Journal of Circuits, Systems, and Computers
Vol. 27, No. 2 (2018) 1850033 (16 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218126618500330

World Scientific
www.worldscientific.com

Efficient Hardware Implementation of KMAP+: An Ultralightweight Mutual Authentication Protocol*

Umar Mujahid[†]

Department of Computer Science, Gwinnett Tech. ANF Campus, Alpharetta, GA 30009, United States
[†]*ukhokhar@gwinnetttech.edu*

M. Najam-ul-Islam[‡] and Madiha Khalid[§]

Department of Electrical Engineering, Bahria University, Islamabad, Pakistan
[‡]*najam@bahria.edu.pk*
[§]*madiha.zoheb@bui.edu.pk*

Received 29 December 2016
Accepted 26 May 2017
Published 7 July 2017

Internet of Things (IoTs) are becoming one of the integral parts of our lives, as all of the modern devices including pervasive systems use internet for its connectivity with the rest of the world. The Radio Frequency IDentification (RFID) provides unique identification and nonline of sight capabilities, therefore plays a very important role in development of IoTs. However, the RFID systems incorporate wireless channel for communication, therefore have some allied risks to the system from threat agents. In order to prevent the system from malicious activities in a cost effective way, numerous Ultralightweight Mutual Authentication Protocols (UMAPs) have been proposed since last decade. These UMAPs mainly involve simple bitwise logical operators such as XOR, AND, OR, etc., in their designs and can be implemented with extremely low cost RFID tags. However, most of the UMAP designers didn't provide the proper hardware approximations of their UMAPs and presented only theoretical results which mostly mislead the reader. In this paper, we have addressed this problem by reporting our experiences with FPGA and ASIC-based implementation of UMAP named psuedo Kasami code-based Mutual Authentication Protocol (KMAP+). Further, we have also improved the structure of the KMAP protocol to overcome the previously highlighted attack model. The hardware implementation results show that KMAP+ successfully conform to EPC-C1G2 tags and can be implemented using less than 4 K GE (for 32-bit word length).

Keywords: Ultralightweight; RFID; ASIC.

*This paper was recommended by Regional Editor Piero Malcovati.
[†]Corresponding author.

1. Introduction

Currently, the barcodes and the magnetic tapes are the widely deployed identification schemes in supply chain management and Europay, Mastercard & Visa (EMV) cards because of their extremely low cost. However, these identification systems have certain limitations e.g., these systems can't provide unique identification and are limited to Line of Sight (LoS) only. On the other side, the Radio Frequency IDentification (RFID) systems provide automatic and unique identification and operate on Radio Frequency (RF). Hence it can be operated over much larger range. The RFID systems mainly consists of three components: tag, reader and backend database. The tag acts as a transponder that can be implanted onto the objects which needs to be identified. The reader acts as a scanner and reads the contents of the tags while the backend database contains the detailed information of all associated readers and the tags. Usually, it is assumed that the channel between the reader and the backend database is secure, since there is no power constraint and therefore can be secured using traditional cryptographic algorithms. However, because of limited computational capabilities at tag side, only lightweight operations and algorithms can be used.

To ensure the security and privacy of the RFID systems in a cost effective manner, Pedro Peris Lopez[1] introduced a new field of cryptography named "Ultralightweight Cryptography" which efficiently addresses the security issues of extremely low cost IoTs specifically RFID. The Ultralightweight Protocols involves simple bitwise logical operations such as XOR, AND, OR, etc., in their designs that don't add much in overall cost of the tags.

In 2006, Pedro Paris *et al.*[1–3] proposed three new Ultra Lightweight Mutual Authentication Protocol (UMAPs): Lightweight Mutual Authentication Protocol (LMAP), Extremely Lightweight Mutual Authentication Protocol (EMAP) and Minimalist Mutual Authentication Protocol (M2AP). All of the three UMAPs involve simple bitwise logical operations (Triangular functions) in their designs and can be efficiently implemented within 1K logical gates. However, the authors provided only theoretical approximations of their UMAPs and didn't present any proper hardware implementations to justify their claims. Moreover, Tieyan *et al.*[4–6] exploited the poor diffusion properties of T-functions and highlighted many security attacks including desynchronization, Denial of Service (DoS) and full disclosure attacks for these UMAPs. Tieyan's security analysis raised many questions on the future and security claims of ultralightweight cryptography. In 2007, Chein[7] revised the definition of ultralightweight cryptography and suggested the incorporation of nontriangular function in protocol designs. Chein proposed a new ultralightweight nontriangular primitive "Rot" in its protocol: to provide Strong Integrity and Strong Authentication (SASI). Like predecessors (UMAP designers), Chein also presented a theoretical hardware approximation of the protocol and claimed that the "Rot" function is basically a left rotation function which requires only two registers for its proper implementation. However, the proper hardware implementation of the Rot function performed in Ref. 8 has shown the clear antinomy between the theoretical and practical approximations. The SASI protocol also found to be vulnerable against many desynchronization, traceability, DoS and full disclosure attacks[9–12] because of weak diffusion properties of protocol messages.

Later, many other UMAPs[13–18] e.g., GOASSMER, David-Prasad, RAPP, RCIA and R2AP, etc., were also proposed but shortly reported to be vulnerable against desynchronization or full disclosure attacks.[19–27]

The basic reason of this drastic failure is either using of poor security analysis model or informal (*adhoc*) verification tools to validate the security claims of the UMAPs. Recently, a new UMAP namely psuedo-Kasami code based Mutual Authentication Protocol (KMAP) has been proposed.[18] The KMAP protocol incorporates simple bitwise logical operators and lightweight version of Kasami codes in its design. The protocol successfully passes the rigorous and comprehensible security analysis framework and proves to be robust against all possible attack scenarios. But Masoumeh Safkhani and Nasour Bagheri[28] highlighted a desynchronization attack model that is applicable to many UMAPs including KMAP. The proposed desynchronization attack can make both the legitimate reader and the tag permanently desynchronize. In this paper, we have presented the modified version of KMAP protocol: KMAP+. The KMAP+ protocol introduces a novel variable updating mechanism which not only avoids the attack presented in Ref. 28 but all possible desynchronization attacks as well. The protocol avoids linear operations and involves only two ultralightweight operators: bitwise XOR and pseudo-Kasami encoder in its design.

