

Enhanced Ownership Transfer Protocol for RFID in an Extended Communication Model

Jorge Munilla, Alberto Peinado, Guoming Yang and Willy Susilo

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 overcome most of the security weaknesses detected in previously published protocols. Still, this paper reviews that work and points out that such schemes still present some practical and security issues. In particular, they do not manage to guarantee the privacy of the new owner without the presence of a Trusted Third Party, and we find that the assumed communication model is not suitable for many practical scenarios. We then propose here a lightweight protocol that can be used in a wider range of applications, and which incorporates recently defined security properties such as Tag Assurance, Undeniable Ownership Transfer, Current Ownership Proof and Owner Initiation. Finally, this protocol is complemented with a proposed Key Change Protocol, based on noisy tags, which provides privacy to the new owner without either resorting to a Trusted Third Party or assuming an Isolated Environment.

Index Terms—RFID, ownership transfer, privacy, key change, authentication.

I. INTRODUCTION

RADIO Frequency Identification (RFID) is a technology that is widely deployed for supply-chain and inventory managements, retail operations and more generally for automatic identification. The advantages of RFID over barcode technology are that it is wireless, and thus there is no necessity of direct line-of-sight, and they can be interrogated at greater distances, faster and concurrently [1].

Typical RFID architecture involves three main components: i) Tags or transponders, which are electronic data storages that are attached to the objects to be identified; ii) Readers or interrogators, which manage tag population, read data from and write data to tags; and iii) a Back-end Server, which is a trusted entity that exchanges tag information with the readers and processes these data according to the specific intended application. Tags are passive; i.e. they do not have any kind of battery and receive the energy that they need from the reader. Thus, tags are inactive till they pass through the electromagnetic field generated by a reader which is tuned to the same frequency.

To promote the adoption of RFID technology and to support interoperability, EPCGlobal [2] and the International Organization for Standards (ISO) [3] have been actively engaged in defining standards for tags, readers and the communication

protocols. EPC Class-1 Generation-2 (EPCGen2) is the ratified standard for low cost tags which work in the UHF band (860-960 MHz).

Although initial designs of RFID identification protocols focused on performance with little attention being paid to resilience and security, this technology has matured, and nowadays it has found use in many applications that require the implementation of security mechanisms which: i) take into account its special characteristics such as vulnerabilities of the radio channel, power-constrained devices, low-cost tags with limited functionalities and reply upon request; and ii) make them resistant to the different risks that they face such as lack of privacy or confidentiality, malicious traceability and loss of data integrity.

Apart from the aforementioned requirements, key management also becomes a concern when the owner of a tagged item changes. Thus, some RFID applications require the implementation of a Ownership Transfer Protocol (OTP), which allows the secure transfer of the ownership of a tag from a current owner to a new owner. When designing this kind of protocols one of the main issues is to prevent that the current owner can access the tag once it has been transferred to the new owner. As far as we know, only two different mechanisms have hitherto been proposed to guarantee the privacy of the new owner [4]: the presence of an external 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. A centralized scheme with TTP is valid when all tags are identified by readers belonging to the same company, but a trust issue arises when this is not the case. On the other hand, an IsE cannot be always guaranteed, and therefore this assumption is considered unrealistic for many practical cases. Ownership Transfer Protocols are sometimes designed to provide extended capabilities such as *Ownership Delegation* [5] and *Authorized Recovery* [6]. Additionally, Ng et al. [7] have introduced some other interesting novel security properties: *Tag Assurance* –to guarantee that the tag is as described, *Undeniable Ownership Transfer* –previous ownership cannot be denied, *Current Ownership Proof* –a means to prove the current ownership– and *Owner Initiation* –certain process only can be initiated by the current owner.

Recently in this Journal, Kapoor and Piramuthu [8], after reviewing the vulnerabilities of previously published protocols, have proposed two new schemes, based on TTP and IsE respectively, which are lightweight and secure. A variant of

J. Munilla and A. Peinado are with the Department of Communication Engineering, University of Málaga, Málaga, Spain.

G. Yang and W. Susilo are with School of Computer Science and Software Engineering, University of Wollongong, Wollongong, Australia.

Manuscript received —, —; revised —.

these protocols for multiple tags have also been published [9]. These proposals are interesting, and they overcome most of the security problems present in previous protocols, but one still can find some practical and security issues. In particular, we consider that the assumed connectivity setting is not suitable for many practical scenarios. Moreover, although the authors themselves consider that the assumption of an IsE is too optimistic and state that their own protocol, like many others (e.g. [6], [15]), should not be considered as an OTP in the strict sense, since previous owners may continue to have access to the tag, they did not find another alternative. They explain that, while it is not easy to accomplish the ownership transfer in the presence of a TTP, if the assumption of an IsE is not made, achieving the same without TTP remains as a challenging problem with a difficult solution.

Contributions

The main contributions of this paper and its corresponding organization are as follows:

- I. Previous works assume different models whose practical settings are omitted or described at relatively informal levels, making it difficult to provide side-to-side comparisons between alternative proposals. This paper describes a framework for RFID OTPs. An extended communication model is introduced, and a practical example is provided –Section II.
- II. In Section III, Kapoor and Piramuthu’s schemes are reviewed, and some practical and security issues are described.
- III. Then, Section IV proposes a new Ownership Transfer Protocol without TTP, which has the following characteristics:
 - It is designed to work in the extended communication model.
 - It can be implemented in low cost RFID tags which work in the UHF band.
 - It also provides the recently defined properties of *Tag Assurance*, *Undeniable Ownership Transfer*, *Current Ownership Proof* and *Owner Initiation*.
- IV. Finally, in Section V, a Key Change Protocol (KCP) is introduced. This protocol manages to solve the new owner’s privacy problem for the case where the TTP is absent without assuming an IsE.

II. RFID OWNERSHIP TRANSFER FRAMEWORK

Ownership Transfer Protocols allow transferring the rights over a tag from the current owner to a new owner in a secure and private way. Three different roles or entities are always present in an OTP:

1. The item or tag T whose rights are going to be transferred.
2. The seller or Current Owner (CO). At the inception of the protocol, only this entity can identify and trace T.
3. The buyer or New Owner (NO). When the protocol finishes, T can only be identified and traced by this entity.

In some protocols, as explained previously, there exists additionally a TTP which controls the execution and in which every entity trusts.

A. Security Properties

Besides the general security goals of confidentiality, integrity and availability, we can highlight the following security properties for RFID OTPs:

- *Location Privacy* or *Untraceability*. T cannot be traced by external adversaries.
- *Forward Secrecy*. Once T is transferred to NO, an external adversary cannot trace past communications between T and CO, even if the current private information stored in T is revealed (e.g. by physical attacks).
- *Backward Secrecy* or *NO’s Privacy*. Once T is transferred to NO, T cannot be identified by CO anymore.
- *CO’s Privacy*. NO cannot trace past communications between T and CO.
- *Tag Assurance*. NO can verify that the tag which is going to be transferred is indeed the requested tag; i.e. T is according to the description previously provided by CO.
- *Undeniable Ownership Transfer*. This property ensures that the seller (CO) cannot deny having sold T, for example, when it is defective; i.e. the buyer (NO) has a means to prove that the seller was the previous owner of the item.
- *Current Ownership Proof*. CO can generate a proof so that a third party can be convinced that he is the legitimate (current) owner of the item.
- *Controlled Delegation*. CO gives an authorized reader a derived key which allows it to interact with T for a limited number of times or until CO revokes this privilege; i.e. the delegated reader can control T for a period of time, but it does not become the new owner.
- *Authorized Recovery*. This property allows a previous owner to gain back the control of a tag without requiring the execution of the OTP. Ng et al. [7] also define *Temporary Authorization Recovery*. It is a combination of controlled delegation and authorized recovery that provides instant authorized recovery to the previous owner, but CO maintaining the full ownership simultaneously.
- *Owner Initiation*. CO is the only one capable of initiating an ownership transfer, key change or delegation.

B. Entities Capabilities

1) *Tags*: Although there are many types of tags, we can classify them into two large groups according to the coupling mechanism that they use: inductive (or magnetic) coupling for tags which operate in the near field (LF and HF bands), and backscattering for tags which operate in the far field (UHF and microwave bands) [10]. This physical mechanism will determine the available power and the operating range. Broadly speaking, inductive tags have shorter operating ranges and they are more expensive, but they are not so power constraint, and therefore conventional cryptographic schemes –symmetric and asymmetric– can be implemented [11]. By contrast, with backscattering, tags are able to work at higher

distances, but the delivered power is scarce, and therefore only simple or lightweight cryptographic primitives can be utilized. This more challenging case is the target of this paper, and more concretely, it focuses on the inexpensive tags which communicate in the UHF band. Public key cryptosystems, tamper-resistant shielding and on-board clocks are not considered realistic for these low-cost devices. Since proper tamper protection is not possible, it is assumed that these tags can be physically attacked and controlled by adversaries; i.e. all the stored internal secret keys inside a tag can be compromised if tags fall into an adversary's hands. We will just presume that tags are able to implement a hash function $H() : \{0, 1\}^* \rightarrow \{0, 1\}^n$ with the following properties:

One-wayness. The computation of the hash value is efficient, while it is hard to find a pre-image.

Collision resistance. Given any hash value, it is hard to find another message, not equal to the pre-image, which gives the same result.

2) *Readers*: Readers are considered without any special limitation about computation capabilities; thus, they are able to perform complex cryptographic operations like asymmetric encryption and decryption, signing and signature verification. For our protocol, we will assume that there exists a Public Key Cryptosystem for the users to create publicly verifiable digital signatures such that for any given message msg , a public key PK and its corresponding private key SK , we have:

$$\sigma = Sig(msg, SK) \text{ and } msg \stackrel{?}{=} Ver(\sigma, PK)$$

where $Sig()$ is the signing operation, σ the signature itself and $Ver()$ the signature verification operation.

C. Extended Communication Model

We will define the communication model according to two different aspects: connectivity between the parties; i.e. presence or absence of communication channels between them and the security of these channels.

