Global Journal of Computer Science and Technology: E
Network, Web & Security
Volume 17 Issue 5 Version 1.0 Year 2017
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Symmetric Encryption Algorithm for RFID Systems using a Dynamic Generation of Key

By Zouheir Labbi, Mohamed Senhadji, Ahmed Maarof & Mostafa Belkasmi

Mohammed V University

Abstract- The security of RFID systems come to be a major subject today, notably for low cost RFID tags. Multiple Cryptographic algorithms were proposed to insure the information security and in the same time respect the resource-constrained RFID systems (limited computational, storage capability…). In this paper, we proposed a hybrid encryption approach as symmetric key encryption that generate the key dynamically, together with integrity check factors. The key stream generation step follows the chained method, starting from the initial pre-shared key. The approach uses logical operations on intermediate cipher texts and intermediate Keys generated in each stage (named derive function), to generate successive keys. This is improving the chaining technique which use only cipher text to generate the successive keys. Finally, our approach will diminish the computational complexity as well as improve performance.

Keywords: encryption, RFID, dynamic key, integrity, hashing algorithm.

GJCST-E Classification: E.3

Symmetric Encryption Algorithm for RFID Systems using a Dynamic Generation of Key

Zouheir Labbi [α], Mohamed Senhadji [σ], Ahmed Maarof [ρ] & Mostafa Belkasmi [ω]

Abstract- The security of RFID systems come to be a major subject today, notably for low cost RFID tags. Multiple Cryptographic algorithms were proposed to insure the information security and in the same time respect the resource-constrained RFID systems (limited computational, storage capability…). In this paper, we proposed a hybrid encryption approach as symmetric key encryption that generate the key dynamically, together with integrity check factors. The key stream generation step follows the chained method, starting from the initial pre-shared key. The approach uses logical operations on intermediate cipher texts and intermediate Keys generated in each stage (named derive function), to generate successive keys. This is improving the chaining technique which use only cipher text to generate the successive keys. Finally, our approach will diminish the computational complexity as well as improve performance.

Keywords: encryption, RFID, dynamic key, integrity, hashing algorithm.

I. Introduction

RFID (Radio Frequency Identification) system is becoming quite popular and very useful and convenient to identify objects automatically through wireless communication channel. However, since this channel is not secure from the various security violations and attacks, a security mechanism is needed which necessitating the use of data encryption algorithms [1]. However, the storage capabilities and less power of low cost RFID tags restrict and limit to perform sophisticated cryptographic operations [2] like RSA, AES and DES which provide a strong confidentiality. They are very heavy algorithms in terms of computation costs and required high computing capacity and they do not support authentication and integrity alone. As a result, several lightweight encryption algorithms were proposed by researchers to develop new solutions to the challenging problem of providing security to tags with restricted resources (for example RBS (Redundant Bit Security algorithm) [3], Hummingbird [4], etc).

Basically, the security can be reached by encrypting the message using cryptographic algorithms. In symmetric key Encryption, the initial message is divided into blocks of identical sizes using Block ciphers and will be encrypted with stream Ciphers bit-by-bit using the logical XOR operation.

Author α σ ρ Ѡ: Laboratory of Information, Communication and Embedded Systems (ICES), Mohammed V University, ENSIAS, Rabat, Morocco. e-mails: zouhir.labbi@gmail.com, m.senhadji@um5s.net.ma, ahmed.maarof@gmail.com, m.belkasmi@um5s.net.ma

The key stream can be remained the same or derived from initial key using function such as Pseudo Random Number Generator (PRNG) [5]. But this technique is vulnerable to attacks such as distinguishing and key recovery attacks [6]. The key steam bits are generated by stream ciphers using nbit Linear Feedback Shift Register (LFSR). The inconvenience with LFSR is that an n-bit pattern can be repeated in the key stream before reaching 2 ▪ possible cases [7]. To resolve this Weakness, a Non-Linear Feedback Shift Register (NLFSR) has been developed. But there are still no generic designs for NLFSR [8]. In addition, in block ciphers, all keys are derived from the initial key, therefore, recover the initial key (which we know to be the secret key) provides information about the original message. To dominate this Weakness, new technique has been used to generate multiple keys dynamically.

Engels et al. [4] proposed a symmetric key Encryption (SKE) algorithm named Hummingbird as hybrid encryption of the block cipher and key stream dedicated for RFID systems. This algorithm uses the similar classic encryption method of block ciphers such as looping and substitution. whereas, the key is derived via the stream cipher principle which keep this encryption process computationally expensive. Nevertheless, our approach will be compared with Hummingbird without inherit any feature from it. To offer integrity, any encrypted message should be hashed by using a one-way mathematical function called hashing.

