Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 9648971, 14 pages
http://dx.doi.org/10.1155/2016/9648971

Research Article

A New Ultralightweight RFID Mutual Authentication Protocol: SASI Using Recursive Hash

Umar Mujahid, M. Najam-ul-Islam, Atif Raza Jafri, Qurat-ul-Ain, and M. Ali Shami

Department of Electrical Engineering, Bahria University, Islamabad, Pakistan

Correspondence should be addressed to Umar Mujahid; umarkhokhar1@hotmail.com

Received 6 September 2015; Revised 28 November 2015; Accepted 7 December 2015

Academic Editor: Luciano Tarricone

RFID is one of the most prominent identification schemes in the field of pervasive systems. Nonline of sight capability makes RFID systems much better choice than its contended systems (such as barcode, magnetic tape, etc.). Since the RFID uses wireless channel for communication with its associated devices, there should be some optimal encryption methods to secure the communicating data from adversaries. Several researchers have proposed ultralightweight mutual authentication protocols (UMAPs) to secure the RFID systems in cost effective manner. Unfortunately most of the previously proposed UMAPs are later found to be vulnerable against various desynchronization, Denial of Service (DoS), traceability, and full disclosure attacks. In this paper, we present a more sophisticated UMAP to provide Strong Authentication and Strong Integrity (SASI) using recursive hash function. The proposed protocol incorporates only simple bitwise logical operators XOR, Rot, and nontriangular function (recursive hash) in its design, which can be efficiently implemented with a low cost passive RFID tag. The performance analysis of the protocol proves the conformation of the proposed protocol with EPC-C1G2 passive tags. In addition to privacy and security, small chip area (miniaturization) is another design constraint (which is mandatory requirement for a protocol to be considered as ultralightweight authentication protocol). We have also proposed and implemented the efficient hardware design of the proposed protocol for EPC-C1G2 tags. Both the FPGA and ASIC implementation flows have been adopted. The FPGA flow is primarily used to validate the functionality of the proposed hardware design whereas ASIC flow (using TSMC 0.35 μm library) is used to validate the gate count. To the best of our knowledge, this is the first FPGA and ASIC implementation of any ultralightweight RFID authentication protocol.

1. Introduction

Currently barcodes and RFID are the two widely used identification systems. The efficient functional haste and prevailing features (automation and nonline of sight) of the RFID systems are main causes of its massive and rapid deployment compared to other contended schemes. The RFID systems comprise Radio Frequency (RF) tags and RF readers. The reader periodically broadcasts the RF signals and the tags (present in the reader's vicinity) share their internal data with the reader. Table 1 describes the main differences between RFID systems and barcode schemes.

Moreover, the RFID systems can uniquely identify each item/product (tag), while mostly barcodes schemes can only identify the type of the item/product (not unique identification). The only hindrance in rapid growth of RFID technology is its security concerns and overall cost of the tag, which should be \$0.05 to 0.1 to be considered comparable with the barcodes [1, 2]. The demand of low cost tags limits us to use passive RFID tags which only involve simple computational operations for security and other functions. Typically such tags can store only 32–1K bits and can support 250–4K logic gates for security related tasks. So conventional cryptographic algorithms (such as AES and Triple DES) and primitives (such as Hash function) cannot be used to secure the system. In spite of the certain limitations, RFID systems are evolving rapidly (9 billion USD revenue in 2014 [3]) with diverse applications. Although barcode is currently dominant identification technology RFID is an emerging successor and has captured marketplace in various sectors (such as animal tracking, cattle identification, and supply chain management). The RFID systems mainly consist of

Table 1: RFID versus barcode.

Parameters	RFID	Barcode
Line of sight (LOS)	Not required	Required
Reading range	*Active RFID*: 100 ft. or more *Passive RFID*: Up to 40 ft.	Several inches up to several ft.
Reading frequency (rate)	1000 tags/sec	Only one at a time
Read/write capability	Read/write	Only read
Technology	Radio Frequency (RF)	LASER (optical)
Automatic/manual	Automatic	Manual (needs a human for scanning)

three main components: tag, reader, and backend database. A tag is small electronic chip (transponder) implanted on an object, which needs to be identified. A reader scans the tags, collects identification information, and forwards this information towards backend database (server) for final verification.

Security and privacy are the two major concerns of RFID based identification systems, which are associated with the tag's cost. On the basis of the tag's cost and computational capabilities, the RFID tags can be classified into two types: high and low cost tags. The high cost tags are resourceful enough to support traditional cryptographic algorithms and primitives such as AES, hash functions, and stream ciphers for security. These conventional cryptographic algorithms and primitives have excessive power, memory, and silicon (chip) area requirements which are transcendent from the low cost tag's computational capabilities. So a new field of *ultralightweight cryptography* has been introduced to ensure the security of the low cost RFID tags in recent years. Numerous UMAPs have proposed that support only simple T-functions [4] and some special purpose ultralightweight primitives perform security related tasks. To the best of our knowledge, all of the previously proposed UMAPs are reported to be vulnerable against numerous desynchronization and full disclosure attacks. In order to avoid all possible security attacks, this paper presents a new UMAP to provide Strong Authentication and Strong Integrity using recursive hash for extremely low cost RFID systems.

The organization of the paper is as follows. Section 2 describes the literature review, which is followed by the basic working of the novel UMAP in Section 3. Section 4 presents the functional analysis of the protocol and Section 5 lists the security analysis of the proposed UMAP. Section 6 discusses the performance analysis of the proposed UMAP. The efficient hardware implementation of the proposed scheme is described in Section 7. Finally Section 8 concludes the paper.

2. Literature Review

In 2006, Peris-Lopez et al. laid the foundation of ultralightweight cryptography and proposed three UMAPs: LMAP (lightweight mutual authentication protocol) [1], EMAP (extremely lightweight mutual authentication protocol) [5], and M2AP (minimalist mutual authentication protocol) [6]. All three UMAPs involve simple bitwise logical operations (XOR, AND, and ADD) in their designs and hence can be implemented within approximately 1K logical gates. However, in 2007, Tieyan et al. [7, 8] highlighted the vulnerabilities in all three UMAPs listed above and proposed desynchronization and full disclosure attacks on them. They also reported that the combinations of T-functions return another T-function (linear), which are considered to be insecure for computation of the encrypted messages.

In 2007, Chien [9] reported that the assimilation of nontriangular function in UMAP designs makes UMAP more robust and reliable. Chien integrated the hamming weight based cyclic left rotation function, Rot, in protocol messages and proposed a new UMAP to provide Strong Integrity and Strong Authentication (SASI). However right after the introduction of protocol, several researchers highlighted the pitfalls in SASI protocol design. Sun et al. [10] reported two simple desynchronization attacks and Avoine et al. [11] presented full disclosure attack to reveal all the concealed secrets of the SASI protocol. Avoine's full disclosure attack requires 2^{17} authentication sessions to fully disclose the tag's ID.

Peris-Lopez et al. [12] extended Chien's concept of assimilation of nontriangular function in UMAP designs. They proposed a new Genetic Programming based nontriangular primitive, MixBits, and introduced a new UMAP: GOASSMER. To the best of our knowledge, the GOASSMER protocol is the most robust protocol of 2008 and it has received only desynchronization attack till date [13]. However, the authors have not clarified the hardware requirements of MixBits function.

Later David-Prasad [14], Yeh et al. [15], and Lee et al. [16] proposed UMAPs, which are also reported to be vulnerable against various desynchronization, traceability, and full disclosure attacks.

In 2012, Tian et al. [17] introduced a new ultralightweight primitive, permutation, and proposed RFID authentication protocol using permutation (RAPP). The inclusion of the permutation function not only enhanced the diffusion properties of the protocol messages but also increased the randomness

in the messages. In 2013, Ahmadian et al. [18] exploited the poor properties of the permutation function and highlighted the desynchronization attack on the protocol. Wang et al. [19] made several observations on RAPP protocols and proposed a sequential full disclosure attack on the RAPP. The proposed full disclosure attack revealed all the concealed secrets of the tag and thus challenged the security claims of the RAPP.

