Security Framework of Ultralightweight Mutual Authentication Protocols for Low Cost RFID tags

Madiha Khalid
Department of Electrical Engineering
Bahria University, Islamabad
Email: madiha.khalid88@gmail.com

Umar Mujahid
Department of Cyber Security
Gwinnett-Tech, Alpharetta, Georgia 30005
Email: ukhokhar@gwinnetttech.edu

Abstract—**RFID is one of the rapidly growing identification schemes. Unique identification, non-line of sight capability and functional haste result its massive deployment in many supply chain applications. Because of limited computational capabilities at tag side Ultra lightweight protocols are the only solution to ensure secure communication in RFID systems. In this paper we have presented a detailed working of three eminent UMAPs. Further, a generic desynchronization attack model on these UMAPs has been discussed. Then finally, a novel security framework for low cost UMAPs has been presented to avoid all possible desynchronization attacks. The proposed security model is generic in nature therefore can be implemented to all upcoming UMAPs as well.**

1. Introduction

The RFID systems are one of the widely deployed identification systems in the field of ubiquitous computing. The main advantage of RFID tags over bar codes and magnetic tapes is non-line of sight requirement for scanning. There are three main components of RFID network; RFID Tags, a Reader and Database. Tag on entering the vicinity of reader communicates with it on wireless channel. Database provides reader information of tag for identification [1]. Applications of network includes identification, tracking and inventory of item to which tags are attached.

Massive deployment of RFID systems is mainly limited by confidentiality issues. If the communication channel between reader and tag is not secure, then based on the information being exchanged a third party can invade tag′s location and tracking privacy. Modern ciphers using hash functions and random number generators despite being effective, cannot be implemented, as low cost passive RFID systems have limited power and computational resources. A typical systems can only store few hundreds of bits and have 5K to 10K logic gates available, but only 250 to 3K logic gates can be used for implementation of security protocol[2],[3]. Ultralight weight Mutual Authentication Protocol (UMAP) provide the only solution to the problem. In this class of protocol only bit wise operations like XOR, AND, OR and addition mod 2^m are used.The resources consumed by UMAPs are within the 3K logic gates limit. Three main types of algorithms falling under the umbrella of UMAPs using triangular functions are i.e. Lightweight Mutual Authentication Protocol (LMAP), Minimalistic Mutual Authentication Protocol (M^2AP) and Efficient Mutual Authentication Protocol (EMAP) [2], [4], [5] .UMAPs provide extremely low security due to the use of imbalance functions (AND,OR) but the resources being consumed are minimum.

In order to improve the security, UMAPs are evolving with time. There is an elaborated list of UMAPs using ‘Non-Triangular Ultralightweight primitives’ (rotation, recursive hash, permutation) which includes Strong Authentication Strong Integrity Protocol (SASI), Robust Confidentiality, Integrity, and Authentication (RCIA) Protocol and Succinct and Lightweight Authentication Protocol (SLAP) [3], [6], [7]. This class of protocols provide comparatively better security than UMAPs using T-functions.

Desynchronization Attack is one of the most common security threat to tags. Under this attack, tag is deactivated for the valid reader and hence cannot be identified or tracked. In this paper, first we will discuss some major non triangular UMAPs, and then propose a generic security patch for protocols to avoid desynchronization attack. Security analysis of this patch with respect to SASI, RCIA and SLAP will also be presented.

The organization of paper is as follows: In Section 2, we will discuss eminent UMAPs followed by desynchronization attacks on these protocols in Section3. In Section 4, a novel security patch to avoid this security threat is presented. Finally conclusion has been discussed in Section 5.

2. Ultralightweight Mutual Authentication Protocols (UMAPs)

UMAPs using T-functions are more prone to desynchronization attacks. Recent UMAPs use non triangular functions such as rotation, recursive hash and conversion function [3],[6],[7]. These are relatively stronger ciphers but desynchronization attack is still very common. In all protocols to be discussed we will consider following assumption:

1. UMAPs are applied on communication channel between reader and tag.
2. Length of all pseudonyms, identifiers and keys is 96 bits as per EPC global standard [2.25].
3. Rot (x,y) used in protocols is defined by cyclic left rotation of x by hamming weight of y.

