Generalized Desynchronization attack on UMAP: application to RCIA, KMAP SLAP and SASI$^+$ protocols

Masoumeh Safkhani and Nasour Bagheri

[1] Computer Engineering Department, Shahid Rajaee Teacher Training University,Tehran, Iran
Postal code: 16788-15811, Tel/fax:+98-21-2297006 Safkhani@srttu.edu

[2] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran
Postal code: 16788-15811, Tel/fax:+98-21-2297006 NBagheri@srttu.edu

Abstract. Tian *et al.* proposed a permutation based authentication protocol [19] entitled RAPP. However, it came out very soon that it suffers from several security treats such as desynchronization attack. Following RAPP, several protocols have been proposed in literature to defeat such attacks. Among them, some protocols suggested to keep a record of old parameters by both the reader and the tag. In this paper we present a genrilized version of all such protocols, named GUMAP, and present an efficent desynchronization attack against it. The complexity of our attack is 5 consequences sessions of protocol and the success probability is almost 1. Our attack is applicable as it is to recently proposed protocols entitled RCIA [10], KMAP [12], SASI$^+$ [9] and SLAP [11]. To the best of our knowledge, it is the first report on the vulnerability of these protocols.

keywords: RFID, Authentication, RAPP, RCIA, KMAP, SLAP, SASI$^+$, GUMAP, Desynchronization Attack.

1 Introduction

To identify or trace an object, a common approach could be to attach a tag to the object and use Radio Frequency IDentification (RFID) technology, as a wireless identification method that uses radio frequency to communicate with the tag and identify/trace the tag holder (object). However, this approach may compromise the tag holder's privacy duo the fact that malicious readers also can communicate with the tag. Assuming that it is possible to find a relation between consequences responses of the tag, it would be possible to use such responses to trace the tag holder. Hence, the used communication protocol between the tag and the reader is considered from security point of view and the protocol is urged to satisfy some requirements, where security against traceability attacks are among such requirements. To address this requirement, protocol's parties randomize sessions by introducing nonces and may also update common parameters between the tag and the reader. However, in this case, if an adversary forces the tag and the reader to update a common parameter to different values, they will not authenticate each other in later sessions and we say they have been desynchronized. If a tag desynchronize form the reader, the tag holder will not be able to receive the service which is provided by the reader which contradicts availability. Hence, for any proper authentication protocol, it should not be possible for the adversary to desynchronize the tag and the reader; otherwise the protocol is suffers from desynchrnoization attack.

On the other hand, most of the tags that are attached to objects are passive. A passive tag is a highly constrained microchip with antenna that contains unique information related to the object that the tag has been attached to [8]. In the last decade, targeting passive tags, several ultralightweight RFID protocols have been proposed, e.g. [6, 13–15, 18], but all these suffer from various vulnerabilities such as desynchronization, traceability, replay and secret disclosure attacks (e.g. [4, 7] and [16]).

Later, Tian *et al.* proposed a permutation based ultralightweight mutual authentication protocol called RAPP [19]. Although it has been demonstrated very soon that RAPP is not a secure protocol [1, 5, 20], but several successors protocols attempted to improve it, e.g. R^{2}AP [21], RCIA [10], KMAP [12] and SLAP [11]. Among them RCIA, KMAP and SLAP are the newest protocols in this trails. Similar to RAPP, in these protocols also tags only use three simple operations: bitwise XOR, left rotation and a very lightweight nonlinear function to provide desired confusion. The general framework of RCIA, KMAP and SLAP mostly follows the framework proposed by Tian *et al.* in RAPP [19] and its successor R^{2}AP [21]. However, to overcome the desynchronization attack that have been proposed against RAPP [1] and R^{2}AP [11], both the tag and the reader keep a history of old date. For example in SLAP protocol, the reader and the tag share secret parameters K_1^{old},K_2^{old},IDS^{old} and K_1^{new},K_2^{new},IDS^{new} that are updated after each successful run of the protocol. In addition, each tag has a static identifier denoted by ID. In this paper we introduce and and analyse the security of a generalized version of RAPP-based ultralightweight mutual authentication (GUMAP).

