Missouri University of Science and Technology
Scholars' Mine

Electrical and Computer Engineering Faculty Research & Creative Works

Electrical and Computer Engineering

01 Jun 2018

Tag Ownership Transfer in Radio Frequency Identification Systems: A Survey of Existing Protocols and Open Challenges

Eyad Taqieddin

Hiba Al-Dahoud

Haifeng Niu

Jagannathan Sarangapani
Missouri University of Science and Technology, sarangap@mst.edu

Follow this and additional works at: https://scholarsmine.mst.edu/ele_comeng_facwork

 Part of the Electrical and Computer Engineering Commons

Recommended Citation

E. Taqieddin et al., "Tag Ownership Transfer in Radio Frequency Identification Systems: A Survey of Existing Protocols and Open Challenges," *IEEE Access*, vol. 6, pp. 32117-32155, Institute of Electrical and Electronics Engineers (IEEE), Jun 2018.
The definitive version is available at https://doi.org/10.1109/ACCESS.2018.2842778

This Article - Journal is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Electrical and Computer Engineering Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 19, 2018, accepted May 19, 2018, date of publication June 1, 2018, date of current version June 29, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2842778

Tag Ownership Transfer in Radio Frequency Identification Systems: A Survey of Existing Protocols and Open Challenges

EYAD TAQIEDDIN[1], (Senior Member, IEEE), HIBA AL-DAHOUD[1], HAIFENG NIU[2], AND JAGANNATHAN SARANGAPANI[3], (Fellow, IEEE)

[1]Department of Network Engineering and Security, Jordan University of Science and Technology, Irbid 22110, Jordan
[2]Amazon, Seattle, WA 98032, USA
[3]Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65401, USA

Corresponding author: Eyad Taqieddin (eyadtaq@just.edu.jo)

This work was supported by the Jordan University of Science and Technology under Project 259/2016.

ABSTRACT Radio frequency identification (RFID) is a modern approach to identify and track several assets at once in a supply chain environment. In many RFID applications, tagged items are frequently transferred from one owner to another. Thus, there is a need for secure ownership transfer (OT) protocols that can perform the transfer while, at the same time, protect the privacy of owners. Several protocols have been proposed in an attempt to fulfill this requirement. In this paper, we provide a comprehensive and systematic review of the RFID OT protocols that appeared over the years of 2005–2018. In addition, we compare these protocols based on the security goals which involve their support of OT properties and their resistance to attacks. From the presented comparison, we draw attention to the open issues in this field and provide suggestions for the direction that future research should follow. Furthermore, we suggest a set of guidelines to be considered in the design of new protocols. To the best of our knowledge, this is the first comprehensive survey that reviews the available OT protocols from the early start up to the current state of the art.

INDEX TERMS Authentication, delegation, EPC, ownership transfer, RFID, trusted third party.

I. INTRODUCTION

Radio Frequency Identification (RFID) is a system that employs radio waves to identify assets [1]. This system has many features that allow it to outperform the traditional identification systems such as barcodes [**?**]. This has led to the widespread proliferation of RFID systems and their deployment in several application environments [3].

Any RFID system consists of three main entities; the Radio Frequency (RF) tag, the RF reader (interrogator), and the back-end server (or database) [1]. Each asset in the supply chain is embedded with an RF tag. The tag has an electronic silicon chip that stores a unique identifier (ID) in its non-volatile memory. This ID uniquely identifies the tag and its associated assets. The reader has one or more antennas which are used to send an RF signal to the surrounding tags to interrogate them. Once the tags receive the RF signal, they harvest power from it and respond by sending their IDs. The back-end server stores the required information for all items and tags in the system. Thus, when the reader gets the ID information from one or more tags, it can identify the corresponding item(s) by passing the ID(s) to the back-end server.

The computation and storage capabilities of readers and back-end servers are higher than those of tags. Thus, they can perform strong cryptographic functions. Therefore, the communication channels between readers and servers are assumed to be secure and are treated as one entity. On the other hand, the communication channels between any entity and the tags are typically insecure and susceptible to security threats.

Tags in RFID systems are classified into three categories depending on the power source: active, semi-passive (or semi-active), and passive tags. Active tags use on-board batteries to activate the circuitry and to transmit data. The semi-passive tags power up their circuitry using their own batteries yet they harvest the power required for data transmission from the incident RF signal. Passive tags do not have any power source and depend solely on the received RF signal for power harvesting. Therefore, in active tags the communication ranges are longer than passive tags and their computation

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

levels are much better which makes them the most expensive type.

In wireless networks, the coverage area between two end points is unknown and uncontrollable and this causes vulnerabilities at different levels for all wireless communications. Thus, the security problems of general wireless communication are also present in RFID systems.

In addition to the problems of wireless communication, RFID systems suffer from other inherent security issues. Any reader (regardless of the holder's identity) can interrogate the tag which gives a chance to adversaries to collect confidential information, insert harmful data into the tag, or even kill the tag to disable it permanently. Moreover, since RFID tags always respond with the same static ID whenever they are queried, the tagged item can be tracked. This could expose of the owner's physical location.

Another point of concern is that passive tags have limited power and computational capabilities. The number of available logic gates ranges from 5,000 to 10,000 gates of which only 250-3000 can be utilized for security purposes [4]. Thus, strong encryption techniques such as the Advanced Encryption Standard (AES) which needs approximately 3000 gates and hash functions such as SHA-256 which require about 8000-10000 gates cannot be implemented on passive RFID tags. According to the EPC Class-1 Generation-2 (EPC-C1G2) standard, passive tags can only perform 16-bit Cyclic Redundancy Check (CRC) and 16-bit Pseudo Random Number Generation (PRNG). In addition, they can execute simple bitwise operations such as the XOR, AND, and OR. Hence, developing security protocols for passive RFID systems is considered a difficult challenge.

Even if we disregard the scarcity of available gates to implement the security protocol, the limited power available for communication and computation is still an issue. Since most of security protocols are performed in terms of several sessions during a limited time period, it should be guaranteed that the provided power is enough to fully complete the protocol session. Otherwise, the protocol may be aborted before completion, which can result in de-synchronization problems.

The time required to execute the security protocol is another challenge. Protocol message exchanges should be performed in a short computation/communication time. For example, in supply chain inventory, the tags are identified as groups. Therefore, the employed security protocol should not slow the identification process down.

As demonstrated by these challenges, designing a security protocol for passive RFID system is not a simple task as there is a trade off between the needed security and minimizing the computational load.

A. EPC STANDARD

Several standards were proposed for RFID systems. One of the well-known standards is the Electronic Product Code (EPC). This standard has several classes; one of which is the EPC-C1G2.

The EPC-C1G2 standard specifies the properties for tags in the passive RFID system as follows:

- Tags are passive.
- The communication range of the tags is limited up to 9 meters.
- Strong cryptographic techniques cannot be implemented on tags as they are extremely constrained in the resources and storage capacity.
- Each tag's chip has on board circuitry for a 16-bit *CRC* and 16-bit *PRNG*.
- In these tags, two passwords are stored. The first one is the Access password which is issued when protected data need to be accessed. The second is the Kill password which is used to permanently disable the tag.
- Each tag has a unique identifier known as EPC that must consist of 96 or 192 bits.

B. SECURITY GOALS FOR RFID SYSTEMS

Since RFID systems are susceptible to various threats, several security protocols have been proposed to secure these systems. In order to determine whether a protocol is secure, it should satisfy the following requirements:

1) The security protocol should guarantee that any two communicating entities are legitimate. Generally, this requirement is known as mutual authentication and can be achieved by ensuring that both parties share the same secret information such as the nonces, keys, and hash values.

   If any of the communicating entities is malicious, the consequences may be severe. For instance, a malicious reader can acquire sensitive information by interrogating an RFID-tagged credit card. On the other hand, a merchant might approve a transaction that is based on false data when dealing with a fake tagged credit card.
2) RFID systems are constrained in power resources when compared to other devices such as PCs and cellphones. This fact makes the supported encryption methods in RFID systems more susceptible to attacks such as eavesdropping. Hence, while taking into account the power limitation issue, the security protocol should protect the privacy of RIFD systems by achieving the following properties:
   - Confidentiality: Sensitive information such as passwords, keys, and tag data should be encrypted to conceal them from adversaries.
   - Integrity: The security protocol should protect the exchanged data between legitimate entities from modification. Moreover, it should provide a way for the participating parties to verify the validity of the received data.
   - Untraceability: In some situations, the adversaries are interested in the location of the tagged item rather than disclosing its secret information. By tracking the tag's responses (whether

encrypted or not), an attacker may expose its location.

To prevent tracking, the security protocol should ensure that the tags always respond with fresh data whenever interrogated. Randomizing the tag's response is generally performed by employing PRNGs or true random number generators such that each response includes a fresh nonce.

3) One last requirement is related to changing the owner of the tagged item. Obviously, in these cases, the access rights of the tag need to be transferred to the new owner in a secure manner. The process of completely transferring the read and write privileges from a reader or group of readers to another reader or group of readers is known as the ownership transfer (OT). In different scenarios, the access rights need to be transferred from one owner to another (called delegate) for a specified time period. For instance, the tagged item is to be rented or to go under warranty maintenance. In this case, the process of transferring the access privileges is called ownership delegation (OD). OD is considered a special case of the OT.

   In contrast to OT, the current owner and the delegate in OD share the access rights during the specified delegation period. However, once the need for delegation no longer exists, the current owner revokes the access rights of the delegate.

C. CONTRIBUTIONS

In this work, we survey the existing OT protocols that have been proposed through the literature over the period of 2005 - 2018. We analyze these protocols and highlight their properties, advantages, and disadvantages. Moreover, we compare between the proposed OT protocols based on several characteristics and show the results in a tabular form. We also provide a suggestion for the future direction of the work in this field and examine the possible improvements on the current state of the art. This is further supported by a set of guidelines that would be useful for the future designs of OT protocols.

The rest of this paper is organized as follows: In Section II we specify the security goals of the ownership transfer, give an introduction to the ownership transfer protocols, present our unified notations, and then demonstrate the idea of ownership transfer through a general scenario. After that, we present the TTP-based ownership transfer protocols in Section III. Next, in Section IV we introduce the second category of ownership transfer protocols, namely the Two-party protocols. In Section V we go though the third class of the protocols which are called the universal ownership transfer protocols. Moreover, in Section VI we present the concept of ownership delegation and further clarify the idea with an example. After that, we compare the protocols from the three categories in a tabular form based on their properties and vulnerabilities to attacks in Section VII. We provide our discussion and recommendations in Section VIII. We conclude our work in the conclusions Section IX.

II. OWNERSHIP TRANSFER

OT is the process in which the ownership of an RFID tagged item is transferred from the current owner to a new one. OT is considered an important aspect of RFID systems because most of the time the tagged item will change owners at least once. In OT, the secret information such as keys and passwords will be exchanged by the owners and tags. This may allow the adversaries to perform several attacks if not carried out securely. Therefore, it is essential to guarantee that OT is performed in a secure manner which protects the privacy of both the current and new owners from violation by each other or by adversaries.

A. SECURITY GOALS OF OWNERSHIP TRANSFER

Since OT is one of the needed requirements for secure RFID systems, the properties that an OT protocol should satisfy are classified into two aspects: properties for securing RFID systems in general and properties for securing the OT process in particular. Both aspects of properties are listed below.

General Properties of OT Protocol:

- Resistance to tag tracking: The protocol should prevent the adversaries from tracking the tags' location.
- Resistance to tag impersonation: The protocol should guarantee that no fake tag can successfully pass the authentication process with a legitimate server/reader.
- Resistance to server/reader impersonation: The protocol should guarantee that no fake server/reader can successfully pass the authentication process with a legitimate tag.
- Resistance to replay (or spoofing) attack: Assuming that an adversary eavesdropped on the exchanged messages between the server and the tag, the protocol should ensure that the adversary cannot complete a successful session with either the server/reader or the tag by replaying the captured messages or a modified version of these messages.
- Resistance to Man-in-the-Middle (MitM) attack: The protocol should prevent the adversaries from inserting new messages or changing those originally exchanged.
- Resistance to De-synchronization attack: This attack happens when the server/reader and the tag are forced to change their states differently by interrupting the message flow between them. This break in synchronization leads them to have different secret values and will no longer be able to communicate. Thus, an OT protocol should ensure that the server/reader and the tag are always synchronized, even if the message flow was interrupted by the adversaries. It should be noted that the de-synchronization attack is considered a type of Denial-of-Service (DoS) attack.
- Resistance to disclosure attack: The protocol should prevent adversaries from obtaining the sensitive information held or exchanged by the participating parties.

Specific Properties of OT Protocol:

- New owner privacy: The protocol should guarantee that the old owner (the entity who previously owned the tag) can no longer access or track the tag after successfully completing the OT process.
- Old owner privacy: The protocol should guarantee that the new owner cannot track the previous transactions of the tag (i.e.; it should protect the privacy of the old owner).
- Windowing problem: The protocol should ensure that there is no time slot where the old and new owners can simultaneously access the tag during the OT, otherwise the protocol becomes a sharing protocol rather than an OT one. The windowing problem should be prevented during OT only. In OD, both the owner and the delegate should be able to access the tag simultaneously so that the owner can terminate the delegation when necessary.
- Exclusive OT: The protocol should prove that the new owner has completely taken over the ownership from the old owner (i.e. provide a proof for the exclusive ownership of the new owner). Violating the new owner privacy (which means that the new owner may not be the only owner of the tag) would also mean violating the exclusive OT property.

In addition to the previously mentioned properties, the OT protocol should satisfy the following properties in order to be considered a complete protocol

- Controlled delegation: The protocol should allow for controlled delegation in addition to the complete transfer of ownership. Controlled delegation indicates that the delegate can access the tag for a predetermined number of times or until the owner terminates the delegation.
- Authorization recovery: In certain situations, the protocol should allow the old owner to access the tag again after its ownership has been transferred. For example, when a tagged item needs to go under warranty maintenance, the retailer (i.e. old owner) should be able to access the tag again to perform the maintenance. In the meantime, the new owner is still the owner of the tag. Thus, the same approach that is used to carry out the controlled delegation can be used to perform the authorization recovery (AR). In other words, AR is considered a special case of the controlled delegation where the delegate is a previous an old owner.
- Supporting Mobile Readers: In mobile RFID systems, the readers and tags are not fixed and change their location regularly [5]–[7]. Moreover, in these systems the readers communicate with the back-end servers via wireless networks [8]. In the literature, one can find some OT protocols that support mobile readers. These protocols take into consideration the characteristics of the mobile RFID systems and are designed to perform OT in them.

It must be mentioned that the new owner's privacy notion has been represented by other terms such as forward untraceability, backward privacy, and forward security. All of these denote the same meaning and can be used interchangeably. The same applies for the old owner privacy where it can be replaced by other terms such as backward untraceability, forward privacy, and backward security. The use of different terminology in the literature is quite confusing when the same terms refer to different things. As such, in this survey, we solely use the terms new owner privacy and old owner privacy; respectively, even if different terms were used in the original papers.

In 2006, a survey on RFID security and privacy was published by Juels [**?**]. In his survey, Juels discussed issues related to general RFID systems, cryptography and possible threats. At that time, OT was in its earliest stages and few protocols were considered in the survey. The earliest OT protocols that were reported in the literature are the protocols of Saito *et al.* [9] and Molnar *et al.* [10] in 2005. After these two protocols, several works that aim to develop OT protocols or analyze the existing ones were proposed. In his thesis, Kapoor assessed some of the existing OT protocols in 2008 [11]. In the same year, the OT protocols that were proposed around that time were briefly presented in Langheinrich's survey on RFID privacy approaches [12]. What appeared in both works is useful although a more thorough and broader investigation of the literature that appeared later is needed.

In this work, we present and analyze the OT protocols from the earliest proposed ones up to the current state of the art. The objective of this survey is to help in providing a thorough and systematic understanding of the OT protocols by presenting a classification for these protocols and discussing their advantages and vulnerabilities.

The main focus of this survey is to discuss each protocol from the OT perspective only. However, in Table 10 in Section VII we specify whether the protocol supports OD, AR, or both.

B. OWNERSHIP TRANSFER PROTOCOLS

Several OT protocols were proposed to secure the OT process in RFID systems. An entity is considered an owner of a tag by exclusively sharing certain secrets with it. Therefore, when the tag's ownership is to be transferred to a new owner, the shared secrets should be updated to new values that are only known by the new owner and the tag. Generally speaking, any OT protocol should involve at least the following three steps:

1) The current owner (who eventually becomes the old owner when the OT is completed) updates the shared secrets with the tag to new values.
2) The current owner forwards the updated secrets to the new owner via a secure channel.
3) The new owner employs the received secrets to generate new ones and then exchanges them with the tag. The exchange process between the new owner and the tag should be performed securely without the intervention of the current owner (i.e. old owner).

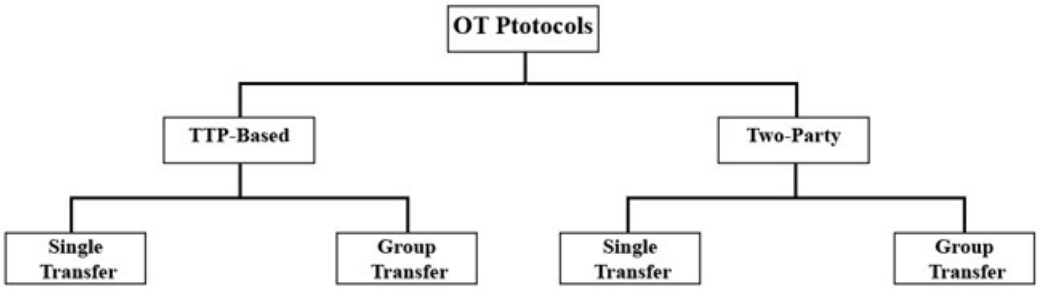

FIGURE 1. Classification of OT protocols

Based on the reviewed literature, the proposed OT protocols are classified into two main categories: protocols that are based on the existence of a trusted third party (TTP-based OT protocols) and protocols that depend only on the communication between the owners (Two-party OT protocols).

In TTP-based OT protocols, there exists a trusted third party (*TTP*) in addition to the main entities of the RFID system. The purpose of the *TTP* is to help in transferring the tag's ownership rights from the current owner(s) to the new owner(s). Thus, in this class, the *TTP* supervises the exchange of secret keys between the new owner(s) and the tag(s). It is also responsible of confirming that the current owner has completely released the tag's ownership (in some situations this can be carried out implicitly by updating the old keys) and that the new owner has become the exclusive owner. In some protocols, the back-end server carries out the role of the *TTP*.

On the other hand, in Two-party OT protocols there are no *TTP*s and the current and new owners complete the OT process by communicating directly with each other and with the tag(s). Moreover, some protocols assume that there is an isolated (i.e. secure) environment (ISE) where the new owner(s) can exchange the secret keys with the tag(s) with no interference from the old owner(s). Therefore, this category is sometimes called ISE-Based.

Each of the two categories has its own supporters. Those who develop their protocols based on the existence of a *TTP* argue that it is necessary to rely on a *TTP* to secure the OT process without using heavy cryptographic functions on the tags. Otherwise, the computational and power cost would be high. Moreover, they consider that the assumption of the existence of an isolated environment in the Two-party protocols is unreasonable because if there is such an environment, there would be no need for a security protocol to encrypt the exchanged keys and passwords.

The supporters of Two-party OT protocols point out that there should be no centralized entity such as the *TTP* where all sensitive information is stored in a single location. Because compromising the centralized entity would allow the adversaries to perform all sorts of attacks. In addition, they argue that the new owner can find an isolated environment to complete the OT process. Thus, an OT protocol can be implemented without using heavy cryptographic functions.

The TTP-based and Two-party OT protocols can be further classified into either protocols that perform either single transfer or protocols that perform group transfer. In single transfer, the ownership of only one tag is transferred from one current owner to one new owner (i.e. single-tag to single-owner relation). On the other hand, in group transfer the protocol simultaneously transfers the ownership of a group of tags from a single current owner to a single new owner, transfers the ownership of a single tag from a group of current owners to a group of new owners, or transfers the ownership of a group of tags from a group of current owners to a group of new owners (i.e. multi-tag to single-owner, single-tag to multi-owner, or multi-tag to multi-owner relation).

The classification of OT protocols is shown in Fig. 1.

C. NOTATION

One issue to consider at the beginning of surveying the literature is the non-consistent notation and operations used in different papers. To better present the ideas of the protocols and to explain all protocols with the same terminology, we adopt a unified representation of the notation and operations.

The unified notation and operations used in this work are given in Table 1 and Table 2.

D. GENERAL SCENARIO OF OWNERSHIP TRANSFER

In 2005, Saito *et al.* proposed one of the earliest OT protocols in the literature [9]. In this section, we cover this protocol in order to lay the foundation and demonstrate the main concepts of OT protocols and the threats that they may face.

Saito *et al.* proposed two OT protocols, one of which is TTP-based. We start by covering the TTP-based protocol, shown in Fig. 2.

In the TTP-based protocol, four entities are involved: the *TTP*, current owner (R_c), new owner (R_n), and tag (T). The authors assumed that the communication between the owners is secure as well as that between the owners and the *TTP*. In this protocol, R_c who currently owns T, will transfer its ownership to R_n with the assistance of the *TTP*. T shares with R_c the secret key $K_{R_c,T}$, and with the *TTP* the secret key $K_{TTP,T}$. R_c initiates the OT by sending message M_1 which holds $K_{R_c,T}$ to R_n through a secure channel. Upon receiving

TABLE 1. Unified notation used in this work.

Notation	Meaning
T	RFID tag
R	RFID reader/owner
R_o, R_c, R_n	Old, Current, New reader/owner
CR_{R_c}, CR_{R_n}	Coordinator reader/owner of the old/current and new owners; respectively
BS	RFID Back-end Server (or database)
TTP, R_{TTP}	Trusted Third Party, Reader of TTP
BS_{R_x}	The back-end server trusted by R_x, where R_x can be R_o, R_c or R_n
PX, PX_X	Guardian Proxy, Guardian Proxy owned by entity X
X_i	The i^{th} entity X. For example, T_i denotes the i^{th} tag, R_i denotes the i^{th} reader/owner
$Group_X$	Group of members, where X represents an entity. For example $Group_T$ denotes a group of tags, $Group_R$ denotes a group of readers/owners.
ID_T	Unique static identifier of tag T
$Info(ID_T)$	Related information of tag T
Ent_T^{old}, Ent_T^{new}	The old and new record of the tag T at the back-end sever
$IDE(X)$	Identity of entity X
IDS	Index Pseudonym
C_X, $C_X^{'}$	Ciphertext belonging to entity X , Re-encrypted C_X
$Cert_X$, $Cert_X^{'}$, $Cerd_X$	Certification of entity X, Randomized certification of entity X, Credentials of entity X
N_X, $(N_i)_X$	Random number generated by entity X, the i^{th} N_X
$f_{X,K}$, $b_{X,K}$	Forward key chain and Backward key chain computed using key K and belong to entity X
Cnt, Cnt_{max}	Counter, Maximum counter value
F_{OT}, F_U	Ownership transfer flag/command/request , Update flag/command/request
PW_X	Password that belongs to entity X
SN	Serial Number of the product
SG	Signature of the brand company
PIN	Unique Personal Identification Number
ACK_X	Acknowledgement sent by entity X
M_i, $(M_i)_X$	A message to be exchanged or employed in the computation of another message, A message computed by entity X
Msg	Confirmation message to confirm a transaction
Adv	Adversary
Δ	Some random value used by Adv
K_X	Key generated by/ belongs to entity X
$K_{X,Y}$	Key shared/will be shared between entity X and entity Y
K_{all}	Key shared by all entities
K_i	The i^{th} key
SK_X	Symmetric (or secret) key of entity X
PK_X	Asymmetric (or public) key of entity X
L_x	The length (in bits) of x
$(x)^{new}$	The new value of x
$(x)^{old}$	The old value of x

TABLE 2. Unified operations used in this work.

Notation	Meaning
$E_X(m)$, $LE_X(m)$,$D_X(m)$	Encryption, Lightweight Encryption and Decryption functions that take m as input and X as key
$Sign_X(m)$	Sign m with X, where X is the private key of an entity
$F_X(m)$	Pseudo Random Function that takes m as input and key X as a seed
$F_1(m)$, $F_2(m)$	Pseudo Random Functions that take m as input. The length of the output of F_1 is different than that of F_2
RNG	Random Number Generator. The output can be a pseudorandom or a true random number.
$H(m)$	One way hash function that takes m as input
$H_X(m)$	Keyed hash function that takes m as input and X as key
$\oplus$, $\|$	XOR, Concatenation
CRC	Cyclic Redundancy Check
$EXT_L(m)$	Extract L bits from m
$\{EXT_L(m)\}_{(i=0)}^{(j-1)}$	Group of extractions, where $0 < i \leq j-1$
p, q	Large prime numbers
$Rep(X,Y)$	Successive execution of operation X for Y times. The output of each execution becomes the input for the next one.

M_1, R_n randomly generates a new key K_{R_n} and then it sends it, using message M_2, along with the received M_1 to the *TTP* via a secure channel. The *TTP*, in turn, encrypts the received messages by computing message $M_3 = E_{K_{TTP,T}}(M_1 \| M_2)$ and sends it to R_n which in turn forwards the message to T. Once T receives M_3, it decrypts it by computing $D_{K_{TTP,T}}(M_1 \| M_2)$ in order to obtain M_1 which holds $K_{R_c,T}$ and M_2 which holds K_{R_n}. After that, T checks whether the extracted $K_{R_c,T}$ matches the stored one and if both values match, it verifies the legitimacy of the received message and updates its key to the extracted K_{R_n} and by that R_n becomes the new owner.

At first sight, one may assume that this protocol satisfies the requirements of a secure OT since the new owner has safely shared its key with the tag and the old owner can no longer access the tag or track the new owner's transactions with it. Thus, Saito *et al.*'s TTP-based protocol achieves new owner privacy.

However, this protocol violates the old owner privacy because the old owner shares its secret key with the new owner. Therefore, using the old owner's key, the new owner can decrypt all previous transactions between the old owner and the tag.

Moreover, the tag does not send an acknowledgment to indicate that it has updated its key and this can lead to de-synchronization problems. If the last message sent from the new owner to the tag is blocked, the tag will not update its key. On the other hand, the new owner assumes that it has become the owner of the tag and so does the *TTP*. Hence, the new owner will not be able to communicate with the tag as it still uses the old key.

This protocol is not suitable for low-cost tags as it uses a symmetric key cryptographic function, which is not supported as indicated by the EPC-C1G2 standard. Furthermore, Saito *et al.*'s protocol does not perform mutual authentication between the participating entities before issuing the OT.