2.1. SASI

In SASI protocol, reader has access to index pseudonym (IDS), two keys (K_1/K_2) and ID associated with tag. Tag stores all the variables that reader has as $(IDS_n, K_{1n}, K_{2n}, ID)$. It also keeps a copy of previous pseudonyms and keys as (IDS_o, K_{1o}, K_{2o}). SASI protocol executes in three steps [3]:

I. **Tag Identification:** When Tag enters the vicinity of reader, it receives a "$Hello''$ message. Tag replies with its IDS_n. Reader searches for this value in the database. If match is found, protocol proceeds to next step. Otherwise IDS is enquired again by reader. This time tag sends IDS_o. In case of mismatch at reader end protocol is terminated.

II. **Mutual Authentication:** Once tag is identified, reader generates two random numbers n_1 and n_2. With the help of these numbers following values are calculated and $A\|B\|C$ is transmitted to the reader.

$$A = IDS \oplus K_1 \oplus n_1$$
$$B = (IDS \vee K_2) + n_2$$
$$\overline{K_1} = rot(K_1 \oplus n_2, K_1)$$
$$\overline{K_2} = rot(K_2 \oplus n_1, K_2)$$
$$C = (K_1 \oplus \overline{K_2}) + (\overline{K_1} \oplus K_2)$$

Tag extracts value of n_1 and n_2 from A and B respectively and calculates C locally (C'). If $C' = C$, reader is authenticated else the protocol is terminated. After reader authentication, tag send message D. The protocol proceeds to third step only when D is equal to locally calculated value $D(D')$ at reader end.

$$D = (\overline{K_2} + ID) \oplus ((K_1 \oplus K_2) \vee \overline{K_1})$$

III. **Pseudonym and Key Updating:** In the last stage of SASI, the values related to tag are updated on both sides. On the tag side following computation is performed.

$$IDS_o = IDS_n, K_{1o} = K_{1n}, K_{2o} = K_{2n}$$
$$IDS_n = (IDS + ID) \oplus (n_2 \oplus \overline{K_1})), K_{1n} = \overline{K1}$$
$$K_{2n} = \overline{K_2}$$

Finally the reader is updated according to following equation.

$$IDS = (IDS + ID) \oplus (n_2 \oplus \overline{K_1})), K_1 = \overline{K_1}, K_2 = \overline{K_2}$$

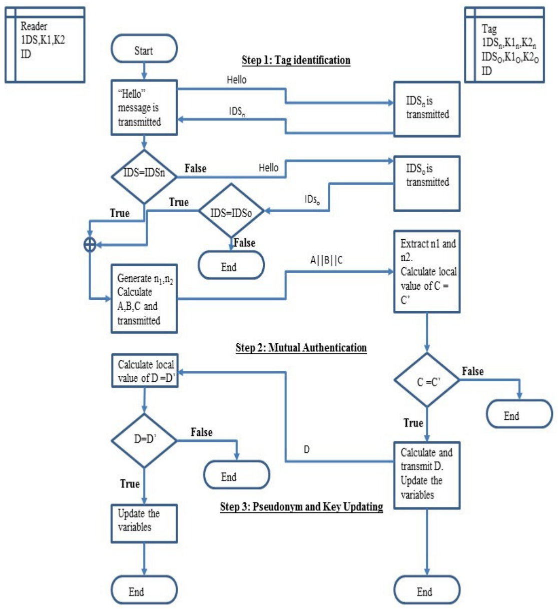
Reader
1DS,K1,K2
ID
Start
Step 1: Tag identification
Tag
$1DS_n,K1_n,K2_n$
$IDS_O,K1_O,K2_O$
ID
Hello
"Hello" message is transmitted
IDS_n
IDS_n is transmitted
IDS=IDSn
False
Hello
IDS_o is transmitted
True
True
IDS=IDSo
IDs_o
False
End
Extract n1 and n2. Calculate local value of C = C'
A||B||C
Generate n_1,n_2 Calculate A,B,C and transmitted
Step 2: Mutual Authentication
Calculate local value of D =D'
C =C'
False
End
True
D=D'
False
End
D
Calculate and transmit D. Update the variables
True
Update the variables
Step 3: Pseudonym and Key Updating
End
End

Figure 1. Working Of SASI Protocol

TABLE 1. Steps For Recurrsive Hash

Step 1: Division in *l* bit Chunks
Group *l* bits of variable to form K (K= *n/l* =8) chunks of data.
Step 2: Calculation of Seed
$Seed = I = wt(R) mod K$
Step 3: XOR and Rotation operation
(i) Calculate XOR of K_{Ith} block with all other block except for itself (ii) $Rot(K_I, K_I)$ Concatenating all the blocks output will form the result of recursive hash

SASI Protocol execution steps are presented in Figure 1.