This function converts a variable sized message to a fixed size output called Message Digest (MD). The receiver will compute the MD using the received message to verify the integrity.

Our approach, proposes a hybrid algorithm that involves a simple XOR operation. A key stream is derived from a precedent key and an intermediate cipher text blocks, which is encrypted block-by-block (using the XOR operation) with message. To apply the integrity check to the message, a fixed size final key in each round is used as the MD after encryption process. As our algorithm use only the XOR operation during the encryption process and furnish an integrity check without using any external hashing algorithm, we can except that our approach will increase performance as well as reduce the computational complexity.

This paper is organized as follows. In Section II, we present a literature review of existing algorithms. In Section III, we explain our proposed algorithm. In

Section IV, we show some potential attacks, provide a security analysis of our approach R and compare our approach with existing algorithms. Section V concludes the paper.

II. Background and Related Work

The recent research is focused on hybrid approach that combines the benefits of the stream and block ciphers. Hybrid cryptographic approaches are gaining popularity in the applications which require high security over data transmission with efficient computation. Recently, a combination of block cipher and the stream approach he was proposed called Hummingbird [4]. This algorithm encrypts the message block by block and key stream is updated using internal register. The block size of the register is 64-bits which is divided into 4 smaller blocks of 16-bits. In initial state, the internal state register is loaded by random bit. Furthermore, the LFSR and the cipher text are used to regenerate the internal state register bits. It uses a 256-bit key that is divided in to four 64-bit keys to encrypt the message. Two more secret keys, namely, K5 and K6, are introduced and these are derived from the initial four keys. The Hummingbird algorithm has four rounds of permutation and substitution and on each round, uses four keys to encrypt the message. In the fifth round, the output is encrypted with key K5 followed by a final substitution and then encryption with key K6. For resource constraints of RFID systems, this approach is more adapted and suitable as it is resistant to linear attacks as well as differential [4]. However, our algorithm will be compared with Hummingbird algorithm as an example without take any property from it in our approach.

Krupa and Kutuboddin [9] have extended Hummingbird algorithm in order to increase the computational effectiveness, for which they involved an additional XOR operation. In conclusion, their proposed approach had a better computational efficiency compared to the original version.

Integrity is the important pillar of networking security which obstructs unnamed entity from data manipulation. Several hashing algorithms are used to check the integrity of the message such as MD5, SHA1, SHA2, SHA3, etc. using hashing function with encryption (decryption) will lead to elevated computational on the system [10].

In recent years, Jeddi et al. [4] proposed a SKE for RFID systems using low-cost tags called redundant bit security algorithm (RBS) to offer confidentiality with integrity. The Message Authentication Code (MAC) algorithm is used as hashing technique to generate the redundant bits for RBS. This increases the computational load of the tag.

In summary, the existing algorithms are feeble when they come to generating a key or/and the encryption process leads to computational overhead. In this paper, we proposed our approach to remedy this gap between computational overhead and to support a strong dynamic key generation process to encrypt the message, which uses a simple XOR operation for better computation, and Intermediate Cipher Text (ICT) to generate the key dynamically.

III. Proposed Approach

a) Overview

Using cipher text as part of the encryption can conduct to information about the original message being uncovered to an attacker. To avoid this weakness, our algorithm uses an ICT to generate a dynamic key which will be used on each block of the message encryption, to generate at the end a cipher text bits completely random. Table 1 presents a summary of notations used in our proposed approach.

Table 1: Symbols

K_a	Keys to encryption Round One
K_b	Keys to encryption Round Two
M	Last block
L	Current block
i_x	Prediction bit
M	Message to be encrypted
$M_1, m_2 .., m_1$	Message blocks
XOR	Logical bit-by-bit XOR operation
δ ()	Derivation function
ρ ()	Prediction function
γ ()	Inversion function
M_x	Message block x
K_{ax}	Round One key block x
K_{bx}	Round Two key block x
ICT_x	Intermediate Cipher Text block x
CT_x	Cipher Text block x
$C_{\alpha\beta}$	Segment β of component α (α can be M_X, ICT_X, K_{AX}, K_{BX}, CT_X)
KG_1	Key generation for Round One
KG_2	Key generation for Round Two

In our hybrid encryption algorithm, we include both features, stream cipher and block cipher. This approach consists of two parts: key generation part based on stream ciphers and two rounds of encryption (or decryption) using a basic XOR operation (employing message blocks as in block ciphers). Each message block is encrypted block-by block with different (block) keys. The size of one block in the message is 128-bits in sixteen 8-bit segments.