The rest of the paper is organized as follows: Section 2 presents the related works which is followed by the KMAP+ protocol in Sec. 3. Section 4 describes the hardware architecture (top level design) of KMAP+ protocol and Sec. 5 discusses the hardware approximation results using both FPGA and ASIC design flows and finally Sec. 7 concludes the paper.

2. Related Works

One of the most important concerns that affects the commercialization of the UMAPs is their ambiguous hardware approximation. Most of the authors theoretically approximate the hardware utilization of their proposals and thus the hardware implementation of ultralightweight protocol has long been neglected. It was unclear whether such UMAPs are practically compatible with low cost passive EPC-C1G2[29] tags or not. Since last decade, only few researchers tried to fill this research gap. The summary of hardware implementation of RFID protocols are described as follows:

In 2010, Yu-Jung Huang *et al.*[30] presented the first proper hardware implementation of RFID authentication protocols. First, they implemented three pad-generation functions that can be incorporated in UMAP designs (to improve their security) and performed the comparative analysis on the basis of hardware utilization. Then the hardware implementation of mutual authentication protocol based on pad-generation function, tag access and kill password schemes has also been performed using Field Programmable Gate Array (FPGA) design flow. The Altera Cyclone II FPGA board has been used for their approximations. However, the problem of hardware approximation still remained unanswered, since the FPGA based implementation can give us the verification of the implementation but not the exact hardware approximation.

In 2012, Yu-Jung Huang *et al.*[31] improved their previous protocol design and proposed two new pad-Gen function based RFID authentication protocols. They used XOR and MOD operations in pad-Gen function designs for conformance to International Standards Organization 18000-6 Type-C protocols. Like their previous implementation, they used Altera Cyclone II FPGA for hardware approximations of their designs which again make the hardware approximation ambiguous.

In order to compute the exact number of gates and area of the chip, Honorio Martín *et al.*[32] used Application Specific Integrated Circuit (ASIC) platform to justify their results. They explored the design space and provided a detailed analysis of the area occupied by the synthesized circuits, their power consumption, and the throughput in terms of the proposed protocol. This was the first hardware implementation of lightweight Mutual Authentication Protocols using ASIC design flow. They implemented two EPCC1G2 protocols (Burmester-Munilla and Chien-Huang) for three different word lengths (32, 64 and 128 bits). Both the protocols incorporate Pseudo Random Number Generators (PRNGs) in their designs which enhance the diffusion properties of the exchanged messages. Therefore, the authors put most of the efforts in optimal designing of PRNGs (AKARI-I and AKAR-II).

Later many some other hardware implementations of UMAPs were also reported,[33,34] but almost most of them used FPGA design flow for verification of their results.

Recently, Zilong Liu *et al.*[35] implemented a new Variable Linear Feedback Shift Register (VLFSR) based lightweight authentication protocol using ASIC design flow. They mainly explored the power consumption and silicon area of the proposed ASIC, however they have also targeted the lightweight authentication protocols.

From the above discussion, we can observe that most of the protocol designers either use theoretical or FPGA-based hardware implementations for approximating their designs. Moreover, most of the UMAP designers don't use formal security analysis models to validate their UMAPs and therefore are reported to be vulnerable against many adversarial models. Recently, a new UMAP (KMAP) has been proposed[18] which extensively used the ultralightweight primitive pseudo-Kasami codes (K_c) in their design and eventually encrypts the secrets optimally. The authors used many formal and structural security analysis models to verify the robustness of the KMAP protocol. Although, desynchronization attack model presented in Ref. 28

affects the smooth functionality of the KMAP, however, that particular attack related to the structure of KMAP doesn't disclose any of the concealed secrets. In this paper, we first improve the structure and some of the mathematical equations of the KMAP protocol to avoid the possible desynchronization attacks and second present the hardware implementation of the improved KMAP; KMAP$^+$ uses both the ASIC and FPGA design flows.

3. KMAP$^+$ Protocol

In this section, we improve the structure of the KMAP protocol and propose a new UMAP; KMAP+ which keeps the robust security properties of the KMAP but avoids all the unbalanced triangular functions such as AND, OR, etc. The KMAP+ protocol involves two bitwise operations: XOR and left rotation (Rot) operations in its design. A new ultralightweight primitive: pseudo-Kasami code (K_c) has been extensively used in protocol messages which increases the diffusion properties of the protocol messages. The computation of pseudo-Kasami codes involves three simple steps and hence can be easily implemented with extremely low-cost RFID tags. These steps are:

(i) Extraction of pseudorandom numbers (n_1, n_2) from the protocol messages and then computation of seed for pseudo-Kasami coding.
(ii) Left rotate the selected number of the bits (starting from LSB) which in return generates another string.
(iii) Take XOR between the rotated string and the original string to get the pseudo-Kasami code.

To better understand the computation of pseudo-Kasami code, consider the following example (with reduced bit size):

Example: Assume 8-bit string 0 0 1 0 1 0 1 1 and seed = 3, then according to 2nd step, the new string will be

$$\boxed{0\,1\,1}\,0\,0\,1\,0\,1.$$

We can observe that the new string, is basically the shifted version of string (last 3 bits left rotated). Now, to get the final result, we just need to take XOR between both strings.

$$\begin{array}{r} 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1 \\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1 \\ \hline K_c = 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0 \end{array}$$

Figure 1 presents the theoretical framework of pseudo-Kasami encoder.