In the rest of the paper, in Section 2 we introduce required notations and background. In Section 3 we introduce the generalized version of RAPP-based ultralightweight mutual authentication protocols, i.e. GUMAP. The generalizes desynchronization attack against GUMAP is presented in Section 4. Finally, we conclude the paper in Section 5.

2 preliminaries

2.1 Notation

Given bit strings x and y, we denote their bitwise XOR by $A \oplus B$, bitwise AND by $A \wedge B$. In this paper $Rot(x, y)$ assigned to bitwise right/ left rotation of x as a factor of y. For example, in SLAP [11] In this protocol, $Rot(x, y)$ denotes a circular left rotate string x by $wt(y)$ bit(s), where $wt(y)$ is the Hamming weight of string y (amount of bit(s) in y that are equal to 1)

2.2 SLAP

Description of SLAP [11] is depicted in Fig.1. In this protocol $Con(x, y)$ is a very lightweight function introduced by the designer of SLAP and $Rot(X, Y)$ denotes left rotation of X according to the Hamming weight of Y. If the Hamming wight of B is odd then R sends A and B_L to T, where B_L is the left halve of B; otherwise R sends A and B_R to T. Similarly, T sends either C_L or C_R depends on the Hamming weight of C. In the updating phase of parameters the halve of B and C that have not been transfered over the channel are used. For more details of SLAP we refer to [11].

Reader R		**Tag** T
$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$		$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$
	$\xrightarrow{Hello}$	
Generates n	$\xleftarrow{IDS}$	
$A = Con(K_1, K_2) \oplus n$		
$B = Con(Rot(K_1, n), K_1 \oplus K2) \oplus Rot(Con(K_2, K_2 \oplus n), K_1)$		$n = A \oplus Con(K_1, K_2)$
	$\xrightarrow{A, B_{L/R}}$	Verifies $B_{L/R}$
		$C = Con(Con(B, K_1^{new}), Con(K_1^{new}, K_2^{new} \oplus n)) \oplus ID$
Verifies $C_{L/R}$	$\xleftarrow{C_{L/R}}$	
Updates IDS, K_1, K_2		Updates IDS, K_1, K_2
$IDS^{old} = IDS$		
$K_1^{old} = K_1$		
$K_2^{old} = K_2$		
$IDS^{new} = Con(IDS, n \oplus (B_{R/L} \Vert C_{R/L}))$		
$K_1^{new} = Con(K_1, n) \oplus K_2$		
$K_2^{new} = Con(K_2, n) \oplus K_1$		

Fig. 1. Mutual authentication phase of SLAP [11]

2.3 KMAP

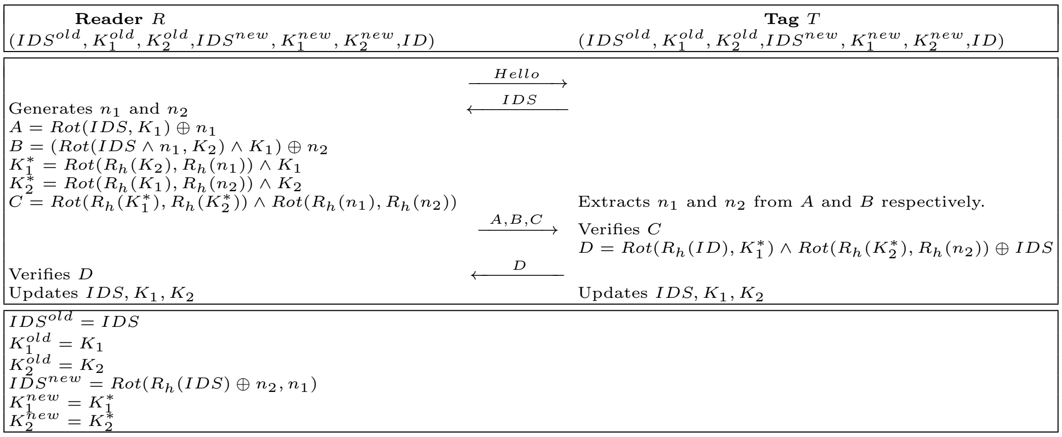
Description of KMAP [12] is depicted in Fig.2. In this protocol $K_c(x)$ is an ultralightweight function introduced by the designer of KMAP, named pseudo Kasami code, and $Rot(X, Y)$ denotes left rotation of X according Y. In this protocol, reader generates two random number, i.e. n_1 and n_2, and they influence the calculation of $K_c(x)$. In addition tag keeps a counter i which has no impact on our attack. For more details of KMAP we refer to [12].