Finally, in this protocol, the tag stores the secret key $K_{TTP,T}$ that is shared with the *TTP*. If this key is exposed by an *Adv*, it can compromise the whole system. Using $K_{TTP,T}$ and messages M_1 and M_2 which contain the old and new keys; respectively, *Adv* can decrypt all previous and upcoming transactions in the system. In addition, it can forge its own tags and lead the *TTP* to authenticate them as legitimate tags.

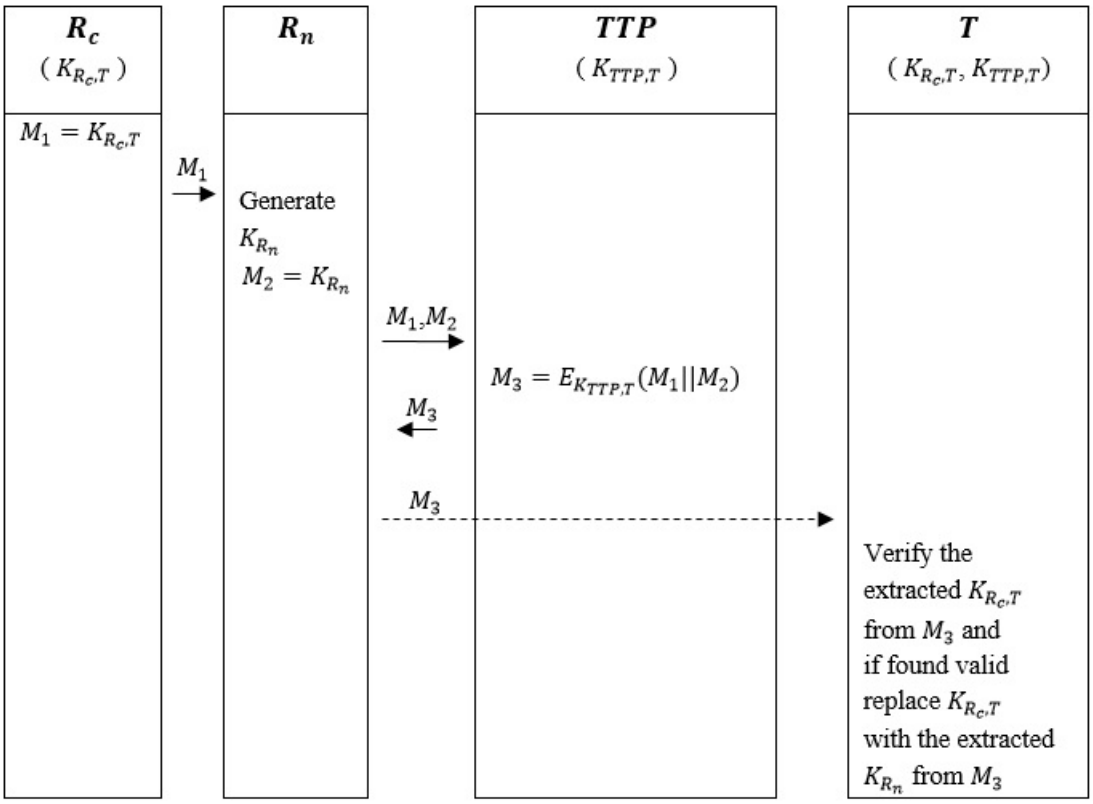

FIGURE 2. Saito *et al.*'s TTP-based protocol

As can be seen from this quick analysis, Saito *et al*.'s TTP-based protocol suffers from several vulnerabilities which may threaten the privacy and security of the involved entities.

Next, we consider the second protocol in the work of Saito *et al*., which does not involve a *TTP* and is only based on the communication between the current and new owners. In this protocol, three entities are involved; current owner R_c, new owner R_n and T, as shown in Fig. 3.

OT is started by R_c which securely sends message M_1 to R_n through the secure channel. R_n, in turn, transmits a query to T, which responds by generating a random nonce N_T and sends it as message M_2. When R_n receives M_2, it generates a new key K_{R_n} and then computes message $M_3 = E_{M_2}(M_1 \| K_{R_n})$ and sends it to T. At last, T decrypts the received message to obtain $K_{R_c,T}$ and K_{R_n} and if the received $K_{R_c,T}$ is verified, T updates its key to K_{R_n}.

The second OT protocol suffers from a serious shortcoming in addition to the previously mentioned ones. In this protocol, message M_2 which holds N_T is used by the new owner to encrypt its and the old owner keys. Since M_2 is transmitted in plaintext, any entity that captures this message can decrypt message M_3, which hides the owner's keys and this violates their privacy. In other words, if the old owner captures M_2, it can obtain the new owner's key. Thus, Saito's second protocol breaches the new owner privacy.

In the following sections, we provide an overview of the OT protocols based on the taxonomy given in Fig. 1.

III. TTP-BASED OWNERSHIP TRANSFER PROTOCOLS

In this section, we cover the TTP-based protocols present in the literature. We start by demonstrating the single transfer protocols and then go to the group transfer.

A. TTP-BASED SINGLE OWNERSHIP TRANSFER PROTOCOLS

TTP-based Single OT protocols are classified into two categories; namely EPC-compliant and non-EPC-compliant. The EPC-compliant protocols adhere to the EPC-C1G2 standard which was previously discussed in I-A. Mainly, the computational capabilities of the tags in the passive RFID systems are limited to simple bitwise operations, 16-bit PRNG, and 16-bit CRC functions only. Furthermore, the standard states that the tag ID should be of 96-bit length and that any tag must store two passwords; the KILL and ACCESS passwords. On the other hand, non-EPC-compliant protocols break at least one of the rules specified by the standard. In this section, we first explain the EPC-compliant protocols and then move to the non-EPC-compliant protocols. For each protocol, we start with a brief description of its procedure and then discuss its vulnerabilities.

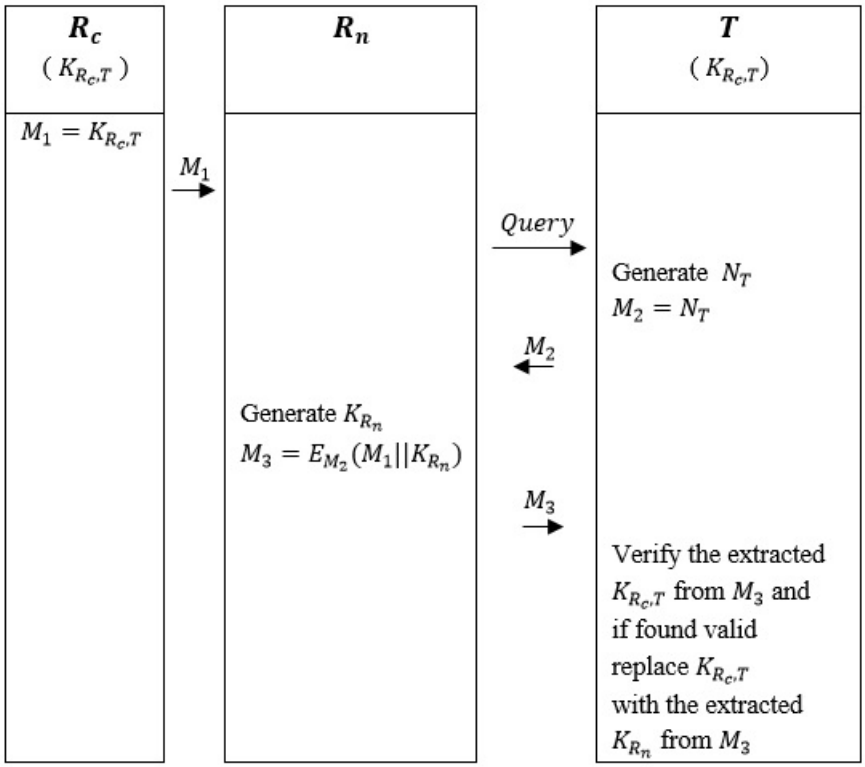

FIGURE 3. Saito *et al.*'s Two-Party protocol

1) EPC-COMPLIANT TTP-BASED SINGLE OWNERSHIP TRANSFER PROTOCOLS

We start with Lim and Kwon's protocol which emphasizes that an OT protocol should achieve new owner privacy in addition to the old owner privacy before and after the tag is compromised [13]. The main operations employed in this protocol are bitwise operations, random number generators (*RNG*), and pseudorandom functions that are implemented from a single lightweight block cipher. To achieve new owner privacy in their protocol, Lim and Kwon perform mutual authentication between the *TTP* (represented as the back-end server (*BS*) in the protocol) and T before OT takes place.

Fig. 4 shows the messages that are exchanged to mutually authenticate *TTP* and T in Lim and Kwon protocol. The mutual authentication protocol is initiated by R_c in which it queries T by sending message M_1 which holds the random nonce N_{R_c}. T in turn replies with M_2, M_3, and M_4. R_c then forwards the received messages along with M_1 to the *TTP* for validation. If the *TTP* verifies the received messages, it transmits M_5 to R_c and updates the tag's records (i.e. Ent_T^{old} and Ent_T^{new}). R_c then forwards M_5 to T which in turn verifies its integrity. If M_5 is verified (i.e. the mutual authentication succeeded), T updates the backward key chain $b_{T,K_{TTP}}$ and the secret key $K_{TTP,T}$ ($K_{TTP,T}$ is refreshed using $b_{T,K_{TTP}}$ and the exchanged random nonces) and sets its counter *Cnt* to 0. *Cnt* is used to limit the number of queries to the tag to prevent the adversaries from querying it endlessly. If the mutual authentication results in a success, the secrets at the *TTP* and T sides are simultaneously renewed in order to protect the new owner privacy.

In case of failure of mutual authentication, if *Cnt* is less than j (the maximum number of allowable authentication failures), T deterministically updates the secret key $K_{TTP,T}$ only by using a forward key chain (i.e. $K_{TTP,T} = F_1(K_{TTP,T})$ and increments *Cnt* by 1. When $Cnt = j$, T stops updating its secret key.

After the *TTP* and T have successfully authenticated each other, OT can be carried out in two steps. At first, R_n communicates with the *TTP* through a secure channel in order to obtain the tag's information (i.e. Ent_T^{old}, Ent_T^{new}, and ID_T). After that, it queries T (using the random nonce N_{R_n}) which eventually leads the tag to update its secrets and this makes it readable by R_n only.

However; as shown in [14], the tag in Lim and Kwon's protocol can be tracked by forcing it to stop updating its secret. In other words, *Adv* continues querying T for more than j times which causes its secret to remain static.

A year later, Seo *et al.* [15] tried a different approach by proposing a protocol for granular data access and OT. The proposed protocol employs Public Key Infrastructure (*PKI*) [16] and a proxy [17]. A proxy is a personal RFID-privacy device that writes data into tags and validates

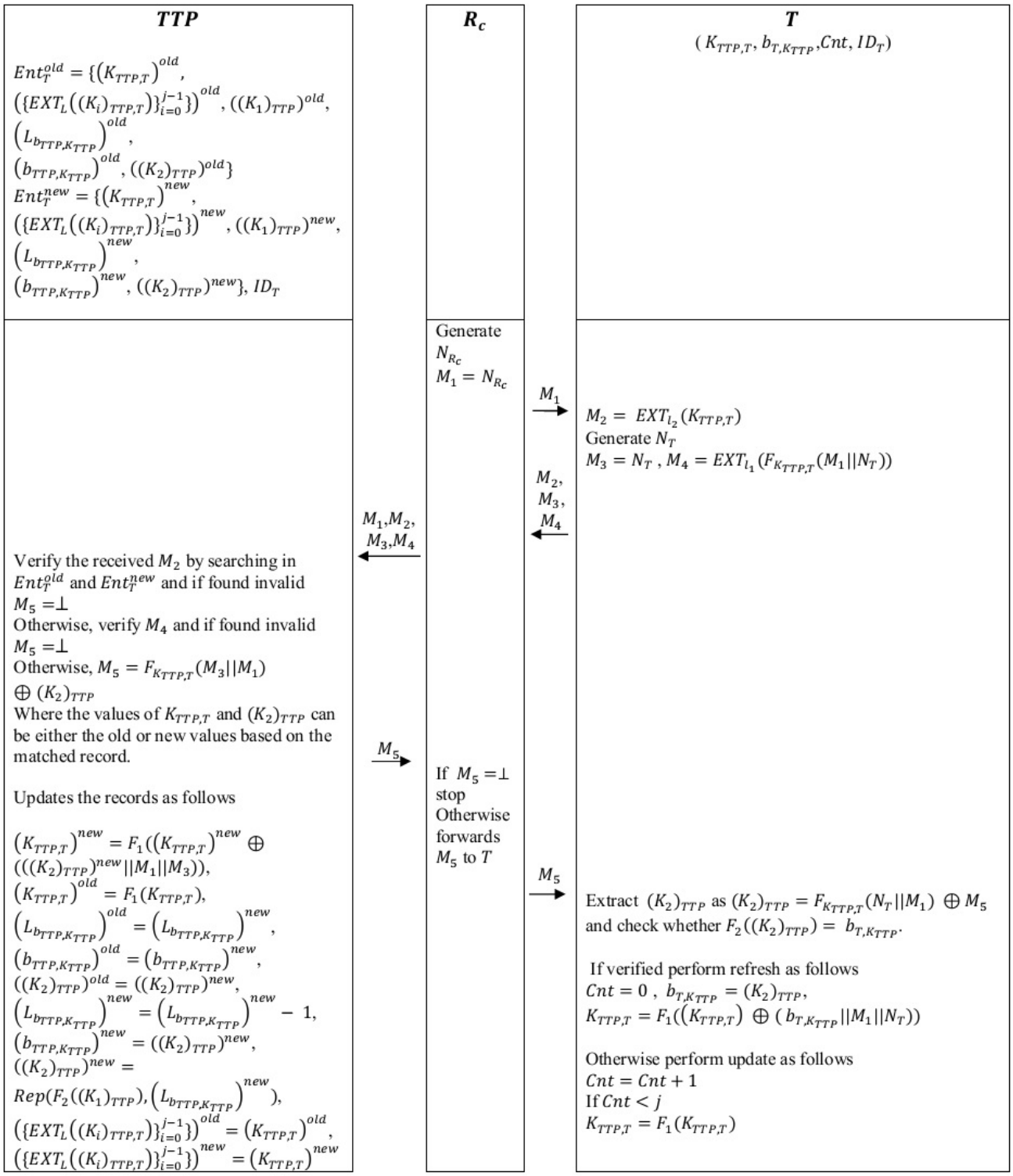

FIGURE 4. Lim and Kwon protocol.

the data received from owners. If the received data are verified, the proxy updates the tag secrets. Moreover, some proxies have access control lists which are used to implement owners' privacy policies.

In Seo *et al.*'s protocol, each owner has its own proxy which performs six functions: Tag Acquisition, Information Management, Relabeling, Authentication, Access Control and Tag Release. In addition, the proxy stores the server

location (*SL*) which helps the owners to communicate with correct *BS* directly to reduce the searching time. The *PKI* is performed on the *TTP*, the proxy, and the owners side, whereas the tag carries out only simple bitwise operations and *RNG*. The proxy takes its turn in the authorization phase to achieve granular data access.

As for OT, R_c initiates the process by querying T with a random nonce. In turn, T replies with C_T, where C_T is a ciphertext computed based on El-Gamal cryptography and was previously written into T. Next, R_c transmits the encrypted Personal Identification Number (PIN) to R_n which in turn communicates with the *TTP* in order to be authenticated. Once R_n is verified, it generates C_{R_n} and $(PIN)^{new}$ and forwards them to T so that it updates its secrets.

Nevertheless, *Adv* can track T in Seo *et al*.'s protocol by continuously querying it. The tag simply responds by transmitting C_T that remains constant as long as T's secrets are not updated. Moreover, *Adv* can cause de-synchronization by blocking the last message sent from R_n to T which holds the new secrets. By doing so, R_n will use the C_{R_n} and $(PIN)^{new}$, whereas T will use C_T and *PIN* [18].

Osaka *et al*. [19] followed an approach similar to Seo *et al*. in which they proposed an OT protocol that is based on using a guardian proxy (*PX*) [17], [20]. Their protocol differs from Seo *et al*.'s protocol by using the insubvertible encryption (*IE*) [21] instead of *PKI*. In Osaka *et al*., *PX* is used to validate the owners and update the tag's secrets in case of successful authentication. Moreover, it performs the required cryptographic function on behalf of the tag which means that all the tag needs to do is to store some information. The main operations used by this protocol are *IE*, hash function, and *RNG*. The *TTP* acts as a certification authority that gives certificates to the tag's owner. The *PX* participates in both authentication and OT phases.

For OT phase, the PX_{R_c} starts the process by sending a request to T which responds by sending its information. PX_{R_c} then generates a random nonce and transmits it along with a ciphertext and an encrypted form of R_c's certificate to R_c which, in turn, forwards them to the *TTP*. The *TTP* verifies the received information and, if valid, it releases the ownership of T from R_c and marks its status as ''released''. After that, R_n obtains the ownership of T by sending a request to the *TTP* through PX_{R_c}. When the *TTP* receives the request, it checks the legitimacy of R_c and whether T is in ''released'' state. If all conditions are met, R_n can be granted the ownership of T by letting its PX_{R_n} communicate with the *TTP* to write the new data into T and generate a public-private key pair for R_n. After that, R_n becomes the new owner of T and T's status becomes ''available''.

Another approach for OT using digital signatures was followed by Chen *et al*. in [22]. Chen *et al*. proposed a protocol that combines online authentication with digital signatures in order to resist counterfeiting. Online authentication means that the *BS* (represented by the brand company in the protocol) authenticates the Original Equipment Manufacturers, the commercial agent, and the *TTP* (given as the authorized agent in the protocol) and gives them certificates if the authentication succeeds. Therefore, only the entities that are authorized by the *BS* can verify the digital signatures. Similar to Osaka *et al*.'s protocol, T in this protocol is only required to store information and does not perform any computation. Moreover, all the current and new owners need to do is to enter passwords and compute their hash values. As for the other entities (*BS* and *TTP*) they use concatenation, hash function, and digital signatures.

Before starting OT, R_c needs to prove to R_n that the product is not a fraud and that it can trust the *TTP*. The *TTP* communicates with T and *BS* to get verified. After that, both owners enter their passwords to the *TTP* which in turn transmits them in an encrypted form to the *BS* for validation. If the *BS* verifies the received passwords, it generates new secrets, updates T's stored records and sends the new secrets to the *TTP*. As a final step, the *TTP* verifies the received information and, if found valid, it writes the new secrets into T. Moreover, the *TTP* prints for R_n the transaction receipt which includes the necessary information.

An *Adv* can track T in Chen *et al*.'s protocol because it sends its secret information to the *TTP* in clear text. In addition, once *Adv* knows this secret information, it can clone the tag. A last note on this protocol is that because R_c and R_n are required to enter passwords manually in a secure environment, there is no need for an OT protocol in the first place and the protocol is not suited for mass production [23].

In [24], Chen *et al*. proposed another OT protocol as an improvement over their anti-counterfeit protocol. The proposed protocol works in a similar approach except that the tag stores information and also performs computations using simple bitwise operations and lightweight operations such as the *CRC* and *RNG*. These operations are used by the tag to send its information in an encrypted form and to verify the received messages. As for the *TTP* and the *BS*, they employ symmetric key cryptography (*SKC*) in addition to the same operations used by the tag to encrypt the exchanged messages and verify their correctness. Moreover, it adds the 32-bit Kill and the 32-bit Access passwords (as specified by the EPC-C1G2 standard) to the structure of the messages and avoids sending sensitive information in clear text.

Another protocol based on using the *CRC* and *RNG* on the tag's side was proposed by Chen *et al*. [25]. The *CRC* and *RNG* are also used by the authorized reader, the *TTP*, and the owners. However, the *TTP* and the owners further employ *PKI* and a hash function. R_c starts the OT by sending tagged-item certificate ($Cert_T$) encrypted to R_n. After that, the authorized reader performs mutual authentication with T. When the mutual authentication is completed successfully, R_n decrypts the received message to obtain $Cert_T$ and, if found valid, it forwards it to the *TTP* through the authorized reader. The *TTP* then verifies the received $Cert_T$ and generates a new certificate with its hash value. It sends the new certificate and its hash to the authorized reader which writes the hash value into T and then forwards the new certificate to R_n.

Abyaneh showed in [26] that T in Chen *et al.*'s protocol can be traced by R_c who can eventually impersonate it. Therefore, Chen's *et al.*'s protocol does not guarantee the new owner privacy.

Kulseng *et al.* [27] focused on proposing a protocol that supports mutual authentication and OT while simultaneously being efficient from a hardware perspective. Thus, they proposed a protocol that is based on using minimalistic cryptographic primitives such as Physically Unclonable Functions (*PUF*) [28] and Linear Feedback Shift Registers (*LFSR*) [29]. Moreover, the authors used simple bitwise operations such as the *XOR*. Using *PUFs* and *LFSRs*, which are implemented efficiently in hardware, this protocol requires only 784 gates for 64-bit variables. Hence, it is suitable for the cheapest low-cost tags.

Before starting the OT, R_c and T should mutually authenticate each other. After successfully completing the mutual authentication protocol, R_c forwards to R_n the needed information including the key $(K_2)_{R_c}$ that belongs to R_c for the OT. In the meantime, it notifies the *TTP* about the verification pair $((K_1)_{R_c}, (K_2)_{R_c})$ that is used to verify R_n and T. R_n starts the OT by securely transmitting a request to the *TTP* along with $(K_2)_{R_c}$. In its turn, the *TTP* verifies the received request in order to authenticate R_n. If the authentication succeeds, the *TTP* communicates with T through R_n, which works as mediator, and forwards the messages between them in order to update the secrets at both sides. The *TTP* transmits the new secrets to R_n so that it becomes the new owner of T and can perform mutual authentication with it.

Nevertheless, in [30], Kardas *et al.* revealed several vulnerabilities in Kulseng *et al.*'s protocol. The authors demonstrated that the mutual authentication protocol is susceptible to de-synchronization attack which can be accomplished by message blocking in the third round of the protocol or by message injection attack. The first de-synchronization attack can be solved by storing the old secret value at T's side which will be used by R_c if the response from T is not known to it.

Moreover, they showed that ID_T can be revealed in one session through a disclosure attack launched against the mutual authentication protocol which eventually leads *Adv* to trace T. This attack can be carried out because of the misuse of *LFSR* in hiding the secrets. In order to resist the proposed disclosure attack, Kardas *et al.* suggest using a shrinking generator instead of the *LFSR*, where shrinking generator is a lightweight cryptographic primitive that is stronger than *LFSR* [31], [32].

The authors also proved that the proposed OT protocol does not guarantee the new owner privacy because R_c can track the tag using ID_T -which remains constant even after the OT is completed- and eavesdropping on the messages exchanged between R_n and T.

Therefore, in [33], Cui proposed mutual authentication and OT protocols that are considered an enhancement over Kulseng *et al.*'s protocols. Cui's protocols have the same flow of messages as Kulseng *et al.*'s protocols. However, the message structure is modified in order to resist the previously mentioned attacks against the mutual authentication protocol and to increase the security of the OT protocol. Moreover, Cui suggests storing the old secrets at the tag's side as a remedy for resisting the de-synchronization attacks.

Similar to the work in [27], Lo *et al.* [34] proposed an OT protocol that focuses on achieving hardware efficiency by making the tag perform lightweight operations only. The tag in this protocol employs the *XOR* operation, a *RNG*, and a lightweight *Nun* function which is built from shift and addition operations. Lo *et al.*'s *Nun* function is a modified version of the original *Nun* function which is used to generate a pseudonym that depends on the tag's identity [35]. The modified *Nun* function, however, is used to validate the integrity of the transmitted messages. This function accepts two operands: the value to be verified and the random value used to reshuffle the latter. As for the owners and the *TTP*, they use the same operations used by the tag. The *TTP* adds the use of a hash function.

Before starting the OT in Lo *et al.*'s protocol, it is assumed that R_c and T have mutually authenticated each other. After that, R_c initiates the OT by sending an OT request to the *TTP* which in turn checks if R_c is authentic. If the verification succeeds, the *TTP* generates a temporary key and sends it in an encrypted form to R_c. R_c then re-encrypts the received key and forwards the result to T. In its turn, T verifies the received key and, if found valid, it update its secret key to the received value. After that, the *TTP* sends the temporary key and ID_T to R_n which then generates a new key and transmits it encrypted to T. T then verifies the received key and if found valid, it updates its secret key to the received value and by that R_n becomes the new owner.

Niu *et al.* [37] proposed a protocol that employs simple bitwise operations (such as the permutation [36]) in addition to a *RNG* while taking into consideration the importance of efficient hardware implementation. All parameters are of 96-bit length and to simplify the implementation, each 96-bit value is divided into 16-bit words. Thus, the computations are performed six times to obtain the 96-bit result.

In Niu *et al.*'s protocol, R_c starts by performing mutual authentication with T and when completed successfully, the OT can be initiated. The OT is started by the *TTP* which updates the master key that is shared with R_c and T to a new value and then it sends the new master key via a secure channel to R_n. After that, the *TTP* transmits the new master key in an encrypted form to T which checks its validity. If the verification succeeds, it updates its master key to the new value. As a last step, R_n and T need to mutually authenticate each other in order confirm the success of OT

Based on the analysis of Bagheri *et al.* [38], it was shown that Niu *et al.*'s protocol cannot resist secret disclosure and de-synchronization attacks.

The authors proposed several disclosure attacks that are based on one fact that the messages in Niu *et al.*'s protocol are generated by 16-bit *RNG*s and each 96-bit message is broken into 16-bit words. *Adv* computes, in off-line mode, a table which holds the pairs $(x, RNG(x))$ for $0 \leq x < 2^{16}$.

Thus, if *Adv* has $RNG(x)$, it can find the associated x value by looking it up in the table.

In addition, the authors showed that a de-synchronization attack can be launched against the mutual authentication phase of the protocol.

On the other hand, Taqieddin *et al.* [39] proposed an enhancement to Niu *et al*.'s protocol such that it can resist the attacks proposed in [38]. The proposed improvement involves using the bitwise conversion operation [40] and Hamming Weight-based left rotation instead of the 16-bit *RNG* and the permutation operation in order to secure the protocol from disclosure attacks. The enhanced protocol adds extra messages to Niu *et al*'s mutual authentication protocol and employs the old-IDS-denial mechanism [41].

We end this section with the cloud-based RFID OT protocol (CROP) which was recently proposed by Cao *et al*. [42]. As the name implies, CROP is an OT protocol that is designed for cloud-based applications. It is based on the quadratic residue property and on the existence of a cloud provider that is responsible for storing the owners and tags information. In other words, in cloud-based systems, the provider carries out the role of the *TTP*. The main operations employed in the protocol are bitwise operations, *RNG* and a hash function. Since the tag does not perform hash functions, the protocol is compliant with the EPC-C1G2 standard.