Since, there is no resource constraint at the reader' side, therefore to avoid desynchronization attacks, we have proposed a novel memory mechanism to store

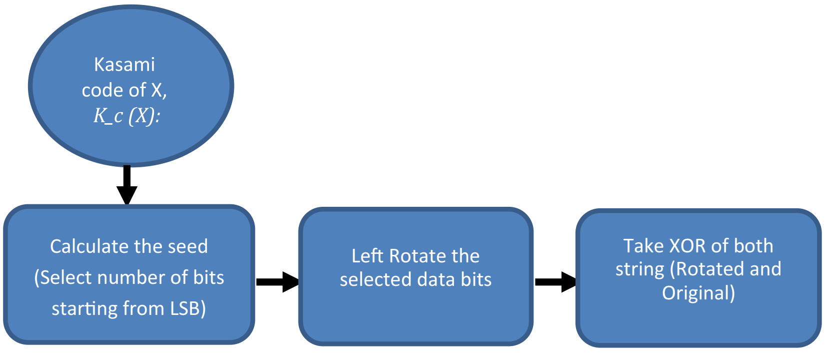
Kasami code of X, K_c (X):
Calculate the seed (Select number of bits starting from LSB)
Left Rotate the selected data bits
Take XOR of both string (Rotated and Original)

Fig. 1. Theoretical framework of pseudo-kasami encoder.

variables at the reader/backend database side. In most of the desynchronization attacks, the adversary blocks the last communication message which is either the reader sends to the tag or the tag sends to the reader for final authentication and variables updating. So, the next time, when the reader again enquires such tags, then it responds with its IDS^{Old} instead of the updated one. In our scheme, whenever the reader receives the old value of the IDS, then it will do two main tasks:

(1) Uses old variables for authentication with the tag.
(2) Increases its memory size and will not discard the updated variables.

It means, if an adversary blocks one update message (last communication message), the reader will have $\text{IDS}^i, \text{IDS}^{i-1}, \text{IDS}^{i+1}$ in its database. Similarly, after second blocking, the reader's database will be $\text{IDS}^i, \text{IDS}^{i-1}, \text{IDS}^{i+1}, \text{IDS}^{i+2}$. In case of smooth communication between the reader and the tag, this database will update current and the previous IDS otherwise will increase sequentially. By using this mechanism, both the reader and the tag will always remain synchronized regardless of the adversarial strength. The presented model is generic in nature and applicable to any UMAP.

Like KMAP, KMAP$^+$ also involves two main components: Reader ($\mathcal{R}$) and tag ($\mathcal{T}$). Both the $\mathcal{R}$ and $\mathcal{T}$, pre-share index Pseudonyms (IDS), keys (K_1, K_2) and secret ID. Figure 2 shows the detailed specification of the KMAP$^+$.

The working of KMAP$^+$ is as follows:

Step 1: The reader ($\mathcal{R}$) continuously broadcasts beacon signals, so whenever a tag ($\mathcal{T}$) enters in its vicinity, it receives a query ($\mathcal{P}$) from $\mathcal{R}$.

$$\mathcal{R} \rightarrow \mathcal{T} : \mathcal{P}$$

Step 2: Upon receiving the query ($\mathcal{P}$), the $\mathcal{T}$ responds with its current IDS.

$$\mathcal{T} \rightarrow \mathcal{R} : IDS$$

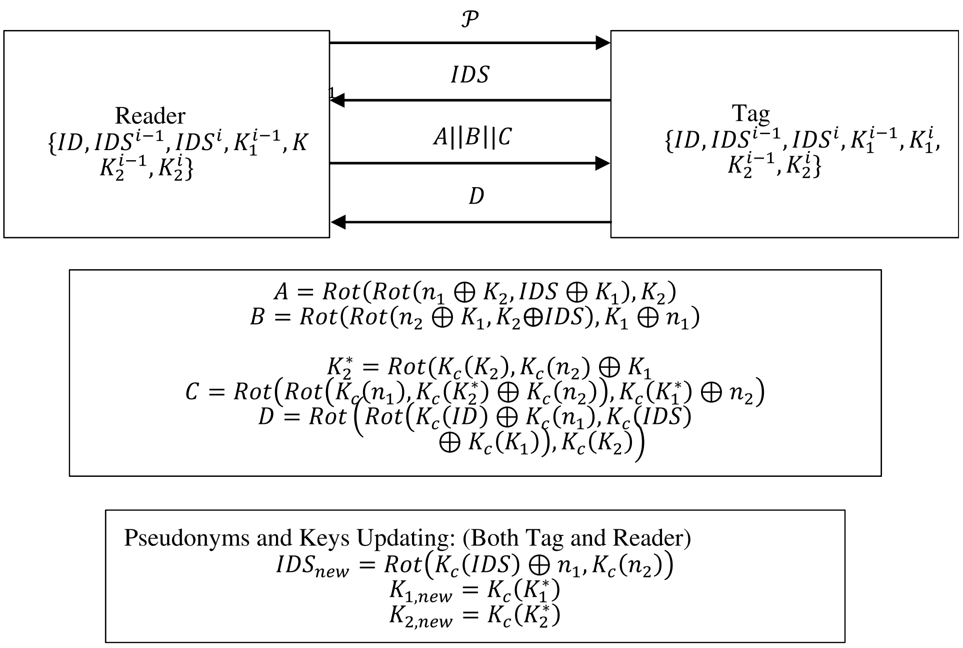
$\mathcal{P}$
IDS
$A||B||C$
D
Reader
$\{ID, IDS^{i-1}, IDS^i, K_1^{i-1}, K$
$K_2^{i-1}, K_2^i\}$
Tag
$\{ID, IDS^{i-1}, IDS^i, K_1^{i-1}, K_1^i,$
$K_2^{i-1}, K_2^i\}$
$A = Rot(Rot(n_1 \oplus K_2, IDS \oplus K_1), K_2)$
$B = Rot(Rot(n_2 \oplus K_1, K_2 \oplus IDS), K_1 \oplus n_1)$
$K_2^* = Rot(K_c(K_2), K_c(n_2) \oplus K_1$
$C = Rot\big(Rot\big(K_c(n_1), K_c(K_2^*) \oplus K_c(n_2)\big), K_c(K_1^*) \oplus n_2\big)$
$D = Rot\Big(Rot\big(K_c(ID) \oplus K_c(n_1), K_c(IDS) \oplus K_c(K_1)\big), K_c(K_2)\Big)$
Pseudonyms and Keys Updating: (Both Tag and Reader)
$IDS_{new} = Rot\big(K_c(IDS) \oplus n_1, K_c(n_2)\big)$
$K_{1,new} = K_c(K_1^*)$
$K_{2,new} = K_c(K_2^*)$