Reader R		**Tag** T
$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$		$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$
		counter $i++ \leq$ threshold
	$\xrightarrow{Hello}$	
Generates n_1 and n_2	$\xleftarrow{IDS}$	
$A = Rot(Rot(n_1, IDS \oplus K_1), K_2)$		
$B = Rot(Rot(n_2, IDS \oplus K_2), K_1 \oplus n_1)$		
$K_1^* = Rot(K_c(K_1), K_c(n_1)) \oplus K_2$		
$K_2^* = Rot(K_c(K_2), K_c(n_2)) \oplus K_1$		
$C = Rot(Rot(K_c(n_1), K_c(K_2^*) \oplus K_c(n_2)), K_c(K_1^*) \oplus n_2)$		Extracts n_1 and n_2 from A and B respectively.
	$\xrightarrow{A, B, C}$	Verifies C
		$D = Rot(Rot(K_c(ID) \oplus K_c(n_1), K_c(IDS) \oplus K_c(K_1)), K_c(K_2))$
Verifies D	$\xleftarrow{D}$	
Updates IDS, K_1, K_2		Updates IDS, K_1, K_2
$IDS^{old} = IDS$		
$K_1^{old} = K_1$		
$K_2^{old} = K_2$		
$IDS^{new} = Rot(K_c(IDS) \oplus n_1, K_c(n_2))$		
$K_1^{new} = K_c(K_1^*)$		
$K_2^{new} = K_c(K_2^*)$		

Fig. 2. Mutual authentication phase of KMAP [12]

2.4 RCIA

Description of RCIA [10] is depicted in Fig.3. The structure of this protocol is almost similar to KMAP [12]. In this protocol $R_h(x)$ is an ultralightweight function introduced by the designer of RCIA, named recursive hash function, and $Rot(X, Y)$ denotes left rotation of X based on Y. Similar to KMAP, in this protocol, reader generates two random number, i.e. n_1 and n_2, and they influence the calculation of $R_h(x)$. In addition tag keeps a counter i which has no impact on our attack. For more details of RCIA we refer to [10]. It worth to note the protocol uses bitwise AND operation such that its secret keys congruent to fully zero after a few sessions of protocol. However, details of such weaknesses in this protocol and other protocols are out of this paper's scope.



Reader R
$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$

Tag T
$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$

$\xrightarrow{Hello}$

$\xleftarrow{IDS}$

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

Extracts n_1 and n_2 from A and B respectively.

$\xrightarrow{A,B,C}$

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

$\xleftarrow{D}$

Verifies D
Updates IDS, K_1, K_2

Updates IDS, K_1, K_2

$IDS^{old} = IDS$
$K_1^{old} = K_1$
$K_2^{old} = K_2$
$IDS^{new} = Rot(R_h(IDS) \oplus n_2, n_1)$
$K_1^{new} = K_1^*$
$K_2^{new} = K_2^*$

Fig. 3. Mutual authentication phase of RCIA [10]

2.5 SASI$^+$

Description of SASI$^+$ [9] is depicted in Fig.4. The structure of this protocol is almost similar to RCIA [10]. In this protocol $R_h(x)$ is an ultralightweight function introduced by the designers, also named recursive hash function, and $Rot(X, Y)$ denotes left rotation of X based on Y. In this protocol also, reader generates two random number, i.e. n_1 and n_2, and they influence the calculation of $R_h(x)$. For more details of SASI$^+$ we refer to [9].

