Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02732-2

ORIGINAL RESEARCH

Ultralightweight resilient mutual authentication protocol for IoT based edge networks

Madiha Khalid[1] · Umar Mujahid[2] · M. Najam-ul-Islam[1] · Hongsik Choi[2] · Imtiaz Alam[1] · Shahzad Sarwar[3]

Received: 22 January 2019 / Accepted: 20 November 2020

Abstract
Confidentiality and authentication are primary concerns of RFID (Radio Frequency IDentification) based identification systems. Low cost, non sight-line scanning capability, unique identification and functional haste make RFID systems much more suitable for supply chain management and IoT networks, compare to barcodes and magnetic tapes. Since, RFID tags can be scanned from a greater distance over a radio channel which is easily accessible for adversaries, therefore numerous Ultralightweight Mutual Authentication Protocols (UMAPs) have been proposed to ensure the security of the RFID systems in a cost effective way. These UMAPs mainly involve low cost bitwise logical (*Triangular*) operations such as *XOR*, *ANDOR* etc., usually these logical operations offer poor diffusion properties and hence result in disclosure of the concealed secrets. Almost all of the previous UMAPs have certain pitfalls and reported to be susceptible to Denial of Service, traceability, full disclosure and cloning attacks. In this paper, we propose a new UMAP that is resilient to structured and unstructured cryptanalysis techniques. The proposed Ultralightweight Resilient Mutual Authentication Protocol (URMAP) avoids all the pitfalls and unbalanced operators in its design that were the main reasons of failure of previous UMAPs. The incorporation of ultralightweight primitives; *Per-XOR* (P_x) and *InversePer-XOR* (P_x^{-1}) makes the protocol messages more resistant to all types of adversaries. The security and privacy capabilities of URMAP are evaluated through formal analysis. The results of GNY Logic analysis prove that the proposed protocol is resilient to all possible attack scenarios thus ensuring optimal confidentiality, integrity and availability services.

Keywords RFID · UMAP · Ultralightweight · Permutation

1 Introduction

Radio Frequency IDentification (RFID) is an object discovery technology that consists of three main components; Tag($\mathcal{T}$), Reader ($\mathcal{R}$) and the back end database ($\mathcal{D}$). The '$\mathcal{T}$' acts a small electronic chip which contains the information about the object onto which it is implanted. The '$\mathcal{R}$' acts a scanner which reads the contents of all associated tags. The 'D' contains the detailed information of all the tags associated with the RFID system.

In a normal scenario of RFID handshaking, the '$\mathcal{R}$' continuously broadcasts the beacon signals and whenever '$\mathcal{T}$' enters in its vicinity, it receives a "*Hello*" message (pilot signal). Upon receiving of this message, the '$\mathcal{T}$' sends current *IDS* to the '$\mathcal{R}$'. The '$\mathcal{R}$' searches the matched entry (received *IDS)* in its 'D' and if a match is fount then the '$\mathcal{R}$' generates some pseudorandom numbers, conceals them in messages using pre shared secrets and sends to '$\mathcal{T}$.' The legitimate '$\mathcal{T}$' retrieves the pseudorandom numbers and authenticates the sender. Finally, the '$\mathcal{T}$' conceals its secret *ID* within the cipher message and sends to '$\mathcal{R}$' so, the '$\mathcal{R}$' can authenticate the tag. After successful authentication, the communicating parties i.e. tag and reader, update their pseudonyms (*IDS*, *Keys*) for future correspondence.

The communication channel that connects the '$\mathcal{R}$' and 'D' is considered to be secure since the channel is wired with minimal resources constraints; hence the traditional cryptographic suites can be used to encrypt their communications. However, the channel between the '$\mathcal{R}$' and '$\mathcal{T}$'

✉ Umar Mujahid
ukhokhar@ggc.edu

[1] Department of Electrical Engineering, Bahria University, Islamabad, Pakistan

[2] School of Science and Technology, Georgia Gwinnett College, Lawrenceville, GA 30043, USA

[3] College of Information Technology, Punjab University, Lahore, Pakistan

Published online: 02 January 2021

is wireless and requires cost effective security solutions because of limited computational capabilities at the tag's side. According to EPCC1G2 standard (Air et al. 2013), the tags can store only $32-1K$ bits and $10K$ logical gates can be used for computations. For security related task, such tags can use only $250-4K$ gates. Based on the computational capabilities of the tags, Chein (2007) classified the security protocols into four main types: Full Fledged Protocols, Simple Protocols, Lightweight Protocols and Ultralightweight Protocols.

a) *Full Fledged* may involve all traditional cryptographic algorithms such as AES, ECC and Hash Functions etc. and there is no power constraint at any side.
b) *Simple* involves pseudorandom number generators such as LCG (Linear Congruential Generators), Kasami sequences etc. and one-way Hash functions only.
c) *Lightweight* only supports CRC (Cyclic Redundancy Check) and lightweight random number generators such as LFSR (Linear Feedback Shift Register), LAMED etc.
d) *Ultralightweight* involves only bitwise logical operators and some special purpose ultralightweight primitives such as *Rotation* (Chien 2007), *RecursiveHash* (Mujahid et al. 2015) and *MixBits* (Peris-Lopez et al. 2008) etc.

The focus of the paper will remain on the ultralightweight class of protocols. Typically, most of the ultralightweight protocols involve only bitwise logical operators therefore offer extremely low security. However, incorporation of non-triangular primitives increases the robustness of the protocol against various attacks. Over hundreds of UMAPs (Chien 2007; Lee et al. 2009; Luo et al. 2018; Mujahid et al. 2017, 2015; Peris-Lopez et al. 2006a, b, c, 2008; Tian et al. 2012; Yeh et al. 2010; Zhuang et al. 2014) have been proposed since last decade, but almost all of the protocols were found to be vulnerable mostly within one (1) year right after their invention(Avoine et al. 2012). There are many reasons for this radical failure of UMAPs but validation of the protocol's functionalities using poor security analysis model is top of the list. A brief overview of the UMAPs and their vulnerabilities is presented below.

1.1 Overview of UMAPs

Pedro Peris et al. (2006a, b, c) introduced the concept of using *T-functions* in security algorithms. They proposed three *T-functions* based UMAPs: Lightweight Mutual Authentication Protocol (LMAP), Minimalist Mutual Authentication Protocol (M^2AP) and Extremely Lightweight Mutual Authentication Protocol (EMAP). All of the three UMAPs involve simple bitwise logical operations such as *AND*, *OR&XOR* in their designs and can be implemented within $1K$ logic gates. The basic functionalities of the protocols were also analyzed using some informal security models. However, Li and Wang (2007), Li et al. (2008) highlighted many issues of using *T-functions* in security protocols. They used the weak diffusion properties of *T-functions* to report simple full disclosure and desynchronization attacks on all of the three UMAPs. Henceforth, the concept of using *T-functions* in security protocol seemed to be insecure and impractical.