Fig. 2. The KMAP$^+$ protocol.

Step 3: The $\mathcal{R}$ uses the novel mechanism and compares the received IDS with its database and if a match occurs, then $\mathcal{R}$ generates messages A, B and C and sends to $\mathcal{T}$:

$$A = \mathrm{Rot}(\mathrm{Rot}(n_1 \oplus K_2, IDS \oplus K_1), K_2) \tag{1}$$

$$B = \mathrm{Rot}(\mathrm{Rot}(n_2 \oplus K_1, K_2 \oplus IDS), K_1 \oplus n_1) \tag{2}$$

$$K_1^* = \mathrm{Rot}(K_c(K_1), K_c(n_1)) \oplus K_2 \tag{3}$$

$$K_2^* = \mathrm{Rot}(K_c(K_2), K_c(n_2) \oplus K_1 \tag{4}$$

$$C = \mathrm{Rot}(\mathrm{Rot}(K_c(n_1), K_c(K_2^*) \oplus K_c(n_2)), K_c(K_1^*) \oplus n_2). \tag{5}$$

$$\mathcal{R} \to \mathcal{T} : A, B \;\&\; C$$

However, if a match doesn't occur, then $\mathcal{R}$ resends $\mathcal{P}$ and expects $\mathrm{IDS}^{\mathrm{OLD}}$. If $\mathcal{R}$ receives $\mathrm{IDS}^{\mathrm{OLD}}$ in the second attempt, then the protocol will work smoothly, otherwise, $\mathcal{R}$ will follow the novel memory mechanism.

Step 4: On successful reception of messages A, B and C the T first checks the authenticity of the sender and then responds to the authentic sender only. The $\mathcal{T}$ involves following four steps to validate the authenticity of the $\mathcal{R}$:

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

$$n_1 = \mathrm{Rot}^{-1}(\mathrm{Rot}^{-1}(A, K_2), \mathrm{IDS} \oplus K_1) \oplus K_2 \tag{6}$$

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

(2) Calculate the seed $(\mathcal{S})$ for the computation of C message:

$$\mathcal{M} = n_1 \oplus n_2 \tag{8}$$

$$\mathcal{S} = \mathrm{wt}(M) \bmod K. \tag{9}$$

Here (wt) represents the hamming weight.

(3) After computation of the message $\mathcal{C}$ (using Eq. (5)), the $\mathcal{T}$ compares locally computed $\mathcal{C}$ with the received $\mathcal{C}$ (sent in step 3). If both values coincide, then $\mathcal{T}$ believes that it is communicating with the legitimate $\mathcal{R}$. Otherwise, $\mathcal{T}$ will consider the sender as an illegitimate $\mathcal{R}$ and will abort its protocol session.

(4) After successful authentication, T conceals its secret ID in message D and sends to R:

$$D = \mathrm{Rot}(\mathrm{Rot}(K_c(ID) \oplus K_c(n_1), K_c(\mathrm{IDS}) \oplus K_c(K_1)), K_c(K_2)). \tag{10}$$

$$\mathcal{R} \to \mathcal{T} : D$$

$\mathcal{T}$ also updates its IDS and keys for next interaction using the following Equations:

$$\mathrm{IDS}_{\mathrm{new}} = \mathrm{Rot}(K_c(\mathrm{IDS}) \oplus n_1, K_c(n_2)) \tag{11}$$

$$K_{1,\mathrm{new}} = K_c(K_1^*) \tag{12}$$

$$K_{2,\mathrm{new}} = K_c(K_2^*). \tag{13}$$

Step 5: Finally, $\mathcal{R}$ authenticates $\mathcal{T}$, by checking the correctness of message D and if received D coincides with the locally computed D, then $\mathcal{R}$ will also update its IDS and keys using (11–13).

Since KMAP$^+$ involves the similar internal structure (encryption mechanism) as that of KMAP, therefore, it can satisfy all the formal and structure security analysis models discussed in Ref. 18.

4. Hardware Architecture of UMAPs

In this section, we present the hardware architecture (Top level module) of the KMAP+ protocol. The presented hardware architecture is similar to generic architecture presented in Ref. 8. For efficient and compact hardware implementation (4 K GE), the low-level designs of internal components are optimally designed and logical components have been extensively reused. The top-level design of KMAP+ mainly involves three components: Arithmetic Logic Unit (ALU), Register block and Finite State Machine. The detailed working of all components is described as follows:

4.1. *Register block*

This block contains all the registers (memory blocks) required to store intermediate computations (results), permanent variables and long-term values.