In cloud-based environments there are usually two assumptions:

- The cloud database is insecure and the adversaries can compromise it.
- The communications between the readers, tags, and cloud database are performed through wireless channels. Thus, *Adv* can eavesdrop on the exchanged messages.

Hence, two security requirements need to be considered in cloud-based systems:

- Data Privacy of the Cloud-storage service: The cloud database protects the privacy of the stored information by encrypting the keys of the readers and tags.
- Access anonymity of the inquiry service: The cloud database achieves anonymity of the access and inquiry service by encrypting the messages exchanged between the readers and the tags.

In designing CROP, the authors took into account the importance of achieving these two security properties.

The cloud provider stores an encrypted hash table (*EHT*). The *EHT* pre-stores the encrypted keys for every reader and its purpose is to secure the access anonymity and the stored information from untrusted cloud providers. The *EHT* consists of several records where each record is identified by a unique index. The index is a hash digest that uniquely indicates the current record (or session) of the owner.

CROP begins with an off-line authentication phase in which R_n sends an OT request to T which replies with its encrypted information. R_n then forwards the information necessary to authenticate T to R_c. It also sends a pre-stored random nonce in an encrypted form to R_c through a secure channel. In turn, R_c employs the received information to verify T and, if found authentic, it responds with an acknowledgment (ACK_{R_c}). After that, R_c sends a hash value of its secrets to the *TTP* (represented by the cloud provider in the protocol) via a Virtual Private Network (*VPN*) channel in order to get authenticated. *TTP* in turn searches in the *EHT* for R_c's record and then sends the contents of the record securely to R_n through a *VPN* channel. It should be noted that this step is the only step that is performed on-line by interacting with the *TTP* in this phase. R_n then checks the validity of the received information and, if verified, it authenticates R_c. Next, R_n sends the previously received ACK_{R_c} along with the encrypted pre-stored random nonce to T. By that the off-line authentication phase is completed.

The OT phase starts immediately once the previous phase is completed. T employs the received information to authenticate both R_c and R_n. If the authentication results in a success, it sends its key $K_{R_c,T}$ in an encrypted form to R_n which in turn verifies its validity. If found valid, R_n generates a new key and uses it to compute the acknowledgment, Ack_{R_n}. After that, R_n performs an on-line authentication with *TTP* in the same way R_c did. If the *TTP* authenticates R_n, it replies with both the record's index and content. In turn, R_n decrypts the received index to obtain the necessary information and then it generates a ciphertext by encrypting the received record's content and Ack_{R_n} and update its key K_{R_n}. Next, R_n employs the new K_{R_n} to update its record's index and content and then forwards them to the *TTP* through a *VPN* channel. Upon receiving the information, the *TTP* updates R_n's record accordingly. R_n transmits the ciphertext, Ack_{R_n}, and its original record's content (i.e.; before the update) to T which verifies the received ciphertext and then updates its secrets to new values.

TABLE 3. Main themes of EPC-compliant TTP-based single OT protocols.

Protocol		Main Theme
Lim	[13]	Backward key chain to simultaneously refresh the secrets at the authentic server and tag sides or forward key chain to update the tag secrets deterministically in case of authentication failure.
Seo	[15]	Gurdian Proxy and PKI
Osaka	[19]	Gurdian Proxy and Insubvertible encryption
Chen	[22]	Online authentication and digital signatures
Chen	[24]	Online authentication, SKC, CRC and kill and access keys
Chen	[25]	PKI and CRC
Kulseng Cui Lo Niu Taqieddin	[27] [33] [34] [37] [39]	Lightweight operations such as PUF, $LFSR$, Nun, Permutation, Conversion and RNG for efficient hardware implementation.
Cao	[42]	Quadratic residue property and a cloud provider

Table 3 lists the main themes of the EPC-compliant TTP-based single OT protocols.

In this section, we discussed the EPC-compliant TTP-based single OT protocols. Approximately half the protocols violate at least one of the following properties: the old owner privacy, the new owner privacy, and the resistance to the windowing problem. Moreover, only four protocols

provide OD or AR, whereas only two can support mobile readers. On the other hand, none of the discussed protocols provide exclusive OT.

Half of the presented EPC-compliant TTP-based single OT protocols cannot defend against at least one of the security attacks which are summarized in Section II-A.

2) NON-EPC-COMPLIANT TTP-BASED SINGLE OWNERSHIP TRANSFER PROTOCOLS

One of the earliest OT protocols that appeared after Saito *et al.*'s protocol, is the work of Molnar *et al.* [10]. This is a scalable protocol that is based on using a series (in the form of tree) of encrypted pseudonyms (secrets). The pseudonyms are generated by both the *TTP* and tag using a *PRF* and are updated each time the tag is queried. The employed *PRF* is implemented using *SKC*. In addition to using *PRF*, the participating entities can use *RNG*.

In order for R_c to query T, it requests pseudonyms from the *TTP*. The *TTP* gives R_c a series of upcoming q pseudonyms so that it can decrypt the pseudonyms encrypted by T without referring to the *TTP*. This procedure is followed for scalability reasons. Once the series of q pseudonyms is consumed, R_c communicates again with the *TTP* to request another series and so on.

To initiate the OT, R_n sends an OT request to the *TTP* which updates its security policy so that only R_n can request pseudonyms. Therefore any subsequent requests from R_c will not be honored. Occasionally, before the OT takes place, a series of pseudonyms could have been given to R_c. Hence, if this series of pseudonyms is not exhausted, T can still be accessed by R_c which violates the new owner privacy. In order to handle such situation, two approaches can be used. The first one is called soft killing where R_n continuously queries T until the series of pseudonyms that belong to R_c is consumed. In the second approach, R_n communicates with T in order to lead it to increase its counter such that it leaps over R_c's pseudonyms. The first approach is carried out by performing several queries without any key exchange, whereas in the second, two key exchanges are needed.

Nevertheless, if an *Adv* compromises T in Molnar el al.'s protocol, it may reveal the other tags' information and employs it for tracking [43].

Similar to Molnar *et al.*, Fouladgar and Afifi's work followed an approach in which an owner can identify T's encrypted pseudonyms without referring to the *TTP* for a specified number of times, *Cnt* [44]. However, the owner does not receive a series of pseudonyms, rather it receives the key necessary to decrypt them. Moreover, the participating entities encrypt the pseudonyms by using a hash function. In addition to the hash function, this protocol employs simple bitwise operations and a *RNG*.

In Fouladgar and Afifi's protocol, R_n initiates the OT by sending a query to T which in turn increments its *Cnt* and replies with N_T and an encrypted form of its secret key $(K_1)_{TTP,T}$. R_n then forwards the received information along with an OT flag (F_{OT}) and its credentials ($Cerd_{R_n}$) to the *TTP* in order to get authenticated. If R_n is authenticated, the *TTP* sends the stored T's keys ($(K_1)_{TTP,T}$ and $(K_2)_{TTP,T}$) encrypted to it. The purpose of the key $(K_1)_{TTP,T}$ is to decrypt T's pseudonyms, whereas $(K_2)_{TTP,T}$ is used to update its secrets. In turn, R_n forwards the received message to T so that it sets its *Cnt* to the maximum value. After that, R_n queries T again which, in turn, responds by another N_T and an encrypted form of $(K_2)_{TTP,T}$. R_n passes the received information along with $Cerd_{R_n}$ to the *TTP* which then verifies them and then retrieves ID_T and updates T's keys by using the random nonce N_{TTP}. The *TTP* then sends N_{TTP} and $(K_2)_{TTP,T}$ in encrypted form to R_n which forwards them to T to allow it update its keys.

After that, R_n sends a query to T to increment its *Cnt* and then replies with another N_T and an encrypted form of $(K_1)_{TTP,T}$. Upon receiving the information, R_n forwards it along with $Cerd_{R_n}$ to the *TTP* which verifies it and then responds with ID_T and $(K_1)_{TTP,T}$. Finally, R_n employs the received information to identify the tag. R_n can continue accessing the tag until the *Cnt* reaches the maximum value.

The authors proposed an enhancement to their work in [45]. In the enhanced protocol, *SKC* is employed instead of the hash function. In addition, the authors stated that if R_n does not trust the *BS* of R_c (BS_{R_c}), then T's keys must be transferred to its *BS* before the OT can take place. However, BS_{R_c} should keep a copy of the keys in case R_c needs to access T after the OT. In such a case, BS_{R_n} can force T to change its keys back to the old values.

However, the work in [46] indicates that an *Adv* can impersonate R_n to obtain ID_T by capturing the clear text of $Cerd_{R_n}$ and replaying it to the *TTP*. The attack succeeds in case R_n uses a mobile reader in Fouladgar and Afifi's protocols. In addition, *Adv* can impersonate T by capturing N_T and the encrypted $(K_1)_{TTP,T}$ and replaying them to the owner since it does not keep track of the replayed values. A solution is suggested to handle the tag impersonation by having the owner use a counter to limit the maximum number of received replays that use the same $(K_1)_{TTP,T}$ value.

Moreover, Jin *et al.* [47] noted that Fouladgar and Afifi's enhanced protocol [45] is subject to tracking attacks that violate the new owner privacy because $(K_1)_{TTP,T}$ and $(K_2)_{TTP,T}$ remain static after the OT unless the server intentionally changes them. Moreover, since this same issue is present in their hash-based protocol (i.e. SiDeS), we find that SiDeS is vulnerable to the same tracking attacks.

Another approach that is based on using hashing was proposed by Ahamed *et al.* in [48]. YA-SRAP is a protocol that does not require the existence of a *BS* for the authentication of owners and tags or for the OT. In this protocol, the *TTP* is responsible of deploying the tags and authorizing the owners. If R_c is authorized by the *TTP*, it obtains a contact list that contains information about the tags that are accessible to it. For each T, the contact list contains ID_T and the secret key $K_{TTP,R_c,T}$. Using the contact list, R_c can identify T without a need for a *BS* and this makes YA-SRAP scalable. In addition

to the hash function, this protocol employs simple bitwise operations and a *RNG*.

Before starting the OT process, R_c and T have to authenticate each other. Therefore R_c sends a request and a random nonce to T which in turn responds by transmitting its secret key $K_{TTP,R_c,T}$ in an encrypted form. R_c then tries to authenticate T by searching all records in the contact list until a match is found. If T is verified, R_c updates $K_{TTP,R_c,T}$ to a new value and then it sends an encrypted random value to T. If T validates the received information, it updates its key similarly.

After that, R_c initiates the OT by sending T's information (i.e.; ID_T and the new key) to the *TTP*. The *TTP* then updates the contact list of R_n (after being authorized) with T's information and removes this information from R_c's contact list and by that only R_n can access T.

In [49], Xei *et al*. discussed the main advantages and vulnerabilities of YA-SRAP protocol. There are two advantages of the protocol. First, all readers are identical (i.e. there is no unique identifier for each reader) and this protects the privacy of the holders. Second, the protocol provides OT in an offline environment and ensures the old and new owner privacy because after transferring T's ownership to R_n, T's secret will continue to be updated in each authentication session.

As for the vulnerabilities, the first one is that the protocol is device-dependent. That is, if the reader that was loaded with the contact list is missing, T's owner in an offline location (i.e.; where it cannot communicate with the *TTP*) may not be able to authenticate the tags unless it reloads a different reader with the contact list in an online environment. The second drawback is that the protocol is not scalable because all records in the owner's contact list need to be searched (i.e. brute force search) in order to authenticate a tag. This needs $O(n)$ complexity, where n is the number of tags in the system. Thus, increasing the number of tags decreases the efficiency of the protocol. On a final note, YA-SRAP is susceptible to de-synchronization attack by blocking or modifying the last message sent from R_c to T. This results from the fact that T does not send an acknowledgment to R_c.

As for Rekleitis *et al*., they proposed a protocol that employs hashing as in Fouladgar and Afifi [44] and Ahamed *et al*.'s [48] protocols. However, they added the use of timestamps to their protocol [50]. In their work, the authors discussed the security and privacy policies that should be adhered to in order to achieve fine granularity and context-aware information in RFID systems. Based on these policies, they proposed a tag management protocol that has a modular design and can perform tag authentication and OT operations. Modular design means that all operations in the protocol are designed in a unified form in order to enhance the implementation efficiency. The protocol uses simple bitwise operations and *RNG*.

Since this protocol uses timestamps in the computation of the messages, one of the shared secrets between the *TTP* and T is a time value called *horizon*. This values specifies a certain point in time and is publicly known. The employed time standard in Rekleitis *et al*.'s protocol is ISO 8601 international standard [51]. This protocol does not need synchronized clocks between the *TTP* and T.

OT is initiated with an authentication phase in order to authenticate T. The *TTP* starts the authentication phase by sending its identity $IDE(TTP)$, a random nonce, and the current time to T. T then compares the received time value with the stored *horizon* and if it is older, it sets the current time value to *horizon*. After that, T encrypts its key $K_{TTP,T}$ using the current *horizon* and other values then transmits the result to the *TTP*. The *TTP* then verifies the received encrypted key using its secret and, if found valid, it confirms that the concerned tag has been identified and authenticated.

After that, the *TTP* starts the second phase in which it forwards to T a new *horizon* value (i.e. $(horizon)^{new}$) and the desired operation (*Oper*) (*Oper* in this case is OT). T, in turn, responds by sending a random nonce N_T. After receiving T's response, the *TTP* computes a checksum value equal to $H(Oper, N_T, Rep(H(K_{TTP,T}), currenttime - horizon) \oplus (horizon)^{new})$ and sends it to T. Next, the *TTP* employs the computed checksum to update the secret key $K_{TTP,T}$ and then sets its *horizon* value to $(horizon)^{new}$. T checks whether the received checksum value is valid. If the verification succeeds, it updates $K_{TTP,T}$ and *horizon* in the same way the *TTP* did.

It should be noted that each time the second phase is executed, T's time is set to the current time.

In addition, the proposed protocol supports tag re-initialization operation. This operation is performed by the owner in order to interrupt the linkability between the successive secret key values and can be carried out using the second phase of the protocol. The authors stated that this operation should be performed in a safe period where the adversaries that know the current secret key value cannot eavesdrop on the communication.

Osaka *et al*. tried a different approach in which they used hashing and *SKC* to propose an OT protocol that was later used as the base for many other protocols [52]. Their protocol also employs simple bitwise operations and a *RNG*.

The protocol exchange is started by R_c who sends a query and a random nonce N_{R_c} to T. T responds by sending its secret key $E_{K_{all}}(ID_T)$ in an encrypted form to R_c which in turn forwards it along with N_{R_c} to the *TTP* for verification. Moreover, R_c generates a new key K_{R_c} and transmits it to the *TTP*. *TTP* verifies the received encrypted $E_{K_{all}}(ID_T)$ and if found valid, it decrypts it and then retrieves the item's related information $Info(ID_T)$. Moreover, the *TTP* encrypts ID_T using K_{R_c} (i.e. computes $E_{K_{R_c}}(ID_T)$ and then it updates its K_{all} to K_{R_c} and $E_{K_{all}}(ID_T)$ to $E_{K_{R_c}}(ID_T)$ respectively. After that, the *TTP* sends the $E_{K_{R_c}}(ID_T)$ in an encrypted form along with $Info(ID_T)$ to R_c. In turn, R_c uses $Info(ID_T)$ to identify T and then forwards $E_{K_{R_c}}(ID_T)$ to it. T then extracts $E_{K_{R_c}}(ID_T)$ from the received message and employs it as its new secret key.

Next, R_c changes its key K_{all} to K_{R_c} in order to protect its own privacy and then forwards the new key along with ID_T and $Info(ID_T)$ to R_n through a secure channel. R_n then

changes the received key to a new value (i.e. K_{R_n}) so that its privacy is protected and then employs this key for subsequent transactions.

T in Osaka *et al.*'s protocol can be tracked by blocking the first message sent from R_c to T and then transmitting (query, $N_{R_c} = 0$) instead. Moreover, when the *TTP* updates the secret key to $E_{K_{R_c}}(ID_T)$, *Adv* can record the last message sent from R_c to T and use it track the tag by sending it in place of N_{R_c} during the next session. In addition to tracking, *Adv* can cause de-synchronization by blocking the last message sent from R_c to T or by adding some noise to it [18].

In order to secure Osaka *et al.*'s protocol from these attacks, several works have tried to improve the protocol by modifying the original messages, adding new messages, or changing the protocol's structure [53]–[58]. However, some of these protocols still suffer from tracking and de-synchronization attacks. Even worse, some of them violate the new owner privacy and are susceptible to disclosure and impersonation attacks [18], [23].

Similar to Osaka *et al.*, Kapoor and Piramuthu [46] proposed a protocol that employs simple bitwise operations, *RNG*, *SKC*, and a hash function. The protocol is designed for scenarios that involve multi-tagged objects and the authors demonstrated their protocol using a generic case where only one tag is involved. Thus, we consider the work of Kapoor and Piramuthu as a single OT protocol.

R_n starts the OT process by sending an OT request to the *TTP* which, in turn, generates a new key $K_{TTP,R_n,T}$ and sends it in an encrypted form to T. T decrypts the received message to extract the new key and then it re-encrypts it using its secrets and sends the result to the *TTP* as an acknowledgment. After that, the *TTP* transmits to R_c the key K_{TTP,R_c} in an encrypted form and a revoke message to inform it that its privileges have been revoked. As for R_n, it sends to it the encrypted $K_{TTP,R_n,T}$ and a grant message to give it a full permission over T. R_n in turn decrypts the received message to get $K_{TTP,R_n,T}$ and acknowledges the reception by encrypting the obtained key using its secret K_{TTP,R_c} and sending the result back. R_n performs an authentication handshake with T to confirm the OT success. Thus, R_n generates a random nonce and sends it encrypted to T to confirm that it has obtained the new key. T then decrypts the received message and sends an acknowledgment to R_n to confirm the same.

Another approach was taken by Yang [8] who proposed the lightweight OT protocol (LOTP) in order to perform OT across different authorities in a mobile RFID environment. The term ''different authorities'' means that R_c and R_n are under the authority of different back-end servers. This protocol employs lightweight cryptography (*LE*) on T's side such as DESLite [59] and Grain [60] in an attempt to reduce the complexity of the performed operations. As for the other parties, they employ *SKC*. In addition to *LE* and *SKC*, the protocol uses simple bitwise operations and a *RNG*.

Yang noted that the key to secure OT in mobile RFID environments is to secure the communication between the participating entities. Therefore, the OT protocol starts with a mutual authentication phase to authenticate T. In order to achieve mutual authentication in this protocol, techniques such as Kerberos [61] and *PKI* are used.

R_c starts the OT process with the authentication phase by sending an OT request to T which then responds with its information (its secret keys and identity) in an encrypted form. Upon receiving the information, R_c then re-encrypts the received information and transmits it to BS_{R_c} for validation.

If the verification succeeds, BS_{R_c} initiates the next phase by sending T's information that is necessary for OT and $IDE(R_n)$ encrypted to BS_{R_n}. From the received information, BS_{R_n} knows that T's ownership will be transferred to R_n. Thus after verifying $IDE(T)$, it transmits T's information and $IDE(BS_{R_c})$ in an encrypted form to the *TTP* as a request to update the key $K_{BS_{R_c},T}$. The *TTP* verifies the received information and, if found valid, it randomly generates a new key, encrypts it using the key K_{TTP,R_c}, and sends the result to BS_{R_n}. Moreover, the *TTP* encrypts the new key using its key $K_{TTP,T}$ and then re-encrypts the result using key K_{TTP,R_c} and transmits it to BS_{R_c}.

When BS_{R_n} receives the message, it extracts the new key and employs it to manage T from this time forth. Also, BS_{R_n} assigns a new identity for T and registers R_n as the new owner for T. As for BS_{R_c}, it validates the received information and, if verified, it sends the encrypted new key and an F_{OT} in an encrypted form to R_c which, in turn, forwards them to T. T verifies the received information and then it updates K_{TTP,R_c} to the received new key to complete the OT.

Note that although the main focus of LOTP is on OT in different authorities, it can also support OT in the same authority.

As for Zhou *et al.* [62], they proposed a protocol for health care supply chains [62] using *SKC*. The proposed protocol involves third party logistics providers (TPL) that transfer the tagged items between R_c and R_n when they are at different locations. Thus, T's ownership needs to be transferred temporarily from R_c to the TPL until the item is delivered to R_n. Both the owner and the TPL need to access T and this can be carried out using two keys; the primary (main) key for the owner and the temporary (sub) key for the TPL. The main-key is shared between the owner and T, whereas the sub-key is shared between the owner, the TPL, and T. Therefore, in this protocol the owner's key is a composition of both the main-key and sub-key. Other operations used in this protocol are simple bitwise operations and a *RNG*.

Zhou *et al.*'s protocol aims to achieve two goals: OT between R_c and R_n and granting permission to the TPL to directly access T by giving it the sub-key. The TPL obtains the sub-key at the point-of-origin and returns it back at the destination point, hence it is assumed that there are two *TTP*s, one in each point. The *TTP*s are treated as one entity because they are assumed to be identical and can communicate directly with each other through secure channels.

This protocol consists of four loops: In the first loop, R_c informs the *TTP* about a change in the tagged item status, the change can be an OT between R_c and R_n or transfer

through the TPL or both. Thus in this loop, R_c sends its composite key in an encrypted form to the *TTP*. In situations where only the sub-key needs to be changed (i.e. there is no owner change) the sub-key value is set to null. In the second loop, the *TTP* generates a new main-key and informs R_n about it. R_n responds by generating a new sub-key for T when necessary (i.e. when there is no owner change but a TPL is involved) and sends it in an encrypted form to the *TTP*. Next, the *TTP* informs T about the new main-key and the new sub-key (if it has been changed) in the third loop. After that, the *TTP* informs the TPL about the new sub-key in the fourth loop. This loop is optional because it is carried out only if a new sub-key has been generated. When the *TTP* completes the optional fourth loop, it sends an acknowledgment to R_c. By this, the first loop is completed and so is the OT.

Another *SKC*-based approach was proposed by Chen *et al*. in [63]. The authors proposed mutual authentication and OT protocols for mobile RFID systems in supply chains. These protocols aim to provide secure OT for the different roles in the supply chains. The *TTP* is responsible of initializing the parties in the system, controlling the goods' positions, connecting the supply chain with good mobility, assigning permissions to parties and performing quick transfer. In addition to *SKC*, the proposed protocols utilize a *RNG*.

The authors proposed three protocols; the mutual authentication protocol, the OT protocol and the tag key update protocol. The purpose of the mutual authentication protocol is to ensure the legitimacy of the owner and to assign the user's authority by the administrator.

As for the OT protocol, it takes into consideration the different parties that exist in the supply chain environments. Thus, the authors demonstrated three cases for OT; server-to-server, reader-to-server and reader-to-reader.

In the server-to-server case the ownership is transferred from the upstream firm BS_1 to the midstream firm BS_2. BS_2 starts the process by generating a random nonce N_{BS_2} and sending it along with an OT request in encrypted form to the *TTP*. The *TTP* in turn replies with the encrypted $IDE(BS_1)$, K_{TTP,BS_1} and N_{BS_2}. In addition, the *TTP* informs BS_1 about the transfer by transmitting to it $IDE(BS_1)$, $IDE(BS_2)$, N_{BS_2} and a transferring message (info) in an encrypted form. In its turn, BS_1 encrypts $IDE(BS_1)$, the received N_{BS_2} and T's key K_T and sends the result to BS_2. At last, BS_2 verifies the received information and if the validation succeeds, it extracts K_T and records it for future use.

As for the reader-to-server case the ownership is transferred from the downstream server *BS* to the owner R. To initiate the OT, the *TTP* generates a random nonce N_{BS} and then transmits it along with info and $IDE(BS)$ in an encrypted form to R. In its turn, R decrypts the received message and then encrypts $IDE(R)$, info and N_{BS} and sends the result to *BS*. After *BS* receives the message, it re-encrypts it and forwards the result to the *TTP*. The *TTP* then validates the received information and, if verified, it sends the key K_T in an encrypted form to *BS*. *BS* obtains K_T by decrypting the received message.

The reader-to-reader case covers the OT from R_c to R_n. The OT is started by R_c which transmits $IDE(R_c)$ and info encrypted to R_n. R_n, in turn, replies by generating a random nonce N_{R_n} and sending it along with the received message and an OT command in encrypted form to the *TTP*. The *TTP* verifies the integrity of the received information and then encrypts K_T and transmits the result to R_n. R_n then verifies whether the received message is valid or not. If the validation succeeds, it extracts K_T and by that the OT is completed.

When the OT (in any of the three mentioned cases) is completed successfully, T's key should be updated. Thus, BS_{R_n} initiates the update process by transmitting a request to T. T, in turn, generates a random nonce N_T and then encrypts along with $IDE(T)$ and transmits the result to the BS_{R_n}. Upon receiving the message, BS_{R_n} re-encrypts it and sends the result to the *TTP*. When the *TTP* receives the message, it checks the integrity of IDE(BS_{R_n}) and $IDE(T)$. If the verification succeeds, the *TTP* generates a new key and then forwards it to BS_{R_n} in an encrypted form. BS_{R_n} in turn verifies the received information and if found valid, it updates its K_T to the received new key. After that, BS_{R_n} encrypts the new key and sends it to T. T verifies the integrity of the received message and then updates its key K_T to the new key value.

TABLE 4. Main themes of non-EPC-compliant TTP-based single OT protocols.

Protocol		Main Theme
Saito Fouladgar Chen	[9] [45] [63]	SKC
Molnar	[10]	Tree of encrypted pseudonymous and SKC
Fouladgar	[44]	Hashing
Ahamed	[48]	Hashing and Contact list to identify the tags
Rekleitis	[50]	Hashing and timestamps
Osaka Jappinen Yoon Chen Lei Wang Gui Kapoor	[52] [53] [54] [55] [56] [57] [58] [46]	SKC and hashing
Yang	[8]	LE and SKC
Zhou	[62]	SKC and composite key (main-key for owner and sub-key for TPL)

Table 4 summarizes the main themes for non-EPC-compliant TTP-based single OT protocols.

Unfortunately, most of the non-EPC-compliant TTP-based single OT protocols do not achieve old owner privacy, new owner privacy, nor resistance against the windowing problem. Furthermore, only five papers provide OD and none offer AR. As for the exclusive OT and support of mobile readers, we see that none of the discussed protocols offer the former and only two can support the latter.

Although these protocols use cryptographic functions such as hashing and *SKC*, most of the proposed protocols in this category cannot resist at least one of the aforementioned attacks.