2.2. RCIA

Robust Confidentiality, Integrity and Authentication (RICA) [6] protocol offers confusion and diffusion feature in data to be transmitted. This ability in protocol is due to a new primitive called recursive hash $(R_h(X))$. Recursive hash (R_h) of any n bit variable X is elaborated in Table 1.
Inputs required for the function.
Input binary sequence $X = x_n x_{(n-1)} x_{(n-2)} .. x_1$
Number of bit in variable = *n* = 96 bits
Number of bits per chunk = *l* = 12 bit
$R = n_1 \oplus n_2$ (where n_1 and n_2 are the random numbers generated by the reader)

RCIA executes in following steps:

I. **Tag identification:** Reader sends "$Hello''$ message to the tag. Tag replies with its latest index pseudonym

IDS_n. If reader finds data against IDS_n, protocol moves towards authentication phase otherwise tag send IDS_o to reader on second request for index pseudonym. If match is not found in database protocol terminates.

II. **Mutual Authentication:** In this step, reader generates two random numbers n_1 and n_2. By using these numbers and keys corresponding to accepted IDS following variables are calculated.

$$A = Rot(IDS, K_1) \oplus n_1$$
$$B = (Rot(IDS \wedge n_1, K_2) \wedge K_1) \oplus n_2$$
$$R = n_1 \oplus n_2$$
$$K_1^* = Rot(R_h(K_2), R_h(n_1)) \wedge K_1$$
$$K_2^* = Rot(R_h(K_1), R_h(n_2)) \wedge K_2$$
$$C = Rot(R_h(K_1^*), R_h(K_2^*)) \wedge Rot(R_h(n_1), R_h(n_2))$$

Message transmitted to tag is $A\|B\|C$. A and B are used to extract random numbers n_1 and n_2. C is calculated at tag (C') locally. If $C' = C$, reader is authenticated and D message is sent to reader. If C' is not equal to received value, protocol end.

$$D = (Rot(R_h(ID), K_1^*) \wedge (Rot(R_h(K_2^*), R_h(n_2)) \oplus IDS)$$

Local value of $D(D')$ is calculated at reader side for tag authentication. After successful authentications, protocol proceeds to next step.

III. **Variable Updating:**Values of index pseudonyms and keys are updated according to given equation on both sides.

$$IDS_o = IDS_n, K_{1o} = K_{1n}, K_{2o} = K_{2n}$$
$$IDS_n = Rot(R_h(IDS) \oplus n_2, n_1), K_{1n} = K_1^*, K_{2n} = K_2^*$$

2.3. SLAP

Succinct and Lightweight Authentication Protocol (SLAP) [7] aims to provide enhanced diffusion in exchanged messages between reader and tag. The protocol uses a new primitive called conversion $(Con(X, Y))$. Four major properties of this function are: Irreversibility, Sensibility, Full confusion and Diffusion. Given that X and Y are n bit numbers, definition of $Con(X, Y)$ given in Table 2. The description of input to function is as follows.
$X = x_n x_{(n-1)} x_{(n-2)} .x_1$
$Y = y_n y_{(n-1)} y_{(n-2)} y_1$
Threshold (maximum size of grouped bits) = T

SLAP working can be divided in two parts.

I. **Tag Identification:**Reader enquiries the index pseudonym from tag by sending "$Hello''$ message. Tag replies the reader with IDS_n . If this value is not present at the server, reader resends "$Hello''$ message. This time tag sends IDS_o . If record is found against IDS_o, protocol move towards mutual authentication phase otherwise it gets terminated.