Chein (2007) improved the structure of *T-functions* based security algorithms. Instead of using simple bitwise operations, Chein incorporated a non-triangular primitive Rotation (*Rot*) in its SASI (Strong Authentication and Strong Integrity) protocol. The author classified the security protocol into four types and formally named such *T-functions* based security algorithms as "Ultralightweight Protocols". Although, the newly proposed non-triangular primitive "*Rot*" was extremely lightweight and robust but because of poor cipher text structure, the SASI protocol was also reported to be vulnerable against many traceability, desynchronization and full disclosure attacks (Avoine et al. 2010; Hernandez-Castro et al. 2008; Phan 2008).

Peris-Lopez et al. (2008) also acknowledged the suggestion of Chein and extended their previous work on the UMAPs. They proposed a new genetic programming based UMAP: GOASSMER in 2008. The GOASSMER protocol involves two bitwise operators (*Modulo-2addition&XOR*) and one non-triangular primitive (*MixBits*). The proposed UMAP proved to be resistive against all traceability, replay and full disclosure attacks, however failed to resist against DoS (Denial of Service) and desynchronization attack (Avoine et al. 2012; Bilal et al. 2009) models. Moreover, the authors didn't present any hardware implementation of their novel primitive; *MixBits*, which could justify whether or not this non-triangular primitive falls within ultralightweight class (Air et al. 2013).

Later, many other UMAPs (e.g. Yeh et al., David-Prasad and Lee et al. etc.) were proposed using either "*Rot*" function or simple bitwise operations and therefore found to be vulnerable against similar attack models (Ahmadian et al. 2013b; Avoine et al. 2012; Hernandez-Castro et al. 2010).

Tian et al. (2012) proposed a new Ultralightweight Primitive (UP); Permutation (*Per*) based UMAP named RAPP (RFID Authentication Protocol using Permutation). This newly proposed primitive (*Permutation)* was extremely lightweight and offered excellent confusion & diffusion properties. However, the only risk with *'Per'* function was that, it discloses the information of Hamming Weight (H_w) of the input string which actually needs to be concealed. Many cryptanalysts exploited this mathematical weakness of the permutation primitive and reported numerous attacks including full disclosure attacks (Ahmadian et al. 2013a). Because of this weakness, Tian et al. were unable to fully utilize the permutation function in their protocol design and

used "*Rot*" function for the computation of cipher text. The presence of two entirely different non-triangular primitives in one UMAP requires more resources and exceeds from the computational capabilities of the EPCC1G2 tag. In this paper, we improved the structure of permutation function and efficiently used it for computation of Cipher and Plain text on both sides (Reader & tag).

Recently, many UMAPs (Luo et al. 2018; Mujahid et al. 2015, 2017; Zhuang et al. 2014) e.g. RAPLT, RCIA, R^2AP, KMAP and SLAP have been proposed with various non-triangular primitives, however the cryptanalysis of almost all the existing ultralightweight authentication schemes highlights their serious security flaws and pitfalls. Therefore, in order to have a robust security protocol which resists against all possible attacks and can be implemented within $4K$ GE, some more research is required.

1.2 Contribution

To fill this research gap, in this paper, we propose a new UMAP; URMAP (Ultralightweight Resilient Mutual Authentication Protocol) using Per-XOR (P_x) and Inverse Per-XOR (P_x^{-1}). These primitives are inspired from the operators used in the Extremely Good Privacy (EGP) protocol proposed in (Khalid et al. 2019), however this protocol ensures enhanced confusion and diffusion capabilities of public messages. In the URMAP, both of these primitives have been fully utilized for the encryption and decryption and no other primitive (like RAPP used '*Rot*') has been used in protocol design.

1.3 Organization

The organization of the paper is as follows; Sect. 2 describes the basic working of novel UMAP (URMAP) which is followed by security and privacy analysis of URMAP over formal and some Adhoc security attack models in Sect. 3. Section 4 presents the comparative performance analysis and finally Sect. 5 concludes the paper.

2 Ultralightweight Resilient Mutual Authentication Protocol (URMAP)

The URMAP is a non-triangular UMAP that provides security and privacy to the low-cost communicating nodes. The protocol primitives used in URMAP are permutation ($P_x(a,b)$) and inverse permutation ($P_x^{-1}(d,b)$). The details of memory requirement, primitives and algorithm associated with the authentication scheme are as follows:

2.1 Memory architecture

The memory architecture designed for the URMAP ensures protection against all the existing desynchronization attacks. The basic idea of the architecture is to store dynamic variables form x latest authentication sessions on the resource sufficient communication end, making the execution time of denial of service attack infeasible for the adversary (Khalid and Mujahid 2017).

The '$\mathcal{R}$' maintains two buffers of size x (each) for every '$\mathcal{T}$'. These buffers (arrays) store x latest values of pseudonyms i.e. *IDS* and K respectively. The key feature of dynamic arrays at the $\mathcal{R}$'s side is non-replication i.e. latest value is stored in the buffer after every successful authentication and the buffer is updated on First In First Out (FIFO) basis. In addition to that, the *ID* of '$\mathcal{T}$' is stored in a read only register. In total, the memory allocation of the server for a single '$\mathcal{T}$' is $(2x+1)$ L bit registers (L being the size of *ID*).

The '$\mathcal{T}$' consists of five L bit registers. The register file associated with '$\mathcal{T}$' consists of read only tag's *ID* and read/write register stores pair of latest dynamic variables $[(IDS^{new}, IDS^{old}),(K^{new}, K^{old})]$. Table 1 represents the URMAP'S memory architecture.

2.2 UMAP protocol primitives

The UMAP protocol uses only one non-triangular primitive along with its inverse for mutual authentication i.e. Per-XOR ($P_x(a,b)$) and inverse Per-XOR ($P_x^{-1}(d,b)$). The working principle for both operators is as under:

2.2.1 Computation of Per-XOR function ($P_x(a,b)$)

The ($P_x(a,b)$) operation is computed in two steps:

(a) The bit positions of a are shuffled on the basis of b. The algorithm for shuffling is given in Fig. 1a.
(b) The resulting value is $XOR-ed$ with b to obtain the final output i.e. d.

