KP+ : Fixing Availability Issues on KP Ownership Transfer Protocols

Jorge Munilla

Abstract—Ownership Transfer Protocols for RFID allow transferring the rights over a tag from a current owner to a new owner in a secure and private way. Recently, Kapoor and Piramuthu have proposed two schemes which solve most of the security weaknesses detected in previously published protocols. However, this paper reviews this work and points out that such schemes still present some practical and security issues. We then propose some modifications in these protocols that overcome such problems.

Index Terms—RFID, privacy, unlinkability, DoS, forward secrecy, de-synchronization, protocol failure.

I. INTRODUCTION

RADIO Frequency Identification (RFID) is a well-established wireless technology for inventory, retail and supply-chain management. However, this technology faces different risks such as lack of privacy or confidentiality, malicious traceability and loss of data integrity, which can only be prevented with the implementation of security mechanisms that take into account its special characteristics: vulnerabilities of radio channel, power-constrained devices, low-cost tags with limited functionalities and data promiscuously transmitted when excited by being in close proximity to the reader [1].

Ownership Transfer Protocols (OTPs) allow the secure transfer of the (digital) ownership of a tag from a current owner to a new owner. Thus, three different roles or entities are always present in an OTP: the item or tag \( T \) whose rights are going to be transferred, the seller or Current Owner, who has the initial control of \( T \), and the buyer or New Owner, who will have the control of \( T \) when the protocol succeeds. In order to prevent previous owners can access the tag once it has been transferred, two different mechanisms are usually used: the presence of a Trusted Third Party (TTP), which coordinates the transaction, and the assumption of an Isolated Environment (IsE), where, after the private information has been transferred, the new owner can update the keys without being eavesdropped by the previous owner. Both schemes make sense depending on the application [2]. The first provides higher security for strong adversarial scenarios and the second is more appropriate when tags belong to independent authorities or companies.

The first works dealing with Ownership Transfer in the RFID framework were published in 2005 by Molnar et al. [3] and Saito et al. [4]. Recently, Kapoor and Piramuthu have reviewed these and other subsequent proposals [5]–[8] and have proposed two new schemes [9], based on TTP and IsE respectively. A variant of these protocols for multiple tags have also been published [10]. Other OTPs can be found in the literature (e.g., [11]–[15]) but many of them present flaws or vulnerabilities [16], [17]. Thus, in this letter, we review Kapoor and Piramuthu’s schemes, which are claimed to be more secure than those currently existing and yet just as lightweight, and we will show that although they address most of the problems encountered in previous proposals, they still raise other practical and security issues which should be corrected. Thus, our goal in this letter is to propose enhanced versions of these protocols, KP+, that with slight modifications, overcome the mentioned problems. We consider that it is important that potential implementers of the prominent KP protocols know the results of the analysis conducted in this paper and hope that it can help in the development of new designs.

II. KAPOOR AND PIRAMUTHU’S PROTOCOLS

These schemes use two keyed encryption (key \( k \)) functions: \( g_k \), between the high-level entities, and \( f_k \), between the tag and the other entities; and a secure hash function \( H_k() \). In the description of the protocols, \( \mathcal{T} \) will stand for the tag which is going to be transferred, and for the sake of simplicity, we will use \( \mathcal{R}1 \) and \( \mathcal{R}2 \) to refer to the readers of the current and the new owner respectively.

A. Kapoor and Piramuthu’s Protocol with TTP

In the KP protocol with Trusted Third Party, \( \mathcal{TTP} \) shares static secret keys \( r_1 \) and \( r_2 \) with \( \mathcal{R}1 \) and \( \mathcal{R}2 \) respectively, and a secret key \( t_i \) with \( \mathcal{T} \), different for each tag. Additionally, \( \mathcal{TTP} \) knows the key \( s_1 \) that \( \mathcal{T} \) currently shares with \( \mathcal{R}1 \), and it will generate the key \( s_2 \) that \( \mathcal{T} \) will share with \( \mathcal{R}2 \). This protocol is accomplished as follows (see Fig. 1):