In 2014, Zhuang et al. [20] proposed a new UMAP (R2AP) based on their new nontriangular primitive: reconstruction. Although the mathematical formulation of the protocol is quite robust the poor designing of the protocol structure makes the tag traceable. In R2AP protocol, if an adversary blocks the last two messages between tag and reader from reaching at tag's side, then next time both the reader and the tag use the same variables for computation of messages. Now repeated blocking of these messages restricts both parties to communicate with the same variables (static) and hence the adversary can easily track the movements of the tags through publically disclosed messages.

Most of the previous UMAPs [1, 5, 6, 9, 12, 14–17, 20–22] have similar pitfalls in their designs and are hence vulnerable against various cryptanalysis attacks [2, 7, 8, 10, 11, 13, 18, 19, 23–33]. This raises the need of a new more sophisticated UMAP, which should overcome all the previously highlighted problems in cost effective manner.

3. The SASI Protocol Using Recursive Hash

As we have already discussed, the SASI was the first UMAP that introduced the nontriangular function "Rot" in its design and was proposed in 2007 [9]. Many researchers highlighted the various loopholes and vulnerabilities in SASI protocol and challenged its claim of Strong Authentication and Strong Integrity (SASI) [10, 11, 24, 29, 30, 33].

In this section, we improve the overall description of the SASI protocol and make it more robust against all the highlighted attacks to date. In our proposed scheme, both the reader and the tag preshare two copies of IDS and keys (K_1, K_2) (old and current). This storage of an extra copy of IDS and keys at both sides avoids all possible desynchronization attacks. Furthermore, like other UMAPs, this new protocol also updates its pseudonym and keys after each successful authentication session. To strengthen the diffusion properties of the publically disclosed messages, we have incorporated two new ultralightweight primitives (recursive hash (R_h) [22] and double rotation) in protocol design. The recursive hash function (R_h) can be computed as follows.

Computation of Recursive Hash Function. Suppose Y is a "n" bit string, where

$$Y = y_1, y_2, \ldots, y_n,$$
$$y_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n. \tag{1}$$

Then the computation of recursive hash of Y, $R_h(Y)$, involves the following four steps:

(a) Decimate the string Y into "S" number of chunks (memory blocks) with equal number of bits "l" per memory block ($S = n/l$).

For EPC-C1G2 tags [9] $n = 96$ bits and, for efficient hardware implementation, $l = 12$ bits and $S = 8$ $(s_8, \ldots, s_1)$ [22].

(b) Extract random numbers (n_1, n_2) from messages sent by the reader; then the tag computes the seed (index of the memory block) for recursive hash in following manner:

(i) Rand = $n_1 \oplus n_2$;

(ii) seed (index of memory block) = wt (Rand) mod S,

where wt (Rand) represents the hamming weight of Rand.

(c) Select the corresponding memory block (s_i) of decimated string Y using seed calculated in step (b) and take XOR between selected memory block (s_i) with all other blocks except the block itself.

(d) Rotate left the selected memory block (s_i) with itself by its hamming weight; Rot(s_i, $wt(s_i)$).

Finally concatenate both strings (XORed and left rotated (s_i)) to have the final recursive hash of the string.

To better understand the concept of recursive hash function, consider the following example.

Example. Given $Y = 11001011010100101001101100101011$, $n = 32$, $l = 8$, $S = s_4, s_3, s_2, s_1$ and assume seed = 3 then recursive hash of Y, $R_h(Y)$, is

$$R_h(Y) = 10011001100100101100100101111001. \tag{2}$$

Algorithm 1 shows the stepwise computation of the above example.

3.1. The Protocol. Figure 1 shows the basic working of the protocol. We have assumed that the channel between the reader and the database is secure as we can incorporate typical cryptographical algorithms to the secure that channel. The description of the protocol is as follows.

Step 1. The reader initiates the protocol by transmitting the message "*hello*" towards the tag.

Step 2. The tag responds with its current IDS.

Step 3. Upon receiving IDS, the reader uses it as an index and searches for a matched entry in its database. If the reader does not find the IDS in its database, then it sends an error message towards the tag (which means the tag should send IDS^{old}). However if a match occurs, then the reader generates two pseudorandom numbers (n_1, n_2) and conceals them in messages A, B:

$$A = \text{Rot}\left(\text{Rot}\left(n_1 \oplus K_1, \text{IDS} \oplus K_2\right), K_1\right), \tag{3}$$

$$B = \text{Rot}\left(\text{Rot}\left(n_2 \oplus n_1, K_2 \oplus K_1\right), K_2\right). \tag{4}$$

Assume $Y = 11001011010100101001101100101011$, $n = 32$, $l = 8$, $S = s_4 s_3 s_2 s_1$ and Seed = 3

Step 1.

s_4	s_3	s_2	s_1
11001011	01010010	10011011	00101011

Step 2. Since, Seed = 3 therefore s_3(01010010) memory block will be selected for Recursive hashing.

Steps 3 and 4. Take XOR between s_3 and all other blocks except the block itself, left rotates$_3$, Rot$(s_3, wt(s_3))$ and concatenate all computed results sequentially.

	11001011	01010010	10011011	00101011
$\oplus$	01010010		01010010	01010010
	10011001	—	11001001	01111001
Rot$(s_3, wt(s_3))$		10010010		
	10011001	10010010	11001001	01111001

So,

$$R_h(Y) = 10011001100100101100100101111001$$

Algorithm 1: The computation of recursive hash function (example).

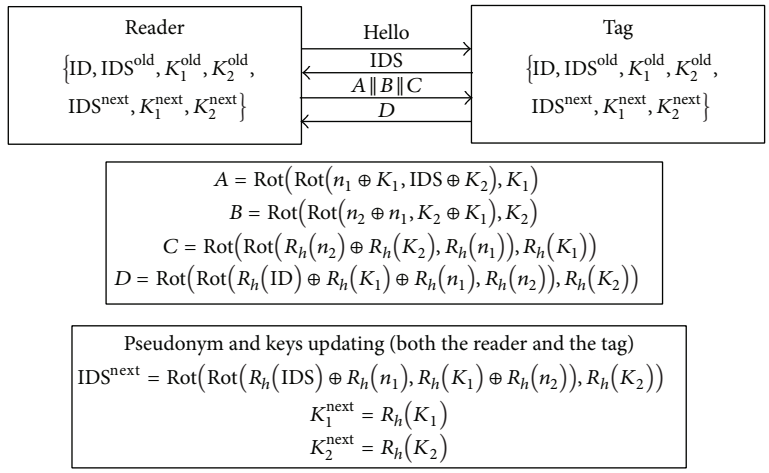

Figure 1: The SASI using recursive hash protocol.

(i) The reader then computes seed for recursive hash (R_h):

$$\text{Rand} = n_1 \oplus n_2.$$
$$\text{Seed (index of memory block)} = wt\ (\text{Rand}) \bmod S \tag{5}$$

(ii) After calculating the recursive hash of the variables (K_1, K_2, n_1, n_2), the reader computes message C:

$$C = \text{Rot}\left(\text{Rot}\left(R_h\left(n_2\right) \oplus R_h\left(K_2\right), R_h\left(n_1\right)\right), R_h\left(K_1\right)\right). \tag{6}$$

(iii) Reader $\rightarrow$ Tag: $A \parallel B \parallel C$.

Step 4. After reception of messages $A \parallel B \parallel C$, the tag performs following tasks:

(i) Extracts pseudorandom numbers (n_1, n_2) from messages A, B:

$$n_1 = \text{Rot}^{-1}\left(\text{Rot}^{-1}\left(A, K_1\right), \text{IDS} \oplus K_2\right) \oplus K_1,$$
$$n_2 = \text{Rot}^{-1}\left(\text{Rot}^{-1}\left(B, K_2\right), K_2 \oplus K_1\right) \oplus n_1. \tag{7}$$

(ii) Computes the seed for recursive hash (R_h) using (5) and calculates the recursive hash of the variables (K_1, K_2, n_1, n_2).

(iii) Computes the local value of message C, C':

$$C' = \mathrm{Rot}\left(\mathrm{Rot}\left(R_h\left(n_2\right) \oplus R_h\left(K_2\right), R_h\left(n_1\right)\right), R_h\left(K_1\right)\right). \quad (8)$$

(iv) The tag authenticates the reader as follows:

If $C = C'$

the tag authenticates the reader successfully

else

the protocol aborted

end if

(v) On successful authentication, the tag computes R_h(ID) and conceals it in message D:

$$D = \mathrm{Rot}\left(\mathrm{Rot}\left(R_h\left(\mathrm{ID}\right) \oplus R_h\left(K_1\right) \oplus R_h\left(n_1\right), R_h\left(n_2\right)\right), R_h\left(K_2\right)\right). \quad (9)$$

(vi) Tag → Reader: D.