II. **Mutual Authentication and IDS, K1, K2 Updating:**Reader generates a random number n. By using K_1 and K_2 associated with index pseudonym, reader calculates A and B.

$$A = Con(K_1, K_2) \oplus n$$
$$B = Con(Rot(K_1, n), K_1 \oplus K_2) \oplus Rot(Con(K_2, K_2 \oplus n), K_1)$$

B is then divided into two equal segments (B_l, B_r). If hamming weight of B is odd, data transmitted to tag is $A\|B_l$ otherwise $A\|B_r$ is transmitted.
Random number is extracted from received message A. Tag verifies the authenticity of reader by calculating local value of B_l or B_r and comparing it with received value. After reader authentication, tag updates its parameters and calculates C.

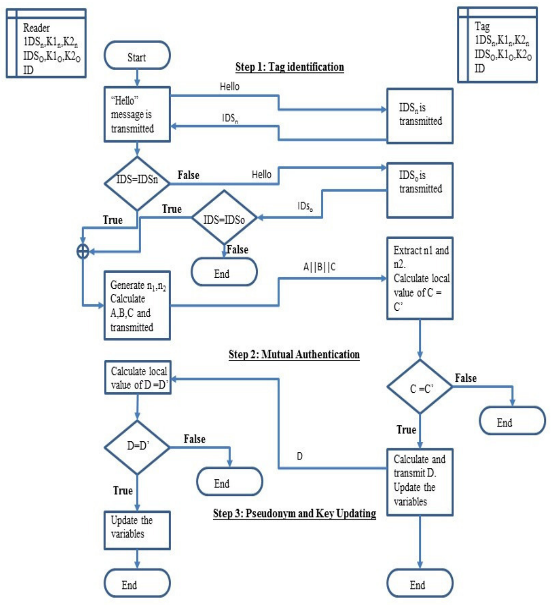
Reader
$IDS_n, K1_n, K2_n$
$IDS_O, K1_O, K2_O$
ID
Tag
$IDS_n, K1_n, K2_n$
$IDS_O, K1_O, K2_O$
ID
Start
Step 1: Tag identification
"Hello" message is transmitted
Hello
IDS_n
IDS_n is transmitted
IDS=IDSn
False
Hello
IDS_o is transmitted
True
True
IDS=IDSo
IDs_o
False
End
A||B||C
Generate n_1,n_2 Calculate A,B,C and transmitted
Extract n1 and n2. Calculate local value of C = C'
Step 2: Mutual Authentication
Calculate local value of D =D'
C =C'
False
End
True
D=D'
False
End
True
D
Calculate and transmit D. Update the variables
Update the variables
Step 3: Pseudonym and Key Updating
End
End

Figure 2. Working of RCIA Protocol

TABLE 2. Steps For Conversion Function $Con(X, Y)$

Step 1: Grouping
If $wt(X) \leq n$ Split the input such that $X_2 = x_n x_{(n-1)} x_{(n-2)} .x_{(wt(X)+1)}$ $X1 = x_{(wt(X))} x_{(wt(X)-1)} x_{(wt(X)-2)} \cdots x_1$ Continue splitting the sub part till size of all subgroups is less than T. Group Y according to same rule
Step 2: Rearrange
Regroup X according to segmentation in Y. Rearrange grouping in Y depending on segmentation pattern in X. Cyclic left rotate each group by its weight. Final outputs will be X'', Y''
Step 3: Composition
$Con(X, Y) = X'' \oplus Y''$

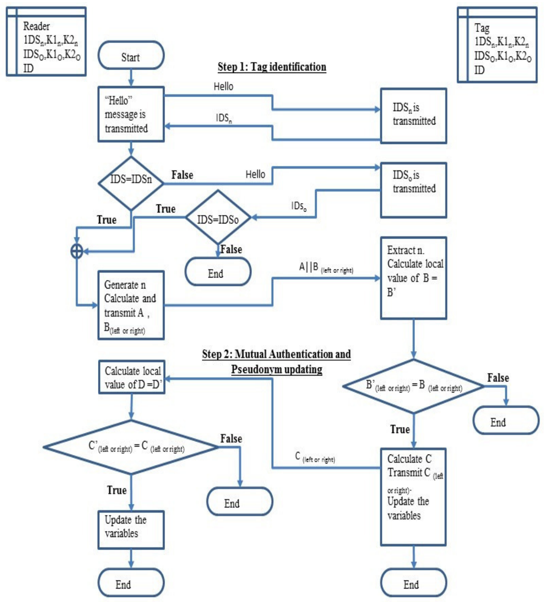
Reader
1DSn,K1n,K2n
IDSO,K1O,K2O
ID
Start
Step 1: Tag identification
Tag
1DSn,K1n,K2n
IDSO,K1O,K2O
ID
Hello
"Hello" message is transmitted
IDSn
IDSn is transmitted
IDS=IDSn
False
Hello
IDSo is transmitted
True
True
IDS=IDSo
IDSo
False
End
A||B (left or right)
Extract n. Calculate local value of B = B'
Generate n Calculate and transmit A , B(left or right)
Step 2: Mutual Authentication and Pseudonym updating
Calculate local value of D =D'
B'(left or right) = B (left or right)
False
End
True
C'(left or right) = C (left or right)
False
C (left or right)
Calculate C Transmit C (left or right). Update the variables
True
End
Update the variables
End
End