1) *Extended Connectivity Model*: Most RFID OTPs (e.g. [7]–[9]) use connectivity settings where the readers of CO and NO and TTP, if it exists, can communicate with T at any moment during the execution of the protocol. This assumption, as we will show next, could be an issue in many practical scenarios. For instance, the operating range of inductive RFID tags which operate in the LF frequency (125 kHz) is around 1.5m, and for those which do in the HF frequency (13.56 Mhz) it is between 10cm (ISO-14443) and 70cm (ISO 15693). Therefore, it may be difficult, even from a physical standpoint, that all the entities could stay within this operating range during the execution of the protocol. The operating range of UHF tags can be quite higher (several meters), but it is still possible to find many practical examples where these connectivity requirements remain too restrictive. Let us imagine a shop in Australia which wishes to buy some shoes from a factory in Spain. Every pair of shoes is equipped with an RFID tag to facilitate tracking and prevent counterfeiting. The Australian shop (buyer) purchases the shoes through Internet, and the shoes are shipped by the

Spanish factory (seller). When the shoes reach Australia, the buyer checks the product and asks the seller to transfer the (digital) ownership of the RFID tags. It is clear from this example that the communication with the tag is not possible simultaneously for the buyer and the seller, and thus a different (extended) connectivity model is required. Fig. 1 illustrates the differences between the conventional and the extended connectivity model, required for this example. This extended model can be more formally defined as follows:

1. Communication between High-Level Entities is always possible; i.e. Readers, Server and/or TTPs are assumed to be permanently connected to each other.
2. Tags only can communicate with Readers, and these must be physically close. Thus, if any other entity (e.g. TTP) wants to communicate with a tag, it must be made via a reader. The communication between tags is not possible since they are passive.
3. Communication between T and the CO's reader is possible from the beginning of the protocol until a certain stage. From this point on, T only can communicate with the NO's reader. Obviously, it is always feasible that a reader relays the messages coming from tags to the other reader, but this is a critical decision, since it increases the communication burden and can cause trust problems; let us imagine, for example, that the NO's reader has to ask the CO's reader for every tag which is not able to singulate, without being sure if that specific tag was transferred by CO or another supplier.

It must be highlighted that those protocols which can work in this extended connectivity model can also work in the conventional one, whereas the opposite is not true.

2) *Security Channel Models*: There are two possible communication channels.

1. Between High-Level Entities. This communication channel is considered to be secure (symbolized by \Rightarrow), since fully-fledged cryptographic techniques can be used to guarantee the security of communications.
2. Between Readers and Tags. This channel, by contrast, is especially vulnerable (symbolized by \rightarrow); it is wireless and tags cannot implement sophisticated cryptographic mechanisms. Thus, we will assume the Dolev-Yao intruder model [12]. In this model, the adversary controls totally the communication channel and may eavesdrop, block, modify and inject messages in any communication between tags and readers. In practice, however, communication from reader to tag –forward channel– is easier to intercept than communication from tag to reader –backward channel, since the signal in this latter case is much weaker.

III. KAPOOR AND PIRAMUTHU'S PROTOCOLS

The first works dealing with Ownership Transfer in the RFID framework were published in 2005 by Molnar et al. [13] and Saito et al. [14]; since then, a number of protocols have been proposed. Kapoor and Piramuthu [8] cryptanalyze some of these proposals and conclude that most of them

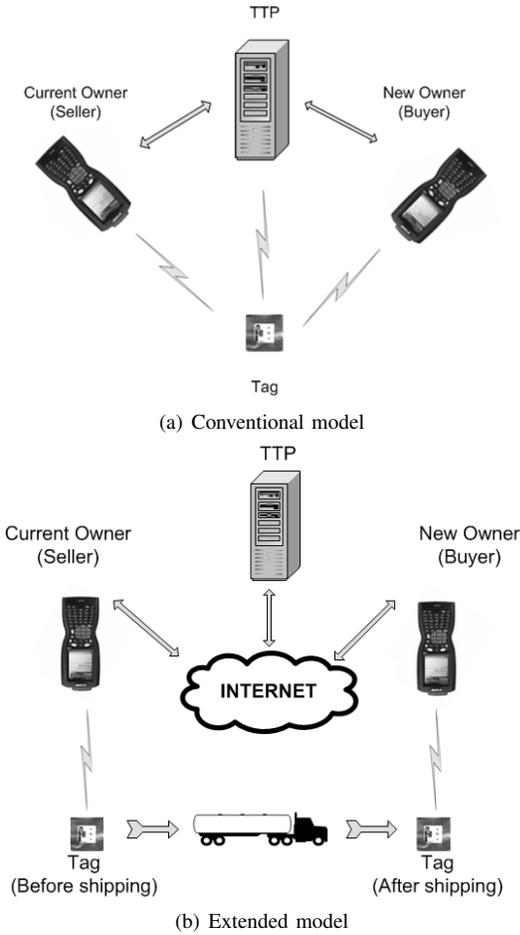


Fig. 1. Connectivity Models

have vulnerabilities. Due to space reasons, we refrain here to describe these protocols again, and we focus on Kapoor and Piramuthu's schemes –with and without TTP, which, according to the authors, are more secure than those currently existing and yet just as lightweight. Thus, we firstly introduce these protocols and then point out some aspects which could mean practical or security drawbacks. From our point of view, these issues are not enough to invalidate these protocols, but, at least, they should be taken into consideration, and in some particular cases, they may well end up being a serious inconvenience.

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 $H_k(\cdot)$. 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 CO's and NO's reader respectively.

A. Kapoor and Piramuthu's Protocol with TTP

The 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 depicted in Fig. 2 and it is accomplished as follows:

- S.1) Upon receiving an Ownership Transfer Request (OTR), \mathcal{TTP} generates a random nonce N_P and a new key s_2 , and it authenticates itself to \mathcal{T} by sending $f_{(N_P \oplus t_i \oplus s_1)}(s_2)$.
 $\mathcal{TTP} \rightarrow \mathcal{T}: N_P, f_{(N_P \oplus t_i \oplus s_1)}(s_2)$
- 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 $N_{\mathcal{T}}$ and using the one-way hash H with this value.
 $\mathcal{T} \rightarrow \mathcal{TTP}: N_{\mathcal{T}}, H_{(t_i \oplus N_{\mathcal{T}})}(s_2 \oplus N_P)$
- 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 revoke message).
 $\mathcal{TTP} \rightarrow \mathcal{R}1: g_{r_1}(s_1)$
- S.4) \mathcal{TTP} generates a new random nonce N'_P and sends it and $g_{(r_2 \oplus N'_P)}(s_2 \oplus r_2)$ to $\mathcal{R}2$.
 $\mathcal{TTP} \rightarrow \mathcal{R}2: N'_P, g_{(r_2 \oplus N'_P)}(s_2 \oplus r_2)$
- S.5) The new owner ($\mathcal{R}2$) sends an acknowledgment with the new key value to \mathcal{TTP} .
 $\mathcal{R}2 \rightarrow \mathcal{TTP}: H_{r_2}(s_2 \oplus N'_P)$
- S.6) Furthermore, $\mathcal{R}2$ generates a random nonce $N_{\mathcal{R}2}$ and sends it to \mathcal{T} encrypted by using the key s_2 .
 $\mathcal{R}2 \rightarrow \mathcal{T}: N_{\mathcal{R}2}, f_{s_2}(N_{\mathcal{R}2})$
- S.7) \mathcal{T} , to acknowledge that the message is correct, sends the hashed value of a new random nonce $N'_{\mathcal{T}}$ along with $N_{\mathcal{R}2}$ and s_2 .
 $\mathcal{T} \rightarrow \mathcal{R}2: N'_{\mathcal{T}}, H_{(N'_{\mathcal{T}} \oplus s_2)}(N_{\mathcal{R}2} \oplus s_2)$

Analysis

Three aspects of this protocol can be pointed out as possible concerns:

- 1) The protocol description does not explain how \mathcal{T} is singulated. It assumes that, from the beginning, \mathcal{TTP} and $\mathcal{R}2$ already know the (supposed) identity of the tag which they are communicating with.
- 2) Desynchronization between the parties. The authors state that \mathcal{TTP} is authenticated to \mathcal{T} (Steps 1 and 2) by checking $f_{(N_P \oplus t_i \oplus s_1)}(s_2)$. Then, \mathcal{T} updates s_1 to s_2 . However, this is not correct as this message does not guarantee the genuineness of \mathcal{TTP} . As a result, an adversary \mathcal{A} could make \mathcal{T} update its key to a fake value s_A . In fact, if \mathcal{A} , impersonating \mathcal{TTP} , sends any two values " N_A, F_A " to \mathcal{T} , then this will decrypt F_A , obtaining $s_A: f_{(N_A \oplus t_i \oplus s_1)}^{-1}(F_A) = s_A$. This desynchronization cannot be detected until Step 7, when $\mathcal{R}2$ finds out that the message sent by the tag is not correct (or not received). Unfortunately, the recovery from this situation

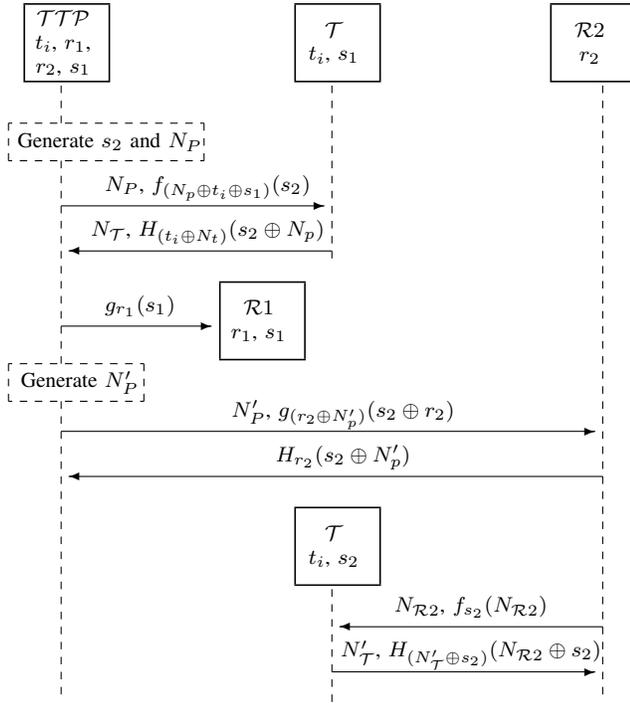


Fig. 2. Kapoor and Piramuthu's OTP with TTP

is not straightforward because s_1 is no longer stored in \mathcal{T} , and therefore a new procedure needs to be designed. This should involve the only value that \mathcal{T} shares with the rest of the system: t_i .