(vii) The tag also updates its pseudonym (IDS) and keys (K_1, K_2) for future communication:

$$\begin{aligned} \mathrm{IDS}^{\mathrm{next}} &= \mathrm{Rot}\left(\mathrm{Rot}\left(R_h\left(\mathrm{IDS}\right) \oplus R_h\left(n_1\right), R_h\left(K_1\right)\right.\right. \\ &\quad \left.\left. \oplus R_h\left(n_2\right)\right), R_h\left(K_2\right)\right), \\ K_1^{\mathrm{next}} &= R_h\left(K_1\right), \\ K_2^{\mathrm{next}} &= R_h\left(K_2\right). \end{aligned} \quad (10)$$

Step 5. On receiving message D, the reader computes the local value of message D, "D":

$$D' = \mathrm{Rot}\left(\mathrm{Rot}\left(R_h\left(\mathrm{ID}\right) \oplus R_h\left(K_1\right) \oplus R_h\left(n_1\right), R_h\left(n_2\right)\right), R_h\left(K_2\right)\right). \quad (11)$$

(i) The reader authenticates the tag as follows:

If $D = D'$

the reader authenticates the tag successfully

else

the protocol aborted

end if

(ii) The reader uses (10) to update the pseudonym (IDS) and keys (K_1, K_2) of the tag in its database for future correspondence with the particular tag.

4. Functional Analysis of the Proposed Protocol

A UMAP can only be considered secure if it provides three basic functionalities: confidentiality, integrity, and authentication. Our proposed protocol ensures these functionalities optimally as described below.

4.1. Confidentiality. One of the main objectives of the UMAPs is the secure transmission of the secret ID from the tag to the reader. In SASI using recursive hash, both the reader and the tag preshare static ID and dynamic keys (K_1, K_2). Therefore only the legitimate party with prior knowledge of these variables can decrypt the A, B, C, and D messages. Increasing the number of rotations increases the level of confidentiality as shown by Khovratovich and Nikolić [32]. In our proposed scheme, the inclusion of double rotation function and recursive hash function causes the extensive overall rotations and makes number of rotations/additions: $P > -t/\log_2(P_r)$ [32], hence making system more secure and reliable against rotational cryptanalysis (where P_r is the logarithmic probability of any ARX (Addition Rotation and XOR) system). Moreover, the optimal composition of the publically disclosed messages (A, B, C, and D) increases the computational complexity and hence makes it impossible for an adversary to retrieve the concealed data. The complexity of recovering conjuncture pseudorandom numbers (n_1, n_2) is as follows.

Computational Complexity of Recovering (n_1, n_2)

(1) The outer rotation from (3) is undone with complexity $O(\log_2 K_1)$:

$$\begin{aligned} F &= \mathrm{Rot}^{-1}\left(A, K_1\right) \\ &= \mathrm{Rot}\left(n_1 \oplus K_1, \mathrm{IDS} \oplus K_2\right). \end{aligned} \quad (12)$$

(2) Further the inner rotation from (12) is undone which increases the complexity to $O(2 \times \log_2 K_1 \times \log_2 K_2)$:

$$\begin{aligned} G &= \mathrm{Rot}^{-1}\left(F, \mathrm{IDS} \oplus K_2\right) \\ &= n_1 \oplus K_1. \end{aligned} \quad (13)$$

(3) Now XORing all the possible values of K_1 from right-hand side doubles the complexity $O(2 \times 2 \times \log_2 K_1 \times \log_2 K_2)$.

(4) After disclosing random number n_1, the adversary takes Rot^{-1} of message B from (4) with K_2, which doubles the overall complexity $O(8 \times \log_2 K_1 \times \log_2 K_2)$:

$$\begin{aligned} H &= \mathrm{Rot}^{-1}\left(B, K_2\right) \\ &= \mathrm{Rot}\left(n_2 \oplus n_1, K_2 \oplus K_1\right). \end{aligned} \quad (14)$$

(5) The inner rotation of (14) is further undone with $K_2 \oplus K_1$; this increases the complexity four times $O(32 \times \log_2 K_1 \times \log_2 K_2)$:

$$\begin{aligned} I &= \mathrm{Rot}^{-1}\left(H, K_2 \oplus K_1\right) \\ &= n_2 \oplus n_1. \end{aligned} \quad (15)$$

(6) Now we take XOR of all possible values of n_1 with right-hand side variable; this further doubles the complexity $O(64 \times \log_2 K_1 \times \log_2 K_2)$.

So with $O(64\times\log_2 K_1\times\log_2 K_2)$ complexity, an adversary may compute the pseudorandom numbers. However these pseudorandom numbers are updated after each authentication session regardless of success or failure of the protocol session. So the proposed protocol ensures the data confidentiality optimally.

4.2. Integrity. In our proposed scheme, all the transmitted messages (A, B, C, and D) are interlinked. If an adversary tries to modify any of the messages by toggling certain bits then the impact of this modification directly transfers to other messages as well. For example, if an adversary modifies certain bits of the message A, then the tag computes invalid n_1 and further invalid n_2 from the message B. Now, the tag is not able to authenticate the reader and hence terminates its protocol session with the reader. The presence of n_1 in message B ensures the integrity of the messages.

4.3. Authentication. As the name suggests, every UMAP should provide a strong mechanism of mutual authentication. In SASI, using recursive hash, the transmitted messages not only ensure the integrity but also provide the mutual authentication. After initial identification of the tag, the reader sends A, B, and C messages towards the tag. Only the legitimate tag can extract the random nonces from A and B messages and hence can authenticate the reader. Moreover, the legitimate tag can only generate such message D which is acceptable to the legitimate reader.

5. Security Analysis of the Proposed Protocol

In this section, we evaluate the resistance of the proposed protocol against various security attacks. The detailed description of the attack models is given below.

5.1. Desynchronization Attack. In our proposed scheme, both the reader and the tag update their pseudonym (IDS) and keys (K_1, K_2) after each successful authentication session. Since the synchronization depends upon the reception of C and D messages, the adversary may interrupt the transmitted messages to desynchronize the legitimate pair (reader and tag). There are two possible desynchronization attack scenarios.

(i) Adversary Interrupts the Message C. In such scenario, the tag does not receive the message C so it does not authenticate the reader and keeps the previous values of the pseudonyms and keys. The tag also abandons this incomplete protocol session and hence does not compute the message D. Therefore both the reader and the tag remain in the same state (synchronized).

(ii) Adversary Interrupts the Message D. In this case, the reader does not receive the message D; therefore it keeps the previous values of pseudonyms and keys while the tag updates its internal variables (since it has received the message C). However our proposed scheme resolves such disruption issues by storing an extra copy of pseudonym and keys (old and new) at both sides. For the next session, the reader and the tag use the old values (pseudonym and keys) for authentication. Hence both legitimate parties remain synchronized.

5.2. Replay Attack. The adversary impersonates as a valid tag and replays the (fake) message D towards the reader. As the message D involves recursive hash values and random nonces (n_1, n_2) (independent of the session), the genuine reader does not authenticate such tag and aborts the protocol session with the particular tag.

Another replay attack scenario: an adversary may interrupt the message D in one session and then replay the previously captured messages (based on old values) $A \parallel B \parallel C$ towards the tag. This scenario does not affect internal secrets and synchronization of the legitimate parties. The storage of two copies of the local variables combats against all possible replay and desynchronization attacks. Moreover the replay attack proposed by Sun et al. for SASI [10] also takes advantage of single copy of pseudonyms and keys and hence is catered by our proposed protocol.

5.3. Traceability Attack. Like RCIA [22], the SASI using recursive hash also avoids all possible traceability attacks. In our proposed protocol, the tag uses dynamic IDS instead of its original static ID for interaction with the reader. For each new authentication session, both the reader and the tag use new (updated) IDS, which makes tracking of a particular tag impossible. Another possibility of traceability attack is tracking of the tag through exchanged messages. However in our proposed scheme, each new authentication session brings new random nonces (integrated in messages), which ensure the freshness of the messages and hence make the protocol untraceable. Raphael proposed one of the most sophisticated formal traceability models to validate the untraceability claims of UMAPs.