Table 1 Memory architecture of the URMAP protocol

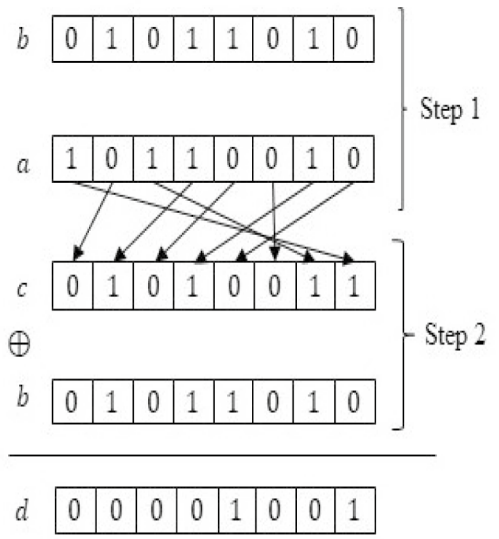
	Reader	Tag
Read only memory	*ID*	*ID*
Read/write memory	IDS, K $(IDS^j, IDS^{j+1}, \ldots IDS^i), (K^j, K^{j+1}, \ldots K^i)$	$IDS^{new}, IDS^{old}, K^{new}, K^{old}$

Algorithm 1: Algorithm for Per-XOR ($P_x(a,b)$)

Input: two n bit strings (a,b)
Output: n bit string d
Initialization:
1. $i \leftarrow 0, j \leftarrow (n-1)$
2. $ptr \leftarrow (n-1)$
   For Loop
3. **for** $m = 0$ to $(n-1)$ **do**
4. **if** $b[ptr] = 1$ **then**
5. $c[j] = a[ptr]$
6. $ptr = ptr - 1$
7. $j = j - 1$
8. **else**
9. $c[i] = a[ptr]$
10. $ptr = ptr - 1$
11. $i = i + 1$
12. **end for**
13. $d = c\ xor\ b$ (c: decimal version of binary sequence c[n])
14. return d

a: Per-XOR algorithm

8-bit example of Per-XOR ($P_x(a,b)$)



b: Per-XOR example

Algorithm 2: Algorithm for Inverse Per-XOR ($P_x{}^{-1}(d,b)$)

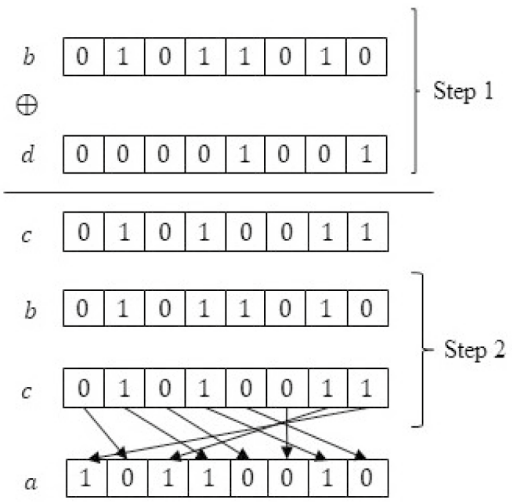
Input: two n bit strings (d,b)
Output: n bit string a

$$c = d\ xor\ b$$

Initialization:
1. $i \leftarrow 0, j \leftarrow (n-1)$
   $ptr \leftarrow (n-1)$
2. For Loop
3. **for** $m = 0$ to $(n-1)$ **do**
4. **if** $b[ptr] = 1$ **then**
5. $a[ptr] = c[j]$
6. $ptr = ptr - 1$
7. $j = j - 1$
8. **Else**
9. $a[ptr] = c[i]$
10. $ptr = ptr - 1$
11. $i = i + 1$
12. **end for**
13. return a

c: Inverse Per-XOR algorithm

8-bit example of Inverse Per-XOR ($P_x{}^{-1}(d,b)$)



d: Inverse Per-XOR example

Fig. 1 **a** Per-XOR algorithm. **b** Per-XOR example. **c** Inverse Per-XOR algorithm. **d** Inverse Per-XOR example

Figure 1b shows the working principle of *Per-XOR* computation with 8-*bit* length.

2.2.2 Computation of inverse Per-XOR function ($P_x^{-1}(d, b)$)

The working principle of the primitive is the exact mirror image of the Per-XOR function. This function is used to

decrypt the message with the help of a symmetric key used as second operand of both functions. The execution also takes place in two steps which are as follows:

(a) The operands are *XOR-ed* to obtain an intermediate value $c(c = dxorb)$.
(b) The bits of intermediate variable c are readjusted according to the algorithm given in Fig. 1c to obtain the final output a.

The Fig. 1d shows the mathematical description of Inverse Per-XOR (P_x^{-1}) function with 8-*bit* length.

2.2.3 Working of URMAP

The protocol can be divided into four main sub functions. The functional description of URMAP is as follows:

(a) The '$\mathcal{R}$' sends the request to the '$\mathcal{T}$' for pseudo-identification number *IDS* in order to retrieve relevant information from the '$\mathcal{D}$'. If the values associated with the '$\mathcal{T}$' (K, ID) are found in the memory, the '$\mathcal{T}$' is considered as identified otherwise the '$\mathcal{R}$' again sends the query for *IDS*.
(b) The mutual authentication requires two symmetric keys (n_1, n_2) which are generated by the '$\mathcal{R}$' and their encrypted versions $(A||B)$ are transmitted to the '$\mathcal{T}$'. The '$\mathcal{R}$' authentication query (C) is also transmitted to the '$\mathcal{T}$'. The mathematical expressions for the messages $A||B||C$ are:

$$A = P_x(n_1, K) \tag{1}$$

$$B = P_x(n_2, [P_x(n_1 \oplus K), n_1]) \tag{2}$$

$$K^* = P_x(n_1 \oplus n_2, [P_x(n_1 \oplus K), n_2]) \tag{3}$$

$$C = P_x(K^* \oplus n_1 \oplus n_2, [P_x(K^* \oplus n_2), (K \oplus n_1)]) \tag{4}$$

(a) The '$\mathcal{T}$' receives the message $(A||B||C)$, extract the symmetric keys,authenticates the '$\mathcal{R}$' and replies with message D for its own identity verification. The protocol proceeds to step (d) in case of successful authentication of both parties. The expressions for the evaluation of n_1, n_2 and message D are given as follows:

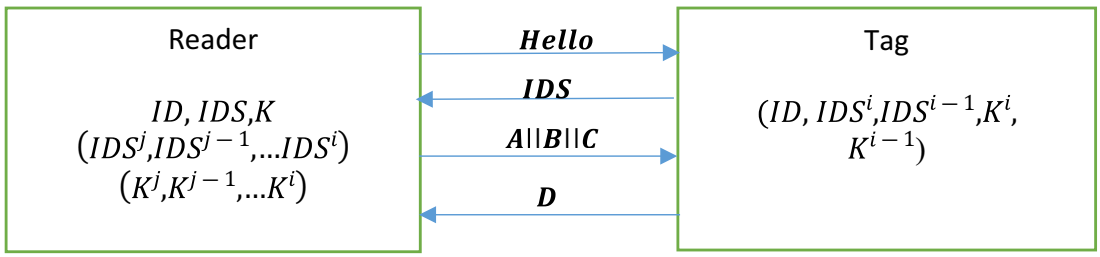
$$n_1 = P_x^{-1}(A, K) \tag{5}$$

$$n_2 = P_x{}^{-1}(B, [P_x(n_1 \oplus K), n_1]) \tag{6}$$

$$D = P_x(K^* \oplus ID \oplus n_1, [P_x(K \oplus n_2), (K^* \oplus n_1)]) \tag{7}$$

(b) In order to ensure freshness in the public messages of the next session, the dynamic variable of the '$\mathcal{T}$' are updated on both sides.

$$IDS^{old} = IDS, K^{old} = K \tag{8}$$



$A = P_x(n_1, K)$	$C = P_x(K^* \oplus n_1 \oplus n_2, [P_x(K^* \oplus n_2), (K \oplus n_1)])$
$B = P_x(n_2, [P_x(n_1 \oplus K), n_1])$	$D = P_x(K^* \oplus ID \oplus n_1, [P_x(K \oplus n_2), (K^* \oplus n_1)])$
$K^* = P_x(n_1 \oplus n_2, [P_x(n_1 \oplus K), n_2])$	$K^{i+1} = K^*$
$IDS^{i+1} = P_x(IDS \oplus n_1 \oplus n_2, P_x(n_1, n_2))$	

Fig. 2 Block diagram of URMAP

$$K^{new} = K^* \tag{9}$$

$$IDS^{new} = P_x\left(IDS \oplus n_1 \oplus n_2, P_x\left(n_1, n_2\right)\right) \tag{10}$$

The block diagram of the URMAP is presented in Fig. 2.

3 Security analysis of URMAP

The security and privacy of the URMAP is analyzed on the basis three features: Basic Functions (security parameters), response of the protocol against existing attacks and formal verification. The details of the security analysis are presented as follows:

3.1 Basic functionality

For basic functionalities, we verify the three main parameters "CIA" (Confidentiality, Integrity and Availability) of the protocol.

3.1.1 Confidentiality

The proposed protocol provides the optimal confidentiality by ciphering the secret variables within the public messages $(A, B, C\&D)$ using pre-shared key and two random nonce (n_1, n_2) for freshness of the each authentication session. Moreover, it is almost impossible for the malicious entity to retrieve the concealed secrets from public messages because the randomizing primitive 'P_x' (used in this protocol) increases the computational complexity significantly.

The computational complexity to retrieve tag's *ID* can be calculated as follows:

(a) In order to compute *ID,* the adversary calculate the n_1 by performing the P_x^{-1} of the message A. This step involves the computational complexity of $O\left(2^K \times log_2(K!)\right)$.
(b) Further, the adversary computes the second random nonce n_2 that doubles the computational complexity:$O\left(2 \times 2^K \times log_2(K!) \times log_2\left(n_1!\right)\right)$.
(c) Computation of K will make the computation complexity to $O(4 \times 2^K \times log_2(K!) \times log_2(n_1!)$.
(d) Computation of K^* will make the computation complexity to $O\left(8 \times 2^K \times log_2(K!) \times log_2\left(n_1! \times log_2\left(n_2!\right)\right)\right)$.
(e) Finally, the retrieval of secret *ID* will make the computational complexity:$O(16 \times 2^K \times log_2(K!) \times log_2\left(n_1!\right)$ $\times \log\left(n_2!\right) \times log_2(K^*!))$.

With this computational complexity, it will become impossible to retrieve the conjecture secret *ID*. Hence the URMAP ensures the confidentiality optimally.

3.1.2 Integrity

In URMAP, the cipher block chain concept ensures interdependencies among public messages (A, B, C, D) to ensure integrity. For instance, if an adversary tries to alter random nonce $\left(n_1\right)$ by tampering the message A, then effect of this tampering will also transfer to message B. Hence the legitimate tag will compute a different nonce $\left(n_2\right)$ and will be unable to verify message 'C'. With the incorporation of ultralightweight primitive 'P_x' and transitional value (K^*) in URMAP structure, it will be impossible for an adversary to calculate such value of message 'C' which will force the tag to authenticate that session. Therefore, any sort of modification in a valid session can't go undetected in URMAP.

3.1.3 Availability

The memory architecture of the URMAP is designed to avoid all the existing desynchronization attacks. The Denial of Service (DoS) attack presented in (Safkhani and Bagheri 2016) claims to desynchronize the '$\mathcal{T}$' and the '$\mathcal{R}$' that stores a pair of (IDS, Key) at both ends in five consecutive session. The basic technique of the attack is to force the '$\mathcal{T}$' and the '$\mathcal{R}$' to update different but valid pair of pseudonyms and keys and therefore making the identification of legitimate '$\mathcal{T}$' impossible.

The URMAP architecture maintains a buffer of size x at the '$\mathcal{R}$' side for each dynamic variable associated with the '$\mathcal{T}$'. At the end of each successful authentication session, the updated value of dynamic variable is added to the '$\mathcal{R}$' memory and the buffer is updated on FIFO basis. Whereas the '$\mathcal{T}$' side stores the latest pair of index pseudonym and key in the dynamic memory. The main aim of this design is to increase the number of sessions intervened by the adversary to launch a successful DoS attack thus making the execution infeasible. To desynchronize a system that has the buffer of size x, the adversary is required to interfere $x + 2$ sessions.