Figure 3. Working of SLAP

$$
\begin{aligned}
IDS_o &= IDS_n, K_{1o} = K_{1n}, K_{2o} = K_{2n} \\
K_{1n} &= Con(K_1, n) \oplus K_2 \\
K_{2n} &= Con(K_2, B) \oplus K_1 \\
IDS_n &= Con(IDS, n \oplus (B''_{lorr} || C''_{lorr})) \\
C &= Con(Con(B, K_{1n}), Con(K_{1n}, K_{2n} \oplus n)) \oplus ID
\end{aligned}
$$

Reader receives a segment of C,C_l or C_r based on hamming weight of C. Tag is validated by calculating and comparing local value of C_l or C_r. At the end of protocol, reader updates the values of index pseudonyms and keys by executing function used by tag side. Steps of SLAP is elaborated in Figure 3.

3. Desynchronization Attack

In Desynchronization attack, the adversary performs eavesdropping and replay attack. The main motive of this security attack is to remove index pseudonyms of tag from the reader database. In this way, protocol fails in tag identification mode. Following are the details of desynchronization attacks on above mentioned protocols.

3.1. Desynchronization Attack on SASI:

Following are the two methods that have been proposed so far to perform desynchronization attack on SASI protocol[8].

3.1.1. The First Attack:. Let us assume that tag and reader are completely in sync. The parameters being saved on both sides are given in Table 3.

TABLE 3. DESYNCHRONIZATION ATTACK 1

	Reader		Tag			
$InitialState$	IDS	IDS^1	IDS_n	IDS^1	IDS_o	IDS^0
$Step1$	IDS	IDS^1	IDS_n	IDS^2	IDS_o	IDS^1
$Step2$	IDS	IDS^3	IDS_n	IDS^3	IDS_o	IDS^1
$Step3$	IDS	IDS^3	IDS_n	IDS^2	IDS_o	IDS^1

TABLE 4. DESYNCHRONIZATION ATTACK 2

	Reader		Tag			
$InitialState$	K_2	K_2^1	K_{2n}	K_2^1	K_{2o}	K_2^0
$Step1$	K_2	K_2^2	K_{2n}	K_2^2	K_{2o}	K_2^1
$Step2$	K_2	K_2^2	K_{2n}	K_2^{2*}	K_{2o}	K_2^1

Step 1: During an authentication cycle, adversary sniffs the values of variables A, B, C. It also blocks message D from tag to reader. Without D message reader cannot verify tag and hence will not update parameters.
Step 2: RFID system run the protocol again without the interference of third party.
Step 3: In this step, adversary pretends to be a reader and sends"$Hello''$ message to tag. It discards IDS^3 value from tag and asks for IDS_o. As a result IDS^1 is received. Then the adversary replays the $A\|B\|C$ messages. Since these messages were generated by valid reader tag authenticates and updates its variables.

As it is evident in the above table 3, different values of variables are stored in reader and tag. After this attack, reader will never be able to identify the tag.