Fig. 1 illustrates the encryption process. The algorithm calls for to initial keys that encrypt the first block in each round. Except for the initial key, each successive key is derived from the ICT block and bits in the current key block.

In general, blocks of message are encrypted using blocks of keys generated using the stream cipher scheme. Encryption and Key generation follows a chained approach, with the current (block) keys and ICTs used to generate (block) keys for the encryption of the subsequent message blocks. The key generation and encryption processes will be described later in this section.

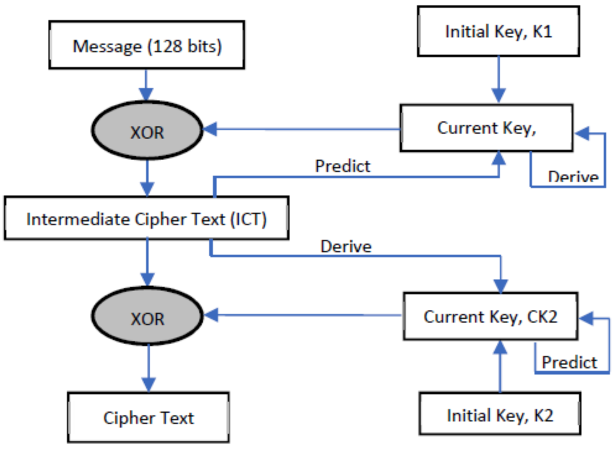

Fig. 1: Encryption process overview

b) *Key Generation*

We suppose that the initial key used to encrypt the first block of the message in each round is generated by the sender and securely transmitted to the receiver.

The successive keys are generated by using the previous keys and the intermediate cipher text (ICT). The term "intermediate cipher text" represents an encrypted message after the first round (Round 1), whereas the final cipher text is generated after Round 2 (will be discussed in the encryption section). In this approach, the process of the keys generation is different for both rounds. Each round uses a Different initial key (K_{a0} & K_{b0}) to encrypt the first block in the message. The encryption of each successive block in the message introduces a combination of derivation and prediction techniques to generate successive or the following keys ($K_{a1\ldots n}$ for Round 1 and $K_{b1\ldots n}$ for Round 2) from a previous key block and an ICT block.

In prediction, three bits in every segment (8 bits) per block are used to choose the i_x^{th} bit within the respective segment. The value of i_x is based on the three binary bits chosen in the segment. The value of i_x is a decimal representation of the 3 binary bits. In a segment, 3-bits are selected in a clock wise direction beginning from the most significant bit (MSB) in the segment to the least significant bit (LSB) to choose one out of eight probable values for i_x, this is repeated 8 times starting from the MSB to the LSB. The value of i represents the position of the binary bit in the segment. Each value of i_x predicts new bit that is used to generate segments per block in every successive key. Fig. 2 shows how the three bits are rotated in a clock wise direction to provide one out of eight possible values for i_x.

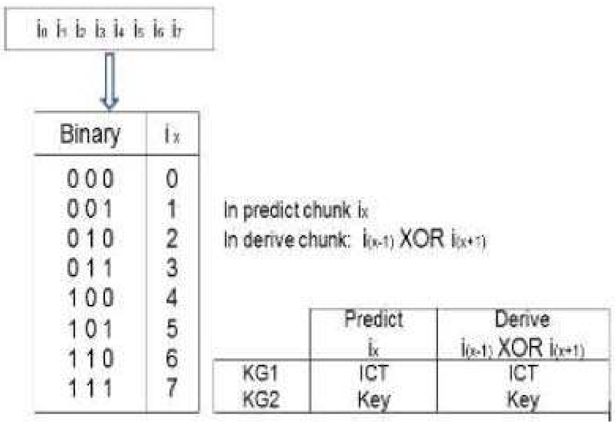

Fig. 2: Key Generation Process

In derivation, the successive key is generated by XORing the (i_x-1)th position of the binary bit with the (i_x+1)th position of the binary bit to form a new bit. The same process is repeated in every segment per block to generate the entire key for encrypting the successive block in the message. The whole key generation process provides randomness in the key stream without creating a cycle.