3 Description of GUMAP

In this section we describe the generalized version of an ultralightweight mutual authentication protocols (GUMAP) where only the reader introduces nonce to protocol and both the tag and the reader keep a record of old data. In our description we denote all dynamic parameters which is transfered in plain-text as IDS, all secret parameters which is updated through the protocol as K, all static secret parameters by ID and all nonces generated by the reader as n. It should be noted we have no restriction on the length of any of these parameters. Hence, for example in the case of SLAP [11], where tag has two keys K^1 and K^2 in our generalized description we model it as $K = K^1 \| K_2$. For a dynamic value $\mathcal{Z}$, its value in stage i is denoted as $\mathcal{Z}^i$, e.g. K^i and IDS^i. Assume that in the beginning of j^{th} session of

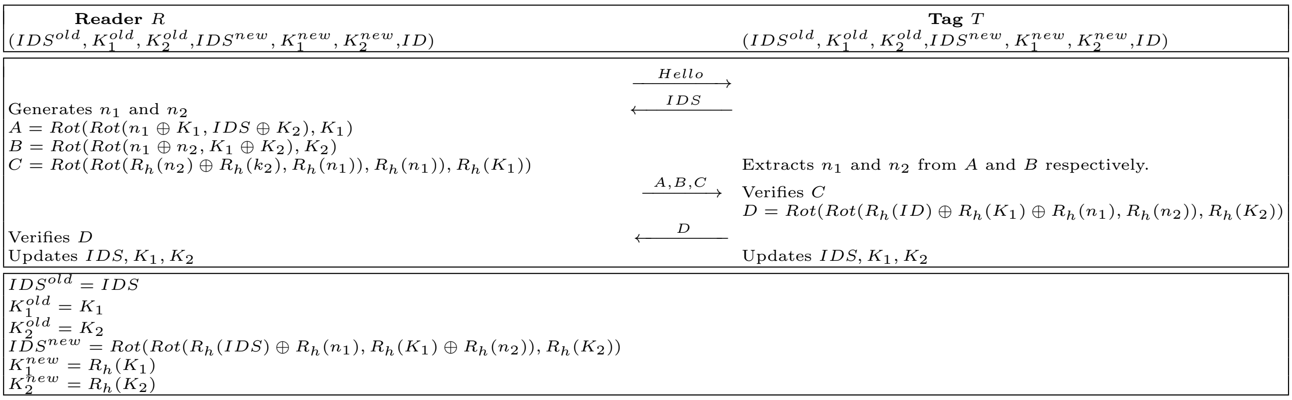

Reader R
$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$
Tag T
$(IDS^{old}, K_1^{old}, K_2^{old}, IDS^{new}, K_1^{new}, K_2^{new}, ID)$
$\xrightarrow{Hello}$
$\xleftarrow{IDS}$
Generates n_1 and n_2
$A = Rot(Rot(n_1 \oplus K_1, IDS \oplus K_2), K_1)$
$B = Rot(Rot(n_1 \oplus n_2, K_1 \oplus K_2), K_2)$
$C = Rot(Rot(R_h(n_2) \oplus R_h(k_2), R_h(n_1)), R_h(n_1)), R_h(K_1))$
Extracts n_1 and n_2 from A and B respectively.
$\xrightarrow{A,B,C}$
Verifies C
$D = Rot(Rot(R_h(ID) \oplus R_h(K_1) \oplus R_h(n_1), R_h(n_2)), R_h(K_2))$
Verifies D
$\xleftarrow{D}$
Updates IDS, K_1, K_2
Updates IDS, K_1, K_2
$IDS^{old} = IDS$
$K_1^{old} = K_1$
$K_2^{old} = K_2$
$IDS^{new} = Rot(Rot(R_h(IDS) \oplus R_h(n_1), R_h(K_1) \oplus R_h(n_2)), R_h(K_2))$
$K_1^{new} = R_h(K_1)$
$K_2^{new} = R_h(K_2)$

Fig. 4. Mutual authentication phase of SASI$^+$ [9]

the protocol the tag's and the reader's records of dynamic parameters IDS and K are both $(IDS^{j-1}; K^{j-1})$ and $(IDS^j; K^j)$, respectively for old and new records. The description of GUMAP, as depicted in Fig. 5, is as follows:

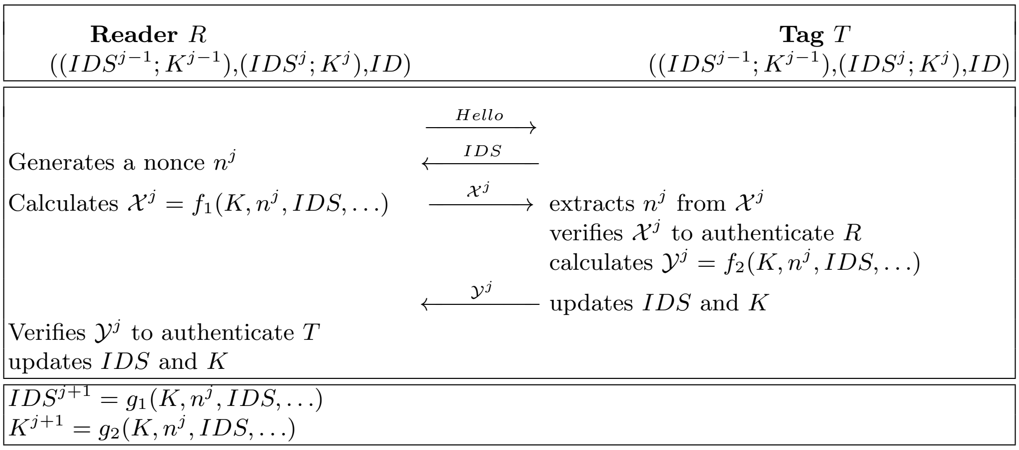
1. The reader R, sends Hello to the target tag T.
2. T replies with its IDS^j; if it has not been recognized by the reader, R resends Hello to the target tag T and T responds R with IDS^{j-1}/
3. R uses the received IDS as an index to find the data related to the tag in its database. If IDS matches the old data of the tag, R uses IDS^{j-1} and K^{j-1} to compute the transfered messages; otherwise it uses IDS^j and K^j. If IDS does not match any record of the reader's database, the protocol's session is terminated. Assuming IDS matches a record in the database, where we denote it as IDS and K, the reader generates a nonce n^j and computes its challenge $\mathcal{X}^j = f_1(K, n^j, IDS, \ldots)$ to be sent to the tag T.
4. T extracts the nonce n^j from the received challenge and verifies $\mathcal{X}^j$ using the extracted n^j and its local records of shared parameters with R. If R has been authenticated, T responds to R by computing its challenge $\mathcal{Y}^j = f_2(K, n^j, IDS, \ldots)$ to be sent to R. In addition, the tag updates its parameters as follows:
   - assigns the current values of K and IDS to K^j and IDS^j respectively and updates the tag's shared parameters as follows:

$$\begin{aligned} IDS^{j+1} &= g_1(K, n^j, IDS, \ldots) \\ K^{j+1} &= g_2(K, n^j, IDS, \ldots) \end{aligned}$$

5. R evaluates the sent challenge $\mathcal{Y}^j$ by T to authenticate T. Assuming the tag has been authenticated, R will updates its records for IDS, K in a way similar to the approach used by T respectively to $(IDS^j; K^j)$ and $(IDS^{j+1}; K^{j+1})$.

In this protocol, to provide forward security, we assume that the updated value of $(IDS^{j+1}; K^{j+1})$ is uniformly randomized by introduced value of nonce $n^j e$.