- 3 The protocol does not work in the extended connectivity model, which, in turn, could increase the trust issues between the parties. \mathcal{TTP} , to communicate with \mathcal{T} , must be physically close, and therefore the past and future locations of this device in a non-neutral position could arise trust problems; i.e. if it usually stays in the buyer's or in the seller's place, or it is carried by one of the parties. Although other bizarre solutions could be envisaged such as the presence of a "third man" or the destruction of the \mathcal{TTP} , it seems clear that the option of \mathcal{TTP} being a server in a neutral location is more convenient (*cf.* Sect. II-C).

B. Kapoor and Piramuthu's Protocol without TTP

This protocol is similar to the previous one except that the TTP is absent. A secure channel between $\mathcal{R1}$ and $\mathcal{R2}$ is assumed. This scheme is sketched in Fig. 3 and works as follows:

- S.1) Upon receiving a request (*OTR*), $\mathcal{R1}$ generates a fresh random number $N_{\mathcal{R1}}$, computes $N_{\mathcal{R1}} \oplus s_1$, where s_1 is the key that $\mathcal{R1}$ shares with \mathcal{T} , and sends the result to $\mathcal{R2}$ on a secure channel.

$$\mathcal{R1} \Rightarrow \mathcal{R2}: N_{\mathcal{R1}} \oplus s_1$$

- S.2) $\mathcal{R1}$ sends the same information to \mathcal{T} but encrypted with s_1 .

$$\mathcal{R1} \rightarrow \mathcal{T}: f_{s_1}(N_{\mathcal{R1}} \oplus s_1)$$

- S.3) \mathcal{T} generates two fresh random numbers: $N_{\mathcal{T}}$ and $N'_{\mathcal{T}}$; and computes the value $N = N_{\mathcal{R1}} \oplus N_{\mathcal{T}}$. Then, \mathcal{T} randomly *flips* one bit in N , creating N' . \mathcal{T} sends the following messages to $\mathcal{R2}$:

$$\mathcal{T} \rightarrow \mathcal{R2}: N_{\mathcal{T}} \oplus s_1, N'_{\mathcal{T}}, f_{(N' \oplus N'_{\mathcal{T}})}(N' \oplus N'_{\mathcal{T}}), H_{(N' \oplus N'_{\mathcal{T}})}(N' \oplus N'_{\mathcal{T}})$$

- S.4) Now, both \mathcal{T} and $\mathcal{R2}$ know N . Knowing N , $\mathcal{R2}$ uses a brute force technique on $f_{(N' \oplus N'_{\mathcal{T}})}(N' \oplus N'_{\mathcal{T}})$ to determine N' , and checks the computed result with the hash value $H_{(N' \oplus N'_{\mathcal{T}})}(N' \oplus N'_{\mathcal{T}})$. Then, $\mathcal{R2}$ generates a new key s_2 and sends the following message to \mathcal{T} :

$$\mathcal{R2} \rightarrow \mathcal{T}: f_{N'}(N' \oplus s_2)$$

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

$$\mathcal{T} \rightarrow \mathcal{R2}: H_{s_2}(N' \oplus s_2)$$

- S.6) $\mathcal{R2}$ sends $f_{s_2}(N' \oplus s_2)$ to acknowledge receipt of the message in the previous step.

$$\mathcal{R2} \rightarrow \mathcal{T}: f_{s_2}(N' \oplus s_2)$$

If the tag does not receive this within a predetermined amount of time, the process is repeated beginning with Step 1.

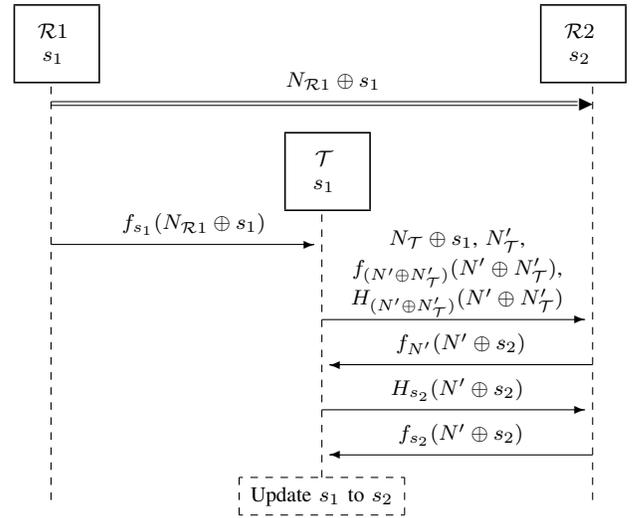


Fig. 3. Kapoor and Piramuthu's OTP without TTP.

Analysis

As previously mentioned, Kapoor and Piramuthu consider unrealistic the assumption of an IsE and state that, without a TTP, the privacy of the new owner is not guaranteed. Apart from this, we still find other possible issues:

- 1 It is again not clear (not described in the protocol) how readers, $\mathcal{R1}$ and $\mathcal{R2}$, singulate \mathcal{T} .
- 2 Tags initiate communications with the readers, which is not according to the general RFID principles. RFID

tags are passive, and therefore the communication must always be initiated by the reader. However, in Step 3, \mathcal{T} initiates the communication with \mathcal{R}_2 , and something similar happens if \mathcal{T} does not receive the last message within a predetermined amount of time (Step 6); in this case, the process must be repeated from the beginning, and it seems (not described) that it should be \mathcal{T} which informs about this situation.

- 3 It could be subject to a practical DoS (Denial of Service) attack. Let us assume an active adversary who, in Step 3, modifies the messages sent from the tag by adding a fixed value Δ to $N_{\mathcal{T}} \oplus s_1$ and $N'_{\mathcal{T}}$, and leaving the encrypted and the hash values without modifying; i.e. the adversary sends: $[N_{\mathcal{T}} \oplus s_1 \oplus \Delta, N'_{\mathcal{T}} \oplus \Delta, f_{(N' \oplus N'_{\mathcal{T}})}(N' \oplus N'_{\mathcal{T}}), H_{(N' \oplus N'_{\mathcal{T}})}(N' \oplus N'_{\mathcal{T}})]$. \mathcal{R}_2 will then compute an incorrect value $N'_{\mathcal{A}} = flip(N_{\mathcal{R}_1} \oplus N_{\mathcal{T}} \oplus \Delta) = flip(N_{\mathcal{R}_1} \oplus N_{\mathcal{T}}) \oplus \Delta = N' \oplus \Delta$; but it will accept these messages as valid since the computed values of the encryption and the hash function are correct, as $N'_{\mathcal{A}} \oplus N'_{\mathcal{T}} \oplus \Delta = N' \oplus N'_{\mathcal{T}}$. \mathcal{R}_2 will then send the message $f_{N'_{\mathcal{A}}}(N'_{\mathcal{A}} \oplus s_2)$ to \mathcal{T} . As a result, \mathcal{T} will not be able to get the correct value of s_2 , and steps 4 and 5 will be repeated indefinitely. Moreover, the use of the *flip* function makes the protocol vulnerable to overload; \mathcal{A} can just send randomly generated messages (Step 3), and \mathcal{R}_2 will need to check these junk messages for every possible tag and for every bit (flipping each of them) before discarding them.
- 4 The protocol does not work in the extended communication model. If it is not successful, the protocol is repeated from the beginning (Step 6), and therefore \mathcal{T} has to interact again with \mathcal{R}_1 .

IV. PROPOSED PROTOCOL

This section describes a novel OTP without TTP which:

- i) overcomes the disadvantages of Kapoor and Piramuthu's protocol,
- ii) captures the security properties defined by [7] and
- iii) works in the Extended Communication Model. Additionally, to guarantee the privacy of the new owner without requiring either TTP or IsE, this protocol is complemented with a proposed KCP, described in Sect. V.

Next, we will introduce some of the notation which will be used to describe the protocol:

Functions

- $H()$ is a secure hash function as defined in Section II-B.
- $H(a, b, c) = H(a||b||c)$; i.e. the hash of the concatenation.
- Sig and Ver are signing operation and signature verification operation respectively, as defined in Section II-B.

Setup (manufacturer)

- ID is the identifying information of \mathcal{T} ; e.g. its EPC code.
- $Info_{ID}$ is manufacturer's information about the tag.
- $M_0 = H(ID, Info_{ID})$, a hash value for the tag \mathcal{T} .
- $\sigma_0 = Sig(M_0, SK_0)$, manufacturer's signature of the product.

After i successful executions, the tag will belong to the reader \mathcal{R}_i , which stores:

- IDR_i , its own identifier.
- ID , the identifier of \mathcal{T} .
- $M_i = H(M_{i-1})$, a hash chain of the information provided by the manufacturer.
- $\sigma_i = Sig(M_i, SK_i)$, the current owner's signature of the (hashed) tag information.
- s_i , a key shared with \mathcal{T} .

At that point, \mathcal{T} , besides its identifier ID , will have four variables with the following assigned values:

- $IDR_{\mathcal{T}} = IDR_i$, the identifying information of the current owner's reader.
- $M_{\mathcal{T}} = M_i$, the hash chain of the tag information.
- $\varphi_{\mathcal{T}} = \varphi_i = H(M_i, \sigma_i)$, a hash value of the tag information and its corresponding signature generated by \mathcal{R}_i .
- $s_{\mathcal{T}} = s_i$, the key shared with \mathcal{R}_i .