The prediction and derivation process are applied in combination to generate a key is as follows. In Round 1, the key is obtained from key K▪ series based on the ICT block's prediction. Whereas, in Round 2, the successive or the following keys are derived from the ICT block based on the key block's prediction. To generate the keys during two rounds, the combination of the both functions is alternatively applied on key block and ICT block.

c) *Encryption or (Decryption)*

Initially, in Round 1, four out of eight bits (0, 2, 3, 5) in every segment in a key K_{a0} are inverted. The first segment in the initial key Ka_0 is XORed with the first segment in the first block of the message, which produces the first segment in ICT_0 (represents the ICT first block). Again, four out of eight bits (2, 4, 5, 7) in every segment in ICT_0 are inverted, which is followed by successive key generation operations. The output result of the Key generation operation will be the first segment in the key K_{a1} to encrypt the first segment of the next message block. The first segment of key K_{a1} is XORed with the second segment of key K_{a0} that will be used to encrypt the second segment in the first block of the message. To resume, every 1st segment in each block of the message is directly XORed with the 1st segment of the respective key, although the successive segments in each block in the message are encrypted with the XORed output of the successive segment in the present key and the currently derived segment for the next key.

The 4th segment in message $C_{M2,4}$ for example, will be encrypted with the XORed output of the 4th segment in the current key $C_{Ka2,4}$, and the 3th segment in the next key $C_{Ka3,3}$.

Round 2 follows the same method like Round 1, apart from that the message and initial key K_{a0} will be replaced respectively by the ICT and key K_{b0}. Moreover, the inversion action is applied on bits 1, 4, 6, 7 and 0, 1, 3, 6 for the key and the ICT respectively. The output of Round 2 is the final cipher text. Fig.3 illustrates the encryption process. A decryption process is symmetric with encryption in reverse manner.

d) *Integrity Check*

The last keys ($K_{a(n+1)}$ and $K_{b(n+1))}$ in each round of encryption will be exploited as the integrity check parameter. In this approach, the key $K_{a(n+1)}$ (128-bit) will be concatenated with key $K_{b(n+1)}$ (128-bit) to form a 256-bit message digest used for integrity check.

e) *Example*

Key Generation and Encryption. This section explains the two rounds of the encryption process with a generic example. Each round used different initial key K_{a0} and K_{b0}. The message M is subdivided into n different sub-blocks. The first block of the plain text message M is encrypted with a key combination of K_{a0} (the present key) and K_{a1} (the currently derived key) at Round 1 that generates the first ICT block. Let the message blocks be:

$$M = M_0, M_1, M_2, M_3, \dots, M_n \qquad (1)$$

Round 1: Encrypting the first block of the message M_o: The message block will be encrypted segment by segment. Before the XOR operation, bit positions 0, 2, 3, 5 will be inverted in the first segment of an initial key. Then, the first segment in the first block of the message will be XORed with the initial key's first segment (generated after the inversion).

$$C'_{K_{ao},0} = \gamma\,(C_{K_{ao},0}) \qquad (2)$$

$$C_{ICT_0,0} = E_{C'_{K_{ao},0}}\,(C_{M_0,0}) \qquad (3)$$

Once again, bits 2, 4, 5 and 7 in the ICT will be inverted before the key generation process and the output will be used to generate a key to encrypt the next segment as well as the block.