3.1.2. The Second Attack:. In this scheme, K_2 at tag side is changed in such a way that protocol running with a legitimate reader always terminates at tag authentication step. Hence tag and reader lose their communication link. Consider an RFID system in which tag and reader are synchronized as given in table 4.
Step 1: Reader and tag completes a successful session. But the communication is recorded and saved as $(A_1 = A, B_1 = B, C_1 = C)$.
Step 2: In this step tag communicates with the adversary on the basis of IDS^1. Message A, B and C are slightly modified version of previously recorded messages (A_1^*, B_1^*, C_1^*). $A_1^* = A_1$ with kth bit flipped, $B_1^* = B_1$ and $C_1^* = C_1$ with msb flipped. Change in A_1 will cause k_{th} bit of n_1 to change. This will flip k_{th} bit of $K_2 \oplus n_1^*$. If the weight of K_1 moves this flipped bit at msb of C then adversary gets authenticated and modifies K_2 of tag as K_2^*.

The results in Table 4 show the desynchronization of tag and reader.

3.2. Desynchronization attack on RCIA and SLAP:

In September 2016, Masoumeh and Nasour proposed a generalized desynchronized attack on RCIA and SLAP UMAPs [9]. The proposed attack model requires only five authentication sessions to make both the reader and the tag desynchronize. The generalized description of the desynchronization attack is presented as follows:

In RCIA and SLAP, both the reader(R) and the tag (T) store two entries of pseudonyms $(IDS^{(i+1)} \& IDS^{i})$ and Keys $(K^{(i+1)} \& K^{i})$ to overcome the possible desynchronization attacks. So, in first session, R initiates the protocol session with the targeted T and receives messages, $A^{(i+1)}, B^{(i+1)}, C^{(i+1)}$. The tag, T then responds with message $D^{(i+1)}$ and also updates its variables $(IDS^{(i+2)}, K^{(i+2)})$. After authentication of message $D^{(i+1)}$, the reader also updates its variables $(IDS^{(i+2)}, K^{(i+2)})$. Now, both the reader and the tag have updated and previous variable values $(IDS^{(i+1)}, IDS^{(i+2)}, K^{(i+1)} \& K^{(i+2)})$ in their databases. The adversary (A) eavesdrops the first authentication session and stores the value of pseudonym and messages $(IDS^{(i+1)}$ and $A^{(i+1)}, B^{(i+1)}, C^{(i+1)} \& D^{(i+1)})$.

In the second authentication session,R receives the $IDS^{(i+2)}$ and sends $A^{(i+2)}, B^{(i+2)}, C^{(i+2)}$ to T. The adversary (A) eavesdrops this conversation and prevents these messages from reaching at T side. Since, the authentication session remains incomplete, so both the R and T will not update their variables and remain stay with previous variables $(IDS^{(i+1)}, IDS^{(i+2)}, K^{(i+1)} \& K^{(i+2)})$. For the third session, R initiates the authentication session and T responds with its $IDS^{(i+2)}$. The adversary (A) interrupts the conversation, blocks the $IDS^{(i+2)}$ from reaching at readers side and sends the random value as IDS^{*} to the R. R will not find the find the matched entry of this random pseudonym and hence sends another"$Hello''$ message towards T. This time, T responds with its old value $IDS^{(i+1)}$ for authentication. Now, R sends $A^{(i+3)}, B^{(i+3)}, C^{(i+3)}$ messages to T and receives $D^{(i+3)}$. After successful authentication both R and T store $(IDS^{(i+1)}, IDS^{(i+3)}, K^{(i+1)} \& K^{(i+3)})$in their database and still remain synchronized .

In fourth session, A impersonate as a valid R and interacts with T. On receiving of "$Hello''$ message, T responds with $IDS^{(i+3)}$. However, A sends another "$Hello''$ message to T and at this time T responds with $IDS^{(i+1)}$. The adversary, A then sends the pre-captured messages $A^{(i+1)}, B^{(i+1)}, C^{(i+1)}$ to T ;which are acceptable for T(Since these messages were captured from valid authentication session). T authenticates the messages and updates its pseudonyms and keys as $(IDS^{(i+1)}, IDS^{(i+2)}, K^{(i+1)} \& K^{(i+2)})$ while at this point the reader has $(IDS^{(i+1)}, IDS^{(i+3)}, K^{(i+1)} \& K^{(i+3)})$ in its database. Finally, in the last authentication session, A once again pretends to be a valid R and sends a "$Hello''$ message towards T. Upon receiving of readers (A) query,T responds with $IDS^{(i+2)}$ and A replays the pre-captured messages $A^{(i+2)}, B^{(i+2)}, C^{(i+2)}$(second authentication session). Since, these messages are extracted from a genuine authentication session, therefore acceptable for T and after successful authentication, T updates its pseudonym and keys as $(IDS^{(i+2)}, IDS^{(i+4)}, K^{(i+2)} \& K^{(i+4)})$.