3.2 3.2 Security attack models

The security resilience of URMAP is evaluated against traceability and full disclosure (structural and adhoc) attack models. The detailed working is as follows:

3.2.1 Traceability attack

One of the primary security concerns is the location and movement privacy of the tag. The traceability attacks are designed to track the tag under attack at any time. Such security breaches can be fatal for the RFID based supply chain and inventory management systems, as a malicious user may monitor the mobility and location of the classified items. To

avoid the traceability attacks, the URMAP ensures the freshness of public messages by generating a set of random number (n_1, n_2). Furthermore the tag's dynamic variables (IDS, K) are restructured after every successful authentication to ensure the response of the tag to the reader's ping message unforeseeable for the adversary. Phan (2008) proposed a structured traceability model to formally verify the un-traceability claim of the authentication protocols. The model is described as follows:

3.2.1.1 Formal traceability model The Raphael formal traceability model is based on four assumptions (parameters) that refers to the computational capabilities of the adversary:

1. **Eavesdropping**: The adversary can listen and copy the public messages being communicated between the reader and the tags.
2. **Impersonation**: The adversary can assume the identity of both the reader $(\mathcal{R})$ and the tag $(\mathcal{T})$
3. **Tampering**: The adversary can modify the public messages (A, B, C, D) transmitted between the reader $(\mathcal{R})$ and the tag $(\mathcal{T})$
4. **Identification**: The malicious user can perform comparative analysis between different queries received from multiple tags to identify the individual tag.

The traceability model involves mainly three steps:

1. In this step, the adversary performs eavesdropping, impersonation and tampering to collect the all detailed information about the tags.
2. The malicious entity tries to make different sets of the collected information so that the tags can be differentiated.
3. Finally, the adversary tries to learn the relationship between tags secret ID and their publically disclosed messages by performing different operations between the public messages:

$$ID^i = A^i :: B^i :: C^i :: D^i \tag{11}$$

where '$::$' is the bitwise logical operator.

If $P_r\left(ID^i\right) > \frac{1}{2}$, then the adversary can successfully track the individual tag. However, in URMAP the optimal message structure of (A, B, C, D) makes that accurate ID estimation via exploiting the structural weaknesses of the protocol primitives impossible. Equation 12 defines the variables and their expression that need to be estimated accurately in order to execute successful traceability attack. The computation complexity to calculate Eq. (12a) is discussed in Sect. 3.1.1.

$$ID = K^* \oplus n_1 \oplus P_x^{-1}\left(D, [P_x\left(K \oplus n_2\right), \left(K^* \oplus n_1\right)]\right) \tag{12a}$$

$$n_1 = P_x^{-1}(A, K) \tag{12b}$$

$$n_2 = {P_x}^{-1}(B, [P_x\left(n_1 \oplus K\right), n_1]) \tag{12c}$$

Considering the computational complexity of Eq. 12a, it will not be feasible for a malicious entity to track and trace particular tag by resolving the equation. Moreover, the dynamic variables (IDS, K) and random numbers (n_1, n_2) are restructured after each identity verification session, to make the tag/reader communication of consecutive authentication sessions untraceable.

3.2.2 Replay attack

In the replay attack model the adversary has the computational capability to sniff the public messages and replay them at any given time without any alteration. The adversary's main objective is to gain access to the network by impersonating as a valid reader or tag. As a consequence, the adversary desynchronizes legitimate tag/reader pair by writing false data on either of the valid party's memory. However, the URMAP avoids all existing replay attacks since the messages involve two pseudorandom numbers (n_1, n_2) to ensure the diffusion of public messages. Moreover, the dynamic variable update after every successful authentication, adds freshness to the consecutive authentication session's public messages. Therefore, in URMAP the replaying of the legitimate captured messages will not make any difference in normal authentication sessions.

3.2.3 Man in the middle attack (MIMA)

The MIMA can be categorized into two type based on the adversary's computational capacity i.e. active and passive. In active MIMA attack, the adversary can sniff, modify, block and replay the public messages from a valid authentication session. In the passive attacks, the malicious user gathers the database of challenge and response messages from numerous identity verification session and tries to calculate the variables associated with the tag locally by exploiting the week diffusion capabilities of protocol's primitives. In the URMAP, the concept of block chain ensures robustness to the active MIMA attacks and the computational complexity evaluated in Sect. 3.1.1 ensures protection against passive attacks.

3.2.4 Impersonation attacks

There can be two possibilities of impersonation attacks; an adversary can impersonate as a reader or as a tag. Following is the description for both possible scenarios:

(a) *Reader impersonation* For the reader impersonation, the adversary sniffs the pseudo-identity (*IDS*) and reader's authentication challenge messages of valid authentication sessions. After establishing a database of eavesdropped messages, the adversary impersonates as a valid reader and forces the tag to communicate on the basis of *IDS* sniffed earlier. The main aim of the adversary is to desynchronize the tag/reader dynamic memory by overwriting the old pseudonym in the tag's memory. In URMAP, the reader maintains a buffer of x latest dynamic variables associated with each tag. This forces the attacker to block/replay $x + 2$ messages to execute the attack therefore, the size of the reader's buffer is directly proportional to the URMAP's resilience against the reader impersonation.

(b) *Tag impersonation* In the tag impersonation, the adversary gathers the pseudonyms and tag authentication messages. In this model, the adversary tries to get verified as a valid tag by replaying the valid tag authentication message eavesdropped from previous session. In URMAP, every session involves random numbers (n_1, n_2) and pseudonyms (IDS, K). These dynamic variables update in every session to ensure the freshness of the tag authentication challenge message of consecutive sessions. Hence due to dynamic variables n_1, n_2, IDS, K; tag impersonation attack is not feasible.

3.2.5 Full disclosure attack

The cryptanalysis of the existing UMAPs exploited the structural weakness of the ultralightweight primitives to estimate the values associated with the tag. The $T - functions$ exisibited very low diffusion capabilities resulting in full disclosure attacks. However, the inclusion of $non - Tfunctions$ such as *Recursivehash*, *Mixbit*, *PerXor* and *Rot* enhances the confidentiality of the public messages ensuring robustness against full disclosure attack.

Despite the protocol specific adhoc attacks, the structured full disclosure attack models i.e. Tango attack (Hernandez-Castro et al. 2010), Recursive Differential Cryptanalysis (RDC) and Recursive Linear Cryptanalysis (RLC) (Ahmadian et al. 2013b) are applied to UMAPs to systematically evaluate the confusion and diffusion capabilities of the public messages. The security analysis of URMAP over these structural models is as follows:

3.2.6 Tango attack

The main purpose of tango cryptanalysis is to identify hidden values corresponding to the tag by making use of the low diffusion properties of public messages. The tango attack can be separated into two parts i.e. *ID* approximation equations derivation and *ID* estimates for the tag. The description of these steps are outlined as follows:

(a) The triangular combinations (XOR, AND, OR) of challenge & response messages (A, B, C, D) are selected in this step. An equation is marked as *ID* approximation equation if the hamming difference between the equation and the *ID* of the tag is less than half the length of the number of identification bits $(L/2)$.

(b) The static *ID* number of any tag enforcing the protocol under observation may be determined after obtaining a set of X *ID* approximation equations. The procedure is as follows:

(i) Sniff Y valid identity verification session among the $\mathcal{R}$ and the $\mathcal{T}$.