The reader keeps two sets of the tuple $\{IDS^x, K_1^x, K_2^x, K_3^x\}$ for $x = old$ (previous values) and $x = new$ (updated values) to prevent desynchronization attacks (e.g., when an active



Reader R
$((IDS^{j-1};K^{j-1}),(IDS^{j};K^{j}),ID)$

Tag T
$((IDS^{j-1};K^{j-1}),(IDS^{j};K^{j}),ID)$

$\xrightarrow{Hello}$

Generates a nonce n^j $\xleftarrow{IDS}$

Calculates $\mathcal{X}^j = f_1(K, n^j, IDS, \ldots)$ $\xrightarrow{\mathcal{X}^j}$ extracts n^j from $\mathcal{X}^j$
verifies $\mathcal{X}^j$ to authenticate R
calculates $\mathcal{Y}^j = f_2(K, n^j, IDS, \ldots)$

$\xleftarrow{\mathcal{Y}^j}$ updates IDS and K

Verifies $\mathcal{Y}^j$ to authenticate T
updates IDS and K

$IDS^{j+1} = g_1(K, n^j, IDS, \ldots)$
$K^{j+1} = g_2(K, n^j, IDS, \ldots)$

Fig. 5. Generalized ultralightweight mutual authentication protocol (GUMAP)

adversary blocks the last message D and E). In addition, it keeps the used random number in the last session n_1 and n_2. Hence, on the next session, if the reader receives IDS^{old}, it will use the same n_1 and n_2 to calculate A, B, D and E. However, it will not give the service access to the tag and is used only to synchronize the tag, it is denoted by *synchronization session*. To receive service, the tag should participate in another session.

Contributions: R^{2}AP is claimed to be resistant against common attacks, including desynchronization and traceability. In this paper, we first propose a desynchronization attack with a success probability of almost 1. Secondly, we introduce a traceability attack that exploits the weak properties of $Rec(\cdot)$. In this attack, we disclose a correlation between the transferred messages and bits of the ID which can be used to trace the tag. The attack can be extended to disclose the whole ID (as in [20]) at the expense of requiring various protocol runs.

4 Desynchronization attack on GUMAP

To overcome the desymnchronization attack on RAPP [1, 5] and its successors such as R^{2}AP [21] that only the tag keeps a record of old and new shared parameters, some protocols such as RCIA [10], KMAP [12] and SLAP [11] suggested both reader and tag to keep old parameters also. Such protocols fits our generalized RAPP. In this section we present a desynchronization attack against GUMAP which can be also applied to such protocols almost as it is, i.e. RCIA, KMAP and SLAP. Given a target tag T with $(IDS^{i-1}; K^{i-1})$ and $(IDS^i; K^i)$, respectively as the old and the new records of its shared parameters with the reader R, our generalized desynchronization attack works as follows:

1. In session i:
   (a) R receives IDS^i form T, generates n^i and sends $\mathcal{X}^i = f_1(K^i, n^i, IDS^i, \ldots)$ to T.
   (b) T authenticates R, send $\mathcal{Y}^i = f_2(K^i, n^i, IDS^i, \ldots)$ to R and updates its record of shared values to $(IDS^i; K^i)$ and $(IDS^{i+1} = g_1(K^i, n^i, IDS^i, \ldots); K^{i+1} = g_2(K^i, n^i, IDS^i, \ldots))$.