For simplicity, we will assume again that the current owner's reader is \mathcal{R}_1 , with identifier IDR_1 , and the new owner's reader is \mathcal{R}_2 , with identifier IDR_2 , being $IDR_1 \neq IDR_2$. According to the definition of the Extended Communication Model (Sect. II-C), initially only \mathcal{R}_1 can communicate with \mathcal{T} ; and from a certain point on (e.g. the product is shipped), it is \mathcal{R}_2 the only entity which can interact with \mathcal{T} . The channel between \mathcal{R}_1 and \mathcal{R}_2 is assumed to be secure.

A. Description

This protocol is depicted in Fig. 4 and works as follows:

Initialization

- S.1) \mathcal{R}_1 sends an *OTR* to \mathcal{R}_2 , indicating that the tag \mathcal{T} , with identifier ID , is going to be transferred. \mathcal{R}_1 also provides it with information about the tag $Info_{ID}$, the hash value M_1 and its corresponding signature σ_1 .

$\mathcal{R}_1 \Rightarrow \mathcal{R}_2$: $OTR(IDR_1, IDR_2, ID), Info_{ID}, M_1, \sigma_1$.

- S.2) \mathcal{R}_2 verifies $M_1 \stackrel{?}{=} Ver(\sigma_1, PK_1)$. If correct, it computes $M_0 = H(Info_{ID}, ID)$ and then $M_i = H(M_{i-1})$, for $1 < i < MAX$, and checks if any of them coincides with the value received from \mathcal{R}_1 (M_1 in this case). MAX is the maximum number of ownership transfers that a tag is expected to have during its lifetime. If a match is found, then it computes $M_2 = H(M_1)$ (M_{i+1} in the general case) and the corresponding signature with its secret key: $\sigma_2 = Sig(M_2, SK_2)$ ($\sigma_{i+1} = Sig(M_{i+1}, SK_{i+1})$ in the general case), and sends it to \mathcal{R}_1 . If a match is not found, the process is aborted.

$\mathcal{R}_2 \Rightarrow \mathcal{R}_1$: σ_2 .

Setup of \mathcal{T}

- S.3) After checking that σ_2 is correct $M_2 \stackrel{?}{=} Ver(\sigma_2, PK_2)$, \mathcal{R}_1 is ready to transfer \mathcal{T} . It regularly broadcasts *Query* messages to detect tags.

$\mathcal{R}_1 \rightarrow$ Tags: $Query(IDR_1)$

- S.4) When \mathcal{T} is within the operating range of \mathcal{R}_1 , it receives $Query(IDR_1)$. Then, it draws a random nonce $N_{\mathcal{T}}$

and replies with the following message:

$$\mathcal{T} \rightarrow \mathcal{R}_1 : H(ID, N_{\mathcal{T}}, s_1), N_{\mathcal{T}}$$

- S.5) \mathcal{R}_1 searches in its Database for a pair $ID_i - s_i$ which matches with the received message. If a match with the pair $ID - s_1$ is found, \mathcal{T} is singulated. If so, it picks a random nonce $N_{\mathcal{R}_1}$, computes $s' = H(IDR_1, IDR_2, N_{\mathcal{R}_1}, s_1)$ and replies with a message to inform \mathcal{T} that its ownership is going to be transferred, along with $H(N_{\mathcal{T}}, s_1), \sigma_2$ and $N_{\mathcal{R}_1}$. If this match is not found, the process (with regards to \mathcal{T}) is repeated from Step 3.

$$\mathcal{R}_1 \rightarrow \mathcal{T} : OTR(IDR_1, IDR_2), H(N_{\mathcal{T}}, s_1), \sigma_2, N_{\mathcal{R}_1}$$

- S.6) \mathcal{T} checks $H(N_{\mathcal{T}}, s_1)$ to authenticate \mathcal{R}_1 . If it is not correct, \mathcal{T} does not reply. Otherwise, it computes $M_2 = H(M_1)$ and then $\varphi_2 = H(M_2, \sigma_2)$, saves $[IDR_2, \varphi_2, N_{\mathcal{R}_1}]$ (until the OTP finishes or a new OTR is received), and acknowledges receipt of this message by sending a hash value:

$$\mathcal{T} \rightarrow \mathcal{R}_1 : H(IDR_2, \varphi_2, N_{\mathcal{R}_1}, s_1)$$

- S.7) If this message is not received correctly after a period of time, the protocol is repeated from Step 3 (\mathcal{T} will replace the stored values $[IDR_2, \varphi_2, N_{\mathcal{R}_1}]$). Upon reception of this message, \mathcal{R}_1 confirms that \mathcal{T} is ready to be transferred and sends s' to \mathcal{R}_2 .

$$\mathcal{R}_1 \Rightarrow \mathcal{R}_2 : ID \text{ is ready, } s'$$

After Shipping

- S.8) After computing $s_2 = H(s', \varphi_1)$, \mathcal{R}_2 is ready to receive \mathcal{T} . It regularly broadcasts *Query* messages to detect tags within its operating range.

$$\mathcal{R}_2 \rightarrow \text{Tags: } Query(IDR_2)$$

- S.9) When \mathcal{T} receives *Query*(IDR_2), it generates a random nonce $N'_{\mathcal{T}}$, computes $s' = H(IDR_1, IDR_2, N_{\mathcal{R}_1}, s_1)$ and then $s_2 = H(s', \varphi_1)$, and replies with the following message:

$$\mathcal{T} \rightarrow \mathcal{R}_2 : H(ID, N'_{\mathcal{T}}, s_2), N'_{\mathcal{T}}$$

- S.10) \mathcal{R}_2 uses the previous message to singulate \mathcal{T} . If so, \mathcal{R}_2 draws a fresh random number $N_{\mathcal{R}_2}$ and replies with the following message:

$$\mathcal{R}_2 \rightarrow \mathcal{T} : H(N'_{\mathcal{T}}, \varphi_2, s_2), N_{\mathcal{R}_2}$$

- S.11) \mathcal{T} checks the previous message. If it is not correct, \mathcal{T} does not reply. Otherwise, \mathcal{R}_2 is authenticated, and \mathcal{T} updates its variables $IDR_{\mathcal{T}}, s_{\mathcal{T}}, \varphi_{\mathcal{T}}$ from IDR_1, s_1, φ_1 to IDR_2, s_2, φ_2 . These values determine the ownership of a tag; hitherto, \mathcal{R}_1 could have aborted the OTP by using s_1 . \mathcal{T} acknowledges the transfer by replying with a message to authenticate itself to \mathcal{R}_2 .

$$\mathcal{T} \rightarrow \mathcal{R}_2 : H(N_{\mathcal{R}_2}, s_2)$$

- S.12) If the previous message is not received correctly, the

protocol is repeated from Step 8 (see next subsection for further details). Otherwise, the protocol has finished successfully, and \mathcal{R}_2 informs \mathcal{R}_1 of it.

$$\mathcal{R}_2 \Rightarrow \mathcal{R}_1 : \text{Ownership Transferred.}$$

- S.13) \mathcal{R}_2 , to prevent \mathcal{R}_1 from accessing \mathcal{T} , execute the KCP described in Sect. V.

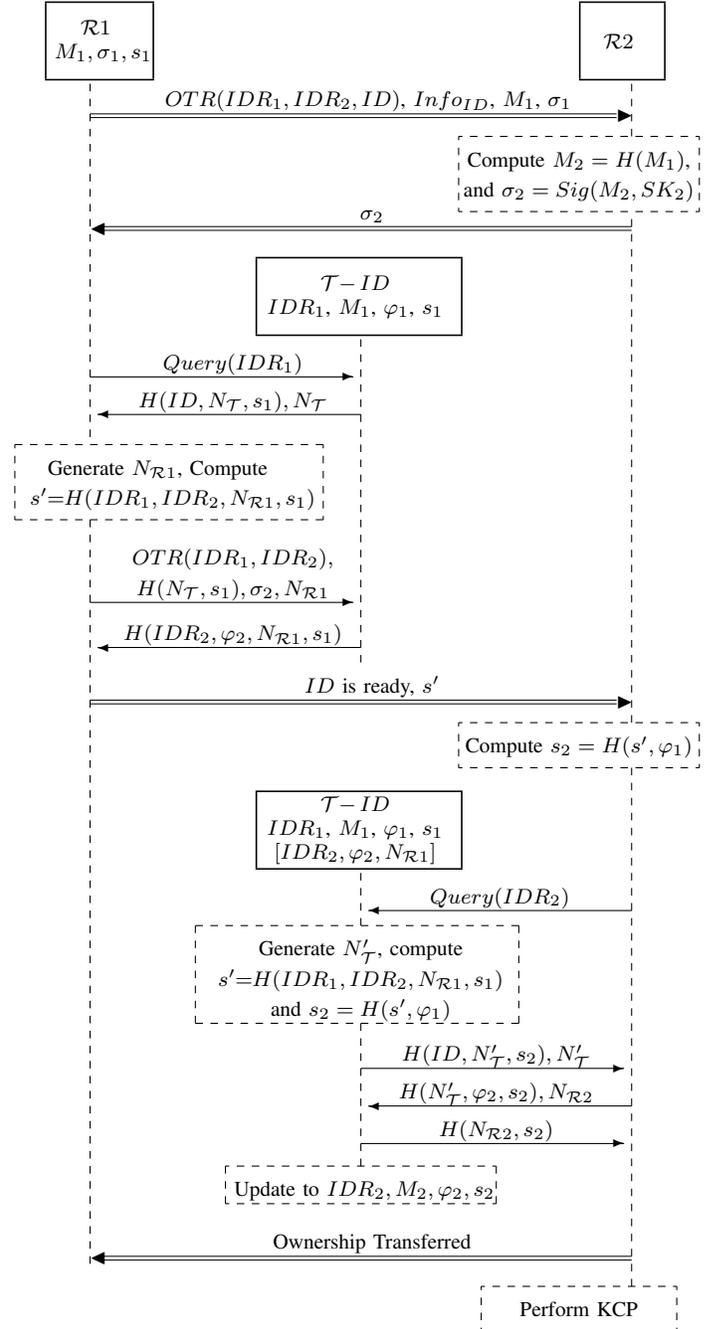


Fig. 4. Proposed OTP

B. Security Analysis

Readers and tag are mutually authenticated, which prevents desynchronization attacks. \mathcal{R}_1 and \mathcal{R}_2 are authenticated to \mathcal{T} in Step 5 and 10 respectively; while \mathcal{T} is authenticated to \mathcal{R}_1 and \mathcal{R}_2 in Step 6 and 11 respectively.