$$C'_{ICT_0,0} = \gamma\,(C_{ICT_0,0}) \qquad (4)$$

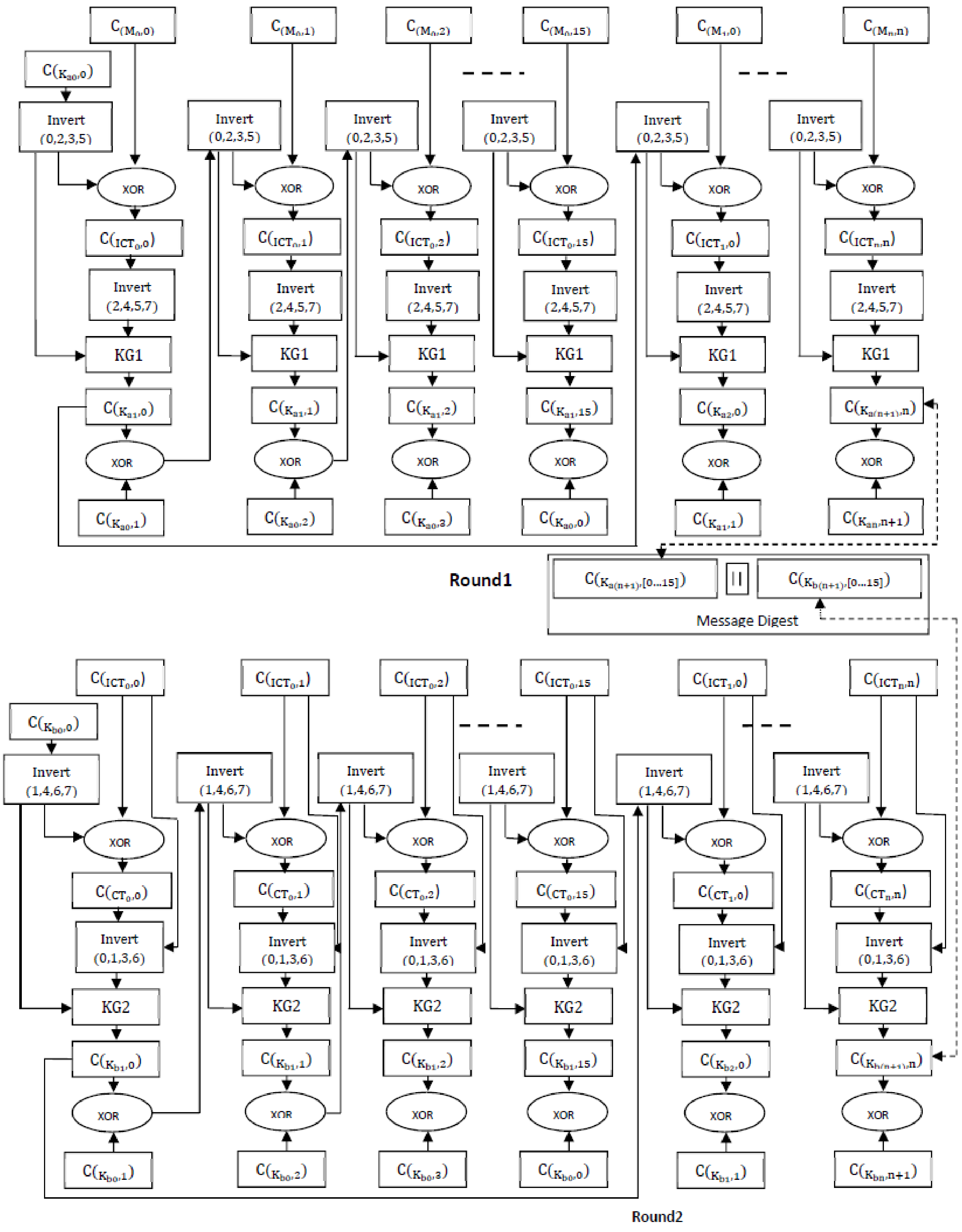

Fig. 3: Encryption process

Each bit of the key C_{Ka1},o, is generated from the previous key $_{C'Kao,o'}$ whereas, n bits in $C'_{ICTo,o}$ chooses a bit in $C'_{Ka0,0}$ to produce the new key CKa1,o. In general, a current key K_{al} is generated from the prediction of a previous ICT block ICT_{1-1} and the derivation of the previous key $K_{a(1-1)}$ in Round 1. Every first segment in the message block is directly XORed with the inversion of the first segment in the key, whereas the 2nd to 15th segment in the message will be encrypted with an inversion of the XORed value of the next segment key with next block's segment's key.

Key Generation: for initial segments only

$$C_{K_{al},0} = \rho(C'_{ICT_{l-1},0}\,[\delta\,(C'_{K_{a(l-1)},0}\,)]) \tag{5}$$

For the 2nd to 15th segments

$$C_{K_{al},1} = [C_{K_{al},1}]\ \mathrm{XOR}\ [C_{K_{a(l+1)l},0}] \tag{6}$$

$$C_{K_{al},15} = [C_{K_{al},15}]\ \mathrm{XOR}\ [C_{K_{a(l+1)l},14}] \tag{7}$$

For encryption in Round 1, the generic equation is as follows:

$$ICT_0 = E_{K_{a0}}(M_0), ICT_1 = E_{K_{a1}}(M_1),\ldots, ICT_n = E_{K_{an}}(M_n) \tag{8}$$

Round 2: In this Round, the encryption process is the same as in Round 1. But in the key generation process, the prediction and derivation functions are swapped between ICT block and key. The initial key (K_{b0}) encrypts the 1st block of the ICT block. To encrypt subsequent blocks, a current key (K_{b1}) is generated from the prediction of the previous Key block ($K_{b(1-1)}$) and the derivation of the previous ICT block ($ICT_{(1-1)}$). In Round 2, the bits chosen to invert are different from Round 1. For key bits 1, 4, 6 and 7 are inverted before encryption, whereas, bits 0, 1, 3 and 6 are inverted for the ICT segments (after the encryption, but before the key generation process).