The analysis of the proposed protocol over Raphael formal traceability model [33] is as follows.

(i) Formal Traceability Test. In RFID systems, protocol parties ($\mathscr{P}$), tags ($\mathscr{T}$), and the readers ($\mathscr{R}$) communicate with each other. An adversary ($\mathscr{A}$) can interact actively and passively and control the communication amongst all the parties. The adversary ($\mathscr{A}$) can run the following queries.

Execute $(\mathscr{R}, \mathscr{T}, i)$ *Query.* This query models the passive adversary ($\mathscr{A}$). The adversary ($\mathscr{A}$) can eavesdrop on the communication channel and can record (obtain) all transmitted messages in protocol session i between the reader and the tag.

Send $(\mathscr{P}_1, \mathscr{P}_2, i, m)$ *Query.* This query models the active adversary ($\mathscr{A}$). The adversary ($\mathscr{A}$) can impersonate either as $\mathscr{P}_1$ ($\mathscr{P}_1 = \mathscr{T}$) or $\mathscr{P}_2$ ($\mathscr{P}_2 = \mathscr{R}$) and send message m in protocol session i to its contended party ($\mathscr{P}_1$ or $\mathscr{P}_2$).

Corrupt $(\mathscr{T}, S')$ *Query.* This query allows the adversary ($\mathscr{A}$) to obtain the concealed secret S of the tags and then set $S = S'$. Since the RFID tags generally are not tamper, resistant, this query assumes that the adversary ($\mathscr{A}$) has physical access to the tag.

Test $(i, \mathcal{T}_0, \mathcal{T}_1)$ *Query.* This query mainly defines the untraceability (UNT) and does not correspond to any adversary ($\mathcal{A}$) capabilities. In the protocol session i, this query gives adversary ($\mathcal{A}$), ID_b from the set $(\mathrm{ID}_0, \mathrm{ID}_1)$ corresponding to the tags ($\mathcal{T}_0$, $\mathcal{T}_1$) where $b \in \{0, 1\}$. The adversary ($\mathcal{A}$) succeeds if it can guess the correct value of b.

Now, the adversary's capabilities are clear so, the untraceability (UNT) problem can be defined as game ($\mathcal{G}$). The game ($\mathcal{G}$) is played between adversary ($\mathcal{A}$) and protocol parties ($\mathcal{P}$).

The SASI using recursive hash protocol proves to be untraceable against the described formal traceability model. Consider the untraceability (UNT) model presented above, where the adversary can send multiple queries to learn the relation between ID and the publically transmitted messages. The adversary returns to being successful, if an adversary computes the ID with probability P_r in the form of

$$[\mathrm{ID}]_i = [N_1]_i \blacksquare [N_2]_i \blacksquare \cdots \blacksquare [N_l]_i, \tag{16}$$

where N_j is the publically disclosed message and $\blacksquare$ denotes the logical operation between the messages. More precisely, if probability $P_r > 1/2$, the adversary wins the game and the protocol does not resist traceability.

Assume that in learning phase, the adversary ($\mathcal{A}$) computes (16) with probability $P_r > 1/2$. Then the adversary ($\mathcal{A}$) is given the challenge identifier ID_b from the set $(\mathrm{ID}_0, \mathrm{ID}_1)$ corresponding to the two fresh tags ($\mathcal{T}_0$, $\mathcal{T}_1$) where $b \in \{0, 1\}$. The adversary ($\mathcal{A}$) sends "execute query" to obtain all the transmitted messages of $\mathcal{T}_0$ in an authentication session. Now the adversary ($\mathcal{A}$) uses (16) and computes ID_i'. If $\mathrm{ID}_i' = [\mathrm{ID}_b]_i$ then $b = 0$; otherwise $b = 1$. The result of the game ($\mathcal{G}$) is $\mathrm{adversary}_{\mathcal{A}}^{\mathrm{UNT}}(k) = |\Pr[\mathcal{A} \text{ wins}] - \Pr[\text{random coin flip}]| = |P_r - 1/2| > \varepsilon(K)$ (since $P_r > 1/2$).

The main reason which allows the adversary to compute (16) is the use of only unbalanced operators in protocol designs. In SASI, using recursive hash, none of the unbalanced operator is used in the messages. Therefore, the adversary may only compute the following ambiguous equation for our proposed scheme:

$$\begin{aligned} &\mathrm{Rot}^{-1}\left(\mathrm{Rot}^{-1}\left(D, R_h\left(K_2\right)\right), R_h\left(n_2\right)\right) \oplus R_h\left(K_1\right) \\ &\quad \oplus R_h\left(n_1\right) = R_h\left(\mathrm{ID}\right). \end{aligned} \tag{17}$$

The adversary wins the game ($\mathcal{G}$) if he can rewrite (17) as follows:

$$[\mathrm{ID}]_i = [N_1]_i \blacksquare [N_2]_i \blacksquare \cdots \blacksquare [N_l]_i \quad \text{With probability } P_r > \frac{1}{2}. \tag{18}$$

All the variables used in (17) only occur in the message D and all operators in protocol design are balanced. It is practically impossible for an adversary to compute the recursive hash of any variable, since for each new authentication session the adversary encounters with new pseudorandom numbers and hence new recursive hash functions. Secondly, merging of these variables with other messages makes equation more complex and ambiguous. Thus the adversary ($\mathcal{A}$) cannot identify the challenge identifier and hence $\mathrm{adversary}_{\mathcal{A}}^{\mathrm{UNT}}(k) = |\Pr[\mathcal{A} \text{ wins}] - \Pr[\text{random coin flip}]| = |1/2 - 1/2| = 0 < \varepsilon(K)$.

Therefore, the SASI using recursive hash protocol proves to be untraceable against the Raphael formal traceability attack.

5.4. Full Disclosure Attacks. Most of the attacks proposed for UMAPs are ad hoc and probabilistic, which are not extensible to a broader class of the protocol. To the best of our knowledge there are only three formal (structural) cryptanalysis models (for UMAPs) that exist, namely, Tango [26, 27], RLC, and RDC [25]. As our proposed scheme involves nontriangular primitives (i.e., recursive hash and double rotation) extensively in its design so none of the formal and ad hoc attack can disclose any concealed secret. This limitation of the formal attacks has also been indicated by their inventors. Hence the optimal designing of the protocol messages makes our proposed scheme robust against all possible full disclosure attacks.

5.5. Formal Security Analysis. For formal security analysis of our proposed scheme, we have used Casper and FDR tools [34]. The FDR uses the assumptions of Dolev-Yao security model to find the possible attacks in the protocols [35], which is considerably more reliable than GNY and BAN (in the case of UMAPs) [36, 37]. Firstly, we describe the proposed protocol in simple and abstract language understandable by Casper, which further produces the Communicating Sequential Processes (CSP) description of the same protocol. After loading of the CSP file protocol in FDR2, it does not identify any attack. Hence successful formal security analysis enlists our proposed protocol among secure UMAPs.

6. Performance Analysis of the Proposed Protocol

This section presents the performance analysis of the proposed UMAP (SASI using recursive hash) in terms of computational operations, memory requirements, communication cost, and security for each tag. As far as computational operations (operators used) are concerned, our proposed UMAP involves only simple bitwise logical operations such as XOR, left rotation (Rot), and recursive hash (R_h) operations.

Regarding memory (storage) requirements, each tag owns one static ID and two dynamic entries of IDS and keys (old and new). Therefore ROM of size "$1L$" is required to store static ID and rewritable memory of size "$6L$" is required for storage of old and newly updated variables: keys (K_1, K_2) and IDS. For simplicity, here we have neglected the transitional registers, which are required to store the intermediate results.

The communication cost of the tag is basically the number of messages sent by the tag during one protocol session. Here in our proposed scheme, each tag transmits altogether two messages; hence the communication cost is "$2L$" bits.