The KMAP protocol stores two copies of $IDS(IDS_{\text{new}}, IDS_{\text{old}})$ and keys $(K_{(1,\text{old})}, K_{(2,\text{old})}, K_{(1,\text{new})}, K_{(2,\text{new})})$ to avoid the desynchronization attacks. Therefore, the KMAP$^+$ protocol requires $8L$ dynamic memory to store its pseudonym, keys (both old and new) and pseudorandom numbers (n_1, n_2) received from the reader. It also requires $1L$ static memory to store its secret ID. For internal logical operations, we use eight General Purpose (GP) registers $(\text{GP}_1, \text{GP}_2, \text{GP}_3, \ldots, \text{GP}_7)$ of L bits; which hold the intermediate results of ongoing computations and will be reused after their previous task.

4.2. *ALU block*

The ALU block mainly comprises of bitwise logical operators (protocol specific) and performs the specified computational operations. Hence the designing of ALU block entirely depends upon the protocol (operational) specifications. As far as optimization is concerned, most of the efforts concentre in designing of cost effective ALU block. In KMAP$^+$, ALU mainly performs four logical bitwise operations: AND, XOR Rotation (Rot) and pseudo Kasami coding (K_c).

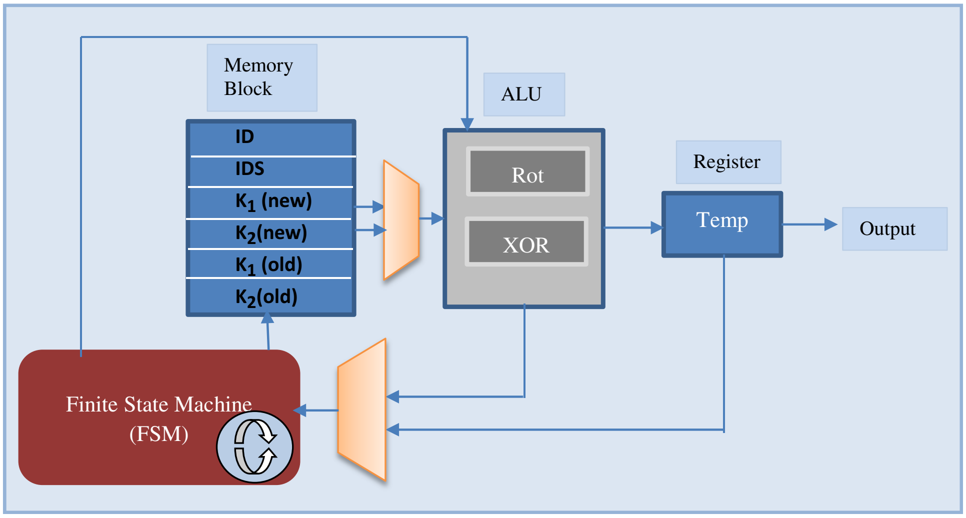

Fig. 3. Top level design of KMAP$^+$.

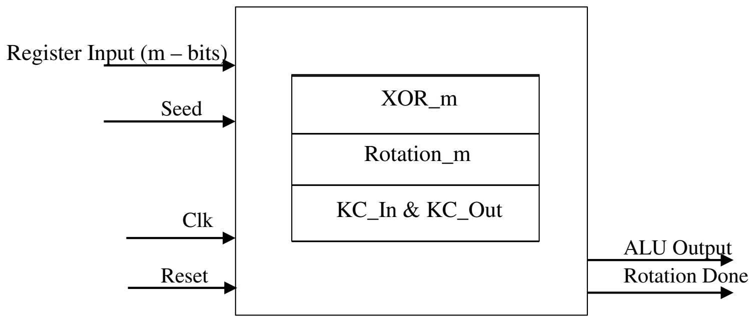

Fig. 4. Top level design of ALU (KMAP).

The KMAP$^+$ protocol incorporates only two nontriangular ultralightweight primitives: Rotation and pseudo Kasami coding. The efficient designing of these primitives plays an important role in overall optimization of the hardware, since the remaining operators are basic logical operators and can't be further optimized.

For optimization of ALU block, we use an efficient Rotation module [8] and propose an efficient architecture for pseudo Kasami encoder. The low level designs of rotation and novel pseudo Kasami encoder are presented in Figs. 5 and 6.

4.2.1. *Rotation module*

Typically a rotation module requires two internal modules (one hamming weight module and one barrel shifter) for its proper execution. However, our rotation module is clock-based which just left shift the "Rotor" string by checking its MSB. If value at MSB is '1', then the value stored at MSB of "to rotate" string will be rotated left otherwise no operation will be performed. Here, we have used two string: "Rotor"

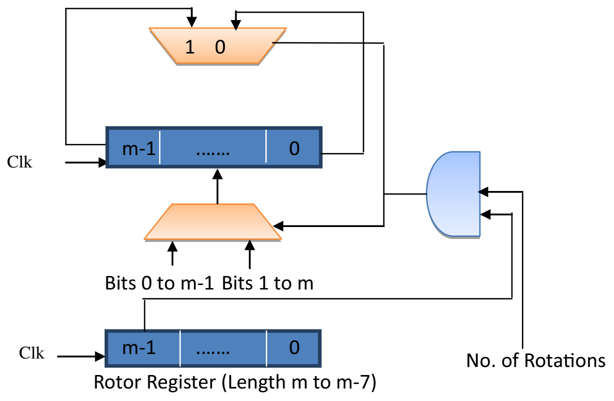

Fig. 5. Low level design of rotation module.

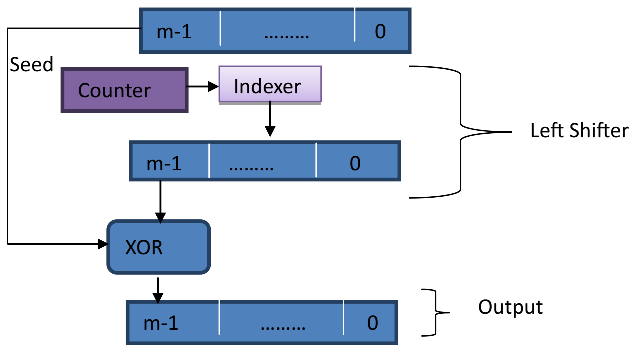

Fig. 6. Low level design of pseudo-kasami encoder module.

and "to rotate". The value of the "Rotor" string causes rotations in the "to rotate" string. Figure 5 shows the low-level design of Rotation module.