(ii) Evaluate *ID* approximation equation for each snooped session.

(iii) As a final step, *ID* approximation equation are analyzed bit by bit. If number of one's at a certain bit position of approximation equations is greater than $\boldsymbol{\beta}$, one is placed as *ID* estimation of bit position under consideration, If number of $1s$ are less than $\boldsymbol{\beta}$, zero is estimated. The equation for $\boldsymbol{\beta}$ is as follows:

$$\beta = 0.5 \times X \times Y \tag{13}$$

By repeating this method for each bit location, estimation of tag identification number can be found.

The tango attack has been proven less effective against *non-Tfunctions* based UMAPs. Moreover, in URMAP the Per-Xor function is used extensively to ensure the random shuffling of the public message strings. Therefore, the random shuffling ensures the diffusion; making the derivation of good approximation equations almost impossible.

3.2.7 Recursive linear (RLC) and recursive differential (RDC) cryptanalysis

The recursive linear/differential cryptanalysis are two symmetric full disclosure techniques proposed by Zahra Ahmedian. The working principle of both models are based on defining the linearized version of A, B, C, D in terms of tag associated variables (dynamic and static).This step facilitates the derivation of tag identifiers through simultaneous equation solution. The only difference in RLC and RDC is defined by the relationship between number of public messages (N_{pm}) per authentication session and number of tag variables (N_{tv}). If $N_{pm} \geq N_{tv}$,the RLC is used for full disclosure attack otherwise RDC is executed. In RDC, the adversary acts in an active manner to find the conjucture tag's *ID*.

In URMAP, the use of nested *PerXor* function makes the linearization of public message almost infeasible. Thus making the attack model unfit for the presented protocol. The inherent limitation of RDC and RLC towards the non-Triangular UMAPs has also been highlighted by Zahra Ahmadin.

3.3 Formal security analysis

The formal security analysis evaluates the soundness and viability of the authentication approach. This method evaluates the structure of public messages and sequence of communication among the communicating parties to evaluate the protocol's functionality on an abstract level. The aspects of the protocol evaluated are as follows:

- The authentication mechanism
- The protocol's output
- The adequacy of basic assumption
- The redundancy analysis

Numerous methods are available in literature for the evaluation of authentication logic (Burrows et al. 1989; Morris 1996; Van Ditmarsch et al. 2015). The NIST-approved methods, which are widely being used are BAN and GNY logic. Both models analyze the protocol step by step and make explicit assumptions required to draw conclusion about the final position it attains. The inclusion of BAN or GNY logic evaluation in our security model will ensure systematic evaluation of the authentication mechanism at the design stage. In this paper, the GNY Logic analysis is used for systemic formal analysis of URMAP.

3.3.1 GNY logic analysis

GNY (Gong–Needham–Yahalom) logic is a systematic mathematical cryptanalysis tool that verifies the security and privacy assumptions of the authentication protocols (Brackin 1996). The GNY analysis is a muti-phase process that initially defines the assumptions and public messages (IDS, A, B, C, D) of the algorithm in an abstract language and then start verifying the defined goals. The rules used for goal verification are; Being Informed, Possession and Freshness Rules.

(a) **Being informed rules**: Any formula that $\mathcal{H}$ receives is considered as "being informed".

$\mathcal{I}_1$: If $\mathcal{H}$ is informed the formula $\mathcal{N}$, which he didn't convey in this run, then $\mathcal{H}$ is informed $\mathcal{N}$

$$\mathcal{I}_1 : \frac{\mathcal{H} \lhd * \mathcal{M}}{\mathcal{H} \lhd \mathcal{M}}$$

$\mathcal{I}_2$: If $\mathcal{H}$ is informed an encrypted formula with symmetric key (K) then $\mathcal{H}$ is informed the formula.

$$\mathcal{I}_2 : \frac{\mathcal{H} \lhd \{\mathcal{M}\}_K, \mathcal{H} \ni \mathcal{M}}{\mathcal{H} \lhd \mathcal{M}}$$

(b) **Possession rules**: If $\mathcal{H}$ possesses a formula then it can possess other associated formulae as well.

$\mathcal{P}_1$: $\mathcal{H}$ can possess any variable which it is being informed

$$\mathcal{P}_1 : \frac{\mathcal{H} \lhd \mathcal{M}}{\mathcal{H} \ni \mathcal{M}}$$

$\mathcal{P}_2$: If $\mathcal{M}$ possess two different formulae then it can possess their concatenation and functions as well.

Table 2 Notations used in GNY logic analysis

S.No	Statement/notations	Interpretation
1	$\mathcal{H}$	Manager/administrator
2	$\mathcal{M}$	Message/variable
3	$\mathcal{R}$	Reader
4	$\mathcal{T}$	Tag
5	ζ	Intermediate step
6	$\mathcal{H} \lhd * \mathcal{M}$	$\mathcal{H}$ is informed $\mathcal{M}$, $\mathcal{H}$ didn't convey in current session
7	$\mathcal{H} \lhd \mathcal{M}$	$\mathcal{H}$ is informed $\mathcal{M}$
8	$(\mathcal{M} \parallel \mathcal{O})$	Concatenation of $\mathcal{M}\&\mathcal{O}$
9	$\mathcal{H}\mid \equiv \#(\mathcal{M})$.	$\mathcal{H}$ believes that $\mathcal{M}$ is fresh
10	$\mathcal{H}\mid \equiv \varphi(\mathcal{M})$	$\mathcal{H}$. believes that $\mathcal{M}$ is computable/recognizable
11	$\mathcal{H}\mid \equiv \mathcal{M}$	$\mathcal{H}$ believes that $\mathcal{M}$ holds
12	$\mathcal{H}\mid \equiv \mathcal{H} \overset{K}{\leftrightarrow} \mathcal{Z}$	$\mathcal{H}$ believes that K is an appropriate secret for $\mathcal{Z}$
13	$\mathcal{H} \ni \mathcal{M}$	$\mathcal{H}$ can possess $\mathcal{M}$
14	$\mathcal{H}\mid \sim \mathcal{M}$	$\mathcal{H}$ once conveyed $\mathcal{M}$
15	$\{\mathcal{M}\}_K$	$\mathcal{M}$ is encrypted using symmetric key K

$$\mathcal{P}_2 : \frac{\mathcal{H} \ni \mathcal{N}, \mathcal{H} \ni \mathcal{O}}{\mathcal{H} \ni (\mathcal{N}, \mathcal{O}), \mathcal{H} \ni \mathcal{F}(\mathcal{N}, \mathcal{O})}$$

Table 2 enlists the notations used in GNY logic analysis.