S.1) Upon receiving an Ownership Transfer Request, \( \mathcal{TTP} \) generates a random nonce \( N_P \) and a new key \( s_2 \), and authenticates itself to \( \mathcal{T} \) by sending \( f(N_P \oplus t_i \oplus s_1)(s_2) \) along with \( N_P \).

\[ \mathcal{TTP} \rightarrow \mathcal{T}: \, N_P, \, f(N_P \oplus t_i \oplus s_1)(s_2) \]

S.2) \( \mathcal{T} \) checks the received message. If it is correct, \( \mathcal{T} \) updates \( s_1 \) to \( s_2 \), and acknowledges it by generating a random nonce \( N_T \) and using the one-way hash \( H \) with this value.

\[ \mathcal{T} \rightarrow \mathcal{TTP}: \, N_T, \, H(t_i \oplus N_T)(s_2 \oplus N_P) \]

S.3) \( \mathcal{TTP} \) informs the current owner (\( \mathcal{R}1 \)) that his privileges are being revoked by sending a value computed with the keyed cryptographic function (along with a simple
B. Kapoor and Piramuthu’s Protocol without TTP

This assumes a secure channel between RA and RB. The protocol is presented in Figure 2 and described below.

S.1) Upon receiving a request for Ownership Transfer, RA generates a fresh random number N_RA, computes N_RA ⊕ s1, where s1 is the key that RA shares with T, and sends the result to RB over a secure channel.

RA → RB: N_RA ⊕ s1

S.2) RA sends the same information to T but encrypted with s1.

RA → T: f_s1(N_RA ⊕ s1)

S.3) T generates two fresh random numbers: N_T and N_T'; and computes the value N = N_RA ⊕ N_T. Then, T randomly flips one bit in N, creating N'. T sends the following messages to RB:

T → RB: N_T ⊕ s1, N_T', f_s1(N_T ⊕ N_T'), H_s1(N_T ⊕ N_T')

S.4) Now, both T and RB know N. Knowing N, RB uses a brute force technique on f_s1(N_T ⊕ N_T') to determine N', and checks the computed result with the hash value H_s1(N_T ⊕ N_T'). Then, RB generates a new key s2 and sends the following message to T:

RB → T: f_s2(N_T ⊕ s2)

S.5) The previous step is repeated after a predetermined time period until T acknowledges receipt of the new key, by using it with the hash function.

T → RB: H_s2(N_T ⊕ s2)

S.6) RB sends f_s2(N_T ⊕ s2) to acknowledge receipt of the message in the previous step.

RB → T: f_s2(N_T ⊕ s2)

If the tag does not receive this within a predetermined amount of time, the process is repeated from the beginning.

III. CRYPTANALYSIS

A. Desynchronization attack on KP Protocol with TTP

According to the authors, the message {N_P, f_s1(N_R1 ⊕ s1)} in Step 1, authenticates the TTP to the tag, which updates s1 to s2. However, this is not correct and an adversary can send forged messages that make T update its key to a fake value s_A, causing desynchronization.
Proof. Let $A$ be an adversary that, impersonating TTP, sends any two values “$N_A$, $F_A$” to $T$ in Step 1. Then, $T$ will decrypt $F_A$ and update $s_1$ to $s_a$, with $s_a = f_{f_1^{-1}} (F_A)$. □

B. DoS attacks on KP Protocol without TTP

The values sent by $T$ in Step 3,
\begin{align*}
\{N_T \oplus s_1, N'_T, f_{(N'_T \oplus N_T')} (N' \oplus N'_T), H_{(N'_T \oplus N_T')} (N' \oplus N'_T)\}
\end{align*}
do not provide integrity on $N'_T$, which causes that an adversary can modify intercepted messages to generate new forged messages that will be accepted by $R$, causing the protocol to go into an endless loop.