B. TTP-BASED GROUP OWNERSHIP TRANSFER PROTOCOLS

Similar to the TTP-based single OT protocols, group OT protocols are also classified into EPC-compliant and non-EPC-compliant. We start this section by covering the EPC-compliant protocols and then discuss the non-EPC-compliant ones.

1) EPC-COMPLIANT TTP-BASED GROUP OWNERSHIP TRANSFER PROTOCOLS

In the reviewed literature, only one protocol that falls under the EPC-compliant group OT category was found [64]. This protocol, proposed by Sundaresan *et al.*, handles the OT in multi-tag multi-owner scenarios and focuses on achieving the privacy-among-owners (or individual-owner-privacy) property. The privacy-among-owners states that when multiple owners share the ownership of tags, each individual owner's privacy should be preserved (i.e. the communication between an owner and its tags should not be known to or accessed by the other owners). The proposed protocol implements the privacy-among-owners concept through the generation of a different secret for each new owner. The main operations employed in this protocol are simple bitwise operations and a *RNG*. Moreover, in order to prevent the adversaries from extracting any information from the exchanged messages, the protocol hides each generated random nonce by using a blind factor (i.e. in a message that contains the random nonce and other secrets).

OT is initiated by sending F_{OT} from the group of the current owners ($Group_{R_c}$) to the *TTP*. After receiving the request, the *TTP* generates a new key for each R_n in the group of new owners ($Group_{R_n}$) and then sends to it the new secret and $Info(ID_T)$ for each T in the group of tags ($Group_T$) in an encrypted form. When each R_n receives the *TTP*'s messages, it checks the validity of the messages and confirms whether it is indented for it. If the verification succeeds, R_n extracts its new secret and $Info(ID_T)$ for each T and then replies with an acknowledgment. After the *TTP* receives and authenticates the acknowledgments from all owners in $Group_{R_n}$, it transmits the identity and the new secret that are associated with each R_n in $Group_{R_n}$ to each T in $Group_T$. In its turn, each T in $Group_T$ verifies the received messages and if found, it extracts the received new secrets and identities. Next, each T in $Group_T$ responds by sending an acknowledgment to the *TTP*. When the *TTP* validates the acknowledgments received from all tags in $Group_T$, it transmits a confirmation message to all owners in $Group_{R_c}$ to confirm the OT.

In addition to the OT protocol, the authors proposed an ownership test protocol so that the new owners can confirm the successful completion of the OT. The new owners perform the test protocol to confirm that they can access all tags in $Group_T$. The ownership test protocol is performed in a safe environment without adversarial interventions and the messages are not encrypted. If test process is to be done in an insecure environment, then a mutual authentication protocol should be used for the ownership test.

This protocol is designed to transfer the ownership of all tags in $Group_T$ to the new owners. However, the authors stated that the protocol can be easily modified to support other scenarios. Two scenarios are given to explain the idea; the first one is for transferring the ownership of some of the tags in $Group_T$ to the new owners. In order to handle this scenario, the current owners send ID_T of the tags whose ownership will be transferred along with F_{OT}. After that, the *TTP* executes the OT protocol only for this subset of tags. The second scenario is for situations where tags need to join $Group_T$ or leave it. To handle this scenario, the current owners cooperate with the *TTP* to remove T's ID_T from $Group_T$ (in case T needs to leave the group) or to add ID_T to it (in case T needs to join $Group_T$).

Nevertheless, in [65] Munilla *et al.* revealed the weaknesses of Sundaresan *et al.*'s protocol and demonstrated that it is susceptible to the following attacks: de-synchronization and/or replay, tag traceability, impersonation, and forward secrecy. In their cryptanalysis, the authors focused on the step in which the *TTP* communicates with the tags in $Group_T$ because it has the vulnerabilities that lead to the proposed attacks. It was proven that Sundaresan *et al.*'s protocol does not guarantee the old and new owner privacy which was the main aim of the protocol.

We covered the EPC-compliant TTP-based group OT protocols. We see that only one protocol [64] falls under this category. This hints towards the need to design other EPC-compliant TTP-based group OT protocols. This is still an open research area, especially with the fact the EPC-compliant protocols are more suitable for supply chain applications of RFID systems.

The main theme of this protocol is based on the use *RNG* and the support of multi-tag multi-owner. However, it achieves neither the old owner privacy nor the new owner privacy. It also does not offer exclusive OT, OD, or AR neither does it support mobile readers. The protocol does succeed in resisting the windowing problem.

This protocol cannot resist any of the threats as has been shown in this section.

2) NON-EPC-COMPLIANT TTP-BASED GROUP OWNERSHIP TRANSFER PROTOCOLS

We start this section with the first group OT protocol, which was proposed by Zuo [66]. Zuo's protocol simultaneously transfers the ownership of the tags in $Group_T$ from R_c to R_n (i.e. handles multi-tag single-owner scenario). This protocol employs grouping-proofs in order to prove the existence of multiple tags at the same time and to ensure that no two parties will hold the authentication information of T at the same time during the ownership process (i.e. solves the windowing problem). The operations involved in the protocol are simple bitwise operations, a *RNG* and a hash function. Moreover, this protocol utilizes *SKC* (e.g. AES or 3DES) on the back-end servers side only.

Although Zuo's protocol can be used with any number of tags, the author demonstrated the working procedure of

the protocol assuming that $Group_T$ consists of two tags (T_1 and T_2) for simplicity. Moreover, it is assumed that all parties in the system are able to identify the tags within a group.

BS_{R_n} starts the OT process by sending F_{OT} to BS_{R_c} which in turn verifies the received request. After validating the request, BS_{R_c} issues an order for R_c to scan all tags in $Group_T$ and to verify whether all members are present. In Zuo' protocol, it is assumed that R_c scans the tags by using ALOHA scheme [67], [68]. If the verification fails, the process is terminated. Otherwise, BS_{R_c} sends an acknowledgment to BS_{R_n} which then generates a new key for each T in $Group_T$

Next, BS_{R_n} chooses two new keys for each T in $Group_T$; the first key is shared with the new owner (i.e. $K_{BS_{R_n},T_1}$ and $K_{BS_{R_n},T_2}$), whereas the second is shared with the other tags in the group (i.e.; $K_{BS_{R_n},R_n,Group_T}$). Afterwards, BS_{R_n} encrypts the new keys and transmits them in a secure manner to R_c with the help of BS_{R_c} and the *TTP*. This protocol encrypts the new keys in way such that only T_1 and T_2 can obtain them. After R_c receives the encrypted new keys, it transmits them to T_1 and T_2 simultaneously.

When T_1 and T_2 receives R_c's messages, they check their validity. If T_1 verifies the message, it computes two messages: $(M_1)_{T_1}$ which hides the key $K_{BS_{R_c},R_c,Group_T}$ and $(M_2)_{T_1}$ which encrypts the key K_{R_c,T_1}. After that, T_1 sends $(M_1)_{T_1}$ and $(M_2)_{T_1}$ to R_c which in turn stores $(M_1)_{T_1}$ (to be used as part of the group OT proof) and forwards $(M_2)_{T_1}$ to T_2. As for T_2, if it has authenticated the message received from R_c, it uses the received $(M_2)_{T_1}$ to authenticate T_1 (i.e. confirms that T_1 belongs to $Group_T$).

After verifying T_1, T_2 updates its $K_{BS_{R_c},T_2}$ to $K_{BS_{R_n},T_2}$ and $K_{BS_{R_c},R_c,Group_T}$ to $K_{BS_{R_n},R_n,Group_T}$ and then hides $K_{BS_{R_n},T_2}$ in $(M_1)_{T_2}$ and $K_{BS_{R_n},R_n,Group_T}$ in $(M_2)_{T_2}$, respectively. Next, it transmits $(M_1)_{T_2}$ and $(M_2)_{T_2}$ to R_c. Again, R_c keeps $(M_1)_{T_2}$ as a part of the group OT proof and forwards $(M_2)_{T_2}$ to T_1. In its turn, T_1 updates $K_{BS_{R_c},R_c,Group_T}$ to $K_{BS_{R_n},R_n,Group_T}$ and checks whether the received $(M_2)_{T_2}$ is valid (i.e. to authenticate T_2). If the verification results in success, T_1 updates $K_{BS_{R_c},T_1}$ to $K_{BS_{R_n},T_1}$ and then transmits $K_{BS_{R_n},T_1}$ and $K_{BS_{R_n},R_n,Group_T}$ encrypted in $(M_3)_{T_1}$ to R_c. When R_c receives $(M_3)_{T_1}$, it is employed it along with previously stored $(M_1)_{T_1}$ and $(M_1)_{T_2}$ to generate the group OT proof and then forwards the proof to BS_{R_n}, the *TTP*, and BS_{R_c} for validation.

It should be noted that in Zuo's protocol, each tag has a timer that specifies the time limit in which a tag can generate a valid proof.

The OT process is completed with a verification process in which BS_{R_n} communicates in a challenge-response mode with all tags in $Group_T$ simultaneously by using a grouping-proofs protocol or with each T individually by using a tag-reader authentication protocol. However, if not all secret keys of the tags have been verified, the entire protocol must be repeated. In this case, BS_{R_c} employs its old secrets with the assistance of the *TTP* to reset the tags.

Zuo's protocol ensures an atomic group OT by confirming that all tags have updated their keys in the same session (i.e. either all successful or nothing at all). Moreover, it follows the concept presented in [69] which states that it is preferable to authenticate group members early rather than leaving it after the protocol terminates. Detecting non-group members early can help in avoiding the waste of resources which may be considered a DoS vulnerability.

As in Zuo's protocol, another set of group OT protocols that employ simple bitwise operations, *RNG*, hashing and *SKC* were proposed by Kapoor *et al*. to handle the OT of the multi-tagged and multi-owner objects in supply chains [70]. However, in Kapoor *et al*., the tags are the only entities that cannot use *SKC*. There are two protocols proposed by Kapoor *et al*.; the first one handles the situation where a group of owners will transfer the ownership of a tagged object to a group of new owners (i.e. single-tag multi-owner scenario), whereas the other transfers the ownership of a group of tags (two or more) from R_c to R_n (i.e. multi-tag single-owner scenario). In fact, the two protocols are modeled after the single OT protocol that was proposed in [46] and previously discussed in Section III-A2.

In addition to the proposed OT protocols, Kapoor *et al*. proposed a mutual authentication protocol so that the new owner(s) can authenticate the tag(s) after OT has taken place. In all protocols, it is assumed that all involved owners are present or reachable from the tag(s) at the same time.

The first protocol works using exactly the same procedure of the protocol discussed in Section III-A2. However, the procedure is repeated for each owner in $Group_{R_c}$ and $Group_{R_n}$ and slightly changes the messages' structure.

It is worth mentioning that the first protocol covers two possible cases. The first one is that the tagged object is considered a base object and its components are just parts of it. Hence, the OT is performed only once for the entire object. As for the second case, the object is not a base one. Thus, the object does not exist without the individual components. For this case, the OT protocol needs to be repeated for each component and this can be performed either sequentially or in parallel.

Nevertheless, in [71], Bagheri *et al*. showed that Kapoor *et al*.'s protocol has a windowing problem and is susceptible to de-synchronization attack. In their analysis, the authors assumed that T updates its key to the new value after receiving R_n response. The proposed de-synchronization attack de-synchronizes the tag forever with a success probability of one while needing only two runs of the protocol sessions. To perform the de-synchronization attack, all an *Adv* needs to do is to block some messages and replay some of eavesdropped ones without modification. Furthermore, the authors proposed an enhancement in order to resist the windowing problem and the proposed attack. The suggested enhancement involves changing the structure of some of the messages and sending the result of Xoring two random nonces instead of transmitting the random nonce as is.

Moreover, a special case of the first protocol where the ownership of only one T is transferred from a single R_c to a single R_n is presented by Kapoor and Piramuthu in [18].

However, the same windowing problem and de-synchronization attack and their remedy which are proposed by Bagheri *et al*. are also applied to the single-version protocol.

Another de-synchronization attack against Kapoor and Piramuthu protocol [18] was reported in [72]. In this attack, it is assumed that T updates its secret after receiving the *TTP* s response. An *Adv* can perform this attack by simply sending a manipulated version of the *TTP* s response to T in order to lead it to update it secret differently from the *TTP*.

Furthermore, since the protocol in [18] is a single version of Kapoor *et al.* [70] group protocol, we can state that Kapoor *et al.* group protocol is also vulnerable to the attack proposed in [72].

On a final note, as mentioned previously, Kapoor *et al*.'s group OT protocol was modeled after Kapoor *et al*.'s protocol which is proposed in [46]. The only difference between the two protocols is that the group OT protocol slightly changes the messages' structure of the protocol proposed in [46]. However, since the two protocols work using exactly the same procedure we conclude that the protocol in [18] suffers from the same windowing problem and de-synchronization attack proposed in [71] and [72].

As for the second OT protocol, it consists of five loops, the *TTP* initiates the first three loops, whereas R_n initiates the last two. The protocol starts when R_c and R_n decide to transfer the ownership of the tags and inform the *TTP* about the transfer. To demonstrate the protocol, it is assumed that the ownership of only two tags (i.e. T_1 and T_2) needs to be transferred.

After being informed about the transfer, the *TTP* starts the protocol with the first loop in which it generates a new key K_{TTP,R_n,T_1} and send it in an encrypted form to T_1. T_1 in turn extracts the new key and acknowledges the reception of the message by re-encrypting the extracted key and sending the result back to the *TTP*. After the first loop is completed successfully, the *TTP* initiates the second loop by generating another new key K_{TTP,R_n,T_2} and transmitting it encrypted to T_2 which in turn acknowledges the reception similar to T_1. Next, in the third loop, the *TTP* sends the new keys (i.e. K_{TTP,R_n,T_1} and K_{TTP,R_n,T_2}) to R_n. However, before R_n acknowledges the reception of the new keys, it carries out the next two loops in order to authenticate the two tags. In other words, the fourth and fifth loops are nested in the third loop. Thus, R_n initiates the fourth loop in which it encrypts the random nonce N_{R_n} using the key K_{TTP,R_n,T_1} and transmits the result to T_1 and waits for the acknowledgment in order to authenticate it. R_n repeats the same procedure with T_2 in the fourth loop. However, it uses K_{TTP,R_n,T_2} instead of K_{TTP,R_n,T_1} to encrypt N_{R_n}.

When R_n receives the acknowledgment from the two tags and authenticates them, it sends the acknowledgment to the *TTP*. When the *TTP* receives the acknowledgment from R_n, it sends the old keys (i.e. K_{TTP,R_c,T_1} and K_{TTP,R_c,T_2}) in an encrypted form to R_c to inform it that the old keys can no longer be used to authenticate the tags. Therefore, R_c can no longer access the tags.

This protocol is a one-pass protocol between every pair of entities. Although the protocol is demonstrated in a case with only two tags, it can be extended to any number of tags. In that case, the messages should be modified by using the correct keys that are associated with each tag and in the correct sequence.

Another approach for multi-tag single-owner OT was followed by Lei *et al*. who proposed a protocol that supports a novel property called independence of old owner (IOO) [73]. He *et al*.'s protocol can transfer the ownership of multiple tags concurrently regardless of the old owner's (R_c) location. The IOO property allows R_n to update T's secrets without the need for any secure environment. Thus, even if R_c can eavesdrop on the exchanged messages between R_n and T during the secrets update procedure, it cannot deduce the new secrets and hence cannot violate R_n's privacy. The main operations employed in this protocol are simple bitwise operations, a *RNG*, and hash functions.

R_c starts the OT process in which it broadcasts the group ownership transfer request with independence of old owner (GOTRIOO) flag and an encryption of the key $(K_1)_{Group_T}$ (which is shared with all tags in $Group_T$) to all tags that are within its scope. It is assumed that $Group_T$ whose ownership will be transferred consist of m tags. Any T that receives the message, checks whether it is valid or not. If the verification succeeds, T authenticates R_c and then updates $(K_1)_{Group_T}$ to $(K_2)_{Group_T}$ and its unique key $(K_1)_T$ to $(K_2)_T$ and then sends the new keys encrypted to R_c. Since all m tags participate in the protocol, they nearly send their responses at the same time to R_c. Thus, when R_c receives a tag's response, it checks its validity by using the data in the *BS*. For each verified response, R_c confirms that the concerned T is preparing for OT and hence it updates its record in the *BS* (i.e. updates the group key and T's unique key). When R_c confirms that all tags in $Group_T$ have prepared for OT, it sends to the *TTP* the updated keys (i.e. $(K_2)_{Group_T}$ and the updated unique keys of all participating tags) in an encrypted form and the group ownership transfer allowance completion (GOTAC) flag.

After that, when R_n receives the tagged items, it sends F_{OT} to the *TTP*. In turn, the *TTP* replies with the group ownership transfer command from *TTP* (GOTCTTP) flag and an encryption of $(K_2)_{Group_T}$. R_n forwards the received message to all tags within its scope. When any T receives the message, it verifies its integrity. If the message is verified, T confirms that R_c and the *TTP* have approved the OT. Therefore, it once again updates $(K_2)_{Group_T}$ to $(K_3)_{Group_T}$ and $(K_2)_T$ to $(K_3)_T$ and then sends the newly updated keys to R_n which in turn forwards them to the *TTP*. The *TTP* then checks the validity of the received messages. After verifying that all tags have updated their keys, the *TTP* updates the group key $(K_2)_{Group_T}$ to $(K_3)_{Group_T}$ for all tags in $Group_T$ and the unique key for each T individually. The *TTP* sends the updated keys (i.e. $(K_3)_{Group_T}$ and the updated unique keys of all participating tags) to R_n which in turn employs them to communicate with the tags.

The main themes of the group non-EPC-compliant TTP-based OT protocols are presented in Table 5.

TABLE 5. Main themes of non-EPC-compliant TTP-based group OT protocols.

Protocol		Main Theme
Zuo	[66]	SKC and hashing, grouping-proofs, multi-tag single-owner
Kapoor hfill Bagheri	[70] [71]	SKC and hashing, multi-tag single-owner and single-tag multi-owner
He	[73]	Hashing, multi-tag single-owner

The non-EPC-compliant TTP-based group OT protocols were reviewed in this section. All protocols guarantee the old and new owner privacies. As for the windowing problem, only one protocol [70] is vulnerable to it. Furthermore, only one protocol [66] can offer AR. On the other hand, none of the discussed protocols offers OD or exclusive OT and none support mobile readers.

From the attacks perspective, only Kapoor *et al*.'s protocol is susceptible to attacks, whereas the other protocols are secure against them, to the best of our knowledge.

IV. TWO-PARTY OWNERSHIP TRANSFER PROTOCOLS

In this section, the Two-party protocols that were proposed throughout the literature are reviewed. We start by demonstrating the single OT protocols and then go to the group OT. For each category, we explain the EPC-compliant protocols first and then move over to the non-EPC-compliant protocols. For each protocol, we start by a brief description of its procedure and then we discuss its vulnerabilities.

A. TWO-PARTY SINGLE OWNERSHIP TRANSFER PROTOCOLS

Similar to the TTP-based single OT protocols, the Two-party single OT protocols are also classified into EPC-compliant and non-EPC-compliant. As a quick remainder, the EPC-compliant protocols are the protocols that meet the requirements specified in the EPC-standard which includes that the tags can only use simple bitwise operations, 16-bit *PRNG*s and 16-bit *CRC* functions. In addition, the tag stores two passwords which are the Kill and Access and the length of the tag's ID should be 96 bits. The non-EPC-compliant protocols, on the other hand, do not conform to at least one of the standard's requirements. We start this section with the EPC-compliant protocols and then go to the non-EPC-compliant ones. For each protocol, we start by describing its procedure briefly and then go over its weaknesses.

1) EPC-COMPLIANT TWO-PARTY SINGLE OWNERSHIP TRANSFER PROTOCOLS

After Saito *et al*'s. Two-party protocol (Section II-D), Ilic *et al.* [74] proposed a model for dual ownership transfer (DOT). The proposed model demonstrates the management process of the access to the information system (IS) data when the tagged-items change owner in supply chains. In DOT, if an owner proves its physical possession of a tagged item, then it is given the right to access the item's data in the IS. Thus, the main idea of DOT is to connect the physical possession of a tagged item with the IS access control. In supply chains, there are predefined relations and common agreements, DOT benefits from them in order to organize the access control to the IS.

There are two key concepts for DOT; namely P1 and P2. P1 states that if you can prove that you have physical possession of the tag, then you can control it. P2 states that if the tag's possession is transferred to another owner, then the access control is also transferred (this includes past, current and future transactions). In order for R_c to access its T's data in the IS, it needs to establish a DOT link to prove the physical possession of the tag. The DOT link connects the physical tagged-item with its data record in the IS. The proof that the owner needs to provide includes a proof for the tagged item's identity (i.e. ID_T), a proof for the owner's identity and a proof for the timeliness of the previous two proofs.

Once the link is created, the owner can access the item's data. The DOT link of R_c remains established until R_n creates a new DOT link with the item. In DOT, the most recent data records are only accessible by R_c (i.e. the owner that has the DOT link) and it cannot change the old data records. In addition, the old owners can keep the read access to the data that were valid during their ownership in case they still need to use them.

Llic *et al*. did not provide a protocol to implement their model. However, they kept the implementation as a future work.

After that, the RFID Delegation and Ownership Transfer protocol (RFIDDOT) was proposed by Dimitriou in [75]. RFIDDOT is based on using a *PRF* to encrypt the exchanged messages. Moreover, it uses simple bitwise operations and a *RNG*. After receiving the secret key $K_{R_c,T}$, R_n initiates the OT process by sending an update request along with a random nonce $(N_1)_{R_n}$ to T. In its turn, T generates N_T and then replies with N_T and $F_{K_{R_c,T}}(N_T \| (N_1)_{R_n})$. Next, R_n generates a random nonce $(N_2)_{R_n}$ and then transmits it along with N_T and $(N_1)_{R_n}$ in an encrypted form to T. Finally, both R_n and T updates the secret key $K_{R_c,T}$ to a new value (i.e. $K_{R_n,T}$).

Nevertheless, RFIDDOT does not fulfill the requirements of a secure OT protocol [23]. First of all, the random nonces $(N_1)_{R_n}$ and N_T are transmitted in cleartext, thus *Adv* can compute $F_{K_{R_c,T}}(N_T \| (N_1)_{R_n})$. After that it can obtain $(N_2)_{R_n}$ from the last message sent from R_n to T. Once *Adv* gets $(N_2)_{R_n}$, it can obtain the new key $K_{R_n,T}$. From this point, *Adv* can decrypt all subsequent transactions and furthermore clone T. Another issue with this protocol is its vulnerability to de-synchronization, which can be carried out by either blocking the last message sent from R_n to T or by modifying the obtained $(N_2)_{R_n}$ to a different value and then using it to compute the last message.

As for Kulseng *et al*., they proposed a Two-party OT protocol in addition to the TTP-based one that is proposed in [27].The Two-party protocol is based on using *PUF* and

LFSR and implemented by using the proposed mutual authentication protocol directly.

To start the OT, R_c gives R_n the information associated with the concerned T. Once R_n receives the information, it performs mutual authentication with T. Therefore, R_n sends a request to T which in turn responds with its IDS. After receiving T's IDS, R_n replies with ID_T and the shared secret key in an encrypted form. T then verifies the received message and if found valid, it verifies R_n. Hence, T updates its secret key and then transmits the updated key in an encrypted form to R_n. In turn, R_n validates the received message and then updates the secret keys. After that, both sides update the IDS value concurrently. As a final step, R_n performs the explained procedure once more in order to update the secret keys to new values that are unknown to R_c. Therefore, completing two sessions of the protocol, R_c cannot access T.

Still, the same de-synchronization and disclosure attacks previously discussed in Section III-A1 are applicable to the Two-party protocol.

Thus, in [33], Cui also proposed a Two-party protocol that enhances the work of Kulseng *et al*. Cui's protocol employs the same message flow as Kulseng *et al*.'s protocol. However, it modifies the structure of the exchanged messages in order to mitigate the attacks proposed against it.

Another approach was followed by Chen and Chien in their OT protocol for the membership-based store systems and mobile RFID environments [76]. The tags in this protocol employ simple bitwise operations, *RNG* and *CRC*. As for the other entities, they use the same operations used by the tags in addition to hashing, *SKC* and *PKI*.

This protocol consists of four stages. In stage 1, all product tags, users' mobile readers IDs, passwords, and fingerprints are registered with the *BS* via a secure channel. As for stage 2, the users employ their mobile readers to query the tags, the tags information can be obtained only by the readers that are authorized by the *BS*. In stage 3, the current user pays for the product through the cash register and then it obtains the private information from the *BS* in order to become the current owner R_c. Finally, in stage 4, R_c can transfer the ownership of T to R_n, when necessary, through the proposed OT protocol.

The proposed OT protocol is divided into five steps. R_c starts the protocol with the first step in which it sends T's privacy information to R_n via a secure channel. After that, in step 2, R_n generates a new key for T and then sends it in an encrypted form to the authorized agent (AA). In step 3, AA forwards the received messages (which hides the new key) to the *BS*. As for the fourth step, *BS* verifies the integrity of the received messages. If the verification succeeds, the *BS* updates T's secret key to the received new key and updates the sold list of R_n. To finish, R_n updates the secret key of T through a secure channel in the fifth step.

However, in [77] Munilla *et al*. exposed the vulnerabilities of Chen and Chien protocol and demonstrated several attacks against it.

To start, in the specification of Chen and Chien's protocol, T does not authenticate *BS* (or reader) and this makes the protocol susceptible to server/reader impersonation.

Secondly, an *Adv* can disclose all bits of the secret key which is shared between *BS* and every T and this means that all tags in the system are compromised.

Furthermore, T in Chen and Chien's protocol can be tracked by two methods. In the first one, *Adv* can track T after querying it for one time only. On the other hand, in the second method, he queries T for number of times enough to discover the secret system parameter and then employs the disclosed parameter to trace T.

In addition to tracking, an *Adv* can also impersonate T. This impersonation attack can be carried out by two methods; the first one allows *Adv* to disclose T's unique secret key and then use it to impersonate it. In the second method, *Adv* can impersonate T without recovering the secret key.

Munilla *et al*. stated that the main reason which helped in performing the attacks is the weakness in the tag authentication process of Chen and Chien's protocol.

We end this section with Doss *et al*. protocols which are based on the quadratic residue property [78]. Doss *et al*. proposed two protocols, the first protocol is for closed loop systems in which T is reachable by both R_c and R_n during the OT process, whereas the second is for open loop systems where T is reachable by R_n only during the OT process. In addition to the proposed OT protocols, the authors proposed an OT test protocol which is used to ensure that the OT process was completed successfully. The main operations used in the proposed protocols are simple bitwise operations, *CRC*, *RNG* and a hash function. The hash function is used by the *BS* only.