4.2.2. *Kasami encoder module*

The pseudo Kasami encoder mainly performs two tasks:

- Shift (left rotate) the string (m-bits) according to the computed seed.
- Take XOR between shifted string and original string.

The detailed working of the pseudo Kasami encoder has already been described in Sec. 2. Figure 6 presents the optimized low-level design of the pseudo Kasami encoder. In the proposed architecture, the computed seed updates the counter $(k+1)$ which gives pilot information to indexer/ shifter to reserve the $(k+1)$ bit positions in m-bits memory. Further, the indexer places the $(k+1)$ bits (starting from LSB) of S memory block (whom pseudo Kasami code is to be taken) at reserved bit positions. Then after taking the XOR between shifted memory block (Temp register) and S memory block, we get the final m-bit pseudo Kasami code of the variable. In order to reduce the size and cost of the system, we have used bitwise shifting method (controlled through FSM) instead of barrel shifter. This module implements all operations, which involve pseudo Kasami codes of variables.

4.3. ***Finite state machine (FSM)***

A Finite State Machine (FSM) is basically a mathematical model of computation which is used to design optimal sequential logic circuits. 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 in one of the finite number of states, where each state defines different computational

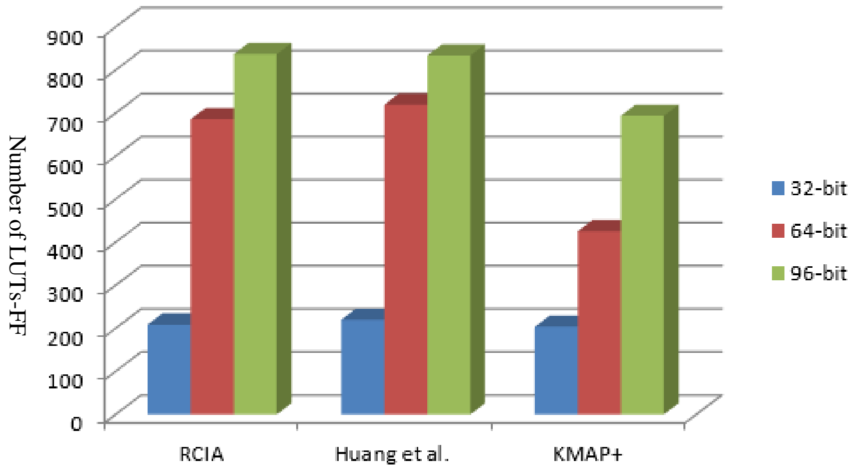

Fig. 7. Comparison of LUTs FF.

tasks. In KMAP protocol, each tag initially resides in 'Idle' state and after receiving the reader's query, the tag moves to the 'Send IDS' state and transmits its current IDS. In order to receive $A||B||C$ messages (from reader), the tag proceeds to the next states (received$_A$, received$_B$, received_C). The computation of pseudorandom numbers (n_1, n_2) requires seven transition states and the computation of each pseudo Kasami code requires only two states (Indexing and *XORing*). After comparison of message "C" (received and locally computed values), the FSM requires two states for the computation of message "D" and two states for pseudonym and keys updating.

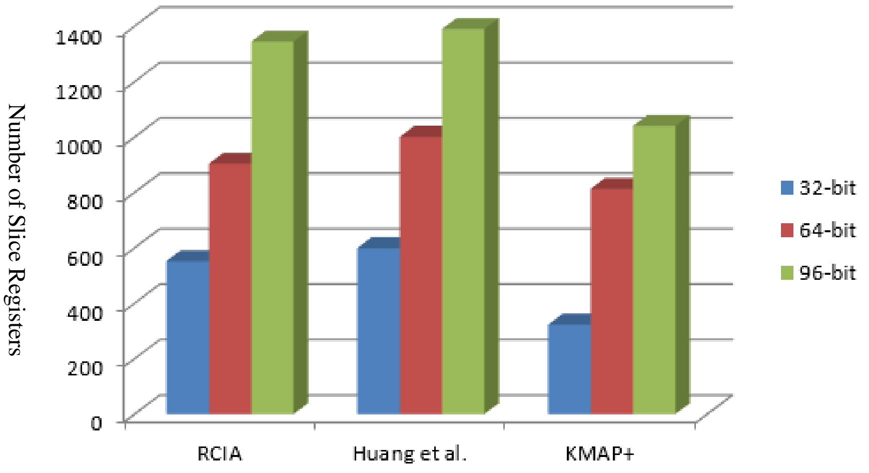

Fig. 8. Comparison of slice registers.

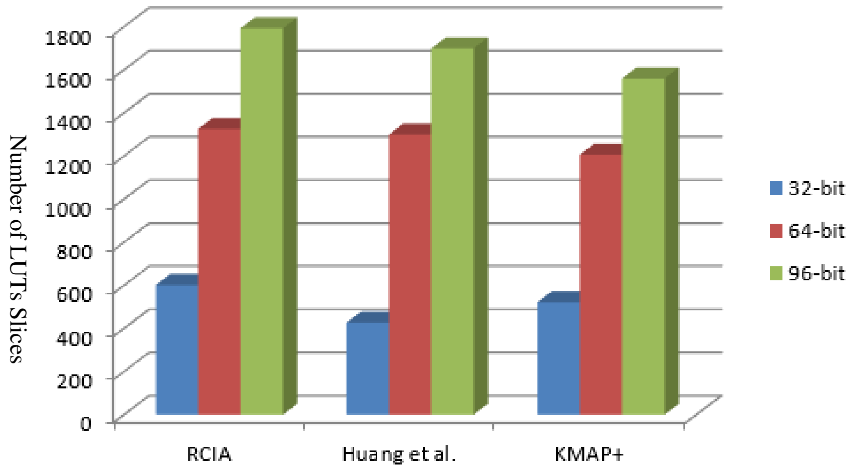

Fig. 9. Comparison of LUTs slices.