Encrypting first block of the message:

$$C'_{K_{bo},0} = \gamma\,(C_{K_{bo},0}) \tag{9}$$

$$C_{CT_0,0} = E_{C'_{K_{bo},0}}\,(C_{ICT_0,0}) \tag{10}$$

$$C'_{ICT_0,0} = \gamma\,(C_{ICT_0,0}) \tag{11}$$

Key Generation: for initial segments only

$$C_{K_{bl},0} = \rho\,(C'_{K_{b(l-1)},0}\,[\,\delta\,(C'_{ICT_{l-1},0})\,)] \tag{12}$$

For the 2nd to 15th segments

$$C_{K_{bl},1} = [C_{K_{bl},1}]\ \mathrm{XOR}[C_{K_{b(l+1)l},0}] \tag{13}$$

$$C_{K_{bl},15} = [C_{K_{bl},15}]\ \mathrm{XOR}[C_{K_{b(l+1)l},14}] \tag{14}$$

The generic equation for encryption in Round 2 is as follows:

$$CT_0 = E_{K_{b0}}(ICT_0), CT_1 = E_{K_{b1}}(ICT_1),\ldots, CT_n = E_{K_{bn}}(ICT_n) \tag{15}$$

Message Digest. The final keys generated from each round will be concatenated to provide the 256-bit Message Digest that will be used as integrity check parameter.

$$MD = K_{a(n+1)} \,||\, K_{b(n+1)} \tag{16}$$

We can use Fig. 4 for the encryption and integrity check processes. For this example, we consider that the message M (32-bits) contains only two 16-bits blocks and each one has only two 8-bit segments.

The initial key, K_{ao} (10011010 11011011), is encrypted with the message (001011010 01100111 01110110 1000011) which produces, in Round 1, the ICT (00000011 1101001010000001 01001111). Then, in Round 1, the ICT block is used to predict a derivation bit in Key block, whereas, in Round 2 a key block predicts the derivation bit in the ICT block. Further, in Round 2, the cipher text (111100110100011100111000 10111101) will be obtained by encrypting the ICT with the key block K_{bo} (1011101100101100). Finally, the concatenation of the last derived keys, K_{a2} and K_{b2}, will act as the message digest (010001111110001010111001 11001111).

Global Journal of Computer Science and Technology (E) Volume XVII Issue V Version I Year 2017