   (c) R authenticates T and updates its record of shared values to $(IDS^i; K^i)$ and $(IDS^{i+1}; K^{i+1})$.
   (d) Adversary A eavesdrops IDS^i and $\mathcal{X}^i$ and stores them.
2. In session $i + 1$:
   (a) R receives IDS^{i+1} form T, generates n^{i+1} and sends $\mathcal{X}^{i+1} = f_1(K^{i+1}, n^{i+1}, IDS^{i+1}, \ldots)$ to T.
   (b) Adversary A eavesdrops IDS^{i+1} and $\mathcal{X}^{i+1}$ and stores them and prevents T from receiving $\mathcal{X}^{i+1}$
   (c) In this point the tag and the reader recodes of shared parameters are $(IDS^i; K^i)$ and $(IDS^{i+1}; K^{i+1})$ yet.
3. In session $i + 2$:
   (a) R sends $Hello$ to T and tag sends IDS^{i+1}.
   (b) Adversary blocks the value sent by T and sends a random value as IDS' to R;
   (c) R will not find a record for IDS'. Hence it sends another $Hello$ to T and this time tag sends IDS^i to R.
   (d) R receives IDS^i form T, generates n^{i+2} and sends $\mathcal{X}^i = f_1(K^i, n^{i+2}, IDS^i, \ldots)$ to T.
   (e) T authenticates R, send $\mathcal{Y}^{i+2} = f_2(K^i, n^{i+2}, IDS^i, \ldots)$ to R and updates its record of shared values to $(IDS^i; K^i)$ and $(IDS^{i+2} = g_1(K^i, n^{i+2}, IDS^i, \ldots); K^{i+2} = g_2(K^i, n^{i+2}, IDS^i, \ldots))$.
   (f) R authenticates T and updates its record of shared values to $(IDS^i; K^i)$ and $(IDS^{i+2}; K^{i+2})$.
4. In session $i + 3$:
   (a) Adversary impersonates R and sends $Hello$ to T and tag sends IDS^{i+1}.
   (b) Adversary sends another $Hello$ to T and this time tag responds with IDS^i to R.
   (c) Adversary receives IDS^i form T, and rsponds with the eavesdropped $\mathcal{X}^i = f_1(K^i, n^i, IDS^i, \ldots)$ from Step 1d.
   (d) T authenticates R, send $\mathcal{Y}^i = f_2(K^i, n^i, IDS^i, \ldots)$ to R and updates its records of shared values to $(IDS^i; K^i)$ and $(IDS^{i+1} = g_1(K^i, n^i, IDS^i, \ldots); K^{i+1} = g_2(K^i, n^i, IDS^i, \ldots))$.
5. In session $i + 4$:
   (a) Adversary once again impersonates R and sends $Hello$ to T and tag sends IDS^{i+1}.
   (b) Adversary receives IDS^{i+1} form T, and responds with the eavesdropped $\mathcal{X}^{i+1} = f_1(K^{i+1}, n^{i+1}, IDS^{i+1}, \ldots)$ from Step 1d.
   (c) T authenticates R, responds with $\mathcal{Y}^{i+1} = f_2(K^{i+1}, n^{i+1}, IDS^{i+1}, \ldots)$ and updates its records of shared values to $(IDS^{i+1}; K^{i+1})$ and $(IDS^{i+4} = g_1(K^{i+1}, n^{i+1}, IDS^{i+1}, \ldots); K^{i+4} = g_2(K^{i+1}, n^{i+1}, IDS^{i+1}, \ldots)$.

To implement the above attack, an adversary eavesdrops a session of the protocol between the reader and the tag and stores all messages but intercepts D and E and sends D' and E' as mentioned in Theorem **??**. Next, it sends a Hello command to the tag. If the tag returns the old record of IDS it means that it has not accepted the sent D' and E' and the adversary can impersonate the reader by sending the stored values of A and B and new values of D' and E'. The adversary repeats this attack up to when the tag accepts D' and E'. Based on Theorem **??**, if the tag accepts the sent D' and E', the updated tags records of secrets do not match the reader records with the probability of 0.9999962. The expected complexity of this attack is eavesdropping a session between the target tag T and a legitimate reader R and $(2^{-10.62})^{-1} \cong= 1574$ impersonation of the reader to the tag.

5 Conclusions

In this paper, we have described a desynchronization attack against a generalized protocol that model a class of ultralightweight mutual authentication protocols for them both the tag and the reader keeps a history of old and new parameters and only reader produces nonce to randomize sessions. This analysis shows that, despite of the details of calculation of messages, any such protocol will be venerable against desynchronization attack. Although previous analysis on ultralightweight protocols based on the details of the calculation of messages, e.g [2, 3, 5, 17, 20], show that it may not be possible to design a secure protocol using only few application of lightweight operations, this analysis more show that both protocol party should contribute to session randomizations.

