networks (ICCCN) 2011*, pages 1–7, 2011.

[6] H. Krawczky, M. Bellare, and R. Canetti. Network working group: RFC 2104. pages 1–12, 1997.

[7] H. Shacham, D. Boneh, and E. Rescorla. Client-side caching for TLS. *ACM Trans. Inf. Syst. Secur.*, 7(4):553–575, 2004.

[8] W. Zhou, E. J. Yoon, and S. Piramuthu. Simultaneous multi-level RFID tag ownership & transfer in health care environments. *Decision Support Systems*, doi:10.1016/j.dss.2012.04.006, 2012.

[9] Y. Zuo. Changing hands together: a secure group ownership transfer protocol for RFID tags. pages 1–10, 2010.

Masoumeh Safkhani is Assistant Professor at Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran. She is the author of 10 articles in cryptology. Her current research interests include RFID security.

Nasour Bagheri is Assistant Professor at Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran. He is the author of over 50 articles in information security and cryptology.

Pedro Peris-Lopez is Visiting Lecturer in the Computer Security (COSEC) Lab at Universidad Carlos III de Madrid, Spain. He holds a M.Sc. in Telecommunications Engineering and a Ph.D. in Computer Science from Universidad Carlos III de Madrid. His research interests are in the design and analysis of cryptographic protocols and primitives and in lightweight cryptography. His current research is focused on Radio Frequency Identification Systems (RFID) and Implantable Medical Devices (IMD). In these fields he has published many papers over the last years in specialized journals and conference proceedings.

Juan E. Tapiador is Associate Professor of Computer Science in the Computer Security (COSEC) Lab at Universidad Carlos III de Madrid, Spain. Prior to joining UC3M, he was Research Associate at the University of York, UK. His work back there was funded by the ITA project (www.usukita.org), a joint effort between the UK Ministry of Defence and the US Army Research Lab led by IBM. His main research interests are in computer/network security and applied cryptography. He holds a M.Sc. (2000) and a Ph.D. (2004) in Computer Science from the University of Granada.