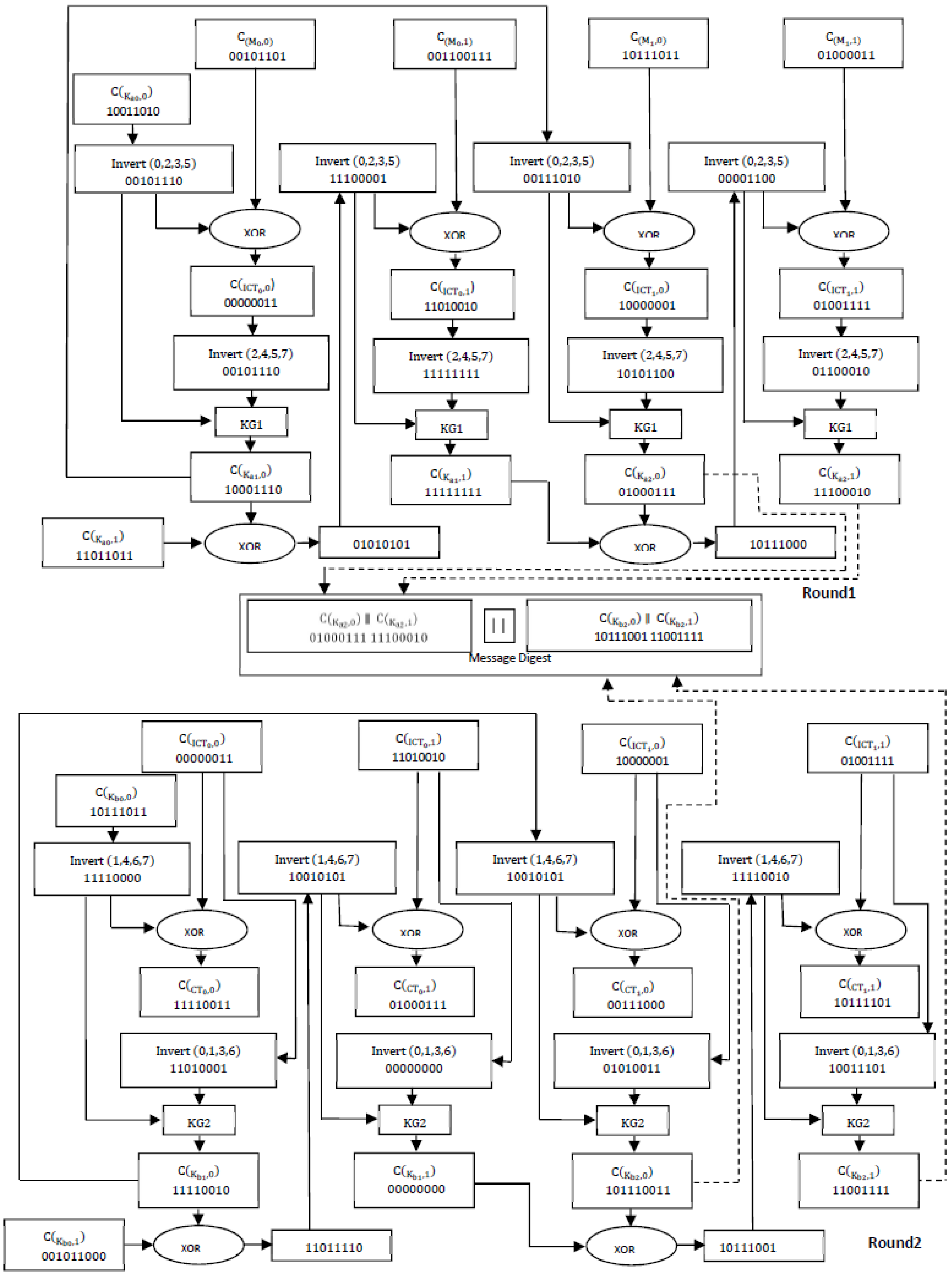

Fig. 4: Encryption and Integrity Check Process Example

IV. Analysis and Discussion

The generation of the successive keys is completely dynamic using message and key based on derivation or prediction function. In fact, there is no regular or fixed cycle in key bits' stream during their generation. Our method is a very simple algorithm based on XOR operations that encrypts the message and generates the keys. It can be unaffected by variety of attacks (replay attack, chosen-plain text attack, chosen cipher attack, differential attack, linear attack and distinguishing attack) [6] [11].

a) *Cryptanalysis*

This section presents the specific cryptanalysis for using this approach on RFID systems [12] like chosen plain text attack, chosen cipher text attack and differential attack. The both first attacks choose one part of information to retrieve the original message for a given Cipher text to obtain the secret key. On another hand, an attacker supposes that a message is in plain text that is encrypted to obtain a corresponding cipher text, and compare it with captured cipher text. As in approach the keys are transformed dynamically for every block encrypted, it's hard to launch chosen plaintext and chosen cipher text attacks.

The differential attack technique compares the difference in an input value with the output to obtain a possible key. Since our approach relies on both a key and an intermediate cipher text, differential attack is difficult to realize. In addition, switch key generation process between prediction and derivation function, lead to remove the linearity in the key cycle avoid revealing some information about the message.

A distinguishing attack focuses on stream ciphers, which compare a given sequence of values to check the randomness. A dynamically generated key ensures that there will be no relation between the current and previous key, so launching a distinguishing attack is difficult.

b) *Security*

In addition to the mentioned attacks above, we used cryptanalysis tool called Cryptool to do some tests. The security analysis of our algorithm was compared with existing block cipher algorithms like DES, AES, Triple DES, etc. Figures below present different scenarios that are based on the cryptanalysis tests (entropy test, periodicity test, frequency test, poker test, run test and serial test). The analysis results show that the security level of the proposed algorithm is no lower than the pool of example algorithms used. Figure 5 display the model graphs, which were taken from the security analysis with sample inputs 128bit.