Reply to *Query* messages (Step 4 and 9) always have the same format and include a nonce drawn by the tag. This prevents traceability, since they will look random for anyone who does not know the secret key, and makes easy the singulation process, since it remains the same independently of the action that the reader will carry out afterwards. Something similar happens if the message sent in Step 11 is not received and the protocol has to be repeated from Step 8; \mathcal{T} will authenticate itself by using the updated values if it completed the previous step, or performing the protocol as described, if it did not. This is possible thanks to the fact that the same messages can be exchanged for tag authentication and secure communication between tags and readers without modification. Most current protocols require multiple suites to perform both actions.

Additionally, the protocol captures the following, specific to OTPs, security properties:

1. *CO's privacy and Forward Secrecy*. CO's privacy is guaranteed because the key s_1 remains unknown to NO. We can prove this by contradiction; suppose there is an attacker who can output the previous tag key s_1 given the values: s' , IDR_1 , IDR_2 and $N_{\mathcal{R}1}$. Then, one can use this attacker to find the preimage of s' in H by computing $IDR_1 || IDR_2 || N_{\mathcal{R}1} || s_1$. This contradicts the assumption that finding the pre-image of a hash value is hard under the one-wayness property. Likewise, *Forward Secrecy* is provided because, again due to the one-wayness property of H , any trace of s_1 is eliminated. Indeed, if the tag is accessed (e.g. physically) and the new secrets of the tags are revealed: s_2 and φ_2 ; the privacy of the previous communications is still guaranteed because the adversary neither knows s_1 nor can compute it. In actual fact, for this protocol in particular, *CO's privacy* implies *Forward Secrecy*.
2. *NO's privacy*. This property will be provided by executing the KCP described in the next section.
3. *Tag Assurance*. $Info_{ID}$ contains information about the tag. This information is hashed M_i and signed σ_i . The collision resistance property of the hash function prevents an adversary from finding another message (pre-image) $Info'_{ID}$ to replace the information given by the manufacturer. The value $M_{\mathcal{T}}$ is computed by the own tag, and therefore, when NO receives the tag, the use of φ_i to compute s_{i+1} makes certain that the tag is according to the expected properties.
4. *Undeniable Ownership Transfer and Current Ownership Proof*. The owner of the tag is defined by the authorship of the signature σ_i whose hashed value is stored in the tag ($\varphi_{\mathcal{T}} = \varphi_i$). However, the owner (or signer) must also know the current tag key $s_{\mathcal{T}} = s_i$ to make \mathcal{T} assign this value. To prove previous ownership and current ownership, it is sufficient to present the previous values: M_{i-1} and σ_{i-1} ; and the current values: M_i , σ_i and φ_i .
5. *Owner Initiation*. This property is provided after executing the KCP described in the next section. Once it is executed, only the current owner is valid to initiate an ownership transfer, key change or delegation process.

In its present state, the protocol does not implement the

following properties, but it can be easily modified to provide them:

6. *Controlled Delegation*. A counter c can be implemented in the tag so that a delegate can use a temporal key Ks for up to c_{max} times. Meanwhile, the proper key will remain valid ($s_{\mathcal{T}} = s_i$) so that the legitimate owner can revoke this permission.
7. *Authorized Recovery*. The session key s' can be restored, which is already known by \mathcal{R}_{i-1} , so that it is not necessary to execute the OTP again.

C. Security Proof

We now prove (in a similar vein to [8]) the correctness of the protocol using GNY Logic [16] and Strand Spaces [17]. GNY Logic is used to verify the correctness of the assumptions with respect to message source as well as the beliefs of the sender and the recipient of the messages. Each principal only can advance his beliefs and increase his possessions based on the physical content of the messages he receives. Strand Spaces, on its hand, is used to exclude vulnerabilities based on the structure of the protocol. It assumes a free encryption algebra, and therefore failures which exploit relations in the algebra are not detected.

Although these analyses have been used to reason about the correctness of the complete protocol, for brevity reasons, we will only show here the proof for the second part of the protocol; i.e. after shipping. The analyses for the other parts of the protocol are similar in structure.

GNY Logic. We begin the GNY Logic analysis of the protocol by listing the initial assumptions. Firstly, it is assumed that the parties believe (\models) in the jurisdiction (\Rightarrow) of $\mathcal{R}1$ over the secrets to be shared between them; i.e. $\mathcal{T} \models \mathcal{R}1 \Rightarrow (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}2)$ and $\mathcal{R}2 \models \mathcal{R}1 \Rightarrow (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}2)$. Additionally: $\mathcal{T} \ni s_2$, $\mathcal{T} \ni N'_{\mathcal{T}}$, $\mathcal{T} \models \#N'_{\mathcal{T}}$, $\mathcal{T} \models \phi(N'_{\mathcal{T}}, \varphi_2)$, $\mathcal{R}2 \ni s_2$, $\mathcal{R}2 \ni N_{\mathcal{R}2}$, $\mathcal{R}2 \models \#N_{\mathcal{R}2}$, $\mathcal{R}2 \models \phi(ID)$, $\mathcal{R}2 \models \phi(N_{\mathcal{R}2})$. That is, both of the participants possess (\ni) the secret s_2 . Each also possesses a nonce and believes in its freshness ($\#$). In addition, \mathcal{T} believes that the pair $(N'_{\mathcal{T}}, \varphi_2)$ is recognizable (ϕ), and $\mathcal{R}2$ believes that ID and $N_{\mathcal{R}2}$ are recognizable. Notation and postulate numbers referred to in the following are from [16].

The description of the messages in GNY Logic is as follows:

- M.1. $\mathcal{T} \triangleleft *Query(IDR_2)$
- M.2. $\mathcal{R}2 \triangleleft *H(*ID, *N'_{\mathcal{T}}, *s_2) \rightsquigarrow \mathcal{T} \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}2), N'_{\mathcal{T}}$
- M.3. $\mathcal{T} \triangleleft *H(N'_{\mathcal{T}}, *\varphi_2, s_2) \rightsquigarrow \mathcal{R}2 \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}2), *N_{\mathcal{R}2}$
- M.4. $\mathcal{R}2 \triangleleft *H(N_{\mathcal{R}2}, s_2) \rightsquigarrow \mathcal{T} \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}2)$

The general goal is to provide \mathcal{T} and $\mathcal{R}2$ with the shared secret s_2 . This can be formally expressed as follows:

$$\mathcal{T} \ni s_2; \mathcal{T} \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}2); \mathcal{T} \models \mathcal{R}2 \ni s_2$$

$$\mathcal{R}2 \ni s_2; \mathcal{R}2 \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}2); \mathcal{R}2 \models \mathcal{T} \ni s_2.$$

Some of these are already given by the initial assumptions, and therefore we just need to prove two of them: i) $\mathcal{T} \models \mathcal{R}2 \ni s_2$, and ii) $\mathcal{R}2 \models \mathcal{T} \ni s_2$.

To prove these goals, we analyze the messages for any run of the protocol:

- M.1. No belief or possession can be derived from this message. This message does not contain any cryp-

tographic information. In particular, \mathcal{T} cannot even be convinced that it was sent by \mathcal{R}_2 .

- M.2. First we note that the extension to the message, $\mathcal{T} \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}_2)$, holds because it is an initial assumption. Besides, \mathcal{T} is sure that \mathcal{R}_2 cannot mistake s_2 as a key for itself and another tag, since the identifier ID is included in the message. Applying postulates T1 and P1, we get $\mathcal{R}_2 \ni N'_\mathcal{T}$. Using the initial assumption $\mathcal{R}_2 \models \phi(ID)$ and applying R5, \mathcal{R}_2 identifies \mathcal{T} . No postulate enables us to further derive new beliefs or possessions from this message. In particular, we cannot derive the freshness of the message.
- M.3. The extension to the message, $\mathcal{R}_2 \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}_2)$ is valid. Applying T1 and P1, we get $\mathcal{T} \ni N_{\mathcal{R}_2}$. Applying the initial assumptions: $\#N'_\mathcal{T}$ and $\phi(N'_\mathcal{T}, \varphi_2)$; and F1 we get $\mathcal{T} \models \#(N'_\mathcal{T}, \varphi_2, s_2)$. Using this belief and applying I3, \mathcal{T} believes that it is indeed \mathcal{R}_2 who sent the message; i.e. $\mathcal{T} \models \mathcal{R}_2 \sim H(N'_\mathcal{T}, \varphi_2, s_2)$. Now, if \mathcal{T} believes that \mathcal{R}_2 conveyed $(N'_\mathcal{T}, \varphi_2, s_2)$ and that it is fresh, then \mathcal{T} is entitled to believe that \mathcal{R}_2 possesses $(N'_\mathcal{T}, \varphi_2, s_2)$ (postulate I6). And applying P3, we prove the first of the goals: $\mathcal{T} \models \mathcal{R}_2 \ni s_2$.
- M.4. The extension to the message, $\mathcal{T} \models (\mathcal{T} \xleftrightarrow{s_2} \mathcal{R}_2)$, is valid. Applying the initial assumptions: $\mathcal{R}_2 \models \phi(N_{\mathcal{R}_2})$ and $\mathcal{R}_2 \models \#N_{\mathcal{R}_2}$; and I3, we obtain $\mathcal{R}_2 \models \mathcal{T} \sim H(N_{\mathcal{R}_2}, s_2)$. Applying I6 and P3 we get the second of the goals: $\mathcal{R}_2 \models \mathcal{T} \ni s_2$

The consistency of the protocol description has also been checked. That is; i) possession consistency: messages only include formula that the sender possesses; and ii) belief consistency: message extensions include only beliefs held by the sender at the time he sends the message.