In the closed loop protocol, BS_{R_n} initiates the OT by sending F_{OT} to T which in turn replies with its information in an encrypted form. BS_{R_n} then generates two large prime numbers and computes their multiplication. After that, it transmits the multiplication result along with the information received from T to BS_{R_c}. When BS_{R_c} receives the information, it checks its integrity. If the verification succeeds, BS_{R_c} retrieves the associated T's information that is stored in the record (i.e. $ID_T, H(ID_T), K_{BS_{R_c},T}, N_{BS_{R_c}}$) and then replies with an acknowledgment. Upon receiving the acknowledgment, T checks its validity and if verified, it confirms that BS_{R_c} and BS_{R_n} share a secure channel and that BS_{R_c} knows the secret $K_{BS_{R_c},T}$.

Thus, T responds by sending its $K_{BS_{R_c},T}$ encrypted to BS_{R_n}. When BS_{R_n} receives T's response, it generates a new key $K_{BS_{R_n}}$ and transmits it in an encrypted form to T. In its turn, T verifies the received information and if found valid it extracts the new key $K_{BS_{R_n}}$ and replies by sending its $H(ID_T)$ and $N_{BS_{R_c}}$ encrypted to BS_{R_n}. BS_{R_n} then extracts $H(ID_T) \oplus N_{BS_{R_c}}$ from the received message and forwards it to BS_{R_c}. In its turn, BS_{R_c} verifies whether the received information is valid.

If the verification succeeds, it locates the concerned T's information and forwards ID_T to BS_{R_n}. Therefore, BS_{R_n} obtains the target T detailed information. After that, BS_{R_n}

updates $N_{BS_{R_c}}$ to a new value and transmits it encrypted to T. As a last step, T validates the integrity of the received information and if the verification succeeds, it confirms that BS_{R_n} has obtained ID_T and that $K_{BS_{R_n}}$ is correct. Hence, T updates its secrets to the new values. By that, the OT process is completed and BS_{R_c} can no longer access T.

As for the open loop protocol, it works using exactly the same procedure of the closed loop protocol. The only difference here is that BS_{R_c} sends the acknowledgment (in the fourth flow) to BS_{R_n} via secure channel. After that, BS_{R_n} encrypts the received acknowledgment and then forwards it to T. The purpose of encrypting the acknowledgment is to prove to T that there is a secure channel between BS_{R_c} and BS_{R_n}.

As mentioned previously, the authors proposed and ownership test protocol. This protocol is based on the quadratic residue property and is launched after the OT process has completed in order to allow an owner to verify whether it has ownership over the concerned T. This protocol is carried out in a virtual environment where there are no adversaries. The owner initiates the test protocol by sending an ownership test message to the target T which in turn replies by sending its information in an encrypted form. The owner validates the received information and if the verification succeeds, it confirms its ownership over T. Therefore, it replies with an encryption of $H(ID_T)$. As a final step, T verifies whether the received information is valid. If so, this confirms that it is owned by a valid owner.

Doss *et al*. stated that because of the quadratic residue property, completing the OT test protocol successfully provides a sufficient condition to prove the owner's exclusive ownership over the tag.

Nevertheless, Cao *et al.* [42] revealed the weaknesses of Doss *et al*.'s protocols. The authors demonstrated that Doss *et al*.'s closed loop protocol suffers from reader impersonation and de-synchronization attacks, whereas the open loop protocol is vulnerable to de-synchronization, tracking, and impersonation attacks (reader and tag). The authors stated that the main reason which helped in performing the attacks is that these protocols do not verify the integrity of each received message. This gives the adversaries the chance to manipulate the exchanged messages and replay them.

Moreover, in order to prevent tag tracking, reader impersonation and de-synchronization attacks, the authors suggested that the protocol should be modified by sending the multiplication result of p and q in an encrypted from to R_c instead of sending it in cleartext.

As a final note, the authors pointed out that Doss *et al*.'s open loop protocol does not guarantee old owner privacy because R_c transmits ID_T to R_n who in turn can employ it to trace T's historical information.

Table 6 shows the main themes of the EPC-compliant Two-party single OT protocols.

From the discussion in this section, we see that all protocols achieve resistance against the windowing problem. However, only two protocols can guarantee both the old and new owner privacies. As for OD or AR and supporting mobile readers, only one protocol [75] can provide just OD and two protocols can offer the latter. Moreover, none of the presented protocols provides exclusive OT.

TABLE 6. Main themes of EPC-compliant two-party single OT protocols.

Protocol		Main Theme
Dimitriou	[75]	PRF and RNG
Kulseng	[27]	PUF and $LFSR$
Cui	[33]	
Chen	[76]	PKI and hashing
Doss	[78]	Quadratic residue property

Based on the reviewed literature, only one protocol (Cui's protocol [33]) can resist the discussed attacks.

2) NON-EPC-COMPLIANT TWO-PARTY SINGLE OWNERSHIP TRANSFER PROTOCOLS

The first non-ECP-Compliant Two-party single OT protocol was proposed by Koralalage *et al*. in [79]. The proposed protocol is called the product-flow ownership-transferring protocol (POP) and can perform several functions such as querying the tags, disabling them and updating their secret keys.

POP informs T about the required function by using the update flag F_U. The main operations employed in POP are *RNG* and *SKC* which is implemented through Grain stream cipher [60]. R_n initiates the process by sending $IDE(R_n)$ and N_{R_n} encrypted to T which in turn verifies whether the received information is valid. If the verification succeeds, it generates a random nonce N_T and sends it in an encrypted form along $IDE(R_n)$ to R_n. When R_n receives the message, it employs it to authenticate T. Once T is authenticated, R_n can access it and perform the needed function. For OT purposes, R_n changes the shared keys $(K_1)_{R_n,T}$ and $(K_2)_{R_n,T}$ and then transmits them encrypted to T. As an acknowledgment, T replies by sending a confirmation message to indicate the success or failure of the process.

Yet, as demonstrated in [18], an *Adv* can trace T by continuously replaying the first message in the protocol to it. Since each tag response contains $IDE(R_n)$ in cleartext, *Adv* can use this value to trace T. Also, the authors stated that it is unknown what will happen in case *Adv* blocks the last message sent from R_n to T, or vice versa. Because this act leads to a situation where R_n has updated $(K_1)_{R_n,T}$ and $(K_2)_{R_n,T}$, whereas T has not.

Song [80] proposed a protocol that uses hashing instead of *SKC* to encrypt the messages. In fact, Song's protocol is based on Song and Mitchell's authentication protocol (denoted by SM) which is proposed in [81]. In addition to *SKC*, the protocol employs other operations such as simple bitwise operations and *RNG*.

This protocol begins with the OT sub-protocol and then goes to the update sub-protocol. The OT sub-protocol is initiated by BS_{R_n} which queries T with a random nonce $(N_{BS_{R_n}})$. T in turn replies by generating a random nonce

N_T and then sends it along with the secret key $K_{R_c,T}$ in an encrypted form (through messages $M_1 = K_{R_c,T} \oplus N_T$ and $M_2 = H_{K_{R_c,T}}(N_{BS_{R_n}} \oplus N_T)$) to BS_{R_n}. Next, BS_{R_n} forwards the received information along with $N_{BS_{R_n}}$ and F_{OT} to BS_{R_c}.

Once BS_{R_c} receives the message, it checks its validity and if the verification succeeds it updates the received N_T value and then re-encrypts the result using its secret key $K_{BS_{R_c}}$. After that, BS_{R_c} updates the old and current values of K_{R_c} and $K_{R_c,T}$ and then transfers them along with $Info(ID_T)$ and the re-encrypted N_T to BS_{R_n} via a secure channel. In turn, BS_{R_n} stores the received secrets and $Info(ID_T)$ and forwards the re-encrypted N_T to T. Finally, when T receives the message, it checks whether it is valid or not. If the received message is found to be valid, T authenticates BS_{R_n} and updates the secret key $K_{R_c,T}$.

After the aforementioned steps are accomplished, R_n becomes the new owner of T. However, it still needs to update the secrets to new values that are unknown to R_c in order to preserve its privacy. Thus, R_n performs the update-sub protocol with T in an environment that is isolated from R_c. BS_{R_n} starts this session by generating two random nonces $(N_1)_{BS_{R_n}}$ and $(N_2)_{BS_{R_n}}$ and a new key (i.e. K_{R_n}). After that, BS_{R_n} transmits K_{R_n}, and K_{R_c} encrypted to T along with $(N_1)_{BS_{R_n}}$. T in turn verifies the received information and then it updates $K_{R_c,T}$ to K_{R_n} and generates a random nonce N_T. Next, T encrypts $(N_1)_{BS_{R_n}}$ and N_T with the key K_{R_n} and sends the result along with N_T to BS_{R_n}.

As a last step, BS_{R_n} verifies the integrity of the received message and if the verification succeeds, it confirms that T has updated its secret. Thus, BS_{R_n} updates $K_{R_c,T}$ to K_{R_n} and K_{R_c} to a new value.

It should be noted that both BS_{R_c} and BS_{R_n} can keep a record of the old secrets (which were given to R_n by R_c) such that R_n can allow R_c to access T again when necessary.

Song's protocol is susceptible to de-synchronization attack as shown in [23]. The de-synchronization happens when *Adv* blocks the last message sent from R_n to T during the OT sub-protocol. In addition to de-synchronization, the authors noted that *Adv* can impersonate T by storing $N_{BS_{R_n}}$, M_1 and M_2 from a session in the OT sub-protocol. After that, when BS_{R_n} sends a new random nonce (for example $(N_2)_{BS_{R_n}}$), *Adv* replies by sending $M_1 \oplus N_{BS_{R_n}} \oplus (N_2)_{BS_{R_n}}$.

Moreover, in [82] Peris-Lopez *et al*. revealed more vulnerabilities of Song's protocol. The authors proved that the OT sub-protocol suffers from tag information leakage and tracking attacks and is susceptible to forward traceability. They also showed that the secret update sub-protocol does not resist de-synchronization attack.

Following after Song, another approach that is based on the SM protocol was proposed by Song and Mitchell in [83]. The proposed protocol employs simple bitwise operations, *RNG* and hash chains. In their work, Song and Mitchell focused on the scalability aspect. Therefore, the proposed protocol makes sure that the tag can be identified and authenticated within a constant time (i.e. no linear search) through the use of a pre-computed lookup table. In fact, the server needs $O(1)$ work to identify and authenticate the tag.

The working procedure of this protocol is divided into three cases; tag authentication, secret update (I) and secret update (II). The tag authentication case is initiated by BS_{R_c} which queries T with a random nonce. Each time T is queried, it checks whether its secret K_T is equal to the last element in the chain (which is stored in its memory). If both values do not match, T updates K_T to the next element in the chain, encrypts $(K_T)^{old}$ using $K_{BS_{R_c},T}$ and then sends the result along with the updated secret (i.e. $(K_T)^{new}$) to BS_{R_c}. BS_{R_c} in turn employs the received information and its lookup table in order to identify and authenticate T. Searching the lookup table requires a constant time (i.e. $O(1)$). The protocol works this way for the first $w-1$ queries of T (where w is the length of the hash chain).

After the number of queries reaches w, the protocol goes to the secret update (I) case. In this case, BS_{R_c} works on updating T's secrets and itself. Therefore, it updates T's secrets and then sends them in an encrypted form to T and then updates its own secrets. Upon receiving the message, T checks its integrity and if found valid, it updates its secrets accordingly. One additional message is required for this case. However, the time complexity for T's authentication is the same as that in the first case (i.e. $O(1)$).

As for the third case, it is performed when the number of queries exceeds w times which is an abnormal situation (i.e. either BS_{R_c} or T or both did not update their secrets successfully during the second case). In this case, T generates a random nonce and uses it to encrypt K_T and then sends the result to BS_{R_c}. In its turn, BS_{R_c} performs a linear search with complexity $O(n)$ in order to validate the received T's secret. If T's secret matches the first element in the chain, BS_{R_c} employs its old secrets to perform the update process. On the other hand, if T's secret matches the last element in the chain, BS_{R_c} carries out the update process using the newly generated secrets.

To perform the OT, BS_{R_c} transfers the secrets and any needed information to BS_{R_n} via a secure channel (after updating the secrets for T as mentioned previously). After that, BS_{R_n} updates T's secrets once more in order to protect its privacy from BS_{R_c} using the same protocol (this update should be performed outside the eavesdropping range of BS_{R_c}).

Song and Mitchell further revised their work as shown in [84]. In the revised version they only changed the structure of the message that is sent from BS_{R_c} to T so that it computes the hash of two values instead of three. Moreover, they proposed another protocol that can be used by BS_{R_n} to update T's state after completing the OT protocol. Compared with the work in [83], the newly proposed protocol requires fewer cryptographic functions' computation on T's side and fewer exchanged messages between BS_{R_n} and T. Therefore, the authors stated that BS_{R_n} should update T's secrets using the new protocol instead of the OT protocol in order to improve the performance. This enhancement can be

achieved assuming that BS_{R_n} knows the identity of the newly obtained T.

However, in [85] Habibi and Aref revealed the weaknesses of Song and Mitchell's revised OT protocol in which they launched de-synchronization, traceability and backward traceability attacks.

The authors stated that the main reason for the de-synchronization attack is a flaw in the OT update mechanism; the flaw is that BS updates the secrets and then forwards them encapsulated in a message to T via an insecure channel. This procedure gives Adv the chance to manipulate this message in the secret update (I) and secret update (II) cases, thus they can lead T to update its secrets to different values than BS. The manipulation of the message is based on representing the length of the employed hash function output which is equal to $l_1 \| l_2 \| l_3$ where l_1 is the first l-bits of the hash function's output, l_2 is the second l-bits of the hash function's output, and l_3 is the last l_m-bits of the hash function's output.

As for the traceability attack, it depends on the nature of the messages' flow in the protocol's cases. In the tag authentication case there are two flows; the first one is from BS to T, whereas the second one is the reply which is sent in the reverse direction. On the other hand, in the secret update (I) and secret update (II) cases there is an additional flow from BS to T in order to perform the update. Using this fact, the adversaries can trace T. Finally, the backward traceability attack can be carried out because the key's value at T's side is updated by using a hash chain of length w which also limits the number of authentication sessions. Therefore, compromising T during the authentication stage, allows Adv to trace all previously executed sessions within this stage.

As discussed previously, Song and Mitchell's protocol in [84] is a revised version of their protocol proposed in [83] and the only difference is that the revised protocol computes the hash of two values instead of three. Moreover, the de-synchronization attack that is proposed by Habibi and Aref [85] depends on the flow of the update mechanism and the representation of length of the hash function's output. Thus, we can state that the same de-synchronization attack applies to Song and Mitchell's protocol which proposed in [83] because it employs the same update mechanism's flow and computing the hash of two values does not change the length of the hash function's output.

Following the same principle, we can state that the protocol in [83] suffers from the same traceability and backward traceability attacks since this protocol and the revised version have the same nature of the messages' flow in the protocol's cases as well as having an identical authentication. Furthermore, any vulnerability reported against Song and Mitchell's revised protocol in Section VII is also applied to the protocol in [83].

To resist the proposed de-synchronization and traceability attacks, the authors modified Song and Mitchell's OT protocol by proposing a new update mechanism where both BS and T participate in the update procedure and by using three flows in all protocol stages.

The proposed update mechanism involves two update states. The first one is used when both BS and T are in a synchronized state. Thus, in this update state both the old and current keys are updated. As for the second state, it is employed when both parties are in de-synchronized state, hence only the current key is updated. This procedure is followed so that BS can synchronize T again. By using this two-state update mechanism, Habibi and Aref's protocol can resist the de-synchronization attack that is launched against Song and Mitchell's protocol. Also, because both cases employ three flows, Adv cannot trace T.

Furthermore, in [86] Wang and Yuan proposed the lightweight scalable de-synchronizing attack-resistant protocol (LSDARP) which is considered an improvement over Song and Mitchell's protocol. LSDARP employs a sliding window mechanism on hash chains in order to reduce the storage at the server's side and to enhance the scalability and performance. Each time the secret update phase in LSDARP is completed successfully, the window slides forward for one step only. Therefore in this protocol, BS needs to store only the two consecutive hash values (eigenvalues) for the tag's old and new hash chains in order to carry out the authentication process. On the other hand, Song and Mitchell's protocol stores all values in the hash chain which requires more storage space.

Because BS in LSDARP stores only the two consecutive old and new eigenvalues, the search complexity for a matched entry is $O(1)$. In addition, this protocol does not need to a keep a record of every possible used eigenvalue or a counter to determine its offset. This helps in avoiding the complexity degeneration problem that happens when all eigenvalues are used up. Moreover, T does not need a counter to perform the authentication phase.

LSDARP is initiated by BS which queries T with a random nonce. T in turn replies with its secrets in an encrypted form. After that, BS tries to authenticate the received messages by searching for a matched value in the new eigenvalues' set and if the authentication succeeds, it updates its secrets, moves the window one step, and then transmits the updated secrets in an encrypted form to T. Otherwise, it performs the search in the old eigenvalues's set and if such a value is found, it sends its current secrets (i.e. does not perform the update) encrypted to T. On the other hand, if no matched value is found in either of the two sets, the authentication results in failure. As for T, once it receives BS's response, it checks the integrity of the received messages and if found valid, it updates its secrets.

It should be noted that the update step at BS's side is always one step behind that at T's side in order to resynchronize T in case a de-synchronization has occurred.

After LSDARP is finished successfully, T's secrets are updated to new values that are unknown to R_c.

Chen *et al.* [87] showed that T in LSDARP can be tracked. Thus, they improve the protocol to LSDARP+ in order to resist the tracking attack. The proposed improvement in LSDARP+ involves the following: changing the messages' structure, allowing T to generate a random nonce and using

it along with the random nonce that is generated by *BS* to encrypt T's output and updating its keys and adding one of the messages that is sent by T to the validation message.

As for Ahamed *et al.* [48], they proposed another Two-party protocol that is based on hashing. In addition to hash functions, this protocol employs simple bitwise operations and *RNG*. The protocol follows the same procedure used in YA-SRAP. However, in this protocol R_c sends T's information to R_n instead of the *TTP* and then removes the information from its list. The authors pointed out that the Two-party approach is preferable since there is no need to communicate with the *TTP* each time there is an OT. Moreover, the authors noted that each reader has a threshold that limits the maximum amount of T's ownership information that it can hold. Thus, if a reader exceeds this limit it becomes overloaded.

In order to relive the overloaded reader, some of T's information can be transferred to a nearby co-operative (non-malicious) reader by using the Two-party OT approach. This procedure helps in ensuring the scalability of the proposed protocol. However, this OT approach has the same vulnerabilities of the YA-SRAP that are discussed in Section III-A2.

Another server-less OT protocol that employs simple bitwise operations, *RNG* and hashing was proposed by Xei *et al.* in [49]. The proposed protocol is named tag-owner-assisting RFID authentication protocol toward access control and ownership transfer (TOA). As the name implies, TOA is an OT protocol that is based on a novel tag-owner-assisting architecture in which the tag's owner has a main role in the authentication process. The main feature of the proposed protocol is that it provides OT, access-control, and mutual authentication without any need for a *BS*. Thus, TOA is a server-less approach that supports OT in offline environments.

Moreover, this protocol offers additional important characteristics such as simplicity, untraceability, scalability, access controllability and device-independence which is a novel contribution by the authors. Access controllability aims to control the access to the reader in order to ensure that no adversaries can access it, whereas device independence means that even if the mobile reader is missing, the authorized owner can still authenticate the tags and transfer their ownership through the use of any unified reader without online initialization.

To start the OT in TOA, R_c and R_n enter their passwords PW_{R_c} and PW_{R_n} into the trusted system reader R; respectively. As a result, R generates a random nonce N_R and transmits it to T which in turn replies by sending PW_{R_c} (that is stored in its memory) in an encrypted form by using its key K_T and the random nonce N_T. After that, R checks the integrity of the received information and if found valid, it sends PW_{R_n} encrypted to T. In turn, T tries to validate the received messages. If the verification results in a success, it confirms that PW_{R_n} is authentic and that R obtains the correct values of PW_{R_c} and K_T which means that it is trusted and authorized by R_c and R_n. Thus, T updates its own version of PW_{R_c} to the received PW_{R_n}. Finally, R_n repeats the previous steps -by using its private reader- to update the password in T to a new value (i.e. $(PW_{R_n})^{new}$). This update procedure is carried out in a secure environment that R_c cannot access or eavesdrop on. After the OT is completed successfully, R_c can no longer access T.

A hash-based OT approach was also proposed by Jin *et al.* in [47]. The protocol is called the Hash-Based Ownership Transfer Protocol for RFID Tags (HBOT). This protocol utilizes simple bitwise operations and a *RNG* in addition to hash functions. As for the OT, it is initiated by R_n which sends a query request to T. When T receives the request, it generates a random number N_T and then sends it encrypted using key $K_{R_c,T}$ to R_n. Next, R_n forwards the received information to BS_{R_n} which in turn retransmits it to BS_{R_c} through a secure channel. Upon receiving the messages, BS_{R_c} verifies their integrity by using the current secret values. If the validation succeeds, it authenticates T and then sends K_{R_c}, $K_{R_c,T}$, and $Info(ID_T)$ to BS_{R_n} -through a secure channel- so that it obtain the ownership of T. On the other hand, if the verification fails, BS_{R_c} performs one of two options which depend on either the security or the efficiency.

For better security, BS_{R_c} terminates the session with failure. Otherwise (i.e. for efficiency purposes), it tries to verify the received information using the old secret value. If the verification succeeds, it transmits $(K_{R_c})^{old}$, $(K_{R_c,T})^{old}$ and $Info(ID_T)$ to BS_{R_n}. Else, BS_{R_c} terminates the protocol session with failure. However, in all cases (i.e. either successful authentication or failure), BS_{R_c} updates the old and current secrets.

After receiving the messages, BS_{R_n} generates new keys $(K_1)_{BS_{R_n}}$ and $(K_2)_{BS_{R_n}}$. Then, it sends $(K_2)_{BS_{R_n}}$ in an encrypted form to R_n. After that, BS_{R_n} updates the old and current secrets. In turn, R_n forwards the received messages to T. As a last step, T checks the validity of the received messages and if found valid, it updates its key to $(K_2)_{BS_{R_n}}$.

Furthermore in [88], Fernandez-Mir *et al.* proposed a hash-based OT protocol that ensures scalability even with the increasing number of tags. The main idea in their protocol is to update the identification key immediately after T has been identified. This idea is employed in the OT procedure in order to update the owner's key. Moreover, the proposed protocol depends on the synchronization between *BS* and T and is designed in a way to prevent the adversaries from determining whether T is synchronized or not. Thus, even if *BS* and T become de-synchronized, the protocol is still scalable and resistant to de-synchronization attacks.

Assuming that BS_{R_c} and T are in a synchronized state, R_c starts the protocol by sending a random nonce N_{R_c} to T. T in turn replies with a random nonce N_T and an encryption for the identification key $(K_1)_{R_c,T}$. After that, T switches to the de-synchronized state (by setting the flag SYNC to 0) and remains in it until the key update process terminates successfully. R_c then forwards the received messages along with N_{R_c} to BS_{R_c}. Upon receiving the messages, BS_{R_c} checks

their validity and if verified, it sends ID_T to R_c and records N_{R_c} and N_T. Next, BS_{R_c} starts the key update process by updating $(K_1)_{R_c,T}$ to $((K_1)_{R_c,T})^{new}$ and then sends the updated key in an encrypted form to T. In turn, T verifies the received message and if found valid, it updates its key to the received $((K_1)_{R_c,T})^{new}$ and sets SYNC to 1.

Finally, R_c can transfer T's ownership to R_n through updating its and T's key $(K_2)_{R_c,T}$ to $((K_2)_{R_c,T})^{new}$ -to protect its privacy- and then forwarding $((K_2)_{R_c,T})^{new}$ to R_n. R_c employs the same updating procedure that is used to update the key $(K_1)_{R_c,T}$. R_n in turn updates $((K_2)_{R_c,T})^{new}$ to a new value in order to protect its privacy by using the same procedure. R_n should perform this key update in a secure environment in which R_c cannot eavesdrop on the exchanged messages.

Following the same approach, Dimitriou proposed another Two-party OT protocol in [89].The protocol employs a key-evolving mechanism to update the tag's key, this mechanism allows direct transfer of ownership between the owners.

Key-evolving means that R_c updates its key to a temporary key and then forwards the temporary key to R_n in order to protect its privacy. After that, R_n updates the received key to a new value. R_n can repeat the update procedure several times in order to ensure that R_c can never obtain the new key.

As a matter of fact, this protocol employs the same structure and messages of the RFIDDOT protocol which was proposed in [75] and discussed in Section IV-A1. However, here Dimitriou employs a hash function to encrypt the exchanged messages instead of a PRF.

As reported in [90], Dimitriou's protocol does not provide full mutual authentication which makes the protocol susceptible to reader impersonation, replay, and disclosure attacks. Moreover, it was shown that Dimitriou's protocol violates the old and new owner privacies.

Moreover, in [91], Munilla *et al*. proposed another OT that employs hashing and PRF. The main difference between Munilla *et al*.'s protocol and the previously discussed protocols is that the former is a Two-party protocol that does not need an isolated environment in order to complete the OT. In order to implement such a protocol, the authors proposed a communication model that can handle spatio-temporal connectivity (i.e. the current and new owners can be at different physical locations) and used Wyner's wiretap channel [92] with noisy tags in order to achieve privacy without an isolated environment.

Noisy tags are reader-controlled interferers and it is assumed that any tag can be used as a noisy tag. Combining wiretap channels with noisy tags produces channels with positive secrecy capacity upon which the protocol is based. Using these channels, R_n and T can exchange the new key securely even if R_c eavesdrops on the exchange process.

Also, the proposed OT protocol supports the following features: secure tag singulation, interrogator-talks-first (i.e. the reader initiates the communication) and tag assurance (confirming that T's information that is given to R_n matches its counterpart at T's side).

R_c starts the OT process by forwarding ID_T and $Info(ID_T)$ to R_n via a secure channel. After that, R_c broadcasts query messages in a continuous fashion to detect the surrounding tags. When a certain T receives the query, it responds by generating a random nonce $(N_1)_T$ and then sending it along with its encrypted key to R_c. Next, R_c tries to authenticate the received response and if found valid, it singulates T. Next, R_c encrypts the received $(N_1)_T$ and then transmits it along with N_{R_c}, $IDE(R_c)$, $IDE(R_n)$ and F_{OT} to T.