As far as the security is concerned (discussed in Section 5 as well), our proposed UMAP provides robust security as compared to its contended (previously proposed)

ultralightweight mutual authentication protocols. None of the previous protocols completely satisfies the proposed comprehensive security model presented in Section 5. The existing UMAPs even fail to provide the basic functionalities (confidentiality, integrity, and authentication) which are the essential requirements for any security protocol. Moreover these UMAPs cannot resist against numerous structural (formal) cryptanalysis (Tango, RLC and RDC) and disclose tag's secret ID to adversaries after only few authentication sessions. Most of the T-function based UMAPs (LMAP, EMAP, and M^{2}AP, David-Prasad et al.) are less resistive against full disclosure attacks. For example, Hernandez-Castro et al. [27] proposed a structural cryptanalysis model (Tango attack) against David-Prasad protocol and retrieved all the concealed secrets (including ID) of the tag. The attack is mainly divided into two phases: selection of Good Approximations (GAs) and comparison of these approximations. In Tango attack, the adversary takes advantage of poor diffusion properties of T-functions (used in the protocol) and constructs the multiple approximations of the secrets using various combinations of the exchanged messages (A, B, D, E, and F). Further, the final selection of the GA equations is scrutinized using their hamming distance with secret itself. In the second phase of the attack, the main idea is to combine the multiple GAs for a particular secret, obtained in various protocol sessions and formation of these multiple session GAs matrixes. The adversary further compares the column wise total number of 1's (A) with precomputed threshold (γ) for final conjecture secret: threshold function of th(A) can be computed as follows:

$$\text{th}(A) = \begin{cases} \text{if } A_i \geq \gamma \quad \text{assign } 1 \\ \text{if } A_i \leq \gamma \quad \text{assign } 0, \end{cases} \tag{19}$$

where $\gamma = 0.5 \times N_A \times N_S$, N_A is the number of Gas for the secrets, and N_S is the number of eavesdropped sessions.

By using the above methodology, the adversary can retrieve all the concealed secrets (ID and keys) with almost 100% success rate.

The nontriangular function based UMAPs (GOASSMER, RAPP, R^{2}AP, RCIA, etc.) are more resistive against full disclosure attacks but researchers have identified many desynchronization and traceability attacks in such UMAPs due to weak protocol designs. For example, the GOASSMER protocol avoids all possible full disclosure attacks but fails to resist against desynchronization attack [38] (related to the structure of the GOASSMER protocol). The attacker stores $A \parallel B \parallel C$ messages of a genuine authentication session and blocks the D message (round-1). Then, in round-2, the adversary lets the pair communicate without any interruption. Since the current IDS does not match, the tag uses its previous IDS and keys for authentication and updating its variables. Finally, the adversary initiates its protocol session with the tag and refuses to accept its current IDS. The tag then uses its previous IDS which is now accepted by the adversary and the adversary sends back the precaptured messages (round-1) $A \parallel B \parallel C$. After accepting these messages, the tag updates its pseudonyms and permanently desynchronizes with its legitimate network.

Likewise, the R^{2}AP protocol avoids many existing adversarial attacks but the alternative approach for avoidance of desynchronization attacks enforces both parties (reader and tag) to compute the previously transmitted messages repeatedly. This repetition of the same messages opens the horizons for several traceability, replay, and Denial of Service (DoS) attacks. For example, if an adversary blocks the messages D and E then the reader updates its pseudonym and keys while the tag keeps the previous values of its pseudonym and keys. Hence, by repeatedly blocking the D and E messages, the adversary can easily identify and track the movement of the tag (because each time the adversary receives the same messages (A, B, C, D, and E)).

The only way to avoid such traceability attacks is to have new pseudorandom number for each new authentication session. The inclusion of new pseudorandom numbers ensures the freshness and anonymity of the messages. On the other hand, as discussed in Sections 4 and 5, our proposed protocol can withstand all the security attacks mentioned in the security model (including formal and ad hoc based attacks). A simple comparison of eminent ultralightweight protocols is listed in Table 2. The analysis depicts that the proposed UMAP outperforms the others while using minimal resources.

7. Design and Hardware Architecture

One of the major challenges in RFID systems is to design efficient and cost effective (ultralightweight) mutual authentication protocol. EPC-C1G2 tags are passive in nature, with no on-chip battery and hence can support fewer resources for routine operations. Only few thousand gates can be allocated to security related tasks, which makes design of such cryptographic protocols more challenging. As the proper hardware implementation of such ultralightweight protocols has been neglected since long hence it was unclear whether such protocols are practically compatible with low cost EPC-C1G2 RFID tags or not.

In this section, we present hardware architecture of the proposed UMAP for low cost passive EPC-C1G2 tags. Figure 2 shows the generic hardware architecture for UMAPs. Most of the ultralightweight authentication protocols follow a similar operational (working) model; the proposed hardware architecture can be used to implement several other UMAPs (such as SASI, Yeh et al., RAPP, RCIA, and R^{2}AP) as well with some operational variations. For efficient implementation and cost effectiveness, we reuse the logical components (gates and registers) frequently. The proposed architecture mainly includes four blocks: registers, Arithmetic Logic Unit (ALU), counter, and Finite State Machine (FSM). The FSM mainly controls the data flow between register and ALU blocks, so it is also protocol specific. The low level optimized designs (registers, ALU, and FSM) of the proposed UMAP are described as follows.

Table 2: Performance analysis of several UMAPs.

	LMAP [1]	EMAP [5]	M^{2}AP [6]	SASI [9]	GOASSMER [12]	David-Prasad [14]	RAPP [17]	Jeon and Yoon [21]	R^{2}AP [20]	SASI using recursive hash
Computational operations on Tag	$\oplus$, OR, +	$\oplus$, AND, OR	$\oplus$, OR, +	$\oplus$, AND, OR, Rot, +	$\oplus$, +, Rot, MixBits	$\oplus$, AND	$\oplus$, Rot, Per	$\oplus$, Sep (), Mer()	$\oplus$, Rec, Rot	$\oplus$, Rot
Memory requirement on Tag	$6L$	$6L$	$6L$	$7L$	$7L$	$5L$	$5L$	$5L$	$5L$	$7L$
Communication messages generated by tag	$2L$	$3L$	$3L$	$2L$	$2L$	$3L$	$2L$	$2L$	$2L$	$2L$
Total number of messages for mutual authentication	$4L$	$5L$	$5L$	$4L$	$4L$	$5L$	$5L$	$4L$	$5L$	$4L$
Resistance to desynchronization attacks	No	No	No	No	No	No	No	No	Yes	Yes
Resistance to full disclosure attacks	No	No	No	No	Yes	No	No	Yes	Yes	Yes
Resistance to traceability attacks	No	No	No	No	No	No	No	No	No	Yes

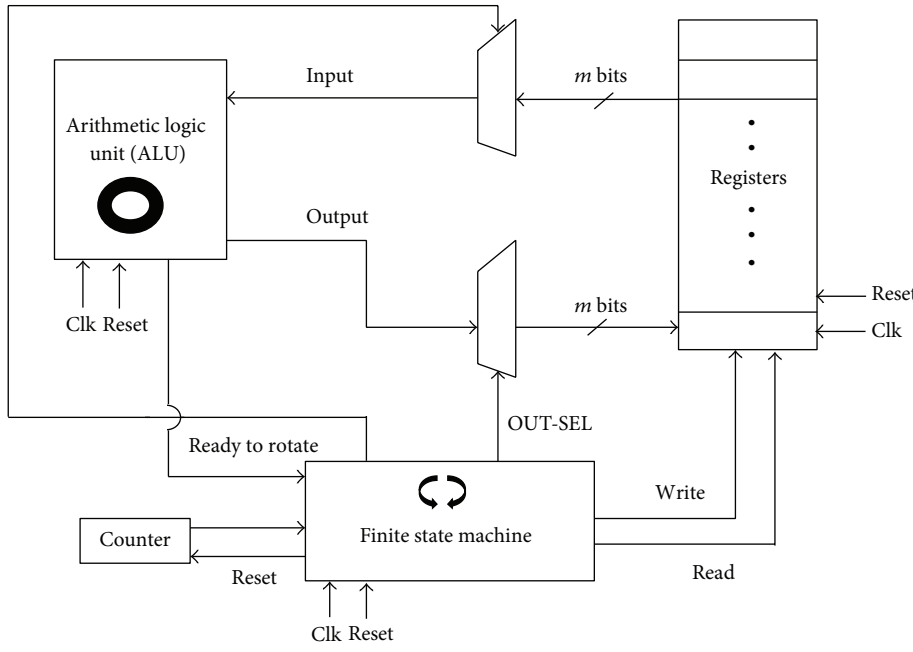

Figure 2: Hardware architecture of SASI using recursive hash.