(c) **Freshness rules**: On the basis of belief, $\mathcal{H}$ determines the freshness of messages.

$\mathcal{F}_1$: If $\mathcal{H}$ believes that a formula is fresh then he also believes that any concatenation and function will be fresh.

$$\mathcal{F}_1 : \frac{\mathcal{H}| \equiv \#(\mathcal{M})}{\mathcal{H}| \equiv \#(\mathcal{M}, \mathcal{O}), \mathcal{H}| \equiv \#\mathcal{F}(\mathcal{M}, \mathcal{O})}$$

3.3.2 GNY logic based formal proof of URMAP:

The implementation of GNY logic is based on three steps i.e. abstraction of URMAP's assumptions, formalization of public messages and URMAP's goal verification by virtue of formal analysis postulates.

The mutual authentication logic mainly depends upon the pseudo-random numbers (n_1, n_2) therefore we apply analysis on first two messages only.

The messages can be formulated as follows:

$$\mathcal{T} \lhd * \left\{n_1\right\}_K \sim> \mathcal{R}| \equiv \mathcal{T} \overset{K}{\leftrightarrow} \mathcal{R}$$

$$\mathcal{T} \lhd * \left\{n_2\right\}_{n_1,K} \sim> \mathcal{R}| \equiv \mathcal{T} \overset{n_1,K}{\leftrightarrow} \mathcal{R}$$

The goal of sending such messages is:

$$\mathcal{G}_1 : \mathcal{T}| \equiv \mathcal{R} \ni (n_1, n_2)$$

By applying the verification postulates, we can validate URMAP's goal as follows:

$$\mathcal{T} \lhd * \left\{n_1\right\}_K \sim> \mathcal{R}| \equiv \mathcal{T} \overset{K}{\leftrightarrow} \mathcal{R}$$

$\zeta_1 : \mathcal{T} \lhd * \left\{n_1\right\}_K$ (Since $\mathcal{R}| \equiv \mathcal{T} \overset{K}{\leftrightarrow} \mathcal{R}$ is already assumed to be satisfied)

By considering $\mathcal{I}_1, \mathcal{S}_4$ and $\mathcal{I}_2$ we can have

$$\zeta_2 : \mathcal{T} \lhd \left\{n_1\right\}_K$$

$$\zeta_2 : \mathcal{T} \lhd n_1$$

Based on rule $\mathcal{P}_1$, we get

$$\zeta_3 : \mathcal{T} \ni n_1$$

Now for the n_2 encryption message:

$$\mathcal{T} \lhd * \left\{n_2\right\}_{n_1,K} \sim> \mathcal{R}| \equiv \mathcal{T} \overset{n_1,K}{\leftrightarrow} \mathcal{R}$$

$\zeta_4 : \mathcal{T} \lhd * \left\{n_2\right\}_{n_1,K}$ (Since ζ_3 : and $\mathcal{R}| \equiv \mathcal{T} \overset{K}{\leftrightarrow} \mathcal{R}$ is already assumed to be satisfied)

If we consider $\mathcal{I}_1, \mathcal{S}_4$ and $\mathcal{I}_2$ then ζ_4 : can be written as:

$$\zeta_5 :: \mathcal{T} \lhd \left\{n_2\right\}_{n_1,K}$$

$$\zeta_7 :: \mathcal{T} \lhd n_2$$

Further ζ_6 : can be inferred as:

$$\zeta_7 :: \mathcal{T} \ni n_2$$

Hence from ζ_3 and ζ_7, it can be proved that URMAP optimally achieve its goal:

$$\mathcal{T}| \equiv \mathcal{R} \ni (n_1, n_2)$$

If a malicious entity attempts to modify n_1, then effect of this tampering directly transfers to n_2 as well. The tag will not verify the reader authentication challenge message C, hence abort such protocol session and will remain synchronized.

4 Performance comparison

This section presents the performance analysis of URMAP in terms of computational operations, communication cost, resistance against security attacks (DoS, Traceability and Full Disclosure), conformance to EPC C1G2 tags and status of formal verification. Table 3 describes the detailed comparison.

At the tag side, the URMAP involves only two bitwise logical operations; *XOR* and *Per-XOR*. The *Per-XOR* operation comprises two functions, *shifting* of bit locations and *XORing*. These operations are extremely lightweight operations and don't require excessive GEs (Gate equivalents) at tag side. Regarding memory storage at the tag side, it requires $5L$ of memory registers to store $1L$ of static and $4L$ of rewriteable memory (*KeysandIDS*). Furthermore, to store the transition values (like random nonce and intermediate results), tag also requires $2L$ memory. Almost most of the previous UMAP designers have ignored this point and just mentioned the main memory size (*ID*, *IDS&keys*).

The communication cost of URMAP is also very minimal, since in URMAP, the tag has to transmit only two messages ($2L$). As far as security is concerned, the URMAP protocol proves to be robust against all existing cryptanalysis techniques. None of the previous UMAPs (discussed here) completely satisfies three basic security attributes

Table 3 Performance comparison of UMAPs

Protocols parameter	SASI (Chien 2007)	GOASSMER (Peris-Lopez et al. 2008)	RAPP (Tian et al. 2012)	RCIA (Mujahid et al. 2015)	SLAP (Luo et al. 2018)	URMAP
Computational operations	$\oplus, OR, AND, Rot, +$	$\oplus, Rot, +, MixBits$	$\oplus, Rot, Per$	$\oplus, Rot, AND$	$\oplus, Rot, Conv(x, y)$	$\oplus, Per\text{-}XOR$
Memory requirements	$7L$	$7L$	$5L$	$7L$	$7L$	$5L$
Communication cost	$2L$	$2L$	$2L$	$2L$	$1.5L$	$2L$
DoS attack resilience	No	No	No	No	No	Yes
Traceability resilience	No	No	No	No	No	Yes
Full disclosure resilience	No	Yes	No	Yes	Yes	Yes
Replay attack resilience	No	No	No	Yes	Yes	Yes
Man in the middle attack resilience	No	Yes	No	Yes	Yes	Yes
Impersonation attack resilience	No	No	No	No	No	Yes
Conformance to EPC C1G2 tags[a]	Yes	No	Yes	Yes	Partial	Yes
Formally verified	No	No	No	No	Yes	Yes

[a]For 32-bit only

(Confidentiality, Integrity and Availability) which make them unsuitable for real world applications. On the other hand, the URMAP protocol can withstand against all types of existing adversarial models.