Once T receives the messages, it employs the received messages to authenticate R_c. If the authentication succeeds, T calculates the key $(K_1)_T = H_{(N_1)_T}(N_{R_c} \| K_{R_c,T})$, stores $[IDE(R_n), (K_1)_T]$ until the termination of the protocol or the reception of a new command from R_c, and then it sends the key $K_{R_c,T}$ in an encrypted form to R_c as an acknowledgment. If R_c receives T's reply, it computes the key $K_{R_c} = H_{(N_1)_T}(N_{R_c} \| K_{R_c,T})$ and then sends it along with the message (ID_T is ready) through a secure channel to R_n as a confirmation that T is ready for the transfer.

If R_n receives the confirmation form R_c, then it is ready to obtain T's ownership. Thus, R_2 calculates $K_{R_n} = H_{K_{R_c}}(Info(ID_T))$ and continuously broadcasts query messages. Once T receives the query message, it computes $(K_2)_T = H_{(K_1)_T}(Info(ID_T))$ and then generates $(N_2)_T$ and sends it along with the encrypted $(K_2)_T$ to R_n. After receiving the message, if R_n succeeds in singulating T, it replies by generating a random nonce $(N_1)_{R_n}$ and sending it along with $H_{K_{R_n}}((N_2)_T)$.

T, in turn, employs the received message to authenticate R_n. If R_n is verified, T updates the stored values $[IDE(R_c), K_{R_c,T}$ which specifies T's ownership rights to $[IDE(R_n), (K_2)_T]$ and then sends $(K_2)_T$ encrypted to R_n as an acknowledgment. Otherwise, it uses the message to authenticate R_c. If R_c is authenticated, T waits for a new command. Other than that, T does not respond.

When R_n receives T's acknowledgment, it performs the key update protocol to ensure that R_c can no longer access T.

The Key update protocol is performed to privately update T's key from $(K_2)_T$ to the new value $K_{R_n,T}$ which is generated by R_n. The entities involved in this protocol are R_n, T, and n_T noisy tags T_t^*, where $1 \leq t \leq n_T$. It is assumed that R_n and T_t^* share the private key K_{R_n,T_t^*}.

R_n initiates the key update protocol by generating $(N_2)_{R_n}$ and then broadcasts it along with a key change request (KCR). When T and T_t^* receive the messages, they generate $(N_3)_T$ and $\{K_{T_t^*}\}_{t=1}^{n_T}$ respectively and then broadcasts them simultaneously. Where $\{K_{T_t^*}\} = H^*_{(N_2)_{R_n}}(K_{R_n,T_t^*})$ and H^* is a cryptographic hash function that can be constructed from function H.

Upon receiving $(N_3)_T$ and $\{K_{T_t^*}\}_{t=1}^{n_T}$, R_n extract $(N_3)_T$, calculates $K_{R_n,T} = H_{(N_2)_{R_n}}((N_3)_T \| K_{R_n})$ and then broadcasts $H_{(N_3)_T}(K_{R_n,T})$. T then calculates its own version of as $K_{R_n,T} = H_{(N_2)_{R_n}}((N_3)_T \| (K_2)_T)$ and uses it to verify the integrity of received message. If the verification succeeds, it updates $(K_2)_T$ to $K_{R_n,T}$ and then it sends $H_{(N_2)_{R_n}}(K_{R_n,T})$ to R_n as an acknowledgment.

Once R_n receives the message, it checks its integrity. If the received message is valid, then the key update protocol is completed successfully and R_n informs R_c. Otherwise, R_n transmits a query to T to check whether it has updated its key. If this in not the case, the key update protocol is repeated.

Ng *et al.* [93] tried a different approach in which they proposed a protocol that is based on *PKI* and hashing in. In their protocol, the authors took into consideration the importance of fulfilling the privacy needs for all parties involved in the OT process (i.e. the company, the buyer (who may become R_n at some point), R_o and R_c). The proposed protocol has countermeasures that will be performed in order to protect the buyer in case the seller (R_c) was found to be a fraud and to protect R_o or R_c in case one of them has tricked the other. Therefore, the authors proposed four new security properties which are supported by their protocol:

- Tag assurance: a property that allows the buyer to confirm that T, whose ownership will be transferred, is the intended one.
- Current ownership proof: a property that allows R_c to prove its ownership of the tag to any third party.
- Undeniable ownership transfer: a property that allows R_c to prove to any third party that T was owned by R_o (i.e. R_o cannot deny owning T previously).
- Ownership initiation: a property to ensure that the OT and the key change can be initiated by R_c only.

In addition to *PKI* and hashing, the proposed protocol uses other operations such as simple bitwise operations, *RNG* and the Auth protocol which is used to authenticate T's response. This protocol outputs true, if and only if, T's response matches the value that is computed by the reader using the T's secret. Otherwise, it outputs false for an authentication failure.

Before carrying out the OT process, the authors assume that the target T should have been authenticated and thus R_c has obtained the necessary information about T (i.e. ID_T, $Info(ID_T)$ and $K_{R_c,T}$). Also, R_c's signature should have been stored in T's memory because this signature will be used to achieve the current ownership and undeniable ownership transfer properties. After that, R_c forwards ID_T and $Info(ID_T)$ to the buyer. Moreover, R_c and the buyer exchange their public keys such that each one of them can verify the public key through the *PKI* system.

The OT protocol is be initiated by R_c which starts by changing the key $K_{R_c,T}$ to a new value in order to protect its privacy and leads T to do the same. After that, R_c generates a new key $K_{R_c,buyer} = H(N_{R_c} \oplus K_{R_c,T})$ and forwards it in an encrypted form to the buyer. Moreover, R_c makes T update its $K_{R_c,T}$ again and changes its maximum number of accesses (Cnt_{max}) to 1 so that the buyer can use the key $K_{R_c,buyer}$ to authenticate T for one time only (after that the key expires).

In turn, the buyer runs the Auth protocol with the key $K_{R_c,buyer}$ in order to confirm that R_c knows the current T's key (i.e. confirm the legitimacy of the tag's current owner) and that T will give a valid response to the current key (i.e. confirms the legitimacy of the tag). Therefore, the buyer sends a query to T and waits for the response. When the buyer receives the response, it uses Auth($K_{R_c,buyer}$) to authenticate T. Next, T executes $Auth(K_{R_c,T})$ to authenticate the buyer and then checks whether $Cnt < Cnt_{max}$ so that Cnt is incremented by 1. Otherwise, it resets $K_{R_c,T}$ to the old value.

After that, R_c encrypts its signature and key K_{R_c} and sends the result along with the Transfer Start (TS) command to T in order to initiate the transfer process. Only R_c can starts the OT process because of the proposed owner initiation property. Upon receiving the information, T verifies its legitimacy and if verified, it responds by broadcasting its secret $(K_1)_T$ to both R_c and the buyer. R_c in turn computes $H((K_1)_T)$ and broadcasts the result along with the Tag Assurance (TA) command to the buyer. When the buyer receives the information, he checks whether it is valid. If verification results in a success, he confirms that the seller is indeed the owner and that T is the targeted one. Otherwise, the buyer may assume that T is not the correct one. Either way he may choose to quit the OT process.

However, if the verification succeeded and the buyer decided to continue the OT to become the new owner (R_n), he computes his signature for $H((K_1)_T)$ and broadcasts it along with the Buyer Signature Verification (VR) command. The tag records the signature and R_c works on validating its integrity. If the verification does not succeed, the protocol is terminated. The authors stated that in real life, the buyer should not give R_c the payment during this step of the protocol. This remark is reasonable because R_c may forfeit and choose not complete the protocol. In such a case, the buyer will not able to access T.

If R_c does forfeit, it generates a new key $(K_{R_c,buyer})^{new}$ and then broadcast it along with its signature and the Transfer End (TE) command to T and the buyer. In turn, T validates the received information and if verified, it updates $(K_1)_T$ to $H((K_1)_T)$ and then updates $(K_2)_T$ using the received $(K_{R_c,buyer})^{new}$ and the signature of the buyer. As for the buyer, he needs to verify the received R_c's signature (this achieves the undeniable ownership transfer property). If the signature is authentic, the buyer now becomes R_n and knows the current T's key and the signature of R_c. Thus, he proceeds with the payment.

Finally, to complete the OT, R_n should change the current T's key again to protect its privacy. The authors stated that R_n should perform this update in a secure environment where R_c cannot eavesdrop on the exchanged messages.

Elkhiyaoui *et al.* [94] proposed an OT protocol that employs *PKI* and hashing. The protocol is called the RFID Ownership Transfer with Issuer Verification (ROTIV). As the name implies, ROTIV supports OT and allows the owners to verify that tags were issued by a trusted party (i.e. issuer). The main idea for achieving issuer verification in this protocol is based on storing the issuer's signature in the tag in an encrypted form using El-gamal ciphertext. Moreover, ROTIV provides mutual authentication between the different parties. In order to perform constant authentication and

issuer verification, this protocol combines an HMAC-based authentication with *PKI* and employs *RNG*, key update, and tag re-encryption techniques. In addition to these techniques, ROTIV participates in subgroups of elliptic curve that supports bilinear pairings.

The OT process in ROTIV is initiated by R_n which queries T with the random nonce N_{R_n}. Once $(N_1)_{R_n}$ is received, T replies by sending its El-gamal ciphertext C_T, a random nonce N_T, and an HMAC encryption $H_{K_{R_c,T}}((N_1)_{R_n} \| N_T \| C_T)$. When R_n receives the messages, it generates another random nonce $(N_2)_{R_n}$ and then sends it in an encrypted form along with the received messages to R_c. Next, R_c authenticates T through the proposed hash-based mutual authentication protocol. If the authentication succeeds, both parties update their state. After that, R_c sends the ownership references (which contain the old and new values of the key $K_{R_c,T}$) and the verification references (which contain T's identifier) to R_n.

In turn, R_n employs the received ownership references to verify the integrity of the received $H_{K_{R_c,T}}((N_1)_{R_n} \| N_T \| C_T)$. If the verification succeeds, R_n confirms that the received $K_{R_c,T}$ is associated with T. After that, R_n uses the received verification references to authenticate the issuer using the proposed issuer verification protocol. The issuer verification protocol aims to validate the integrity of the issuer signature and the El-gamal ciphertext. If the issuer is valid, R_n adds an entry for T in its database BS_{R_n} and by that the authentication of T is completed. Next, R_n generates a new random nonce and uses it to calculate a new El-gamal ciphertext C_{R_n}. After that, R_n transmits C_{R_n} and $H_{K_{R_c,T}}(N_T \| C_{R_n})$ to T and then updates the record in BS_{R_n} which involves updating the key $K_{R_c,T}$ to a new value. As a last step, when T receives the message, it checks whether R_n is authentic. If the verification succeeds, T updates its state and key and by that the OT is completed.

It should be noted that the authors of ROTIV state that their protocol provides exclusive OT property. Exclusive OT in their definition indicates that an *Adv* who does not hold the ownership references of T cannot transfer T's ownership unless he rewrites its content. However; since their definition differs from the general definition of exclusive OT which is employed in this survey and previously clarified in Section II-A, we consider that ROTIV does not provide exclusive OT and reflect that in Table 10 when comparing the properties of the protocols in Section VII.

Nevertheless, in [26], Abyaneh exposed the vulnerabilities of ROTIV. The author showed that T can be traced passively by R_o or actively by *Adv* who can tamper it. Moreover, it should be noted that because the state updating in ROTIV does not depend on the previous values, the owner can trace the tag in the past or the future which violates the old and new owner privacy.

Another Two-party mutual authentication and OT protocol which utilizes *PKI* and hashing was also proposed by Yang *et al*. in [95]. In addition to the protocol, the authors proposed a privacy model for strong adversaries. In their adversarial model, the adversaries can learn all secret information in the reader, authenticate and corrupt tags, observe the full OT procedure, and eventually transfer the tag's ownership to others. *Adv*'s main aim in the proposed model is to be able to differentiate the target tag from other tags. The authors showed that their OT protocol is secure under the proposed model and does not require the extra process in which R_n communicates with T to perform the key update in an isolated environment where there is no eavesdropping.

Before starting the OT protocol, the owners should authenticate each other and prepare a secure channel so that R_c can send $(K_{R_c,T})^{old}$, $(K_{R_c,T})^{new}$, and other tag-related information to R_n.

After that R_n initiates the protocol by transmitting its public key PK_{R_n}, a random nonce N_{R_n} and the access command change to T. Then, T generates a random nonce N_T, encrypts N_T along N_{R_n} and $K_{R_c,T}$ by using PK_{R_n} and then encrypts the result with PK_{R_c} and sends it to R_c. When R_c receives the message, it decrypts it using SK_{R_c} and then forwards the result to R_n.

Once the message is received by R_n, it decrypts it to extract $K_{R_c,T}$, N_{R_n} and N_T. After that, R_n checks whether the received N_{R_n} is valid and if the verification succeeds, it checks whether $K_{R_c,T}$ matches with $(K_{R_c,T})^{new}$ or $(K_{R_c,T})^{old}$. If $K_{R_c,T}$ matches $(K_{R_c,T})^{new}$, R_n updates both $(K_{R_c,T})^{old}$ and $(K_{R_c,T})^{new}$. Otherwise, only $(K_{R_c,T})^{new}$ is updated. After that, R_n calculates $H(N_T)$ and transmits the result to T.

Finally, T checks the integrity of the received message and if found authentic, it changes PK_{R_c} to PK_{R_n} and updates $K_{R_c,T}$ to $K_T = H(N_T \| K_{R_c,T})$ to complete the OT procedure.

We wrap this section up with the Secure Ownership Transfer Protocol (SOTP) which was proposed by Yang and Hu [96]. SOTP utilizes *LE*, *SKC*, simple bitwise operations, and *RNG*. This protocol is designed for mobile RFID systems and provides the ability to assign the transfer target and to transfer the ownership across different authorities. In addition to OT, the protocol performs mutual authentication between the different parties. The mutual authentication is implemented through secure authentication protocols such as Kerberos and *PKI*. In fact, SOTP follows a structure similar to that of LOTP which was proposed by Yang in [8]. However, LOTP is TTP-based whereas SOTP is not.

In addition to performing the OT across different authorities, SOTP can also carry out the OT under the same authority. However, the main focus of this protocol is the OT across different authorities, thus the protocol is demonstrated assuming that R_c who is under the authority of BS_{R_c} will transfer the ownership of T to R_n who is under the authority of BS_{R_n}.

R_c initiates the process by sending F_{OT} to T which replies with its information in an encrypted form. After that, R_c re-encrypts the received information and forwards the result to BS_{R_c}. Upon receiving the message, BS_{R_c} verifies whether T is authentic. If the tag is authenticated, the next step of the protocol is performed. Otherwise, the protocol is aborted.

After T has been authenticated, BS_{R_c} starts the second step (assigning the transfer target) in which it sends $IDE(R_n)$ encrypted to BS_{R_n}. In turn, BS_{R_n} decrypts the received

message and checks whether R_n is under its authority. If the verification succeeds, BS_{R_n} sends a message that confirms the successful verification to BS_{R_c}. Otherwise, it transmits a message that indicates the verification failure. If BS_{R_c} receives a failure message, it aborts the protocol. Other than that, BS_{R_c} encrypts $IDE(R_n)$, $IDE(T)$ and $K_{BS_{R_c},T}$ and transmits the result to BS_{R_n} to indicate that R_n is assigned as the new owner for T. Next, BS_{R_n} changes T's identity to a new value (i.e. $(IDE(T))^{new}$) and registers R_n as the new owner.

The protocol goes into the last step (generating new shared keys for R_n). R_n transmits an F_{OT} to T which in turn responds by transmitting its information encrypted to R_n. Next R_n encrypts the received message and sends the result to BS_{R_n}. Upon receiving the message, BS_{R_n} decrypts it to extract $(IDE(T))^{new}$ and then employs the identifier to identify and authenticate T and to confirm that R_n is the registered owner for T. If the validation succeeds, BS_{R_n} generates two new keys and then sends them in an encrypted form to T through R_n. Next, T checks the integrity of the received message and, if found valid, it updates its keys to the received values. After T updates its secrets, the OT process is completed.

TABLE 7. Main themes for non-EPC-compliant two-party single OT protocols.

Protocol		Main Theme
Saito	[9]	SKC
Koralalage	[79]	
Song	[80]	Hashing
Song	[83]	
Song	[84]	
Habibi	[85]	
Jin	[47]	
Fernandez	[88]	
Xie	[49]	
Dimitriou	[89]	
Ahamed	[48]	Hashing and Contact list to identify the tags
Wang	[86]	Hashing and Sliding window
Chen	[87]	
Munilla	[91]	Hashing, wiretap channels and noisy tags
Ng	[93]	PKI and hashing
Elkhiyaoui	[94]	
Yang	[95]	
Yang	[96]	LE and SKC

Table 7 lists the main themes for the non-EPC-compliant Two-party single OT protocols.

We see that approximately half of the protocols do not achieve at least one of the following properties: old owner privacy, new owner privacy, and resistance of the windowing problem. In addition, there is one protocol that offers OD, one that offers AR, and five that offer both. These five protocols are the only ones that provides both OD and AR in the reviewed literature for all different categories of OT protocols. As for supporting mobile readers and exclusive OT, only three protocols fulfill the former and one achieve the latter (which is the protocol proposed by Ng *et al.* [93]).

Finally, regarding the resistance of the security attacks, it can be seen that the majority of the presented protocols cannot resist at least one of them regardless of the used cryptographic functions which involve hashing, *SKC* and *PKI*.

B. TWO-PARTY GROUP OWNERSHIP TRANSFER PROTOCOLS

We start this section with Kapoor *et al.*'s ownership sharing protocol [70]. The proposed protocol is based on the authors' opinion which states that a protocol without a *TTP* is not considered an OT protocol; rather it is an ownership sharing protocol because the old owners can still access the tags even after transferring the ownership to the new owners. On the other hand, the protocol ensures that the new owners do not know the previous key in order to protect the privacy of the old owners.

Kapoor *et al.*'s protocol employs simple bitwise operations, *RNG*, hash function, and *SKC*. It handles the transfer of a single T from $Group_{R_c}$ to $Group_{R_n}$ (i.e. single-tag multi-owner). In addition, for each group of owners, it is assumed that one of the owners takes the responsibility of being a coordinator. The coordinator is the owner who initiates the communication and propagates the changes in keys or access rights to other owners. Thus, in the initialization phase of this protocol, the communication occurs between the coordinator of the current owners CR_c and the coordinator of the new owners CR_n and in the key change phase it occurs between CR_n and T. Later on, CR_c and CR_n pass the changes to their associated group members.

CR_c starts the initialization phase of the protocol when it receives the OT request F_{OT}. Thus, CR_c transmits $E_{K_{CR_c,T}}(N_{CR_c} \oplus K_{CR_c,T})$ to T and, at the same time, sends $N_{CR_c} \oplus K_{CR_c,T}$ to CR_n over a secure channel. Once T receives the message, it generates a fresh $(N_1)_T$, uses it to encrypt $K_{CR_c,T}$ and then sends the encrypted key to CR_n. After performing this step, T and CR_n knows the value M_1 which is equal to $N_{CR_c} \oplus (N_1)_T$. On the other hand, CR_n is not permitted to know $K_{CR_c,T}$ in order to maintain the privacy of $Group_{R_c}$.

After that, T starts the key change phase by generating a fresh $(N_2)_T$ and then it flips one bit in M_1 randomly to get M_2. After that, it transmits $(N_2)_T$ along with $E_{M_2 \oplus (N_2)_T}(M_2 \oplus (N_2)_T)$ and $H_{M_2 \oplus (N_2)_T}(M_2 \oplus (N_2)_T)$ to CR_n. Since CR_n knows M_1, it obtains M_2 from it by using a brute force technique. After that, CR_n employs M_2 to decrypt the received $E_{M_2 \oplus (N_2)_T}(M_2 \oplus (N_2)_T)$. Next, it uses the received hash value to validate the decryption result. Upon successful verification, CR_n generates a new key K_{CR_n} and transmits it to T in an encrypted form. T in turn acknowledges the reception of the new key by sending $H_{K_{CR_n}}(M_2 \oplus K_{CR_n})$. Finally, CR_n replies with $E_{K_{CR_n}}(M_2 \oplus K_{CR_n})$ as an acknowledgment.

Also, it is worth to be mentioned that Kapoor and Piramuthu [18] proposed a single OT version of this protocol in which the ownership of T is transferred from a single R_c to a single R_n.

The second protocol is the secure multiple group ownership transfer protocol for mobile RFID (SMGOTP) which was proposed by Yang in [97]. SMGOTP transfers the ownership of a group of tags across different authorities in a mobile RFID environment (i.e. handles the multi-tag single-owner OT). The main operations used in the protocol are

simple bitwise operations, *RNG*, hash function, *LE*, and *SKC*. In fact, SMGOTP follows a structure similar to LOTP which was previously discussed in Section III-A2. The proposed protocol is divided into three steps. In step 1, $Group_T$ is authenticated by BS_{R_c}. After that, BS_{R_c} transmits the $Group_T$'s key to BS_{R_n}. In step 3, BS_{R_n} updates $Group_T$'s key and another two secret keys.

The main advantage of the protocol is that it can perform group OT and support mobile environments. In addition, it guarantees the old and new owner privacies and the location privacy and can resists several attacks such as MitM, de-synchronization, and replay. On the other hand, SMGOTP is based on many unrealistic assumptions such as ''tags must be responsive to all the transmission all the time during transfer''. Moreover, there is a step in which R_c needs to delete the keys. However, there is no guarantee that this would be done and an old owner with malicious intents may cause problems.

As for the third protocol, it was proposed by Li *et al.* [98] and handles the single-tag multi-owner transfer. The proposed protocol employs simple bitwise operations, *RNG*, hash functions, *PKC*, and *BLS* signature [99]. The size of the secret keys held by the tag is fixed regardless of the number of owners.

In the proposed protocol, it is assumed that each R_c has a partial ownership of T. Thus, in order to fully transfer T's ownership to $Group_{R_n}$, all owners in $Group_{R_c}$ need to approve the transfer process. Moreover, the owners in $Group_{R_c}$ share a long-term authentication key which allows each owner to generate temporary keys, prove its partial ownership, and authenticate T. As for $Group_{R_n}$, each one challenges all owners in $Group_{R_c}$ in order to obtain T's ownership proof and request the full OT. Once T's ownership is transferred successfully, R_n can update the long-term authentication key.

The protocol is demonstrated in case $Group_{R_c}$ wishes to transfer the ownership of a single T to single R_n. R_{c_i} starts the OT process by leading T to use a temporary key which will be used for a single authentication session. After that, R_n challenges R_{c_i} with a random number, R_{c_i} in turn replies with its signature (which resembles the partial ownership proof). In order to generate a full ownership proof of T, each individual R_c needs to generate its partial ownership proof. R_n checks whether the received signature is valid or not in order to verify the proof of the partial ownership of T. If the signature is verified, it tries to authenticate T before accepting the partial ownership proof.

If R_n needs to verify the full ownership proof, it needs to validate the partial ownership proofs of all owners in $Group_{R_c}$. If all proofs are verified, R_n accepts the full ownership proof. In order to transfer the full ownership of T to R_n, all owners in $Group_{R_c}$ interact in turn with T to update its long-term authentication key. It is assumed that each R_c interacts with T securely (i.e.; the other owners cannot eavesdrop on the communication between the two parties). Afterwards, each R_c sends the new T's keys securely to R_n.

Note that each R_c keeps the old and new keys in order to allow an owner to communicate with T in case the other owners refused the update of the key and to resist de-synchronization attacks.

Also, since the record for the ownership (which is sent from each R_c to R_n) does not contain the private key of the owner, it can use the public/private pair with all its tags.

Finally, if any owner refuses the ownership request, it will be canceled. Also, it is not required for all owners to transfer the ownership simultaneously and they can perform the transfer sequentially.

TABLE 8. Main themes of two-party group OT protocols.

Protocol		Main Theme
Kapoor	[70]	SKC and hashing, single-tag multi-owner
Yang	[97]	LE, SKC and hashing, multi-tag single-owner
Li	[98]	LE, SKC, hashing and BLS signature, single-tag multi-owner

Table 8 provides the main themes of the Two-party group OT protocols.

In Section IV-B, we discussed the Two-party group OT protocols. As mentioned previously, these protocols are non-EPC-compliant. In the reviewed literature, we did not find any EPC-compliant Two-party group OT protocols. This emphasizes the need to design protocols that fall under this category.

Three Two-party group OT protocols were presented and it can be seen that all three protocols protect the privacy of the old owner. Yang's protocol [97] does not achieve the new owner privacy and the resistance against the windowing problem. From another point of view, Yang's protocol is the only one that can support mobile readers. In addition, all three protocols do not offer OD, AR, or exclusive OT.

To the best of our knowledge, all three protocols are secure against the attacks.

V. UNIVERSAL OWNERSHIP TRANSFER PROTOCOLS

Some of the proposed OT protocols are designed to cover all possible transfer situations (i.e. single-tag to single-owner, single-tag to multi-owner, multi-tag to single-owner, multi-tag to multi-owner). These protocols are called universal as they can be used for any type of transfer.

Throughout the literature, only two protocols (proposed in [100] and [101]) can be considered as universal. Both protocols are based on the Two-party model. The work in [101] is an EPC-compliant protocol while [100] is not.

The work of Ray *et al.* [100] is considered an example on the universal OT protocols as it covers all OT scenarios. The OT scenarios in Ray *et al.*'s protocol cover the different transfer situations that result in the business cases. The main operations used in this protocol are simple bitwise operations, *PRF*, hash function, and *PKC*. As for the validation of the ownership request and the authentication of the participating entities, they are performed by the modified Diffie-Hellman algorithm [102].

The proposed protocol supports mobile RFID systems and consists of three stages. In stage 1, the OT is initiated through the first five steps (step 1 to step 5) of the protocol. Stage 2 consists of two steps (step 6 and step 7) in which the mutual authentication and validation is carried out. The OT is finalized and closed in Stage 3 which is performed through steps 8 to 14 of the protocol.

A last note on this protocol is that only the public keys are transmitted between entities. Therefore, even if the adversaries intercept the remote communications between entities, they cannot breach the old and new owner privacies and cannot violate the location privacy. In addition, the proposed protocol can resist the replay, de-synchronization, and MitM attacks.

Another universal OT protocol for the Internet of Things (IoT) systems was also proposed by Ray *et al.* in [101]. The protocol handles the OT in a mobile Networked RFID System (NRS). In addition to the proposed OT protocol, the authors proposed a test protocol to confirm that T's ownership was transferred successfully and that T is exclusively owned by R_n. The proposed protocols are based on number theory, the transitivity property, and the multiplicative inverse of modular arithmetic. As for the operations, it employs simple bitwise operations, arithmetic operations, CRC, PRF, and Hash Function. The tags in this protocol are assumed to be able to perform basic arithmetic operations and 32-bit CRC.