5. Circuit Synthesis and Experimental Results

In this section, circuit synthesis and experimental results of the proposed design on Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) are presented. The FPGA provides the verification of the proposed design and ASIC implementation gives the exact resources estimation. The detailed results on both platforms are described as follows:

5.1. *Experimentation results on FPGA*

We have used Xilinx 12.3 Design Suite and the targeted device was Virtex 6. The main reason for selection this device is the fair comparison of our design with RCIA and Huang *et al.* protocols.[53,55] We have implemented our proposed design for all of the three different bit lengths (32-bits, 64-bits, 96-bits) specified by EPCglobal.[29]

The implementation on Virtex-6 device for 32-bit architecture, the KMAP protocol occupies 312 register slices, 522 Look Up Tables (LUTs) and 188 fully used LUT-FF pairs while the RCIA protocol occupies 438 register slices, 684 Look Up Tables (LUTs) and 205 fully used LUT-FF pairs. Huang *et al.*[31] design requires 599 register slices and 427 LUTs. Furthermore for 64-bit architecture, the RCIA protocol requires 889 registers slices, 1556 slice LUTs and 665 fully used LUT-FF pairs. However, the KMAP protocol requires fewer resources and outperforms on RCIA and Huang *et al.* for 96-bit length as well. This leading pattern in terms of less resources occupancy of KMAP remains same, no matter if we use Virtex or Spartan FPGAs. This clearly states that the occupancy of resources is independent of the FPGA devices. The detailed comparison of results is presented in Figs. 7–9.

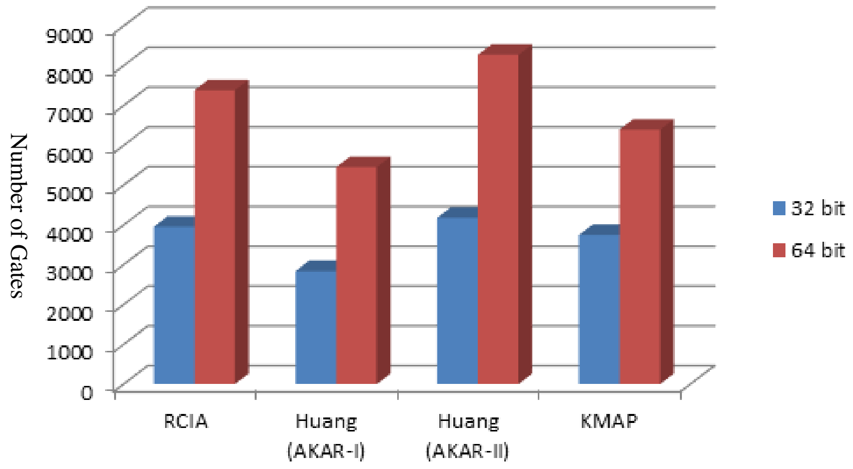

Fig. 10. Comparison of GEs.

5.2. *Experimentation results on ASIC*

For ASIC based resources approximation, we have used Leonardo Spectrum and results were generated with Taiwan Semiconductor Manufacturing Company (TSMC) 0.35 μm library.

Figure 10 shows the comparison of number of gates required to implement KMAP, RCIA and Huang *et al.* (using AKARI-I and AKARI-II). We can observe from the results that KMAP protocol proves to be more economical than RCIA and Huang *et al.* (AKARI-II). Although, Huang *et al.* with AKARI-I requires less resources than KMAP, incorporation of AKARI-I (for generation of randomness) makes protocol less resistive against active adversarial models.[10]

6. Conclusion

One of the most important concerns of RFID is the cost effective RFID tag. Several researchers proposed UMAPs to provide the secure and economical RFID tags. However, only few of them have practically implemented their designs and gave exact picture of required resources and moreover almost all of the previous proposed UMAPs were found to be vulnerable against many security attacks. In this paper, we have addressed this research gap and made two main contributions. Firstly, we have proposed the state of the art UMAP; KMAP+ (advanced version of KMAP protocol) which avoid all the previously highlighted pitfalls of UMAPs and introduces novel desynchronization avoidance mechanism without adding any payload at tag side. Secondly, since this area lags the proper hardware implementations of UMAPs, therefore we used both the FPGA and ASIC platforms for implementations of KMAP+ protocol. We have proposed an optimized design for pseudo-Kasami encoder and reused the modules and wires as much as possible to give an extremely lightweight tag. The ASIC results shows that our proposed design successfully conforms to EPC C1G2 standards and can be implemented within 4K GE constraint. The comparison with existing UMAP designs shows the clear prominence of our proposed design among others.