References

1. Z. Ahmadian, M. Salmasizadeh, and M. R. Aref. Desynchronization attack on RAPP ultralightweight authentication protocol. *Inf. Process. Lett.*, 113(7):205–209, 2013.
2. Z. Ahmadian, M. Salmasizadeh, and M. R. Aref. Recursive linear and differential cryptanalysis of ultralightweight authentication protocols. *IEEE Transactions on Information Forensics and Security*, 8(7):1140–1151, 2013.
3. G. Avoine and X. Carpent. Yet another ultralightweight authentication protocol that is broken. In *Workshop on s Security – RFIDSec'12*, Nijmegen, Netherlands, June 2012.
4. G. Avoine, X. Carpent, and B. Martin. Privacy-friendly synchronized ultralightweight authentication protocols in the storm. *J. Network and Computer Applications*, 35(2):826–843, 2012.
5. N. Bagheri, M. Safkhani, P. Peris-Lopez, and J. E. Tapiador. Weaknesses in a new ultralightweight RFID authentication protocol with permutation - RAPP. *Security and Communication Networks*, 7(6):945–949, 2014.
6. H.-Y. Chien. Sasi: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. *IEEE Trans. Dependable Sec. Comput.*, 4(4):337–340, 2007.
7. P. D'Arco and A. D. Santis. On ultralightweight RFID authentication protocols. *IEEE Trans. Dependable Sec. Comput.*, 8(4):548–563, 2011.
8. GS1 EPCglobal. Epcglobal tag data standards version 1.4. http://www.gs1.org/epc-rfid, last access 12-th Augest 2015.
9. U. M. Khokhar, M. Najam-ul-Islam, A. R. Jafri, Q. U. Qurat-ul-Ain, and M. A. Shami. A new ultralightweight RFID mutual authentication protocol: SASI using recursive hash. *IJDSN*, 2016:9648971:1–9648971:14, 2016.
10. U. M. Khokhar, M. Najam-ul-Islam, and M. A. Shami. RCIA: A new ultralightweight RFID authentication protocol using recursive hash. *IJDSN*, 2015:642180:1–642180:8, 2015.
11. H. Luo, G. Wen, J. Su, and Z. Huang. Slap: Succinct and lightweight authentication protocol for low-cost rfid system. *Wireless Networks*, pages 1–10, 2016.
12. U. Mujahid, M. Najam-ul Islam, and S. Sarwar. A new ultralightweight rfid authentication protocol for passive low cost tags: Kmap. *Wireless Personal Communications*, pages 1–20, 2016.
13. P. Peris-Lopez, J. C. H. Castro, J. M. Estévez-Tapiador, and A. Ribagorda. Emap: An efficient mutual-authentication protocol for low-cost RFID tags. In R. Meersman, Z. Tari, and P. Herrero, editors, *OTM Workshops (1)*, volume 4277 of *Lecture Notes in Computer Science*, pages 352–361. Springer, 2006.
14. P. Peris-Lopez, J. C. H. Castro, J. M. Estévez-Tapiador, and A. Ribagorda. Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol. In *WISA*, pages 56–68, 2008.
15. P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda. Lmap: A real lightwight mutual authentication protocol for low-cost RFID tags. In *Proceedings of RFIDSec06 Workshop on RFID Security*, Graz,Austria, 12-14 July 2006.
16. R. C.-W. Phan. Cryptanalysis of a new ultralightweight RFID authentication protocol - sasi. *IEEE Transactions on Dependable and Secure Computing*, 6(4):316–320, 2009.

17. M. Safkhani and N. Bagheri. Passive secret disclosure attack on an ultralightweight authentication protocol for internet of things. Cryptology ePrint Archive, Report 2016/838, 2016. `http://eprint.iacr.org/2016/838`.
18. A. Tewari and B. B. Gupta. Cryptanalysis of a novel ultra-lightweight mutual authentication protocol for iot devices using rfid tags. *The Journal of Supercomputing*, pages 1–18, 2016.
19. Y. Tian, G. Chen, and J. Li. A new ultralightweight RFID authentication protocol with permutation. *IEEE Communications Letters*, 16(5):702–705, 2012.
20. S.-H. Wang, Z. Han, S. Liu, and D.-W. Chen. Security analysis of RAPP an RFID authentication protocol based on permutation. Cryptology ePrint Archive, Report 2012/327, 2012.
21. X. Zhuang, Y. Zhu, and C. Chang. A new ultralightweight RFID protocol for low-cost tags: R ^{2} AP. *Wireless Personal Communications*, 79(3):1787–1802, 2014.