The object identification model in the IoT is represented by a global pool of objects controlled by several administrative business entities known as NRS. There are four main components in an NRS: the BS, the current and new owners, the coordinator owners, and the tags.

The owners and coordinator owner can access the owned tags in $Group_T$ by using the group's private key which is known to them. Furthermore, the communication between $Group_{R_c}$ and $Group_{R_n}$ is performed through the coordinator owners which represents each group. The communication between two coordinator owners is performed via a secure channel using the secure wireless communication protocol proposed in [103]. In addition to providing a secure channel, the employed wireless communication protocol allows the parties to authenticate each other. Nevertheless, this does not guarantee that R_n has the right to own a new set of tags.

In the proposed protocol, the ownership of $Group_T$ is to be transferred from one NRS represented by $Group_{R_c}$ to another one resembled by $Group_{R_n}$. The owners ($Group_{R_c}$ and $Group_{R_n}$) belong to different administrative domains and before starting the OT protocol, they exchange a token code (TC) value through a secure channel. The purpose of exchanging TC is to formalize the ownership agreement to this OT between the owners (i.e. ensuring that the parties of the OT have agreed on the required monetary, legal and contractual procedures).

Ray *et al.*'s OT protocol can be carried out through two stages; the ownership rights and transferring request validation and the OT and ownership claim process. After the OT is completed, R_n initiates the test protocol to confirm that the ownership was transferred successfully.

BS_{R_n} starts the first stage of the protocol by transmitting $SK_{Group_{R_n}}$, the list of readers that have $SK_{Group_{R_n}}$ ($R_{Group_{R_n}}$), an encryption of TC, and $N_{BS_{R_n}}$ to CR_{R_n}. Once CR_{R_n} receives the information, it requests an OT by transmitting the encrypted TC to CR_{R_c}. In turn, CR_{R_c} polls to all owners in $Group_{R_c}$ in order to collect their status and confirms that they are active and authentic. Next, it sends $PK_{Group_{R_c}}$ and $(K_1)_{CR_{R_c}}$ to CR_{R_n} in order to ask for the verification information that can be employed in the validation of the OT request and the ownership right process. It is worth to be mentioned that $PK_{Group_{R_c}}$ is computed using the modified group Diffie-Hellman equation [102].

After that, CR_{R_n} utilizes the received information along with its secrets to compute $PK_{Group_{R_n}}$. Moreover, it computes the verification key $(K_2)_{CR_{R_c}}$ by using the Diffie-Hellman key generation equation [102]. Finally, CR_{R_n} encrypts $(K_2)_{CR_{R_c}}$ by using $N_{BS_{R_n}}$ and then sends the result along with $PK_{Group_{R_n}}$ to CR_{R_c} in order to verify its OT request and claim of ownership right. When CR_{R_c} receives the verification information, it verifies whether the received $(K_2)_{CR_{R_c}}$ and $N_{BS_{R_n}}$ are valid. If $(K_2)_{CR_{R_c}}$ is valid, CR_{R_c} verifies the OT request and if $N_{BS_{R_n}}$ is verified, it validates the ownership right over the tagged objects.

After verifying the request and ownership right, CR_{R_c} replies by transmitting the stored tags' tokens and associated information (excluding ID_T), the total number of tags that will be transferred, the maximum time delay for a successful protocol execution, $CRC(SK_{Group_{R_c}})$ and $CRC(ID_T)$. Once CR_{R_n} receives the information, it employs the received tags' tokens (TT) to compute a TC value and then compares the result with the stored TC. If both values match, CR_{R_n} verifies that it has received the correct tag set. On the other hand, CR_{R_n} is still unable to read T because it does not have its ID_T. Therefore, CR_{R_n} generates a new T's ID (i.e. $(ID_T)^{new}$) and for each T that will be owned it computes new TT (i.e. $(TT_{CR_{R_n}})^{new}$). Finally, CR_{R_n} computes a new TC (i.e. $(TC_{CR_{R_n}})^{new}$) by using the generated $(TT_{CR_{R_n}})^{new}$ and then transmits it along with the unique ID of the new owners' NRS to CR_{R_c}. By that the first stage is completed.

When CR_{R_c} receives $(TC_{CR_{R_n}})^{new}$ and the ID of NRS, it starts the second stage of the protocol by performing the mutual authentication protocols demonstrated in [104] and [105] with all tags in $Group_T$ to ensure that there are no fraud owners or tags. After that, it calculates new TTs (i.e. $(TT_{CR_{R_c}})^{new}$). Moreover, CR_{R_c} employs the received NRS's ID to retrieve the limit of the new owners. Next, it starts the OT by transmitting the retrieved limit, the session time (t_s) and $(TT_{CR_{R_c}})^{new}$ encrypted along with the state value (which is set to 1 to indicate OT) to the tags that are involved in the OT.

Upon receiving the information, each involved T verifies its legitimacy and if found valid, it authenticates the OT request, changes its state to '1', and determines the correct limit value for R_n. After that, T sends its ID_T in an encrypted

form along with its state value to CR_{R_n} in order to inform it about the change in OT status.

Once CR_{R_n} receives T's response, it tries to validate it. If the received response is verified, CR_{R_n} confirms that it originated from one of its transferring tags. Next, it transmits $(TT_{CR_{R_n}})^{new}$, TC and another two secrets in an encrypted form to T in order to claim the ownership. It should be mentioned that $(TT_{CR_{R_n}})^{new}$ and TC are employed to verify the genuineness of the ownership claim, whereas the other two secrets are used to perform a test suite to claim exclusive ownership. T in turn verifies the received values and if found authentic, it confirms that the ownership claim of CR_{R_n} is valid and then initiates the test protocol to ensure the exclusive ownership to it.

After the test protocol is completed successfully, CR_{R_n} sends an acknowledgment to T. At last, when T receives the acknowledgment, it checks its integrity and if verified, T updates its secrets to $(ID_T)^{new}$ and $CRC(SK_{Group_{R_n}})$, changes its state to '0', and finally deletes all temporary values from its memory. By this, the OT protocol is finished.

TABLE 9. Main themes for universal OT protocols.

Protocol	Main Theme
Ray [100]	PKC and hashing
Ray [101]	Hashing, arithmetic Operations and number theory

The main themes of the universal OT protocols is displayed in the Table 9.

Only two universal OT protocols were reviewed and discussed in this section. Both protocols guarantee the old and new owner privacies and can resist the windowing problem. Regarding the exclusive OT and supporting mobile readers, the protocol in [101] offers both, whereas the protocol in [100] provides only the latter. Neither of the protocols provides OD or AR.

On a final note, based on the reviewed literature, both protocols can resist the security attacks which are summarized in Section II-A.

VI. OWNERSHIP DELEGATION

As mentioned in Section II-A, OD is a special case of OT in which the ownership of the tag is transferred temporarily from the current owner to another owner (called delegate). The delegate can access the tag for a predetermined number of times or until the current owner terminates the delegation. Moreover, as mentioned in Section II-A, if the delegate that needs to access the tag was an old owner of it, the process of temporarily transferring the tag's ownership is called authorization AR.

In order to clarify the idea of OD and AR we demonstrate in this section the OD and AR protocol which was proposed by Dimitriou in [89].

In Dimitriou's protocol, T stores a DELEGATED bit which is set to 1 in OD and AR mode and 0 otherwise. Moreover, T stores a counter (Cnt) in order to limit the number of allowable delegations (i.e. controlled delegation). In addition to OD, Dimitriou's protocol can be used to perform AR.

The OD protocol is executed through the following steps

- R_c generates a random nonce $(N_1)_{R_c}$ and sends it along with a Delegate command and the maximum number of permitted delegations (Cnt_{max}) to T.
- T responds by generating a random nonce N_T and computing $H_{K_{R_c,T}}(N_T, (N_1)_{R_c}, Cnt_{max})$ and then it transmits them to R_c.
- R_c then employs the received message to verify the T's legitimacy. If the verification succeeds, it generates another random nonce $(N_2)_{R_c}$ and computes $H_{K_{R_c,T}}((N_2)_{R_c}, (N_1)_{R_c}, Cnt_{max})$. Finally, R_c transmits $(N_2)_{R_c}$ and $H_{K_{R_c,T}}((N_2)_{R_c}, (N_1)_{R_c}, Cnt_{max})$ to T.
- T in turn uses the received messages to authenticate R_c and the delegation command. If R_c passes the verification, T calculates the delegation key K_D as $K_D = H_{K_{R_c,T}}((N_2)_{R_c}, (N_1)_{R_c}, N_T, Cnt_{max}, \text{“}Delegate''$), sets the DELEGATED bit to 1, and initializes the stored Cnt value to Cnt_{max}.
- R_c also computes K_D and forwards it to the delegate (R_d) through a secure channel. Therefore, the delegate now can query T.

  Each time the delegate queries T, Cnt value is decremented by 1. When Cnt becomes 0, T sets the DELEGATED bit to 0 and destroys K_D. Therefore, the delegation is terminated and again T can only be accessed by R_c.

It should be mentioned that some other protocols start with a Cnt value of 0 and increment it each time T is queried. In these protocols, the delegation is terminated when Cnt value exceeds Cnt_{max}.

In Dimitriou's protocol, R_c can terminate the delegation - before Cnt expires- automatically by two methods. The first one is through updating T's key, whereas the second method is to issue a direct Cancel Delegation command. When this command is received by T and if R_c is verified, T terminates the delegation by setting the DELEGATED bit to 0 and destroying K_D.

As for AR, it is performed by the same OD protocol when R_o needs to query T only. On the other hand, if R_o requires performing more functions on T, the ownership needs to be transferred to it completely.

However, Lee *et al.* [90] stated that because Dimitriou's protocol does not achieve full mutual authentication between the participating entities, the delegation key may be disclosed by *Adv*.

Other OD protocols that follow a procedure similar to Dimitriou's protocol were also proposed in [10], [13], [44], [45], [50], [83], [84], [88], and [93].

A last note on the OD is that in some works the delegation is not controlled (i.e. the number of times that the delegate can query T is not limited). In these works, the delegation can be terminated by R_c only. Examples on this approach are the protocols proposed in [80] and [37].

To further clarify the idea of uncontrolled delegation we briefly explains Niu *et al.*'s OD protocol [37]. In this protocol, the OD process is started with the computation of a secret value, called Ticket, which encrypts the master key (which is shared between R_c and T) and ID_T. The Ticket is computed by R_c and T. After that, the delegate obtains the Ticket from R_c via a secure channel. The purpose of the Ticket is to prove that the delegate is legitimate and has the required credentials to access T. Thus, once the delegate obtains the Ticket, it can query T and update its secret (i.e. K_T) through the proposed mutual authentication protocol. For this protocol, there is no counter and the delegate can query T without limitation. Therefore, to terminate the delegation, all R_c needs to do is change the value of the master key so that the Ticket value at the delegate's side will differ from its counterpart at T's side. Thus, the delegate can no longer be authenticated by T.

Some works propose methods to achieve either OD or AR and some provide both.

VII. COMPARISONS

Table 10 and Table 11 provide a comparison of the discussed protocols based on the properties and security vulnerabilities (attacks); respectively.

The notation used in Table 10 indicates the following:
✓ The protocol satisfies the property.
* The protocol satisfies the property under an assumption.
✕ The protocol does not satisfy the property.

As for Table 11, the notation points out the following:
✓ The protocol can resist the attack.
* : The protocol can resist the attack under an assumption.
✕: The protocol cannot resist the attack.

Note that we did not add the forward and backward traceability attacks to Table 11 because our interest in this survey is their impact on the old and new owner privacies which are presented in the Table 10.

VIII. DISCUSSION

With the abundance of previous work in this field, it is quite essential to use a unified terminology. For example, the presentation of the term old owner privacy varies between backward untraceability, forward privacy, and backward security. The same applies to new owner privacy which is normally substituted by other terms such as forward untraceability, backward privacy, and forward security. These different terms may cause confusion to the novice reader. In this paper, we adopted the terms old owner privacy and the new owner privacy as they give a direct meaning and are easy to follow by the readers.

Also, in comparing the protocols in the literature as being TTP-based or Two-party, we say that approximately there is the same number in both categories. However, if we compare them from the single and group OT perspective, it is obvious that the latter is receiving less attention than the former. We believe that further work is needed to cover the group OT as it is more applicable to the practical uses of RFID systems.

Another point of interest is that if the reviewed protocols in this survey are compared based on the EPC-compliance perspective, we see that the number of protocols that are EPC-compliant these that are non-EPC-compliant. There should be more work towards the EPC-compliant OT protocols, especially for the group OT. Based on our reviewed literature, there is only 1 group EPC-compliant protocol vs seven protocols from the other category. In addition, the only group EPC-compliant protocol falls under the TTP-based OT category, whereas there is none for the Two-party OT category. This area is open are for future research. This couples with the argument that states that EPC-compliance is quite essential for a protocol to be adopted in the supply chain is almost half these that are non-EPC-compliant. There should be more work towards the EPC-compliant OT protocols, especially for the group OT. Based on our reviewed literature, there is only 1 group EPC-compliant protocol vs seven protocols from the other category. In addition, the only group EPC-compliant protocol falls under the TTP-based OT category, whereas there is none for the Two-party OT category. This area is open for future research. This couples with the argument which states that EPC-compliance is quite essential for a protocol to be adopted in the supply chain

Offering exclusive OT and supporting mobile readers is quite essential but is not well-addressed. Based on the reviewed literature, we found that only two papers offer exclusive OT. As for supporting mobile readers, it can be seen that the modern RFID applications have included the use of mobile readers (such as PDAs and mobile phones). On the other hand, we found that only twelve protocols from the presented works can support mobile readers with little support of the required properties/resistance to attacks. This offers a hot area of research for interested researchers to work on, especially with the needs of the modern RFID applications.

Another issue to be addressed is related to the situations where the owners need to rent their tagged items to other, or temporarily transfer ownership to entities such as the TPL provider, or allowing the old owners to access them. In the first two cases the protocol should offer OD, whereas in the third it should provide AR. These situations are common in our lives, however; only thirteen protocols from the presented works afford either OD or AR and just five protocols provide both. Addressing the challenges of such open issues are quite essential for the proper adoption in RFID systems in scenarios that need these properties.

Unfortunately, from the provided discussion for the reviewed protocols, it can be seen that roughly half of them do not achieve at least one of the following: old owner privacy, new owner privacy, and resistance to windowing problem. Furthermore, approximately half of them are susceptible to at least one of the attacks that have been highlighted in this survey.

Generally speaking, the vulnerabilities of most of the presented works are due to one or more of the following:

TABLE 10. Summary of the properties of the investigated protocols.

Reference			Property					
			New Owner Privacy	Old Owner Privacy	Windowing Problem	Exclusive OT	Controlled Delegation Or Authorization Recovery	Support Mobile Readers
EPC-compliant TTP-based Single OT protocols	Lim	[13]	✓	* [84]	× [66]	×	✓ (OD)	✓
	Seo	[15]	× [37]	✓	✓	×	×	×
	Osaka	[19]	✓	✓	✓	×	×	×
	Chen	[22]	✓	✓	✓	×	×	×
	Chen	[24]	✓	✓	✓	×	×	×
	Chen	[25]	×	✓	× [100]	×	×	×
	Kulseng	[27]	×	✓	✓	×	×	×
	Cui	[33]	✓	✓	✓	×	×	×
	Lo	[34]	✓	✓	✓	×	✓ (AR)	×
	Niu	[37]	×	✓	✓	×	✓ (OD)	×
	Taqieddin	[39]	✓	✓	✓	×	✓ (OD)	×
	Cao	[42]	✓	✓	✓	×	×	✓
Non-EPC-compliant TTP-based Single OT protocols	Saito	[9]	✓	×	× [34]	×	×	×
	Fouladgar	[45]	×	✓	✓	×	✓ (OD)	×
	Chen	[63]	✓	✓	✓	×	×	✓
	Molnar	[10]	✓	✓	× [8]	×	✓ (OD)	×
	Fouladgar	[44]	×	✓	✓	×	× (OD)	×
	Ahamed	[48]	✓	✓	✓	×	×	×
	Rekleitis	[50]	✓	✓	✓	×	✓ (OD)	×
	Osaka	[52]	× [47]	× [57]	× [8]	×	×	×
	Jappinen	[53]	* [47]	✓	✓	×	×	×
	Yoon	[54]	× [18]	✓	✓	×	×	×
	Chen	[55]	✓	✓	✓	×	×	×
	Lei	[56]	× [18]	✓	✓	×	×	×
	Wang	[57]	× [23]	✓	✓	×	×	×
	Gui	[58]	✓	✓	✓	×	×	×
	Kapoor	[46]	✓	✓	*	×	×	×
	Kapoor	[18]	✓	✓	*	×	×	×
	Bagheri	[71]	✓	✓	✓	×	×	×
	Yang	[8]	✓	✓	✓	×	×	✓
	Zhou	[62]	✓	✓	✓	×	✓ (OD)	×
EPC-compliant TTP-based Group OT protocols	Sundaresan	[64]	×	*	✓	×	×	×
Non-EPC-compliant TTP-based Group OT protocols	Zuo	[66]	✓	✓	✓	×	✓ (AR)	×
	Kapoor	[70]	✓	✓	*	×	×	×
	Bagheri	[71]	✓	✓	✓	×	×	×
	He	[73]	✓	✓	✓	×	×	×
EPC-compliant Two-Party Single OT protocols	Dimitriou	[75]	×	✓	✓	×	✓ (OD)	✓
	Kulseng	[27]	×	✓	✓	×	×	×
	Cui	[33]	✓	✓	✓	×	×	×
	Chen	[76]	✓	✓	✓	×	×	✓
	Doss	[78]	×†	×†	✓	×†	×	×
Non-EPC-compliant Two-Party Single OT protocols	Saito	[9]	×	×	× [34]	×	×	×
	Koralalage	[79]	✓	✓	✓	×	×	×
	Song	[80]	×	✓	× [34]	×	✓ (AR)	×
	Song	[83]	×	×	✓	×	✓ (OD)	×
	Song	[84]	× [95]	×	✓	×	✓ (Both)	×
	Habibi	[85]	✓	✓	✓	×	✓ (Both)	×
	Jin	[47]	✓	✓	✓	×	×	×
	Fernandez-Mir	[88]	✓	✓	✓	×	✓ (Both)	×
	Xie	[49]	✓	✓	✓	×	×	✓
	Dimitriou	[89]	*	*	✓	×	✓ (Both)	✓
	Ahamed	[48]	✓	✓	✓	×	×	×
	Wang	[86]	✓	✓	✓	×	×	×
	Chen	[87]	✓	✓	✓	×	×	×
	Munilla	[91]	✓	✓	✓	×	×	×
	Ng	[93]	✓	✓	✓	✓	✓ (Both)	×
	Elkhiyaoui	[94]	*	*	✓	×	×	×
	Yang	[95]	✓	✓	✓	×	×	×
	Yang	[96]	✓	✓	× [97]	×	×	✓
Two-Party Group OT protocols	Kapoor	[70]	✓	✓	✓	×	×	×
	Yang	[97]	×	✓	× [100]	×	×	✓
	Li	[98]	✓	✓	✓	×	×	×
Universal OT protocols	Ray	[100]	✓	✓	✓	×	×	✓
	Ray	[101]	✓	✓	✓	✓	×	✓*

✓ The protocol satisfies the property.
* The protocol satisfies the property under an assumption.
× The protocol does not satisfy the property.
† Applied to the open loop scheme only.

TABLE 11. Summary of the vulnerabilities to attacks for the investigated protocols.

Reference			Property					
			Secret Discolsure	Tag Impersonation	Server/Reader Impersonation	Replay	De-synchronization	Tag Tracking
EPC-compliant TTP-based Single OT protocols	Lim	[13]	✓	✓	* [84]	×	✓	×
	Seo	[15]	✓	✓	×	× [37]	×	×
	Osaka	[19]	✓	✓	✓	✓	✓	✓
	Chen	[22]	✓	×	✓	✓	✓	×
	Chen	[24]	✓	✓	✓	✓	✓	✓
	Chen	[25]	×	×	✓	✓	✓	×
	Kulseng	[27]	×	✓	✓	×	×	×
	Cui	[33]	✓	✓	✓	✓	✓	✓
	Lo	[34]	✓	✓	✓	✓	✓	✓
	Niu	[37]	×	✓	✓	×	×	✓
	Taqieddin	[39]	✓	✓	✓	✓	✓	✓
	Cao	[42]	✓	✓	✓	✓	✓	✓
Non-EPC-compliant TTP-based Single OT protocols	Saito	[9]	✓	✓	✓	× [34]	×	✓
	Fouladgar	[45]	✓	×	*	*	× [34]	✓
	Chen	[63]	✓	✓	✓	✓	✓	✓
	Molnar	[10]	✓	✓	✓	✓	✓	*
	Fouladgar	[44]	✓	×	*	*	✓	✓
	Ahamed	[48]	✓	✓	✓	✓	×	✓
	Rekleitis	[50]	✓	✓	✓	✓	✓	✓
	Osaka	[52]	✓	✓	× [80]	×	×	×
	Jappinen	[53]	✓	✓	✓	× [18]	× [18]	✓
	Yoon	[54]	× [18]	× [18]	✓	× [18]	× [18]	× [18]
	Chen	[55]	✓	✓	× [18]	× [18]	✓	× [18]
	Lei	[56]	✓	✓	× [18]	× [18]	* [57]	✓
	Wang	[57]	✓	✓	✓	× [23]	× [23]	✓
	Gui	[58]	✓	✓	✓	✓	✓	✓
	Kapoor	[46]	✓	✓	✓	×	×	✓
	Kapoor	[18]	✓	✓	✓	×	×	✓
	Bagheri	[71]	✓	✓	✓	✓	✓	✓
	Yang	[8]	✓	✓	✓	✓	✓	✓
	Zhou	[62]	✓	✓	✓	✓	✓	✓
EPC-compliant TTP-based Group OT protocols	Sundaresan	[64]	×	*	×	*	*	*
Non-EPC-compliant TTP-based Group OT protocols	Zuo	[66]	✓	✓	✓	✓	✓	✓
	Kapoor	[70]	✓	✓	✓	×	×	✓
	Bagheri	[71]	✓	✓	✓	✓	✓	✓
	He	[73]	✓	✓	✓	✓	✓	✓
EPC-compliant Two-Party Single OT protocols	Dimitriou	[75]	×	×	× [64]	×	×	× [64]
	Kulseng	[27]	×	✓	✓	×	×	×
	Cui	[33]	✓	✓	✓	✓	✓	✓
	Chen	[76]	×	×	×	×	✓	×
	Doss	[78]	✓	×†	×	×	×	×†
Non-EPC-compliant Two-Party Single OT protocols	Saito	[9]	✓	✓	✓	× [34]	×	× [73]
	Koralalage	[79]	✓	✓	✓	×	×	×
	Song	[80]	×	×	×	×	×	×
	Song	[83]	✓	*	*	×	×	×
	Song	[84]	✓	* [64]	* [64]	×	×	×
	Habibi	[85]	✓	✓	✓	✓	✓	✓
	Jin	[47]	✓	✓	✓	✓	✓	✓
	Fernandez-Mir	[88]	✓	✓	✓	✓	× [89]	✓
	Xie	[49]	✓	✓	✓	✓	✓	✓
	Dimitriou	[89]	×	✓	×	×	✓	✓
	Ahamed	[48]	✓	✓	✓	×	×	✓
	Wang	[86]	✓	✓	✓	✓	✓	×
	Chen	[87]	✓	✓	✓	✓	✓	✓
	Munilla	[91]	✓	✓	✓	✓	✓	✓
	Ng	[93]	✓	✓	✓	× [89]	✓	✓
	Elkhiyaoui	[94]	✓	✓	✓	✓	✓	*
	Yang	[95]	✓	✓	✓	✓	✓	✓
	Yang	[96]	✓	✓	✓	✓	✓	✓
Two-Party Group OT protocols	Kapoor	[70]	✓	✓	✓	✓	✓	✓
	Yang	[97]	✓	✓	✓	✓	✓	✓
	Li	[98]	✓	✓	✓	✓	✓	✓
Universal OT protocols	Ray	[100]	✓	✓	✓	✓	✓	✓
	Ray	[101]	✓	✓	✓	✓	✓	✓

✓ The protocol can resist the attack.
* The protocol can resist the attack under an assumption.
× The protocol cannot resist the attack.
† Applied to the open loop scheme only.

- The messages sent by the participating entities are static (i.e. do not contain fresh nonces) which allows for tracking attacks.
- Transmitting the sensitive information in clear text helps the adversaries to perform tracking and impersonation attacks.
- Blocking or modifying the last message sent from the reader/server to the tag which, in turn, does not send an acknowledgment allows for de-synchronization attacks.
- Lack of or improper design of mutual authentication between the participating entities makes it easier for the adversaries to perform various attacks.
- Misuse of the employed cryptographic functions can lead to disclosure attacks.
- Not taking into consideration the cases that may arise if/when a tag has been compromised.

To the best of our knowledge and based on the reviewed literature, the other half of protocols can presumably guarantee the old and new owner privacies and resists the windowing problem and the security attacks. However, these are claims that were made by the authors and the protocols (especially the recent ones) did not undergo rigorous evaluation to prove their security worthiness. Research in this area is needed to guarantee that an adopted protocol would, in reality, provide the claimed security properties and strengths.

We conclude this section by offering a set of guidelines to be considered in the design of new OT protocols. We divide the guidelines into five layers and each layer should be built upon those that precede it (i.e; the protocol complexity increases as we add further layers).

Layer I: This layer presents the minimum requirements that should exist in an OT protocol. First of all, an OT protocol should at least achieve three properties, namely the old owner privacy, new owner privacy, and resistance to the windowing problem. Fulfilling the first two properties is a must in order to achieve a genuine and successful transfer that protects the privacy of both owners. Moreover, it is necessary to ensure that the protocol resists the windowing problem, otherwise the result will be a sharing protocol where both owners can access the tag.

The protocol should offer complete mutual authentication between the participating entities and take into consideration the importance of avoiding the reasons for the vulnerabilities that were summarized above in order to secure the protocol against the attacks in light of the attacks shown in the survey.

The protocol should adhere to the EPC-standard to be compatible with the majority of RFID applications which tend to use low-cost (i.e.; passive tags). Also, the protocol design should be modular such that it can perform several operations with the least possible implementation cost.

Deciding whether the protocol should be designed based on the TTP-based model or the Two-party model depends on the system in which the protocol is to be used. If the owners trust a third party to watch over the transfer, the protocol should be designed based on the TTP-based model. On the other hand, if the owners prefer to perform the transfer directly with each other, the protocol should be modeled after the Two-party model.