Strand Spaces. We next prove the correctness of this part of the proposed protocol using Strand Spaces [17]. To simplify the analysis we remove the first message (Step 8), which, as explained above, does not provide any cryptographic information. A strand space Σ is a set of strands, and a strand is a sequence of events that a party may engage in. For a legitimate party (*principal*) they are their actions in one particular run of the protocol, and for a *penetrator* they are messages that model her capabilities (e.g. type Encryption where she receives a key and a message, and she sends the result of encrypting this message). We refer to the messages that can be exchanged between the principals as *terms*. In a protocol, principals can either send or receive terms, and this is represented with a positive or a negative sign respectively. A *bundle* is a portion of a strand space.

We introduce some of the notation:

- \prec : precedence (transitive closure).
- $a \sqsubset b$ ($a \not\sqsubset b$): a is (not) a subterm of b .
- K_P : set of keys known to the penetrator.
- \wedge : AND logical operation.

Definition 1 An infiltrated space Σ, \mathcal{P} is an EOTP (Extended Ownership Transfer Protocol) space if Σ is the union of three kinds of strands:

1. Penetrator strands $s \in \mathcal{P}$;

2. “Initiator strands” $s \in \text{Init}[\mathcal{T}, \mathcal{R}_2, N'_\mathcal{T}, N_{\mathcal{R}_2}]$, defined to be:

$$\langle +H(ID, N'_\mathcal{T}, s_2), N'_\mathcal{T}, -H(N'_\mathcal{T}, \varphi_2, s_2), N_{\mathcal{R}_2}, +H(N_{\mathcal{R}_2}, s_2) \rangle$$
 where $ID \in T_{name}$ and $s_2 \in K$ but $N'_\mathcal{T} \notin T_{name}$ and $N'_\mathcal{T} \notin K$. \mathcal{T} is the principal associated with this strand.
3. Complementary “responder strands” $s \in \text{Resp}[\mathcal{T}, \mathcal{R}_2, N'_\mathcal{T}, N_{\mathcal{R}_2}]$ defined to be:

$$\langle -H(ID, N'_\mathcal{T}, s_2), N'_\mathcal{T}, +H(N'_\mathcal{T}, \varphi_2, s_2), N_{\mathcal{R}_2}, -H(N_{\mathcal{R}_2}, s_2) \rangle$$
 where $ID \in T_{name}$ and $s_2 \in K$ but $N_{\mathcal{R}_2} \notin T_{name}$ and $N_{\mathcal{R}_2} \notin K$. \mathcal{R}_2 is the principal associated with this strand.

A) AGREEMENT: THE RESPONDER’S GUARANTEE.

Proposition 1. Suppose:

1. Σ is an EOTP space, \mathcal{C} is a bundle in Σ , and s is a responder strand in $\text{Resp}[\mathcal{T}, \mathcal{R}_2, N'_\mathcal{T}, N_{\mathcal{R}_2}]$.
2. $s_2 \notin K_P$; and
3. $N'_\mathcal{T} \neq N_{\mathcal{R}_2}$ and $N_{\mathcal{R}_2}$ is uniquely originating in Σ .

Then \mathcal{C} contains an initiator’s strand $t \in \text{Init}[\mathcal{T}, \mathcal{R}_2, N'_\mathcal{T}, N_{\mathcal{R}_2}]$.

We prove this using the following lemmas (1-4). The node $\langle s, 2 \rangle$ outputs the values $H(N'_\mathcal{T}, \varphi_2, s_2), N_{\mathcal{R}_2}$; for convenience, we will refer to this node as n_0 and to its term as v_0 . The node $\langle s, 3 \rangle$ receives the value $H(N_{\mathcal{R}_2}, s_2)$; we will refer to this node as n_3 and to its term as v_3 . Two additional nodes, n_1 and n_2 , such that $n_0 \prec n_1 \prec n_2 \prec n_3$ are also identified.

Lemma 1: $N_{\mathcal{R}_2}$ originates at $\langle s, 2 \rangle$ (the second node of the reader).

PROOF. We know that $N_{\mathcal{R}_2} \sqsubset v_0$ and the sign of n_0 is positive. Thus, we just need to verify that $N_{\mathcal{R}_2} \not\sqsubset \langle s, 1 \rangle$. Since $\text{term}(\langle s, 1 \rangle) = H(ID, N'_\mathcal{T}, s_2), N'_\mathcal{T}$; we need to check that $ID \neq N_{\mathcal{R}_2}$, which follows from $ID \in T_{name}$ and $N_{\mathcal{R}_2} \notin T_{name}$, that $N'_\mathcal{T} \neq N_{\mathcal{R}_2}$, which is a hypothesis, and that $s_2 \neq N_{\mathcal{R}_2}$, which follows from the stipulation $N_{\mathcal{R}_2} \notin K$. \square

The next lemma establishes that the crucial step is taken by a regular strand and not by a penetrator strand. For this proof we cannot use $N_{\mathcal{R}_2}$ but $H(N_{\mathcal{R}_2}, s_2)$, since in our protocol $N_{\mathcal{R}_2}$ is transmitted in clear, and therefore it could be sent by a penetrator node of type **S** (separation into components). For the same reason, secrecy of $N'_\mathcal{T}$ and $N_{\mathcal{R}_2}$ are not provided.

Lemma 2: The set $S = \{n \in \mathcal{C} : H(N_{\mathcal{R}_2}, s_2) \sqsubset \text{term}(n) \wedge v_0 \not\sqsubset \text{term}(n)\}$ has a \preceq -minimal node n_2 . The node n_2 is regular and its sign is positive.

PROOF. S is non-empty because $n_3 \in S$, as $n_3 \in \mathcal{C}$ and n_3 contains $H(N_{\mathcal{R}_2}, s_2)$ but not v_0 . It is proved ([17]) that if S is not empty, it has at least one \preceq -minimal element n_2 and its sign is positive. Then, we just need to check that n_2 does not lie on a penetrator strand p . Let us examine the possible cases for positive penetrator nodes (cf. [17]).

- M. The trace $\text{tr}(p)$ has the form $\langle +t \rangle$; so we must have $N_{\mathcal{R}_2} \sqsubset t$. In this case $N_{\mathcal{R}_2}$ originates on this strand, but that is impossible because $N_{\mathcal{R}_2}$ originates on the regular node n_0 (Lemma 1).
- F. The trace $\text{tr}(p)$ has the form $\langle -g \rangle$, and thus lacks any positive nodes.
- T, C. These traces have the form $\langle -g, +g, +g \rangle$ and $\langle -g, -h, +gh \rangle$ respectively, so the positive nodes are

not minimal occurrences.

- K. The trace $\text{tr}(p)$ has the form $\langle +K_0 \rangle$ where $K_0 \in K_P$. But $H(N_{\mathcal{R}2}, s_2) \not\sqsubseteq K_0$ (free encryption algebra assumptions), so this case does not apply.
- E. The trace $\text{tr}(p)$ has the form $\langle -K_0, -h, +\{h\}_{K_0} \rangle$. Let us suppose $H(N_{\mathcal{R}2}, s_2) \sqsubset \{h\}_{K_0}$. Hence (free encryption), $h = N_{\mathcal{R}2}$ and $K_0 = s_2$. Thus, there exists a node m (the first of this strand) with $\text{term}(m) = s_2$. However, $s_2 \notin K_P$, and therefore this node should be regular, but no regular node originates s_2 , therefore contradicting the initial assumption.
- D. The trace $\text{tr}(p)$ has the form $\langle -K_0^{-1}, -\{h\}_{K_0}, +h \rangle$. If the positive node is minimal in S , then $v_0 \not\sqsubseteq h$ but $v_0 \sqsubset \{h\}_{K_0}$. However, because $v_0 \neq \{h\}_{K_0}$, if $v_0 \sqsubset \{h\}_{K_0}$ means that $v_0 \sqsubset h$, contradicting the assumption made above.
- S. The trace $\text{tr}(p)$ has the form $\langle -gh, +g, +h \rangle$. Assume $\text{term}(n_2) = h$ (there is a symmetrical case if $\text{term}(n_2) = g$). By the minimality of n_2 , $v_0 \sqsubset gh$. Hence, $g = H(N'_{\mathcal{T}}, \varphi_2, s_2)$ and $h = N_{\mathcal{R}2}$, but $H(N_{\mathcal{R}2}, s_2) \not\sqsubseteq h$, and therefore $n_2 \notin S$ contradicting the initial assumption.

Therefore, n_2 does not lie on a penetrator strand but must lie on a regular strand instead. \square

Lemma 3: Node n_2 follows n_1 on the same regular strand t , and $\text{term}(n_1) = H(N'_{\mathcal{T}}, \varphi_2, s_2), N_{\mathcal{R}2}$.

PROOF. From Lemma 1 and by definition, we know that $N_{\mathcal{R}2}$ originates at n_0 and its uniqueness in Σ . We also know that $n_2 \neq n_0$ since $v_0 \sqsubset \text{term}(n_0)$ and $v_0 \not\sqsubseteq \text{term}(n_2)$. Therefore, $N_{\mathcal{R}2}$ does not originate at n_2 , and there is a node n_1 preceding n_2 on the same strand such that $N_{\mathcal{R}2} \sqsubset \text{term}(n_1)$. By the minimal property of n_2 , $v_0 \sqsubset \text{term}(n_1)$. However, as no regular node contains a combination as a proper subterm, $H(N'_{\mathcal{T}}, \varphi_2, s_2), N_{\mathcal{R}2} = \text{term}(n_1)$. \square

Lemma 4: The regular strand t containing n_1 and n_2 is an initiator strand and is contained in C .

PROOF. n_1 precedes n_2 in the same strand. Node n_2 is a positive regular node with the form $\{H(N_{\mathcal{R}2}, s_2)\}$. Hence, t is an initiator strand since a responder strand would only contain a negative node of this form. Thus, n_1 and n_2 are the second and the third nodes of t respectively. \square

Lemmas 3 and 4 prove Proposition 1.