Now, next time when R wants to interact with the affected T, then it will not recognize him and will always abort its authentication session (permanently desynchronized)..

4. Novel Memory Storage Mechanism for UMAPs:

Since 2007, various memory storage mechanisms have been presented to avoid possible desynchronization attacks (scenarios). In SASI and GOASSMER, the tag stores both the current and new values of the pseudonyms and the keys. However, as discussed in section 3, a very simple confusion mechanism can make both the reader and the tag permanently desynchronized. The RAPP protocol devised another strategy to combat against desynchronization models and instead of storing two entries of pseudonyms and keys at tag side, RAPP stores these two entries at reader side. Although, this memory storage mechanism reduces the memory requirements at tag side but found to be vulnerable against many desynchronization attacks [8] [10]. Later, Umar Mujahid et al.[6] suggested that the storage of two entries (old and new) at both side (tag and reader) is the only way out to get rid from all possible desynchronization attacks. However, as discussed in previous section, Masoumeh and Nasour [9] showed that this two side storage does not also perform well in an active adversarial model. The authors suggested that if we use Pseudo Random Number Generator at both sides only then one can only avoid the desynchronization problems. However, integration of PRNGs at tag side is not practically feasible, since we have resources constraint at tag side (4K GE). In this section, we have proposed a new memory storage mechanism for UMAPs without adding any payload at tags side. The proposed memory storage mechanism is generic, so can be applied to all upcoming UMAPs as well. The detailed description of the proposed model is as follows:

Like RCIA and SLAP, in our proposed memory storage mechanism, the tag (T) stores two entries of the pseudonyms and keys $(IDS^{old}, IDS^{new}, K^{old}, K^{new})$, this reduces the communication cost of the protocol. While the reader (R) has a dynamic memory allocation mechanism and it stores all the pseudonyms and keys $(\cdot, IDS^{(old-1)}, IDS^{old}, IDS^{new}, IDS^{(new+1)} \cdot, K^{old}, K^{new}, K^{(new+1)}, \cdot)$, related to the particular tag. A Real Time Clock (RTC) has also been assimilated at reader side (one for each tag). So, whenever (even in the presence of adversary) a legitimate reader interacts with the valid tag, the response of the tag should be present in readers database. To avoid the buffer overflow at reader side, the RTC has been assimilated with reader for each tag, which monitors and allows an overwriting on the oldest IDS values (starting from MSB). The memory mapping architecture (Reader side) has been presented in the following Figure 4.

By using the above mentioned memory architecture, none of the adversaries can launch desynchronization attack against reader and the tag. For security of the proposed model, let's consider the Masoumeh and Nasour desynchronization attack model.