References

1. P. Peris-Lopez *et al.*, LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags, *Proc. 2nd Workshop RFID Security*, (Austria, 2006), pp. 100–112.
2. P. Peris-Lopez *et al.*, EMAP: An efficient mutual-authentication protocol for low-cost RFID tags, *1st Int. Workshop Information Security (OTM-2006)*, (France, 2006), pp. 352–361.
3. P. Peris-Lopez *et al.*, M2AP: A minimalist mutual-authentication protocol for low cost RFID tags, *6th Int. Conf. Ubiquitous Intelligence and Computing* (Australia, 2006), pp. 912–923.
4. T. Li and G. Wang, Security analysis of two ultra-lightweight RFID authentication protocols, *International Information Security Conference (SEC)* (South Africa, 2007), pp. 109–120.
5. T. Li *et al.*, Vulnerability Analysis of EMAP-An Efficient RFID Mutual Authentication Protocol, *2nd Int. Conf. Availability, Reliability and Security (ARES)*, Austria, 2007, pp. 238–245.
6. T. Li and G. Wang, Security analysis of family of ultra-lightweight RFID authentication protocols, *J. Softw.* **3** (2008) 1–10.
7. H.-Y. Chien, SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity, *IEEE Trans. Dependable Secure Comput* **4** (2007) 337–340.
8. U. Mujahid *et al.*, Efficient hardware implementation of ultralightweight RFID mutual authentication protocol, *J. Circuit Syst. Comput.* **25** (2016) 1650078 [19 pages] doi: http://dx.doi.org/10.1142/S021812661650078X.
9. G. Avoine, X. Gildas, Carpent and B. Martin, Strong authentication and strong integrity (SASI) is not that strong, *6th Int. Conf. RFID Security and Privacy*, Turkey, (2010) pp. 50–64.
10. Z. Ahmadian, M. Salmanzadeh *et al.*, Recursive linear and differential cryptanalysis of ultralightweight authentication protocols, *IEEE Trans. Inf. Forensics Sec.* **8** (2013) 1140–1151.
11. P. Peris-Lopez *et al.*, Quasi-linear cryptanalysis of a secure RFID ultralightweight authentication protocol, *6th Int. Conf. Information Security and Cryptology*, China, (2011) pp. 427–442.
12. H. Daewan, Gröbner basis attacks on lightweight RFID authentication protocols, *J. Inf. Process. Syst.* **7** (2011) 691–706.
13. P. Peris-Lopez *et al.*, Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol, *9th Int. Workshop Information Security Applications*, (Korea, 2009), pp. 56–68.
14. D. Mathieu and N. R. Prasad, Providing strong security and high privacy in low-cost RFID networks, *Int. Conf. Security and Privacy in Mobile Information and Communication Systems* (Italy, 2009), pp. 172–179.

15. Y. Tian *et al.*, A new ultralightweight RFID authentication protocol with permutation, *IEEE Commun. Lett.* **16** (2012) 702–705.
16. U. Mujahid *et al.*, RCIA: A new ultralightweight RFID authentication protocol using recursive hash, *Int. J. Distrib. Sens. Netw.* **2015**, Article Id 642180, (2015). doi: 10.1155/2015/642180.
17. X. Zhuang *et al.*, A new ultralightweight RFID protocol for low-cost tags: R ^{2}AP, *Wirel. Pers. Commun.* **79** (2014) 1787–1802.
18. U. Mujahid *et al.*, A new ultralightweight RFID authentication protocol for passive low cost tags: KMAP, *Wirel. Pers. Commun.* (2016). doi: 10.1007/s11277-016-3647-4.
19. Z. Bilal *et al.*, Multiple attacks on authentication protocols for low-cost RFID tags, *Appl. Math. Inf. Sci.* **9** (2015) 561–569.
20. S. Jeon *et al.*, Cryptanalysis and improvement of a new ultra-lightweight RFID authentication protocol with permutation, *Appl. Math. Sci.* **7** (2013) 3433–3444.
21. H. M. Sun, W. C. Ting and K. H. Wang, On the security of chien's ultralightweight RFID authentication protocol, *IEEE Trans. Dependable Secur. Comput.* **8** (2011) 315–317.
22. Z. Ahmadian *et al.*, Desynchronization attack on RAPP ultralightweight authentication protocol, *Inf. Process. Lett.* **113** (2013) 205–209.
23. J. C. Hernandez-Castro *et al.*, Cryptanalysis of the david-prasad RFID ultralightweight authentication protocol, *6th International Conference on Radio Frequency Identification: Security and Privacy Issues (ACM)*, (2010), pp. 22–34.
24. D. F. Barrero *et al.*, A genetic tango attack against the David–Prasad RFID ultra-lightweight authentication protocol, *Expert Syst.* **31** (2014) 9–19.
25. P. D'Arco *et al.*, On ultralightweight RFID authentication protocols, *IEEE Trans Dependable Secur. Comput* **8** (2011) 548–563.
26. W. Shao-hui *et al.*, Security analysis of RAPP: An RFID authentication protocol based on permutation, Cryptology ePrint Archive, Report 2012/327, https://eprint.iacr.org/2012/327, 2012.
27. M. Zubair *et al.*, Cryptanalysis of RFID ultra-lightweight protocols and comparison between its solutions approaches, *BUJICT J.* **5** (2012) 58–63.
28. M. Safkhani and N. Bagheri, Generalized Desynchronization attack on UMAP:Application to RCIA, KMAP SLAP and SASI++ protocols, *Cryptology* ePrint Archive: Report 2016/905.
29. GS1 EPCglobal tag data standards version 1.4, Available from: http//www.epcglobalinc.org/standards/.
30. Y. J. Huang *et al.*, Hardware implementation of RFID mutual authentication protocol, *IEEE Trans. Ind. Electron.* **57** (2010) 1573–1582.
31. Y. J. Huang *et al.*, Efficient implementation of RFID mutual authentication protocol, *IEEE Trans. Ind. Electron.* **59** (2012) 4784–4791.
32. H. Martín *et al.*, Efficient ASIC implementation and analysis of two EPC-C1G2 RFID authentication protocols, *IEEE Sens. J.* **13** (2013) 3537–3547.
33. Q. Ain and M. Umar, M. Najam-ul-Islam, Hardware implementation of ultralightweight cryptographic protocols, *Int. Conf. Computing, Communication and Security* (*ICCCS*), Lahore, 2015.
34. J. Wu and M. O'Neill, Ultra-lightweight true random number generators, *IET Electron. Lett.* **46** (2010) 988–990.
35. Z. Liu, D. Liu *et al.*, Implementation of a new RFID authentication protocol for EPC Gen2 standard, *IEEE Sens. J.* **15** (2015) 1003–1011.