7.1. Register Block. This block contains all the registers (memory blocks) required to store intermediate computations (results), permanent variables, and long-term values. The SASI using recursive hash protocol requires $8L$ dynamic memory to store pseudonym (IDS^{old}, IDS^{new}), keys (K_1^{old}, K_2^{old}, K_1^{new}, and K_2^{new}), and random nonces (n_1, n_2). It also requires $1L$ static memory to store its secret ID. Moreover, to store the intermediate results of ongoing computations, we use four general purpose registers (GP_0, GP_1, GP_2, and GP_3) of L bits. The SASI using recursive hash protocol is

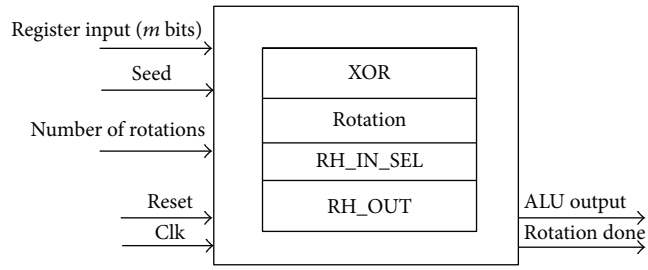

Figure 3: ALU hardware schematic for SASI using recursive hash protocol.

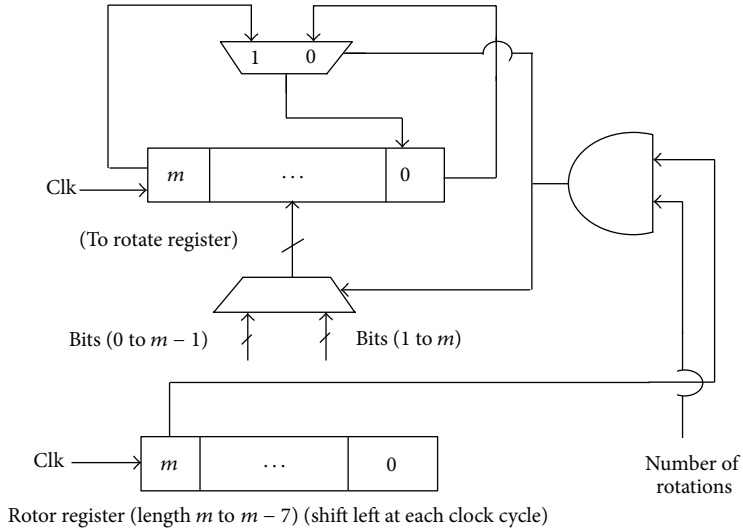

Figure 4: Rotation module (m-bit).

the most memory efficient protocol amongst its contended protocols.

7.2. ALU Block. The ALU block mainly comprises bitwise logical operators (protocol specific) and performs the specified computational operations. The designing of ALU block entirely depends upon the protocol (operational) specifications.

In our proposed UMAP, the ALU mainly performs four logical bitwise operations: XOR, rotation (Rot), and recursive hash (R_h). Figure 3 shows the hardware design of ALU for the proposed protocol. For optimization, most efforts are concentrated on Rot and recursive hash (R_h) blocks. The remaining operators perform basic logical operations and there is not much room for optimization. Moreover the modules have been optimally reused through FSM (Finite State Machine) to reduce the number of gates utilization. The recursive hash (R_h) function is basically the combination of Rot and XOR operations, which requires seed, memory chunk selection (S_i), XORing, and then rotation of selected memory chunk (S_i) with itself.

7.2.1. Rotation Module. Rotation (Rot) is essential and only nontriangular function in ALU, which needs more optimization as remaining functions are basic logical operations (which can be only reused). The Rot(X, Y) is cyclic left rotation of X according to hamming weight of Y [10]. For $Y = [y_1, y_2, \ldots, y_m]$ each bit y_i is observed and if $y_i = 1$ then cyclic left rotation on X is performed $(x_{i+1}, \ldots, x_m, x_i)$; otherwise no operation is performed. The SASI using recursive hash protocol uses the rotation function with two variations: 8 bits for computation of recursive hash and m bits rotations (for other rotation calculations). To reduce the size of silicon, both rotation functions have been implemented within single rotation module. A wire "number of rotations" selects the partial (8 bits) or whole (m bits) registers for cyclic rotations. If we set number of rotations = 0, then it rotates 8 bits (recursive hash computations); otherwise it rotates m bits. The rotation module of m bit rotations and 8-bit rotations is shown Figures 4 and 5, receptively (8-bit rotation module is basically reuse of m-bit module). This module implements all (3), (4), and (6)–(10) which incorporate the rotation functions.

7.2.2. Recursive Hash Module (R_h). Recursive hash (R_h) module mainly performs three tasks:

(i) Decimates string (m-bits) into "S" memory chunks.

(ii) Selects memory chunk using computed seed.

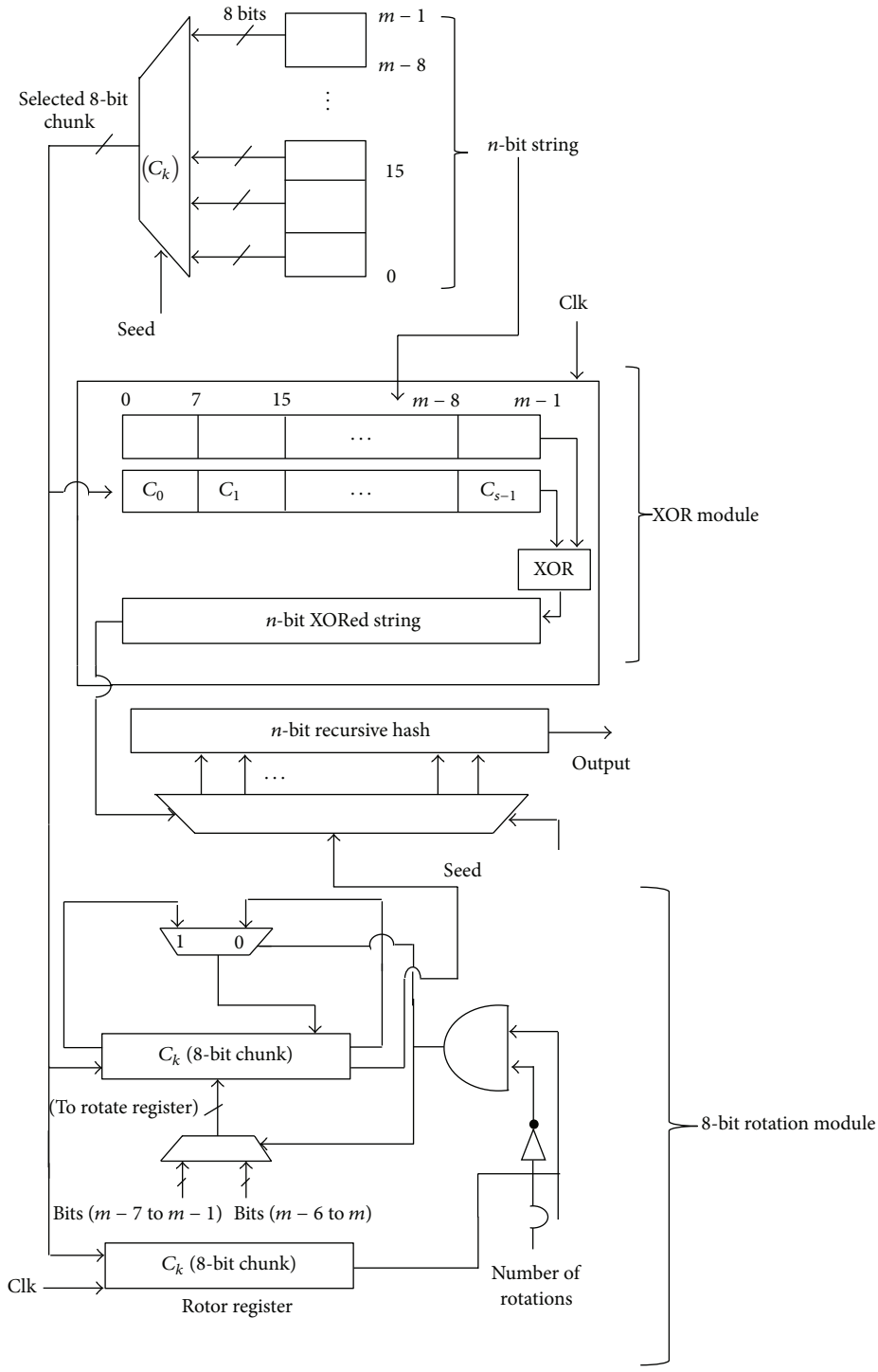

FIGURE 5: Recursive hash module.