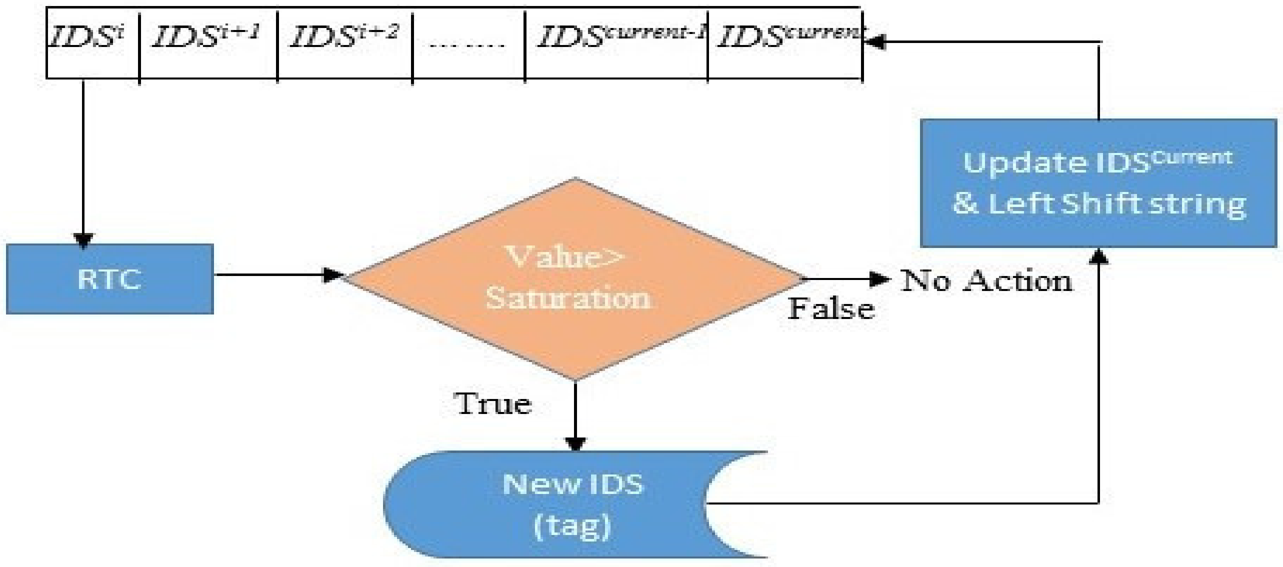

Figure 4. Novel Readers Memory Architecture

In the fifth authentication session, the adversary forces both the legitimate reader (R) and the tag (T) to store $(IDS^{(i+1)}, IDS^{(i+3)}, K^{(i+1)} \& K^{(i+3)})$ and $(IDS^{(i+2)}, IDS^{(i+4)}, K^{(i+2)} \& K^{(i+4)})$ respectively. So, by applying our, if the RTC threshold value is set to 1000 then it means, adversary requires atleast 5000 authentication sessions to make them desynchronize which is practically impossible as counter (present at tag side) resets after extensive wrong queries.

5. Conclusion

This paper presents the comprehensive survey of the ultralightweight cryptography and specifically Ultralightweight Mutual Authentication Protocols (UMAPs). We have discussed the detailed working of two new UMAPs: RCIA and SLAP. Then a new desynchronization attack model has been discussed which highlights the loopholes of both UMAPs. To overcome the highlighted vulnerabilities, a novel security frame work (to avoid desynchronization attacks) has also been proposed. The proposed security framework provides the optimal memory architecture at reader side without adding any payload at the tag‘s side. The security framework is generic in nature so can be applied to all existing and upcoming UMAPs as well to avoid possible desynchronization attacks.

References

[1] M. David and N. R. Prasad, “Providing strong security and high privacy in low-cost rfid networks,” in *International Conference on Security and Privacy in Mobile Information and Communication Systems*, pp. 172–179, Springer, 2009.

[2] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estévez-Tapiador, and A. Ribagorda, “Lmap: A real lightweight mutual authentication protocol for low-cost rfid tags,” in *Workshop on RFID security*, pp. 12–14, 2006.

[3] H.-Y. Chien, “Sasi: A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity,” *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 337–340, 2007.

[4] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, “M2ap: A minimalist mutual-authentication protocol for low-cost rfid tags,” in *International Conference on Ubiquitous Intelligence and Computing*, pp. 912–923, Springer, 2006.

[5] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, “Emap: An efficient mutual-authentication protocol for low-cost rfid tags,” in *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*, pp. 352–361, Springer, 2006.

[6] U. Mujahid, M. Najam-ul Islam, and M. A. Shami, “Rcia: a new ultralightweight rfid authentication protocol using recursive hash,” *International Journal of Distributed Sensor Networks*, vol. 2015, 2015.

[7] J. S. Hanguang Luo, Guangjun Wen and Z. Huang, “Slap: Succinct and lightweight authentication protocol for llow-cost rfid system,” in *The Journal of Mobile Communication, Computation and Informations*, Springer, 2016.

[8] H.-M. Sun, W.-C. Ting, and K.-H. Wang, “On the security of chien’s ultralightweight rfid authentication protocol,” *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, p. 315, 2011.

[9] M. Safkhani and N. Bagheri, “Generalized desynchronization attack on umap: application to rcia, kmap slap and sasi++ protocols,” *Cryptology ePrint Archive*, p. 905, 2016.

[10] Z. Ahmadian, M. Salmasizadeh, and M. R. Aref, “Desynchronization attack on rapp ultralightweight authentication protocol,” *Information processing letters*, vol. 113, no. 7, pp. 205–209, 2013.