An important point to mention regarding this issue is that the ISE assumption which is followed by the Two-party model is considered unreasonable. Therefore, the newly designed protocols that are based on the Two-party model should perform OT between the owners without relying on the ISE assumption. Examples on already proposed protocols that fulfill this concept and discussed in this survey are in [73], [91], and [95]. From another point of view, these three protocols are not EPC-compliant since they use *PKI* and hashing on the tag's side. Thus, the real challenge is to design an EPC-compliant Two-party OT protocol without a need for an isolated environment.

Layer II: This layer presents the additional properties that should be offered by the OT protocol. The additional properties are OD, AR, and exclusive OT. As we argued previously, the OT protocol should also provide OD and AR in order to handle the situations where other parties need to access the tags. Moreover, we believe that the OT protocol should support the exclusive OT property to confirm to the new owner that the protocol session has completed successfully and that it is the sole owner.

Layer III: In our opinion, the newly designed OT protocols need to support mobile readers such that they can be used with the variety of mobile RFID applications that emerged recently.

Layer IV: We believe that the OT protocol should also support group OT in order to transfer the ownership of a group of tags at once from one owner to another (i.e. multi-tag single-owner case). This group transfer is necessary for certain environments such as supply chains systems.

Layer V: The designed OT protocol should afford other types of group transfer which means that it will be a universal OT protocol. An example of such a protocol is the work proposed by Ray *et al.* in [101]. However, this protocol is missing one property in that it neither offers OD nor AR.

IX. CONCLUSION

In this survey, we gave a comprehensive and systematic review of the RFID ownership transfer which is one of the most important aspects of RFID systems. We started by introducing RFID systems and their security goals in general and the RFID ownership transfer and its security goals in particular. After that, we surveyed the protocols in this field from the early-start work up to the current state of the art.

To the best of our knowledge, this paper is the first comprehensive survey for RFID ownership transfer which reviews the proposed protocols that appeared over the period 2005-2018. We classified the protocols based on the mainly used models, namely the TTP-based and Two-party and further we categorized the protocols in each model based on the EPC-standard and the type of transfer (single vs. group).

For each protocol, we presented a brief discussion of its procedure and vulnerabilities and highlighted its main theme. In addition, we clarified the idea of OD and AR through

a general example. After that, we compared all protocols in terms of the security goals which involve the offered ownership transfer properties and security against attacks in a tabular form.

Finally, we gave our insight regarding the discussed work in order to identify open issues that needs to be handled in the future research and we also provide general guidelines to design an OT protocol that would achieve the required security goals.

REFERENCES

[1] R. Weinstein, "RFID: A technical overview and its application to the enterprise," *IT Prof.*, vol. 7, no. 3, pp. 27–33, May/Jun. 2005.

[2] A. Juels, "RFID security and privacy: A research survey," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 381–394, Feb. 2006.

[3] M. Alizadeh, M. Zamani, A. R. Shahemabadi, J. Shayan, and A. Azarnik, "A survey on attacks in RFID networks," *Open Int. J. Inform.*, vol. 1, no. 1, pp. 15–24, 2012.

[4] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "RFID systems: A survey on security threats and proposed solutions," in *Proc. PWC*, vol. 4217. Berlin, Germany: Springer, 2006, pp. 159–170.

[5] H. Lee and J. Kim, "Privacy threats and issues in mobile RFID," in *Proc. 1st Int. Conf. Availability, Rel. Secur. (ARES)*, 2006, p. 5.

[6] C. Seidler, "RFID opportunities for mobile telecommunication services," ITU-T, Geneva, Switzerland, Tech. Paper, 2005.

[7] Y. A. Au and R. J. Kauffman, "The economics of mobile payments: Understanding stakeholder issues for an emerging financial technology application," *Electron. Commerce Res. Appl.*, vol. 7, no. 2, pp. 141–164, 2008.

[8] M. H. Yang, "Across-authority lightweight ownership transfer protocol," *Electron. Commerce Res. Appl.*, vol. 10, no. 4, pp. 375–383, 2011.

[9] J. Saito, K. Imamoto, and K. Sakurai, "Reassignment scheme of an RFID tag's key for owner transfer," in *Proc. EUC Workshops*, vol. 3823. Berlin, Germany: Springer, 2005, pp. 1303–1312.

[10] D. Molnar, A. Soppera, and D. Wagner, "A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags," in *Selected Areas in Cryptography*, vol. 3897. Berlin, Germany: Springer, 2005, pp. 276–290.

[11] G. Kapoor, "Secure ownership transfer and authentication protocols for radio frequency identification (RFID)," Ph.D. dissertation, Univ. Florida, Gainesville, FL, USA, 2008.

[12] M. Langheinrich, "A survey of RFID privacy approaches," *Pers. Ubiquitous Comput.*, vol. 13, no. 6, pp. 413–421, 2009.

[13] C. H. Lim and T. Kwon, "Strong and robust RFID authentication enabling perfect ownership transfer," in *Proc. ICICS*, vol. 6. Berlin, Germany: Springer, 2006, pp. 1–20.

[14] K. Ouafi and R. C. -W. Phan, "Traceable privacy of recent provably-secure RFID protocols," in *Applied Cryptography and Network Security*. Berlin, Germany: Springer, 2008, pp. 479–489.

[15] Y. Seo, T. Asano, H. Lee, and K. Kim, "A lightweight protocol enabling ownership transfer and granular data access of RFID tags," in *Proc. Symp. Cryptogr. Inf. Secur. (SCIS)*, 2007, pp. 1–6.

[16] C. Adams, S. Farrell, T. Kause, and T. Mononen, "Internet x. 509 public key infrastructure certificate management protocol (CMP)," IETF, San Diego, CA, USA, Tech. Rep. RFC 2631, 2005.

[17] M. R. Rieback, B. Crispo, and A. S. Tanenbaum, "RFID guardian: A battery-powered mobile device for RFID privacy management," in *Proc. ACISP*, vol. 5. Berlin, Germany: Springer, 2005, pp. 184–194.

[18] G. Kapoor and S. Piramuthu, "Single RFID tag ownership transfer protocols," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 2, pp. 164–173, Mar. 2012.

[19] K. Osaka, S. Chang, T. Takagi, K. Yamazaki, and O. Takahashi, "A secure RFID protocol based on insubvertible encryption using guardian proxy," in *Proc. 3rd Int. Conf. Availability, Rel. Secur. (ARES)*, Mar. 2008, pp. 733–740.

[20] A. Juels, P. Syverson, and D. Bailey, "High-power proxies for enhancing RFID privacy and utility," in *Proc. Int. Workshop Privacy Enhancing Technol.* Berlin, Germany: Springer, 2005, pp. 210–226.

[21] G. Ateniese, J. Camenisch, and B. de Medeiros, "Untraceable RFID tags via insubvertible encryption," in *Proc. 12th ACM Conf. Comput. Commun. Secur.*, 2005, pp. 92–101.

[22] C.-L. Chen, Y.-Y. Chen, Y.-C. Huang, C.-S. Liu, C.-I. Lin, and T.-F. Shih, "Anti-counterfeit ownership transfer protocol for low cost RFID system," *WSEAS Trans. Comput.*, vol. 7, no. 8, pp. 1149–1158, 2008.

[23] G. Kapoor and S. Piramuthu, "Vulnerabilities in some recently proposed RFID ownership transfer protocols," in *Proc. 1st Int. Conf. Netw. Commun. (NETCOM)*, Dec. 2009, pp. 354–357.

[24] C.-L. Chen, Y.-C. Huang, and J.-R. Jiang, "A secure ownership transfer protocol using EPCglobal Gen-2 RFID," *Telecommun. Syst.*, vol. 53, no. 4, pp. 387–399, 2013.

[25] C.-L. Chen, Y.-L. Lai, C.-C. Chen, Y.-Y. Deng, and Y.-C. Hwang, "RFID ownership transfer authorization systems conforming EPCglobal class-1 generation-2 standards," *IJ Netw. Secur.*, vol. 13, no. 1, pp. 41–48, 2011.

[26] M. R. S. Abyaneh, "On the privacy of two tag ownership transfer protocols for RFIDs," in *Proc. Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2011, pp. 683–688.

[27] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight mutual authentication and ownership transfer for RFID systems," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.

[28] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th Annu. Des. Automat. Conf.*, 2007, pp. 9–14.

[29] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.

[30] S. Kardas, M. Akgün, M. S. Kiraz, and H. Demirci, "Cryptanalysis of lightweight mutual authentication and ownership transfer for RFID systems," in *Proc. Workshop Lightweight Secur. Privacy, Devices, Protocols Appl. (LightSec)*, Mar. 2011, pp. 20–25.

[31] J. D. Golić, "Correlation analysis of the shrinking generator," in *Proc. CRYPTO*, vol. 2139. Berlin, Germany: Springer, 2001, pp. 440–457.

[32] W. Meier and O. Staffelbach, "The self-shrinking generator," in *Communications and Cryptography*. Boston, MA, USA: Springer, 1994, pp. 287–295.

[33] P.-Y. Cui, "An improved ownership transfer and mutual authentication for lightweight RFID protocols," *IJ Netw. Secur.*, vol. 18, no. 6, pp. 1173–1179, 2016.

[34] N.-W. Lo, S.-H. Ruan, and T.-C. Wu, "Ownership transfer protocol for RFID objects using lightweight computing operators," in *Proc. Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2011, pp. 484–489.

[35] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "Solving the simultaneous scanning problem anonymously: Clumping proofs for RFID tags," in *Proc. 3rd Int. Workshop Secur., Privacy Trust Pervasive Ubiquitous Comput. (SECPerU)*, Jul. 2007, pp. 55–60.

[36] Y. Tian, G. Chen, and J. Li, "A new ultralightweight RFID authentication protocol with permutation," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 702–705, May 2012.

[37] H. Niu, E. Taqieddin, and S. Jagannathan, "EPC Gen2v2 RFID standard authentication and ownership management protocol," *IEEE Trans. Mobile Comput.*, vol. 15, no. 1, pp. 137–149, Jan. 2016.

[38] N. Bagheri *et al.*, "Security analysis of Niu authentication and ownership management protocol," Shahid Rajaee Teachers Training Univ., Tehran, Iran, Tech. Rep. 615, 2015.

[39] E. Taqieddin, H. Al-Dahoud, M. Mowafi, and O. Banimelhem, "An enhanced EPC Gen2v2 RFID authentication and ownership management protocol," in *Proc. IEEE 42nd Conf. Local Comput. Netw. (LCN)*, Oct. 2017, pp. 683–689.

[40] H. Luo, G. Wen, J. Su, and Z. Huang, "SLAP: Succinct and lightweight authentication protocol for low-cost RFID system," *Wireless Netw.*, vol. 24, no. 1, pp. 69–78, 2018.

[41] X. Zhuang, Z.-H. Wang, C.-C. Chang, and Y. Zhu, "Security analysis of a new ultra-lightweight RFID protocol and its improvement," *J. Inf. Hiding Multimedia Signal Process.*, vol. 4, no. 3, pp. 166–177, 2013.

[42] T. Cao, X. Chen, R. Doss, J. Zhai, L. J. Wise, and Q. Zhao, "RFID ownership transfer protocol based on cloud," *Comput. Netw.*, vol. 105, pp. 47–59, Aug. 2016.

[43] K. Nohl and D. Evans, "Quantifying information leakage in tree-based hash protocols (short paper)," in *Information and Communications Security*. Berlin, Germany: Springer, 2006, pp. 228–237.

[44] S. Fouladgar and H. Afifi, "A simple delegation scheme for RFID systems (SiDeS)," in *Proc. IEEE Int. Conf. RFID*, Mar. 2007, pp. 1–6.

[45] S. Fouladgar and H. Afifi, "An efficient delegation and transfer of ownership protocol for RFID tags," in *Proc. 1st Int. EURASIP Workshop RFID Technol.*, Vienna, Austria, vol. 66, 2007, pp. 68–93.
[46] G. Kapoor and S. Piramuthu, "Protocols for objects with multiple RFID tags," in *Proc. 16th Int. Conf. Adv. Comput. Commun. (ADCOM)*, Dec. 2008, pp. 208–213.
[47] Y. Jin, H. Sun, and Z. Chen, "Hash-based tag ownership transfer protocol against traceability," in *Proc. IEEE Int. Conf. e-Business Eng. (ICEBE)*, Oct. 2009, pp. 487–492.
[48] S. I. Ahamed, F. Rahman, M. E. Hoque, F. Kawsar, and T. Nakajima. "YA-SRAP: yet another serverless RFID authentication protocol," in *Proc. 4th Int. Conf. Intell. Environ.*, Seattle, WA, USA, Jul. 2008.
[49] W. Xie, L. Xie, C. Zhang, Q. Wang, C. Wang, and C. Tang, "TOA: A tag-owner-assisting RFID authentication protocol toward access control and ownership transfer," *Secur. Commun. Netw.*, vol. 7, no. 5, pp. 934–944, 2014.
[50] E. Rekleitis, P. Rizomiliotis, and S. Gritzalis, "How to protect security and privacy in the IoT: A policy-based RFID tag management protocol," *Secur. Commun. Netw.*, vol. 7, no. 12, pp. 2669–2683, 2014.
[51] *Data Elements and Interchange Formats—Information Interchange—Representation of Dates and Times*, document ISO8601 IS, ISO/TC154, 2000.
[52] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi, "An efficient and secure RFID security method with ownership transfer," in *RFID Security*. Boston, MA, USA: Springer, 2008, pp. 147–176.
[53] P. Jäppinen and H. Hämäläinen, "Enhanced RFID security method with ownership transfer," in *Proc. Int. Conf. Comput. Intell. Secur. (CIS)*, vol. 2, Dec. 2008, pp. 382–385.
[54] E.-J. Yoon and K.-Y. Yoo, "Two security problems of RFID security method with ownership transfer," in *Proc. IFIP Int. Conf. Netw. Parallel Comput. (NPC)*, Oct. 2008, pp. 68–73.
[55] H.-B. Chen, W.-B. Lee, Y.-H. Zhao, and Y.-L. Chen, "Enhancement of the RFID security method with ownership transfer," in *Proc. 3rd Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2009, pp. 251–254.
[56] H. Lei and T. Cao, "RFID protocol enabling ownership transfer to protect against traceability and dos attacks," in *Proc. 1st Int. Symp. Data, Privacy, E-Commerce (ISDPE)*, Nov. 2007, pp. 508–510.
[57] C.-H. Wang and S. Chin, "A new RFID authentication protocol with ownership transfer in an insecure communication environment," in *Proc. 9th Int. Conf. Hybrid Intell. Syst. (HIS)*, vol. 1, Aug. 2009, pp. 486–491.
[58] Y.-Q. Gui, J. Zhang, and H. K. Choi, "An improved RFID security method with ownership transfer," in *Proc. Int. Conf. ICT Converg. (ICTC)*, Sep. 2011, pp. 594–596.
[59] A. Poschmann, G. Leander, K. Schramm, and C. Paar, "New light-weight crypto algorithms for RFID," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 1843–1846.
[60] M. Hell, T. Johansson, and W. Meier, "Grain: A stream cipher for constrained environments," *Int. J. Wireless Mobile Comput.*, vol. 2, no. 1, pp. 86–93, 2007.
[61] J. Kohl and C. Neuman, "The kerberos network authentication service (v5)," IETF, San Diego, CA, USA, Tech. Rep. RFC 1510, 1993.
[62] W. Zhou, E. J. Yoon, and S. Piramuthu, "Simultaneous multi-level RFID tag ownership & transfer in health care environments," *Decis. Support Syst.*, vol. 54, no. 1, pp. 98–108, 2012.
[63] B.-C. Chen, H.-T. Yeh, and C.-C. Wang, "The construction of mobile RFID authentication mechanism and relative ownership transfer protocols for supply chain," *Eng. Comput.*, vol. 33, no. 6, pp. 1825–1834, 2016.
[64] S. Sundaresan, R. Doss, W. Zhou, and S. Piramuthu, "Secure ownership transfer for multi-tag multi-owner passive RFID environment with individual-owner-privacy," *Comput. Commun.*, vol. 55, pp. 112–124, Jan. 2015.
[65] J. Munilla, M. Burmester, and A. Peinado, "Attacks on ownership transfer scheme for multi-tag multi-owner passive RFID environments," *Comput. Commun.*, vol. 88, pp. 84–88, Aug. 2016.
[66] Y. Zuo, "Changing hands together: A secure group ownership transfer protocol for RFID tags," in *Proc. 43rd Hawaii Int. Conf. Syst. Sci. (HICSS)*, Jan. 2010, pp. 1–10.
[67] M. A. Bonuccelli, F. Lonetti, and F. Martelli, "Tree slotted ALOHA: A new protocol for tag identification in RFID networks," in *Proc. Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2006, pp. 603–608.
[68] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *Proc. 2nd Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services (MobiQuitous)*, Jul. 2005, pp. 166–172.
[69] M. Burmester, B. de Medeiros, and R. Motta, "Provably secure grouping-proofs for RFID tags," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Berlin, Germany: Springer, 2008, pp. 176–190.
[70] G. Kapoor, W. Zhou, and S. Piramuthu, "Multi-tag and multi-owner RFID ownership transfer in supply chains," *Decis. Support Syst.*, vol. 52, no. 1, pp. 258–270, 2011.
[71] N. Bagheri, F. Aghili, and M. Safkhani, "On the security of two ownership transfer protocols and their improvements," *Int. Arab J. Inf. Technol.*, vol. 15, no. 1, pp. 87–93, 2018.
[72] J. Munilla, A. Peinado, G. Yang, and W. Susilo, "Enhanced ownership transfer protocol for RFID in an extended communication model," Univ. Málaga, Málaga, Spain, Tech. Rep. 187, 2013.
[73] L. He, Y. Gan, and Y. Yin, "Secure group ownership transfer protocol with independence of old owner for RFID tags," *Comput. Model. New Technol.*, vol. 18, no. 12B, pp. 209–214, 2014.
[74] A. Ilic, F. Michahelles, and E. Fleisch, "Dual ownership: Access management for shared item information in RFID-enabled supply chains," in *Proc. 5th Annu. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2007, pp. 337–341.
[75] T. Dimitriou, "rfidDOT: RFID delegation and ownership transfer made simple," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netw.*, 2008, Art. no. 34.
[76] C.-L. Chen and C.-F. Chien, "An ownership transfer scheme using mobile RFIDs," *Wireless Pers. Commun.*, vol. 68, no. 3, pp. 1093–1119, 2013.
[77] J. Munilla, F. Guo, and W. Susilo, "Cryptanalaysis of an EPCC1G2 standard compliant ownership transfer scheme," *Wireless Pers. Commun.*, vol. 72, no. 1, pp. 245–258, 2013.
[78] R. Doss, W. Zhou, and S. Yu, "Secure RFID tag ownership transfer based on quadratic residues," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 2, pp. 390–401, Feb. 2013.
[79] K. H. S. S. Koralalage, S. M. Reza, J. Miura, Y. Goto, and J. Cheng, "POP method: An approach to enhance the security and privacy of RFID systems used in product lifecycle with an anonymous ownership transferring mechanism," in *Proc. ACM Symp. Appl. Comput.*, 2007, pp. 270–275.
[80] B. Song, "RFID tag ownership transfer," in *Proc. Workshop RFID Secur.*, Budapest, Hungary, 2008, pp. 1–41.
[81] B. Song and C. J. Mitchell, "RFID authentication protocol for low-cost tags," in *Proc. 1st ACM Conf. Wireless Netw. Secur.*, 2008, pp. 140–147.
[82] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Tapiador, T. Li, and Y. Li, "Vulnerability analysis of RFID protocols for tag ownership transfer," *Comput. Netw.*, vol. 54, no. 9, pp. 1502–1508, 2010.
[83] B. Song and C. J. Mitchell, "Scalable RFID pseudonym protocol," in *Proc. 3rd Int. Conf. Netw. Syst. Secur. (NSS)*, Oct. 2009, pp. 216–224.
[84] B. Song and C. J. Mitchell, "Scalable RFID security protocols supporting tag ownership transfer," *Comput. Commun.*, vol. 34, no. 4, pp. 556–566, 2011.
[85] M. H. Habibi and M. R. Aref, "Attacks on recent RFID authentication protocols," *J. Signal Process. Syst.*, vol. 79, no. 3, pp. 271–283, 2015.
[86] X. Wang and C. Yuan, "Scalable and resynchronisable radio frequency identification ownership transfer protocol based on a sliding window mechanism," *IET Inf. Secur.*, vol. 8, no. 3, pp. 161–170, May 2014.
[87] X. Chen, T. Cao, R. Doss, and J. Zhai, "Attacks on and countermeasures for two RFID protocols," *Wireless Pers. Commun.*, vol. 96, no. 4, pp. 5825–5848, 2017.
[88] A. Fernàndez-Mir, R. Trujillo-Rasua, J. Castellà-Roca, and J. Domingo-Ferrer, "A scalable RFID authentication protocol supporting ownership transfer and controlled delegation," in *Proc. Int. Workshop Radio Freq. Identification, Secur. Privacy Issues*. Berlin, Germany: Springer, 2011, pp. 147–162.
[89] T. Dimitriou, "Key evolving RFID systems: Forward/backward privacy and ownership transfer of RFID tags," *Ad Hoc Netw.*, vol. 37, pp. 195–208, Feb. 2016.
[90] C.-C. Lee, C.-L. Cheng, Y.-M. Lai, and C.-T. Li, "Cryptanalysis of dimitriou's key evolving RFID systems," in *Proc. 5th Int. Conf. Netw., Commun. Comput.*, 2016, pp. 229–233.
[91] J. Munilla, M. Burmester, A. Peinado, G. Yang, and W. Susilo, "RFID ownership transfer with positive secrecy capacity channels," *Sensors*, vol. 17, no. 1, p. 53, 2016.
[92] A. D. Wyner, "The wire-tap channel," *Bell Syst. Tech. J.*, vol. 54, no. 8, pp. 1355–1387, 1975.
[93] C. Y. Ng, W. Susilo, Y. Mu, and R. Safavi-Naini, "Practical RFID ownership transfer scheme," *J. Comput. Secur.*, vol. 19, no. 2, pp. 319–341, 2011.

[94] K. Elkhiyaoui, E.-O. Blass, and R. Molva, "ROTIV: RFID ownership transfer with issuer verification," in *Proc. Int. Workshop Radio Freq. Identification, Secur. Privacy Issues*. Berlin, Germany: Springer, 2011, pp. 163–182.
[95] X. Yang, C. Xu, and C. Li, "A privacy model for RFID tag ownership transfer," *Secur. Commun. Netw.*, vol. 2017, Mar. 2017, Art. no. 5084636.
[96] M. H. Yang and Y. H. Hu, "Protocol for ownership transfer across authorities: With the ability to assign transfer target," *Secur. Commun. Netw.*, vol. 5, no. 2, pp. 164–177, 2012.
[97] M. H. Yang, "Secure multiple group ownership transfer protocol for mobile RFID," *Electron. Commerce Res. Appl.*, vol. 11, no. 4, pp. 361–373, 2012.
[98] N. Li, Y. Mu, W. Susilo, and V. Varadharajan, "Shared RFID ownership transfer protocols," *Comput. Standards Interfaces*, vol. 42, pp. 95–104, Nov. 2015.
[99] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer, 2001, pp. 514–532.
[100] B. R. Ray, M. Chowdhury, and J. Abawajy, "Secure mobile RFID ownership transfer protocol to cover all transfer scenarios," in *Proc. 7th Int. Conf. Comput. Converg. Technol. (ICCCT)*, Dec. 2012, pp. 1185–1192.
[101] B. R. Ray, J. Abawajy, M. Chowdhury, and A. Alelaiwi, "Universal and secure object ownership transfer protocol for the Internet of Things," *Future Gener. Comput. Syst.*, vol. 78, pp. 838–849, 2017.
[102] E. Rescorla, *Diffie-Hellman Key Agreement Method*, document RFC 2631, 1999.
[103] J. Kim, D. Jung, Y. Jung, and Y. Baek, "Scalable RTLS: Design and implementation of the scalable real time locating system using active RFID," in *Proc. Int. Conf. Inf. Netw.* Berlin, Germany: Springer, 2007, pp. 503–512.
[104] H. Fernando and J. Abawajy, "Mutual authentication protocol for networked RFID systems," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Nov. 2011, pp. 417–424.
[105] S. Vaudenay, "On privacy models for RFID," in *Advances Cryptology—ASIACRYPT*. Berlin, Germany: Springer, 2007, pp. 68–87.

EYAD TAQIEDDIN (SM'15) received the B.Sc. degree in electrical engineering from the Jordan University of Science and Technology (JUST), Irbid, Jordan, in 1999, and the M.Sc. and Ph.D. degrees in computer engineering from the Missouri University of Science and Technology (formerly the University of Missouri-Rolla) in 2002 and 2007, respectively. He was an Assistant Professor with the Department of Computer Engineering, JUST, from 2007 to 2009. He is currently an Associate Professor and the Chair of the Department of Network Engineering and Security, JUST. His research interests are in the areas of RFID security, wireless multimedia sensor networks, network security, and embedded systems.

HIBA AL-DAHOUD received the B.S. degree in computer engineering from Yarmouk University, Irbid, Jordan, in 2013, and the M.Sc. degree in computer engineering from the Jordan University of Science and Technology, Irbid, in 2016. Her research interests include RFID privacy security.

HAIFENG NIU received the bachelor's degree in electrical engineering from Northeastern University, Shenyang, China, in 2010, the master's degree from the Department of Control and Engineering, Zhejiang University, Hangzhou, China, in 2012, and the Ph.D. degree from the Missouri University of Science and Technology, Rolla, in 2016. Since 2016, he has been with Amazon, Seattle. His research interests include RFID networks, wireless networking security, and cyber-physical systems.

JAGANNATHAN SARANGAPANI is currently with the Missouri University of Science and Technology (former University of Missouri-Rolla), where he is a Rutledge-Emerson Endowed Chair Professor of electrical and computer engineering and the Site Director for the NSF Industry/University Cooperative Research Center on Intelligent Maintenance Systems. He supervised to graduation around 18 doctoral and 29 M.S. level students and his funding is in excess of 14 million from various U.S. federal and industrial members. He has coauthored 129 peer-reviewed journal articles most of them in the IEEE TRANSACTIONS and 235 refereed IEEE conference articles, several book chapters, and three books, and holds 20 U.S. patents. His research interests include neural network control, adaptive event-triggered control, secure networked control systems, prognostics, and autonomous systems/robotics. He is a fellow of the Institute of Measurement and Control, U.K. He received many awards and has been on organizing committees of several IEEE Conferences. He is the IEEE CSS Tech Committee Chair on Intelligent Control. He was a co-editor of the IET book series on control from 2010 to 2013. He is currently serving as the Editor-in-Chief for *Discrete Dynamics in Nature and Society* and on many editorial boards.