(iii) Computes the final recursive hash using logical operations (XOR and Rot) of selected chunk with the whole string.

The detailed computation of recursive hash function has already been discussed in Section 3. Figure 5 shows the low-level design (architecture) of recursive hash module (RH-IN-SEL and RH-OUT). In Step 1, MUX selects the 8-bit memory chunk from the n-bit string (according the seed). In Step 2, selected chunk is forwarded to 8-bit rotation and XOR module. The 8-bit rotation module rotates the 8-bit selected chunk with itself. The XOR module takes XOR

Table 3: Resources utilizations of SASI using recursive hash design on FPGAs.

Target technology	Message length	Number of slice registers	Number of slice LUTs	Number of fully used LUT-FF
Spartan 6	32-bit	288	373	164
	64-bit	512	704	237
	96-bit	847	1003	484
Virtex-5	32-bit	297	402	194
	64-bit	630	841	301
	96-bit	879	1126	723

between $(S-1)$ copies of 8-bit selected chunk and n-bit string. Finally, in Step 3, the results of both modules are applied to demux (control-signal, seed), which injects (places) the result of rotation module at the specified location (8-bit zeroes) in XORed n-bit string. This module implements all (6)–(10) which incorporate recursive hash values of the variables. Figure 5 shows the architecture of recursive hash module.

7.3. Finite State Machine (FSM). The FSM mainly controls the data flow (communications) between various hardware components (ALU and registers) of the circuit. It is considered as an abstract machine that can be operated over finite number of states, where each state defines different computational tasks. In order to achieve miniaturized and cost effective hardware (chip), the optimal designing of FSM is very important. Initially, each tag is in "Idle" state and after receiving "Hello" message; the tag moves to the next state "Send IDS." The tag then proceeds to the next states (receive_A, receive_B, and receive_C) for reception of $A \parallel B \parallel C$ messages. The computation of pseudorandom numbers (n_1, n_2) requires six transition states and computation of each recursive hash function requires two states (XOR and rotation). After comparison of message "C" (received and locally computed values), the tag proceeds to the final states "computation of D state" and "*pseudonym and keys update* states."

7.4. Circuit Synthesis and Experimental Results. In this section, circuit synthesis and experimental results of the proposed design on FPGA (Field Programmable Gate Array) and ASIC (Application Specific Integrated Circuit) are presented. We have first described our proposed protocol in Visual C platform for initial resource estimation and rudimentary working. Then all hardware components of the proposed design were described in VHDL for EPC-C1G2 specified three different bit lengths (32, 64, and 96). The experimental setting, circuit synthesis, and simulation results for both FPGA and ASIC are as follows.

7.4.1. Hardware Implementation on FPGA. FPGA based instantiation and synthesis are performed in XILINX ISE Design Suite 12.3 environment for Spartan 6 and Virtex-6 FPGAs.

We have selected these FPGAs because of their extremely low power process technologies and optimized resource approximation. Spartan 6 FPGA is built on 45 nm low power copper process technology with dual flip-flops (FF) and efficient 6-input look-up tables (LUTs) [39]. Virtex-5 FPGA is considerably larger device, which is built on 65 nm copper CMOS process technology [40]. Table 3 shows the synthesis report (resource utilization) of the proposed designs on Spartan 6 and Virtex-5 FPGAs. We can observe from our synthesis results that increase in bit lengths increases the resource requirement on FPGAs (which means resources occupancy is independent of FPGA device). So, there is a trade-off between level of security robustness and hardware resources requirements, as if we increase the bit length then it provides more security but requires more resources and vice versa.

Martín et al. [41] also implemented and synthesized the similar EPC-C1G2 protocol (CRC-16 based) using PadGen function (for both MOD and XOR) on Virtex-5 XC5VLX30 and Altera Cyclone II. On Virtex-5, XOR scheme requires 599 register slices and 427 slice LUTs and for MOD Scheme 643 register slides and 599 slice LUTs are required. However our proposed UMAPs architecture (for 32-bit) requires less slice registers and slice LUTs on Virtex-5 XC5VLX30 FPGA (as listed in Table 3) which shows the preeminence of our protocols in terms of hardware than PadGen protocol.

7.4.2. Hardware Implementation on ASIC. We have used Leonardo Spectrum for ASIC implementation and resource estimation of our proposed architectures. TSMC (Taiwan Semiconductor Manufacturing Company) 0.35 μm library is used for circuit instantiation and synthesis of the proposed design.

For experimental setup, the operating frequency was set to 100 KHz for signal clock (as per EPC-C1G2 tag's requirement) while power supply is adjusted to 1.3 V. The number of gates (Gate Equivalents, GE) has been used for performance analysis of the proposed architecture. Table 4 summarizes the synthesis report (number of gates) of the proposed designs for 32-bit, 64-bit, and 96-bit lengths. We can observe that the area remains below 4000 GE for 32-bit (which is acquiescent with EPC-C1G2 tags) architecture of all three proposed designs. For larger message length, although the security of the protocol is enhanced, required resources exceed peripheral of ultralightweight class.

8. Conclusion

In this paper, we have proposed a novel ultralightweight mutual authentication protocol (UMAP): SASI using recursive hash. The proposed protocol involves two new

Table 4: Resources utilization of proposed protocol design for ASIC.

Message length	Number of gates			
	Basic logical operations	Rotation	Recursive hash	Total
32 bits	1884	1179	105	3168
64 bits	3198	1867	186	5251
96 bits	5306	2857	297	8460

nontriangular primitives: recursive hash (R_h) and double rotation function, which can be efficiently implemented on low cost passive RFID tags. The optimal design of the protocol avoids all the previous pitfalls and gives the best ultralightweight solution for EPC-C1G2 tags. Compared to the previously proposed UMAPs, which only approximated the hardware utilization of their protocols theoretically, this paper presents the proper hardware implementation using both FPGA and ASIC design flows giving the exact hardware resources utilization. We have also proposed optimized reusable Rot module to perform both variable length rotations and efficient FSM for circuit reuse to reduce the silicon area. Our results show that there is clear trade-off between circuit area and bit length of the protocol messages (directly relates to robust security), so optimization of one may interrupt the other.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. E. Tapiador, and A. Ribagorda, "LMAP: a real lightweight mutual authentication protocol for low-cost RFID tags," in *Proceedings of the 2nd Workshop on RFID Security*, pp. 100–112, Graz, Austria, July 2006.

[2] U. Mujahid and M. Najam-ul-Islam, "Ultralightweight cryptography for passive RFID systems," *International Journal of Communication Networks and Information Security*, vol. 6, no. 3, pp. 173–181, 2014.

[3] R. Das and P. Harrop, "RFID forecasts, players and opportunities 2014–2024," IDTechEx Technical Report, 2014.

[4] A. Klimov and A. Shamir, "New applications of T-functions in block ciphers and hash functions," in *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21–23, 2005, Revised Selected Papers*, vol. 3557 of *Lecture Notes in Computer Science*, pp. 18–31, Springer, Berlin, Germany, 2005.

[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 *Proceedings of the 1st OTM International Workshop on Information Security (IS '2006)*, pp. 352–361, Montpellier, France, 2006.

[6] P. Peris-Lopez, J. C. Hernandez- Castro, J. M. E. Estevez-Tapiador, and A. Ribagorda, "M^2AP: a minimalist mutual-authentication protocol for low-cost RFID tags," in *Ubiquitous Intelligence and Computing: Third International Conference, UIC 2006, Wuhan, China, September 3–6, 2006. Proceedings*, vol. 4159 of *Lecture Notes in Computer Science*, pp. 912–923, Springer, Berlin, Germany, 2006.