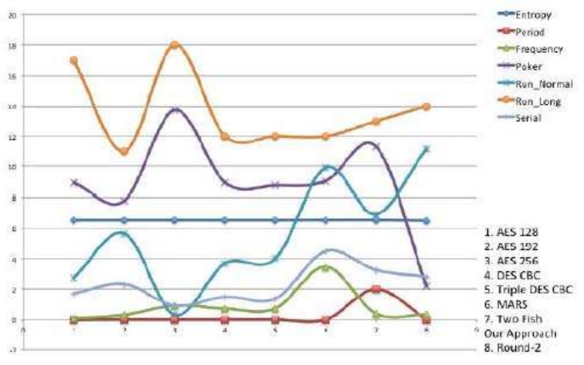

Fig. 5: Sample 128bits

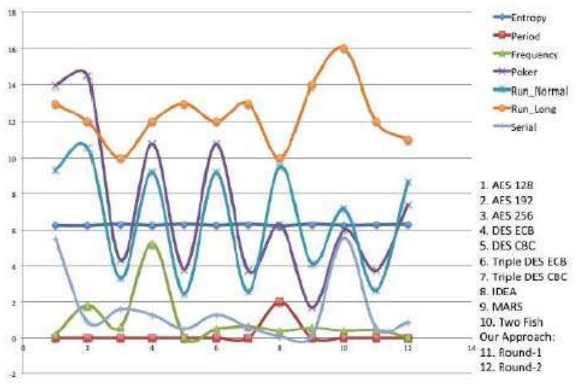

Fig. 6: Sample 480bits

The non-linearity in the key cycle for our proposed algorithm is achieved by using an ICT block. However, using cipher text to encrypt a message is vulnerable, so two rounds of encryption were designed with an ICT as the second-round key generator instead of the cipher text. Applying key generation from a key for the first round and an ICT for the second round ensures a dynamic property in successive keys.

c) *Performance Evaluation*

The performance analysis of this algorithm is expected to be computationally efficient. At first, the computation is required only for the encryption part, but the integrity check does not demand any additional computation process. The number of operations necessary for each bit is only fourteen, including encryption (or decryption), key generation operations and integrity check. According to our research, increase the performance require an increase on the hardware component which cannot be appropriate for some applications. However, our working on the key generation part is expecting to reduce the hardware requirement and in the same time increase the computational efficiency.

V. Conclusions

This paper has presented a different hybrid symmetric key encryption algorithm of stream and block ciphers offering privacy, confidentiality, integrity and generation dynamic key without using supplementary algorithms for limited resource systems such as RFID. In our approach, the message blocs are encrypted using key blocs generated through stream cipher scheme, in which the current block keys and ICTs are used to generate block keys for the encryption of subsequent message blocks.

The newness of the proposed algorithm is that the keys are generated dynamically using intermediate cipher text with integration of the integrity check. Finally, as our approach is not available in real time environment. For this why, improved security analysis, study of hardware implementation will be conducted in the future.

References Références Referencias

1. D. Molnar, and D. Wagner, "Privacy and security in library RFID: issues, practices, and architectures", Proceedings of the 11th ACM conference on Computer and communications security, pp. 210-219, 2004.
2. L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede, "Public-key cryptography for RFID-tags", Pervasive Computing and Communications Workshops, 5th Annual IEEE International Conference, pp. 217-222, 2007.
3. Z. Jeddi, E. Amini, and M. Bayoumi, "RBS: Redundant bit security algorithm for RFID systems", IEEE 21st International Conference, Computer Communications and Networks, pp. 1-5, 2012.
4. D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Hummingbird: ultra-lightweight cryptography for resource constrained devices", Springer Financial Cryptography and Data Security, pp. 3-18, 2010.
5. L. R. Knudsen, "Practically secure feistel ciphers", Springer Fast Software Encryption, pp. 211-221, 1994.
6. M. Hell, T. Johansson, and L. Brynielsson, "An over view of distinguishing attacks on stream ciphers", Cryptography and Communications, pp. 71-94, 2009.
7. Zeng, K., Yang, C.-H., Wei, D.-Y., and Rao, T.: "Pseudorandom bit generators in stream cipher cryptography". Computer, 24(2):8-17 (1991).
8. E. Dubrova, M. Teslenko, and H. Tenhunen, "On analysis and synthesis of (n, k) nonlinear feedback shift registers", IEEE Design Automation and Test in Europe, pp. 1286-1291, 2008.
9. K. Jinabade, and K. Rasane, "Efficient implementation of hummingbird cryptographic algorithm on a reconfigurable platform", International Journal of Engineering Research and Technology, vol. 2, ESRSA Publications, 2013.
10. X. Li, W. Zhang, X. Wang, and M. Li, "Novel convertible authenticated encryption schemes without using hash functions", IEEE International Conference, Computer Science and Automation Engineering (CSAE), pp. 504- 508, 2012.
11. M. Z. W. M. Zulkifli, "Attack on cryptography", 2008.
12. X. Francois, P. Gilles, and Q. Jean-Jacques, "Cryptanalysis of block ciphers: A survey", UCL Crypto Group Technical Report Series, Technical Report CG- 2003/2, 2003.

This page is intentionally left blank