The foremost important parameter that defines the practical suitability of these protocols with low cost tag is their hardware requirements. Since, the URMAP involves extremely lightweight operations; *XOR* and *Per-XOR* (where later requires a simple shifting and former is a bitwise logical operation) therefore can successfully conform to EPC C1G2 tags (requires approx. $4K$ GE for 32-*bit* word length).

Another permanence of URMAP over existing UMAPs is formal verification. Most of the existing UMAPs are not formally verified and hence vulnerable to even simplest attack model. Therefore, it is highly recommended that logic of the security protocol must be verified through either GNY logic or BAN logic models.

5 Conclusion

In this paper, a mutual authentication protocol for Class 1 Generation 2 (C1G2) RFID system titled URMAP (Ultralightweight Resilient Mutual Authentication Protocol) is presented. The proposed protocol is *non-triangular* in nature with two resource effective functions i.e. *Per-XOR* and *XOR*. The security and privacy services of the protocol has been evaluated by systematic formal and functional analysis. The performance comparison of the URMAP shows that it outperforms the contending UMAPs in terms of security, cost effectiveness and computational operations. These features make URMAP an appropriate choice for passive IoT based edge devices and RFID tags.

References

Ahmadian Z, Salmasizadeh M, Aref MR (2013a) Desynchronization attack on RAPP ultralightweight authentication protocol. Inf Process Lett 113:205–209

Ahmadian Z, Salmasizadeh M, Aref MR (2013b) Recursive linear and differential cryptanalysis of ultralightweight authentication protocols. IEEE Trans Inf Forensics Secur 8:1140–1151

Air R, Protocol I, Version M (2013) EPC TM radio-frequency identity protocols generation-2 UHF RFID specification for RFID air interface

Avoine G, Carpent X, Martin B (2010) Strong authentication and strong integrity (SASI) is not that strong. In: International workshop on radio frequency identification: security and privacy issues, 2010. Springer, pp 50–64

Avoine G, Carpent X, Martin BJ (2012) Privacy-friendly synchronized ultralightweight authentication protocols in the storm. J Netw Comput Appl 35:826–843

Bilal Z, Masood A, Kausar F (2009) Security analysis of ultra-lightweight cryptographic protocol for low-cost RFID tags: Gossamer protocol. In: 2009 international conference on network-based information systems, 2009. IEEE, pp 260–267

Brackin S (1996) Automatic formal analyses of cryptographic protocols. In: Proceedings of the 19th national conference on information systems security, 1996

Burrows M, Abadi M, Needham RM (1989) A logic of authentication. Proc R Soc Lond A Math Phys Sci 426:233–271

Chien H-Y (2007) SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. IEEE Trans Depend Secur Comput 4:337–340

Hernandez-Castro JC, Tapiador JME, Peris-Lopez P, Quisquater J-J (2008) Cryptanalysis of the SASI ultralightweight RFID authentication protocol with modular rotations. arXiv:0811.4257

Hernandez-Castro JC, Peris-Lopez P, Phan RC-W, Tapiador JM (2010) Cryptanalysis of the David-Prasad RFID ultralightweight authentication protocol. In: International workshop on radio frequency identification: security and privacy issues, 2010. Springer, pp 22–34

Khalid M, Mujahid U (2017) Security framework of ultralightweight mutual authentication protocols for low cost RFID tags. In: Communication, computing and digital systems (C-CODE), international conference on, 2017. IEEE, pp 26–31

Khalid M, Mujahid U, Najam-ul-Islam M (2019) Ultralightweight RFID authentication protocols for low-cost passive RFID tags. Secur Commun Netw 2019:3295616

Lee Y-C, Hsieh Y-C, You P-S, Chen T-C (2009) A new ultralightweight RFID protocol with mutual authentication. In: 2009 WASE international conference on information engineering, 2009. IEEE, pp 58–61

Li T, Wang G (2007) Security analysis of two ultra-lightweight RFID authentication protocols. In: IFIP international information security conference, 2007. Springer, pp 109–120

Li T, Wang G, Deng RH (2008) Security analysis on a family of ultralightweight RFID authentication protocols. JSW 3:1–10

Luo H, Wen G, Su J, Huang ZJ (2018) SLAP: succinct and lightweight authentication protocol for low-cost RFID system. Wirel Netw 24:69–78

Morris S (1996) The logic of belief and belief change: a decision theoretic approach. J Econ Theory 69:1–23

Mujahid U, Najam-ul-Islam M, Shami MAJ (2015) RCIA: a new ultralightweight RFID authentication protocol using recursive hash. Int J Distrib Sens Netw 11:642180

Mujahid U, Najam-ul-Islam M, Sarwar SJ (2017) A new ultralightweight RFID authentication protocol for passive low cost tags: KMAP. Wirel Pers Commun 94:725–744

Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador JM, Ribagorda A (2006a) EMAP: an efficient mutual-authentication protocol for low-cost RFID tags. In: OTM confederated international conferences "On the Move to Meaningful Internet Systems", 2006. Springer, pp 352–361

Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador JM, Ribagorda A (2006b) M^2AP: a minimalist mutual-authentication protocol for low-cost RFID tags. In: International conference on ubiquitous intelligence and computing, 2006. Springer, pp 912–923

Peris-Lopez P, Hernandez-Castro JC, Estévez-Tapiador JM, Ribagorda A (2006c) LMAP: a real lightweight mutual authentication protocol for low-cost RFID tags. In: Proc. of 2nd workshop on RFID security, 2006.

Peris-Lopez P, Hernandez-Castro JC, Tapiador JM, Ribagorda A (2008) Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol. In: International workshop on information security applications, 2008. Springer, pp 56–68

Phan RC (2008) Cryptanalysis of a new ultralightweight RFID authentication protocol—SASI. IEEE Trans Depend Secur Comput 6:316–320

Safkhani M, Bagheri N (2016) Generalized desynchronization attack on UMAP: application to RCIA, KMAP, SLAP and SASI+ protocols IACR cryptology. ePrint Archive 2016:905

Tian Y, Chen G, Li J (2012) A new ultralightweight RFID authentication protocol with permutation. IEEE Commun Lett 16:702–705

Van Ditmarsch H, Halpern JY, Van Der Hoek W, Kooi B (2015) An introduction to logics of knowledge and belief. arXiv preprint arXiv:1503.00806

Yeh K-H, Lo N, Winata E (2010) An efficient ultralightweight authentication protocol for RFID systems. Proc RFIDSec Asia 10:49–60

Zhuang X, Zhu Y, Chang C-CJ (2014) A new ultralightweight RFID protocol for low-cost tags: R2AP. Wirel Pers Commun 79:1787–1802