Proof. Let $A = N_T \oplus s_1$, $B = N_T$, $C = f_{(N'_T \oplus N_T')} (N' \oplus N'_T)$ and $D = H_{(N'_T \oplus N_T')} (N' \oplus N'_T)$ be valid messages intercepted by an adversary $A$ in Step 3. For any new value $A_A$, the adversary generates and sends (Man In The Middle Attack) a new set of values:
\begin{align*}
\{A_A, B_A = B \oplus A \oplus A_A, C, D\}.
\end{align*}

These values will be accepted by $R$, since $A_A \oplus s_1 \oplus B_A = N_T \oplus N_T$. Thus, the protocol continues normally but $R$ computes an incorrect value $N' = f_{flip} (N_{R1} \oplus N_T \oplus \Delta) = flip (N_{R1} \oplus N_T) \oplus \Delta = N' \oplus \Delta$, with $\Delta = A_A \oplus A$. As a result, $T$ and $R$ assume different values for $N'$ and, according to the description of the protocol, steps 3 will be repeated indefinitely (because tag cannot acknowledge receipt of the new key in Step 4). □

IV. KP+

A. KP+ with TTP

The availability problem of KP protocol described in Section III-A is caused because the flow in Step 1 does not have any authentic TTP to $T$. A new flow whose computation includes session (fresh) randomness provided by the tag must be added, and only after this flow is checked must the tag update its key. We propose this flow takes place between Step 2 and Step 3, while the rest of the protocol remains unchanged (see Fig. 3):

S.2-3) Upon receiving the message $H_{(s_1 \oplus N_T)} (s_2 \oplus N'_T)$, that authenticates $T$ and $N_T$ that provides randomness for this session, TTP computes and replies with $f_{s_2} (s_1 \oplus N_T)$.

\begin{align*}
\text{TTP} \rightarrow \text{T}: f_{s_2} (s_1 \oplus N_T)
\end{align*}

$T$ checks if this is correct and if so, updates $s_1$ to $s_2$. The rest of the protocol remains unchanged. The computation of $f_{s_2} (s_1 \oplus N_T)$ proves the authorship of TTP since it requires the knowledge of $s_1$ and $s_2$, and the use of $N_T$ guarantees its participation in this particular session (preventing replay attacks). Note also that the option that $T$ keeps $s_1$, without updating to $s_2$ until Step 6, when it receives the confirmation from $R$, would not prevent replay attacks with messages (exchanged in Step 1 and Step 6) from interrupted (unsuccessful) sessions.

B. KP+ without TTP

The DoS problem of KP without TTP (Section III-B) is solved by guaranteeing the integrity of $N'_T$ so that it cannot be modified by the adversary without affecting the validity of the other encrypted values. Thus, we propose here to change Step 3 as follows (note that $N'_T$ is not involved in any other flow):

S.3) Upon receiving $f_{s_1} (N_{R1} \oplus s_1)$ from $R$, $T$ generates two random numbers $N_T$ and $N'_T$, and computes $N = N_T \oplus N_{R1}$. Then, $T$ randomly flips one bit in $N$, creating $N'$ and use it to compute $f_{N'} (N'_T)$ and $H_{N'} (N'_T)$. Then, $T$ sends to $R$ the following message:

\begin{align*}
\text{T} \rightarrow \text{R2}: N_T \oplus s_1, N'_T, f_{N'} (N'_T), H_{N'} (N'_T)
\end{align*}

This new flow is simpler than the original and avoids that new valid fake messages can be generated. If $N_T \oplus s_1$ and/or $N'_T$ are modified, new values $f_N (N'_T)$ and $H_{N'} (N'_T)$ must be computed; i.e. previously computed values cannot be reused.

V. Conclusion

We have proven that the two protocols recently defined by Kapoor and Piramuthu suffer from flaws in their design that allow attackers to break the regular behavior of the system by means of desynchronization or denegation of service attacks. This fact is of vital importance in ownership transfer protocols because they are closely related to commercial transactions. Since these protocols were originally designed to overcome security weaknesses of their predecessors, we have proposed modifications in order to fix the problems.
ACKNOWLEDGMENT

REFERENCES