We have just proved the non-injective agreement property for the EOTP responder. Injectivity follows easily on the assumption that $N'_{\mathcal{T}}$ is uniquely originating.

Proposition 2: If Σ is an EOTP space, and $N'_{\mathcal{T}}$ is uniquely originating in Σ , then there is at most one strand $t \in \text{Init}[\mathcal{T}, \mathcal{R}2, N'_{\mathcal{T}}, N_{\mathcal{R}2}]$ for any $\mathcal{T}, \mathcal{R}2$ and $N_{\mathcal{R}2}$.

PROOF: If $t \in \text{Init}[\mathcal{T}, \mathcal{R}2, N'_{\mathcal{T}}, N_{\mathcal{R}2}]$ for any $\mathcal{T}, \mathcal{R}2$ and $N_{\mathcal{R}2}$, then $\langle t, 1 \rangle$ is positive, $N'_{\mathcal{T}} \sqsubset \text{term}(\langle t, 1 \rangle)$, and $N'_{\mathcal{T}}$ cannot possibly occur earlier on t . So $N'_{\mathcal{T}}$ originates at node $\langle t, 1 \rangle$. Hence, if $N'_{\mathcal{T}}$ originates uniquely in Σ , there can be at most one such t . \square

B) AGREEMENT: THE INITIATOR'S GUARANTEE.

Proposition 3. Suppose:

1. Σ is an EOTP space, C is a bundle in Σ , and s an initiator's strand in $\text{Init}[\mathcal{T}, \mathcal{R}2, N'_{\mathcal{T}}, N_{\mathcal{R}2}]$;
2. $s_2 \notin K_P$; and

3. $N'_{\mathcal{T}}$ is uniquely originating in Σ .

Then there exists a responder's strand $t \in \text{Resp}[\mathcal{T}, \mathcal{R}2, N'_{\mathcal{T}}, N_{\mathcal{R}2}]$.

PROOF. Consider the set $\{m \in C : \{H(N'_{\mathcal{T}}, \varphi_2, s_2)\} \sqsubset \text{term}(m)\}$. It is non-empty because it contains $\langle s, 2 \rangle$. So it also contains a minimal member m_0 . If m_0 lies on a regular strand t , then we can show that $t \in \text{Resp}[\mathcal{T}, \mathcal{R}2, N'_{\mathcal{T}}, N_{\mathcal{R}2}]$. If instead m_0 lies on a penetrator strand t , then t should be an E-strand with trace: $\langle -s_2, -N'_{\mathcal{T}}, \varphi_2, +H(N'_{\mathcal{T}}, \varphi_2, s_2) \rangle$, but this contradicts that $s_2 \notin K_P$. \square

V. KEY CHANGE PROTOCOL

Only two solutions have been proposed to guarantee the privacy of the new owner: the use of a TTP and the assumption of an IsE. Although the former entails trust issues, the latter is often considered unrealistic. This IsE would allow the new owner to carry out a KCP away from the interception of the previous owner. However, Kapoor and Piramuthu [8] explains that if one can assume that there is not any other reader in the vicinity, it would not be even necessary to encrypt the messages.

In this section we describe a novel alternative that requires neither a TTP nor an IsE. Our solution is based on the Wire-tap Channel Problem and more specifically in the use of noisy tags. The Wire-tap Channel Problem was defined by Wyner in 1975 [18], and it involves a communication system that is being wire-tapped (passive adversary) via a second noisy channel. The objective is to encode the data in such a way that the wire-tapper's level of confusion is as high as possible. This problem has been the subject of decades of work in the information and coding community, and nowadays we still can find proposals which try to bridge the gap between this body of work and Cryptography [19]. The security of these proposals are based under the sole assumption that the adversary's channel is noisier than the ordinary communication channel between the genuine parties. In the RFID framework, Juels et al. introduced in 2003 the concept of a "Blocker Tag" [20], which, by simulating the full spectrum of possible identifiers, prevents a reader from singulating any tag. It could be used by an adversary to cause DoS attacks but also to protect the privacy of the tags, since it obscures their identifying codes. In 2006, Castellucia and Avoine [21] proposed a protocol that, including a noisy tag, could be used between an RFID tag and a reader to exchange a secret. This protocol involves at least three entities: a reader R and a tag T, which want to agree a secret key K_T , and a tag, which we will call noisy tag NT, which already shares a private key K_{NT} with R. The protocol assumes that collisions are allowed and when several tags reply simultaneously the voltage amplitude of the different bits get added; i.e. if the bit '1' is implemented by a pulse of A mV, and the bit '0' corresponds to a pulse of 0 mV, when both tags send '1's, the total response will be $2A$ mV. Thus, when both tags, T and NT, reply simultaneously two bitstrings, unknown and known respectively, the reader is able to filter out the bitstring sent by NT (known), and thus recover the bitstring sent by T (unknown). By contrast, a passive adversary (eavesdropper) can only recover those bits where both tags replied with the

same value: the total response was 0mV (both tags sent ‘0’) or 2AmV (both tags sent ‘1’).

By using noisy tags, the protocol creates a noisier channel for the adversary than that for the honest reader (Wire-tap Channel). Although this solution could be attractive since it allows establishing a new secret between the parties without any previous information exchange, it has not received much attention due to the fact that it does not provide any protection against active adversaries. More precisely, the protocol cannot ensure authentication, and therefore any active adversary could impersonate the reader or the tag and get private information, or even set a fake key which causes desynchronization between the genuine parties. Fortunately, although assuming passive adversaries is not enough for most security applications, it suits perfectly the adversary model we face here. The adversary model for CO is that of an *honest-but-curious* attacker. That is, CO belongs to the system, shares secret information and appears to be honest, but he wants to continue tracing the tags after his ownership has been transferred. The protection against external active adversaries, on the other hand, could be provided by using the key that NO and the tag share after performing the OTP.

A. Description

This KCP is intended to be performed between $\mathcal{R}2$ and \mathcal{T} once the OTP described in the previous section has been successfully completed. Besides $\mathcal{R}2$ and \mathcal{T} , it involves a noisy tag $\mathcal{N}\mathcal{T}$. Initially, $\mathcal{R}2$ shares with \mathcal{T} and $\mathcal{N}\mathcal{T}$ the keys s_2 (after the execution of the OTP) and $K_{\mathcal{N}\mathcal{T}}$ respectively. And as a result of this KCP, $\mathcal{R}2$ and \mathcal{T} will agree on a new key $K_{\mathcal{T}}$ of m bits. It is accomplished as follows (Fig. 5):

S.1) $\mathcal{R}2$ regularly broadcasts *Query* messages to detect tags.

$$\mathcal{R}2 \rightarrow \text{Tags: } \text{Query}(ID_{\mathcal{R}2})$$

S.2) Upon receiving *Query*($ID_{\mathcal{R}2}$), \mathcal{T} generates a random nonce $N_{\mathcal{T}}$ and identify itself with the following message:

$$\mathcal{T} \rightarrow \mathcal{R}1 : H(ID, N_{\mathcal{T}}, s_2), N_{\mathcal{T}}$$

S.3) $\mathcal{R}2$ singulates \mathcal{T} , and decides to change its key in a secure way. $\mathcal{R}2$ draws a fresh random number $N_{\mathcal{R}2}$ and broadcasts it, $H(N_{\mathcal{T}}, s_2)$ (required to provide *Owner Initiation*) and a Key Change Request (*KCR*).

$$\mathcal{R}2 \rightarrow \mathcal{T}, \mathcal{N}\mathcal{T}: KCR, H(N_{\mathcal{T}}, s_2), N_{\mathcal{R}2}$$

S.4) Upon receiving a *KCR* command, \mathcal{T} and $\mathcal{N}\mathcal{T}$ generate independently two bitstring of length $C \cdot m$ bits: $S_{\mathcal{T}}$ and $S_{\mathcal{N}\mathcal{T}}$; and broadcast them simultaneously. C is a parameter of the protocol, and it will be explained later. $\mathcal{N}\mathcal{T}$ uses $N_{\mathcal{R}2}$ to compute $S_{\mathcal{N}\mathcal{T}} = H'(N_{\mathcal{R}2}, K_{\mathcal{N}\mathcal{T}})$, whereas \mathcal{T} generates $S_{\mathcal{T}}$ randomly. The function H' is a secure hash function whose output has $C \cdot m$ bits, and its implementation could obviously be based on H (or even be the same).

$$\mathcal{T}, \mathcal{N}\mathcal{T} \rightarrow \mathcal{R}2: S_{\mathcal{T}} \text{ and } S_{\mathcal{N}\mathcal{T}} \text{ simultaneously.}$$

S.5) $\mathcal{R}2$ receives the added values of $S_{\mathcal{T}}$ and $S_{\mathcal{N}\mathcal{T}}$. There will be k positions j -th where $S_{\mathcal{T}}(j) \neq S_{\mathcal{N}\mathcal{T}}(j)$ and

$(C \cdot m - k)$ positions i -th where $S_{\mathcal{T}}(i) = S_{\mathcal{N}\mathcal{T}}(i)$; i.e. $k = HW(S_{\mathcal{T}} \oplus S_{\mathcal{N}\mathcal{T}})$, where HW stands for the Hamming Distance. If $k < m$ the protocol must be repeated from Step 1. Otherwise, $\mathcal{R}2$ chooses at random m positions out of the k where $S_{\mathcal{T}}(j) \neq S_{\mathcal{N}\mathcal{T}}(j)$ and generates a bitstring M of length $C \cdot m$, which will have the value 1 for the m chosen positions and 0 for the rest; i.e. $HW(M) = m$. Then, $\mathcal{R}2$ computes the new key: $K_{\mathcal{T}} = S_{\mathcal{T}} \wedge M$ (AND bitwise operator), and broadcasts the following message:

$$\mathcal{R}2 \rightarrow \mathcal{T}, \mathcal{N}\mathcal{T}: H(N_{\mathcal{T}}, s_2, K_{\mathcal{T}}), M$$

S.6) \mathcal{T} computes $K_{\mathcal{T}} = S_{\mathcal{T}} \wedge M$ and checks if the message from $\mathcal{R}2$ is correct. If so, \mathcal{T} updates its variable $s_{\mathcal{T}}$ from s_2 to $K_{\mathcal{T}}$ and sends back the following message:

$$\mathcal{T} \rightarrow \mathcal{R}2: H(N_{\mathcal{R}2}, s_2, K_{\mathcal{T}})$$

S.7) $\mathcal{R}2$ receives the response from the tag and checks if it is correct. If so, the process has finished. Otherwise, $\mathcal{R}2$, sending a new *Query*, checks if \mathcal{T} updated its key, or it is still s_2 (and the entire protocol must be repeated).