[7] T. Li and G. Wang, "Security analysis of two ultra-lightweight RFID authentication protocols," in *New Approaches for Security, Privacy and Trust in Complex Environments: Proceedings of the IFIP TC-11 22nd International Information Security Conference (SEC 2007), 14–16 May 2007, Sandton, South Africa*, vol. 232 of *IFIP International Federation for Information Processing*, pp. 109–120, Springer, New York, NY, USA, 2007.

[8] L. Tieyan and D. Robert, "Vulnerability analysis of EMAP-an efficient RFID mutual authentication protocol," in *Proceedings of the 2nd International Conference on Availability, Reliability and Security (AReS '07)*, pp. 238–245, Vienna, Austria, April 2007.

[9] 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.

[10] H.-M. Sun, W.-C. Ting, and K.-H. Wang, "On the security of Chien's ultralightweight RFID authentication protocol," *IEEE Transactions on Dependable & Secure Computing*, vol. 8, no. 2, pp. 315–317, 2011.

[11] G. Avoine, X. Carpent, and B. Martin, "Strong authentication and strong integrity (SASI) is not that strong," in *Radio Frequency Identification: Security and Privacy Issues: 6th International Workshop, RFIDSec 2010, Istanbul, Turkey, June 8-9, 2010, Revised Selected Papers*, vol. 6370 of *Lecture Notes in Computer Science*, pp. 50–64, Springer, Berlin, Germany, 2010.

[12] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. E. Tapiador, and A. Ribagorda, "Advances in ultralightweight cryptography for low-cost RFID tags: gossamer protocol," in *Information Security Applications: 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23–25, 2008, Revised Selected Papers*, vol. 5379 of *Lecture Notes in Computer Science*, pp. 56–68, 2009.

[13] Z. Bilal, A. Masood, and F. Kausar, "Security analysis of ultralightweight cryptographic protocol for low-cost RFID tags: gossamer protocol," in *Proceedings of the 12th International Conference on Network-Based Information Systems*, pp. 260–267, IEEE, Indianapolis, Ind, USA, August 2009.

[14] M. David and N. R. Prasad, "Providing strong security and high privacy in low-cost RFID networks," in *Security and Privacy in Mobile Information and Communication Systems: First International ICST Conference, MobiSec 2009, Turin, Italy, June 3-5, 2009, Revised Selected Papers*, vol. 17 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 172–179, Springer, Berlin, Germany, 2009.

[15] K.-H. Yeh, N. W. Lo, E. Winata et al., "An efficient ultralightweight authentication protocol for RFID systems," in *Proceedings of the Workshop on RFID Security and Privacy*, pp. 49–60, Istanbul, Turkey, June 2010.

[16] Y.-C. Lee, Y.-C. Hsieh, P.-S. You, and T.-C. Chen, "A new ultralightweight RFID protocol with mutual authentication,"

in *Proceedings of the WASE International Conference on Information Engineering (ICIE '09)*, vol. 2, pp. 58–61, IEEE, Taiyuan, China, July 2009.

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

[18] Z. Ahmadian, M. Salmasizadeh, and M. Reza Aref, "Desynchronization attack on RAPP ultralightweight authentication protocol," *Information Processing Letters*, vol. 113, no. 7, pp. 205–209, 2013.

[19] 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, https://eprint.iacr.org/2012/327.

[20] X. Zhuang, Y. Zhu, and C.-C. Chang, "A new ultralightweight RFID protocol for low-cost tags: R^2AP," *Wireless Personal Communications*, vol. 79, no. 3, pp. 1787–1802, 2014.

[21] I.-S. Jeon and E.-J. Yoon, "A new ultra-lightweight RFID authentication protocol using merge and separation operations," *International Journal of Mathematical Analysis*, vol. 7, no. 49–52, pp. 2583–2593, 2013.

[22] U. Mujahid, M. Najam-ul-Islam, and M. Ali Shami, "RCIA: a new ultralightweight RFID authentication protocol using Recursive hash," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 642180, 8 pages, 2015.

[23] M. Zubair, U. Mujahid, Najam-ul-Islam, and J. Ahmed, "Cryptanalysis of RFID ultra-lightweight protocols and comparison between its solutions approaches," *The Bahria University Journal of Information & Communication Technologies*, vol. 5, no. 1, pp. 58–63, 2012.

[24] G. Avoine, X. Carpent, and B. Martin, "Privacy-friendly synchronized ultralightweight authentication protocols in the storm," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 826–843, 2012.

[25] Z. Ahmadian, M. Salmasizadeh, and M. R. Aref, "Recursive linear and differential cryptanalysis of ultralightweight authentication protocols," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 7, pp. 1140–1151, 2013.

[26] D. F. Barrero, J. C. Hernández-Castro, P. Peris-Lopez, D. Camacho, and M. D. R-Moreno, "A genetic tango attack against the David-Prasad RFID ultra-lightweight authentication protocol," *Expert Systems*, vol. 31, no. 1, pp. 9–19, 2014.

[27] J. C. Hernandez-Castro, P. Peris-Lopez, R. C.-W. Phan, and J. M. E. Tapiador, "Cryptanalysis of the David-Prasad RFID ultralightweight authentication protocol," in *Proceedings of the Workshop on RFID Security and Privacy*, pp. 22–34, Istanbul, Turkey, June 2010.

[28] P. Peris-Lopez, J. C. Hernandez-Castro, R. C.-W. Phan, J. M. E. Tapiador, and T. Li, "Quasi-linear cryptanalysis of a secure RFID ultralightweight authentication protocol," in *Information Security and Cryptology: 6th International Conference, Inscrypt 2010, Shanghai, China, October 20–24, 2010, Revised Selected Papers*, vol. 6584 of *Lecture Notes in Computer Science*, pp. 427–442, Springer, Berlin, Germany, 2011.

[29] D. Han, "Gröbner basis attacks on lightweight RFID authentication protocols," *Journal of Information Processing Systems*, vol. 7, no. 4, pp. 691–706, 2011.

[30] P. D'Arco and A. De Santis, "On ultralightweight RFID authentication protocols," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 548–563, 2011.

[31] U. Mujahid, M. Najam-ul-islam, J. Ahmed, and U. Mujahid, "Cryptanalysis of ultralightweight RFID authentication protocol," Cryptology ePrint Archive, Report 2013/385, 2013, https://eprint.iacr.org/2013/385.

[32] D. Khovratovich and I. Nikolić, "Rotational cryptanalysis of ARX," in *Fast Software Encryption: 17th International Workshop, FSE 2010, Seoul, Korea, February 7–10, 2010, Revised Selected Papers*, vol. 6147 of *Lecture Notes in Computer Science*, pp. 333–346, Springer, Berlin, Germany, 2010.

[33] R. C.-W. Phan, "Cryptanalysis of a new ultralightweight RFID authentication protocol—SASI," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 4, pp. 316–320, 2009.

[34] Casper and FDR Tools Tutorial, http://www.cs.ox.ac.uk/gavin.lowe/Security/Casper/.

[35] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[36] J. Wessels and B. V. Cmg Finance, "Applications of BAN logic," 2001, http://www.win.tue.nl/ipa/archive/springdays-2001/ban-wessels.pdf.

[37] Pieter and Michiel, "Analysis of the OpenPGP and OTR protocols using GNY logic," Online Tutorial, http://www.ai.rug.nl/mas/finishedprojects/2007/PieterMichiel/gny.html.

[38] K.-H. Yeh and N. Lo, "Improvement of two lightweight RFID authentication protocols," *Information Assurance and Security Letters*, vol. 1, pp. 6–11, 2010.

[39] Xilinx, "Spartan-6 Family overview, DS160 (v2.0)," 2011, http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf.

[40] Xilinx, Virtex 5 Family overview, DS100 (v5.0) February 6, 2009, http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf.

[41] H. Martín, E. S. Millan, P. Peris-Lopez, and J. E. Tapiador, "Efficient ASIC implementation and analysis of Two EPC-C1G2 RFID authentication protocols," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3537–3547, 2013.