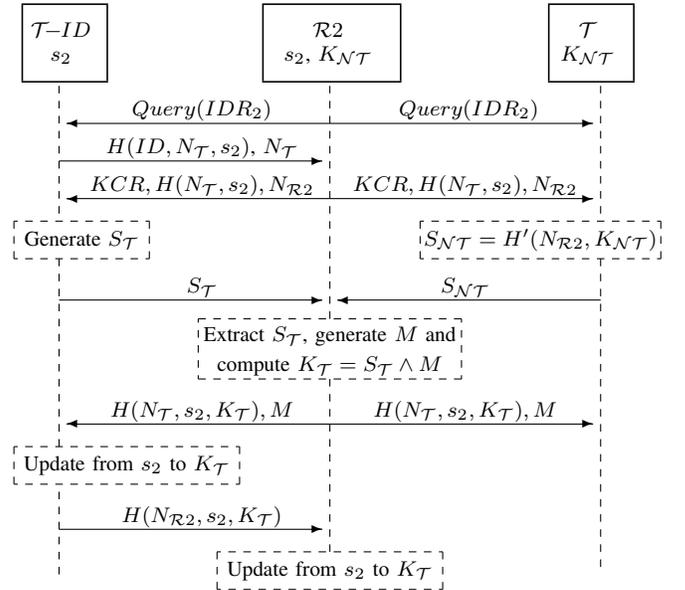


Fig. 5. Key Change Protocol based on Noisy Tags

B. Analysis

In the protocol, there are two important parameters which must be chosen: m which stands for the length of the new key; and C which will determine the length of the generated bitstring, and this, in turn, the probability that the protocol must be repeated after Step 3. This probability can be computed as:

$$Pr_{repeat} = \frac{1}{2^{C \cdot m}} \sum_{i=0}^{i=m-1} \binom{C \cdot m}{i} \quad (1)$$

and for example, for keys of 80 bits ($m = 80$) and strings of 200 bits ($C = 2.5$), it is lower than 0.2%.

The security of the protocol against external adversaries, passive or active, is guaranteed by the use of s_2 and the security properties of the hash function:

- Traceability: \mathcal{T} remains untraceable because the messages that it sends look random for anyone who does not know s_2 .
- Desynchronization: Without knowing s_2 , an adversary cannot compute either $H(N_{\mathcal{T}}, s_2, K_{\mathcal{T}})$ or $H(N_{\mathcal{R}2}, s_2, K_{\mathcal{T}})$, which are required by the parties to update their keys.

The security analysis for $\mathcal{R}1$ must be different. As mentioned above, the adversary model for $\mathcal{R}1$ is that of an *honest-but-curious* attacker. $\mathcal{R}1$ knows s_2 but not $K_{\mathcal{N}\mathcal{T}}$, and therefore he will not be able to filter out $S_{\mathcal{N}\mathcal{T}}$ and recover $K_{\mathcal{T}}$. Without knowing the new key $K_{\mathcal{T}}$, $\mathcal{R}1$ cannot trace \mathcal{T} anymore. The case where $\mathcal{R}1$ uses s_2 to desynchronize \mathcal{T} and $\mathcal{R}2$ is out of our adversary model. Nevertheless, this is only possible before performing the KCP, since once $s_{\mathcal{T}}$ is updated to $K_{\mathcal{T}}$, $\mathcal{R}1$ becomes an external adversary from a security point of view.

Every tag, after updating its key, could act as a noisy tag for future key exchanges, provided that it can implement H' . This can be an advantage in some practical cases; for example, in a conveyor belt, a tag, immediately after updating its keys, could become the noisy tag for the KCP of the next tag on the belt. Although, for the sake of clarity, we have described a simple version of the protocol, it could be modified in several different ways [21]. For example: i) changing the modulation to prevent synchronization problems, ii) increasing the number of tags, which makes the adversary's channel even noisier and more difficult to distinguish the legitimate tag according to the geographical position or physical properties, or iii) increasing the information rate ($1/C$) by making the tags do not send random sequences but appropriate codes [22].

VI. CONCLUSION

In the last years, a myriad of protocols have been published to improve the weak security of the low-cost RFID tags. Apart from security and privacy issues, the ability to change or share ownership of these tags is also relevant. However, most existing Ownership Transfer Protocols have vulnerabilities and assume communication models which are not suitable for many practical cases. To address these issues and, therefore, increase the confidence in this technology and promote its adoption, the contribution of this paper is threefold. First, after defining a framework for RFID OTPs, we scrutinized the protocols, with and without TTP, proposed by Kapoor and Piramuthu in 2012 and identified several security and practical inconveniences. Second, we have proposed an OTP that overcomes these problems, provides additional security properties and is valid for a wider range of practical scenarios. The correctness of this protocol has been proved by using GNY Logic and Strand Spaces. Finally, the current problem to guarantee the privacy of the new owner for those cases where TTP is absent is solved by combining this protocol with a KCP based on the use of noisy tags.

REFERENCES

- [1] K. Finkenzerler, *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*, 2nd ed. John Wiley & Sons, May 2003.
- [2] EPC Global "EPC tag data standards, vs. 1.3," http://www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.3.pdf.
- [3] ISO/IEC, "Standard # 18000 – RFID Air Interface Standard," <http://www.hightechaid.com/standards/18000.htm>.
- [4] A. Fernandez-Mir, R. Trujillo-Rasua, J. Castell-Roca, and J. Domingo-Ferrer, "A Scalable RFID Authentication Protocol Supporting Ownership Transfer and Controlled Delegation." in *RFIDSec*, ser. Lecture Notes in Computer Science, A. Juels and C. Paar, Eds., vol. 7055. Springer, 2011, pp. 147–162. [Online]. Available: <http://dblp.uni-trier.de/db/conf/rfidsec/rfidsec2011.html#Fernandez-MirTCD11>
- [5] D. Molnar, A. Soppera, and D. Wagner, "A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags," in *Proc. Workshop on Selected Areas in Cryptography (SAC 2005)*, ser. LNCS, vol. 3897. Springer, 2006.
- [6] B. Song, "RFID Tag Ownership Transfer," in *Workshop on RFID Security – RFIDSec'08*, Budapest, Hungary, July 2008.
- [7] C. Y. Ng, W. Susilo, Y. Mu, and R. Safavi-Naini, "Practical RFID Ownership Transfer Scheme," *Journal of Computer Security*, vol. 19, no. 2, pp. 319–341, 2011.
- [8] G. Kapoor and S. Piramuthu, "Single RFID Tag Ownership Transfer Protocols," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 42, no. 2, pp. 164–173, 2012.
- [9] G. Kapoor, W. Zhou, and S. Piramuthu, "Multi-tag and Multi-owner RFID Ownership Transfer in Supply Chains," *Decision Support Systems*, vol. 52, no. 1, pp. 258–270, 2011.
- [10] D. Paret, *RFID and Contactless Smart Card Applications*. John Wiley & Sons, 2005.
- [11] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.
- [12] D. Dolev and A. C.-C. Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.
- [13] D. Molnar, A. Soppera, and D. Wagner, "A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags," in *Workshop on RFID Security and Light-Weight Crypto*, Graz, Austria, July 2005.
- [14] J. Saito, K. Imamoto, and K. Sakurai, "Reassignment Scheme of an RFID Tag's Key for Owner Transfer," in *EUC Workshops*, ser. Lecture Notes in Computer Science, T. Enokido, L. Yan, B. Xiao, D. Kim, Y.-S. Dai, and L. T. Yang, Eds., vol. 3823. Springer, 2005, pp. 1303–1312.
- [15] H. Lei and T. Cao, "RFID Protocol Enabling Ownership Transfer to Protect against Traceability and DoS Attacks," in *Proceedings of the The First International Symposium on Data, Privacy, and E-Commerce*, ser. ISDPE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 508–510. [Online]. Available: <http://dx.doi.org/10.1109/ISDPE.2007.113>
- [16] L. Gong, R. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols," in *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1990, pp. 234–248.
- [17] F. Thayer, J. Herzog, and J. Guttman, "Strand Spaces: Proving Security Protocols Correct," *Journal of Computer Security*, vol. 7, pp. 191–230, 1999.
- [18] A. Wyner, "The Wire-Tap Channel," *Bell Syst. Tech. Journal*, vol. 54, pp. 1355–1387, 1975.
- [19] M. Bellare, S. Tessaro, and A. Vardy, "A Cryptographic Treatment of the Wiretap Channel," *CoRR*, vol. abs/1201.2205, 2012.
- [20] A. Juels, R. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy," in *Conference on Computer and Communications Security – ACM CCS*, V. Atluri, Ed., ACM. Washington, DC, USA: ACM Press, October 2003, pp. 103–111.
- [21] C. Castelluccia and G. Avoine, "Noisy Tags: A Pretty Good Key Exchange Protocol for RFID Tags," in *International Conference on Smart Card Research and Advanced Applications – CARDIS*, ser. Lecture Notes in Computer Science, J. Domingo-Ferrer, J. Posegga, and D. Schreckling, Eds., vol. 3928, IFIP. Tarragona, Spain: Springer-Verlag, April 2006, pp. 289–299.
- [22] J. Bringer and H. Chabanne, "On the Wiretap Channel Induced by Noisy Tags," in *European Workshop on Security and Privacy in Ad hoc and Sensor Networks – ESAS'06*, ser. Lecture Notes in Computer Science,

vol. 4357. Hamburg, Germany: Springer-Verlag, September 2006, pp. 113–120.