A NOVEL AUTHENTICATED CIPHER FOR RFID SYSTEMS

Zahra Jeddi, Esmaeil Amini and Magdy Bayoumi

The Centre for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA

ABSTRACT

In this paper, we present RBS (Redundant Bit Security) algorithm which is a low-complexity symmetric encryption with a 132-bit secret key. In this algorithm redundant bits are distributed among plaintext data bits to change the location of the plaintext bits in the transmitted data without changing their order. The location of redundant bits inside the transmitted data represents the secret key between sender and receiver. The algorithm provides integrity and authentication of the original data as well. The implementation comparison of this algorithm with other algorithms confirms that it a good candidate for resource-constraint devices such as RFID systems and wireless sensors.

KEYWORDS

RFID systems, symmetric encryption algorithm, private key, authentication

1. INTRODUCTION

Radio Frequency Identification (RFID) is a method used for identifying items like barcodes. In RFID systems the identification is performed using radio signals and there is no physical contact like barcodes. This way, huge number of items can be identified in a short time with high reliability and low cost which makes this method very attractive for applications like supply chain management, e-health, monitoring objects, electrical tagging, etc.

In general, each RFID system consists of three parts. i) Transponder or tag which is implemented on objects for storing data, ii) Transceiver or reader which provides electromagnetic field in order for activating tags and reading their data through radio frequency waves, and iii) A back-end server which receives and processes data from readers.

Among these three parts, tags have more implementation limitations. There are two types of tags in general: active and passive tags. Active tags are equipped with their own batteries whereas passive tags rely on radio frequency energy transferred from the reader. Compared to active tags, passive tags have longer life time, are smaller and lighter but, their signal range is shorter than active tags. Passive tag systems are severely constrained on chip area and power consumption as they do not have internal power source. This paper focuses on the passive RFID tags and their power limitations.

Since the communication between the tags and the reader is done through an unsecure wireless channel, the transmitted data is vulnerable to attacks by unauthorized readers. These attacks are categorized into two main groups: privacy violation and security violation [1]. In privacy violation, the attacker tries to harvest information from the objects by eavesdropping to the communications between the object and the reader or by tracking them. In security violation, an

DOI:10.5121/ijcis.2014.4102

adversary counterfeit behaviours of a tag or a reader for making undesirable communications. Therefore, a mechanism is required to provide privacy and security during the communication. This demand can be accomplished either physically or cryptographically.

The most known physical methods in RFID security are killing tags, blocking tags, Faraday cage and active interference [2]. Each of these methods has its pros and cons. Killing a tag, the tag will lose its functionality and cannot be reactivated which considerably reduces the life time of tags. Blocking tag method, attacker cannot have access to tags just in a defined range and beyond this range, tags are not protected from attacks. In Faraday cage method, a wrapper shields the tag from radio waves which imposes another cost to the system. In active interference method, unauthorized readers are impeded to have communications with tags, but sometimes legal readers get blocked as well. Based on the limitations and disadvantages of physical mechanisms stated above, these methods are only applicable for some specific applications.

Regarding cryptographic solutions, tags are extremely resource limited for adapting conventional encryption methods. These limitations are strong motivations for investigating light encryption algorithms which can handle the area and power constraints of RFID systems. In this paper, a new light symmetric encryption algorithm is proposed called RBS. In RBS, the message is intentionally manipulated by distributing redundant bits among plaintext bits and the location of redundant bits inside the transmitted data represents the secret key between sender and receiver. Meanwhile, there is a relation between plaintext data and redundant data in RBS algorithm. These redundant bits are generated by Message Authentication Code (MAC) algorithm which its input is the plaintext data. This way, these redundant bits can be used for authenticating the message as well. In RBS, the security level of the algorithm is adjustable through the number of redundant bits. In other words, there is a dependency between the provided security and the authentication part of the system which distinguishes RBS algorithm from other existing algorithms. To have flexibility in the number of redundant bits, the implemented MAC algorithm generates variable length outputs. In addition to the number of redundant bits, their values and their positions in the ciphertext are also determining factors in the security of the generated ciphertext. Furthermore, some plaintext bits are also altered based on the value of the encryption key and redundant bits in order to make the generated ciphertext more secure against attacks.

The rest of this paper is organized as follows: Section 2 presents some related work presented in literature. The relation between redundant bits, security and key space is discussed in sections 3. Then, the algorithm of RBS is presented in section 4. The proposed implementation is presented in section 5. Experimental results are presented in sections 6 and section 7 concludes the paper.

2. RELATED WORK

Cryptographic algorithms are divided into two main groups: Private key or Symmetric algorithms and Public key or Asymmetric algorithms. Public key algorithms such as RSA and Elliptic Curve Cryptography (ECC) are very strong in terms of security and they provide reliability, confidentiality, integrity, availability and non-repudiation services altogether. ECC-based systems offer similar security for smaller key sizes compared to RSA-based systems. Since the computational and area complexities of hardware implementations for cryptographic algorithms are proportional to their key sizes, ECC-based systems are smaller, faster, and consume less power compared to RSA-based systems [3]. Therefore, ECC algorithm is a better choice than RSA algorithm for resource constrained systems.

A lot of research has been done on hardware-efficient ECC implementations [4-13]. In [4, 5] the authors have tried to make a relation between power consumption of their ECC architectures and the requested security level. In [6], the authors have tried to adapt ECC algorithm with RFID

systems by reducing the number of registers, operations and the operation frequency and have used restructured formulas as much as possible in order to meet the resource limitations of RFID systems. Making ECC algorithm lighter is another solution. Reducing the flexibility of ECC algorithm by limiting the parameters such as using only one special elliptic curve [7], selecting specific field sizes [8] or choosing specific prime numbers [9-12] are other ways to make ECC algorithm lighter. Although applying dedicated hardware with these limitations leads to meet the power limitation, but, any change in security parameters imposes replacement of all tags with new ones. Authors of [13] addressed the design of an ECC-based M-PKI usable for mobile phones/devices by some modifications on the existing PKI.

Since Public key algorithms have still significant challenges for RFID systems' implementation, recently researches have been directed towards Private Key schemes. Private Key algorithms are divided into three categories: block ciphers, stream ciphers and hybrid ciphers which are the combination of block and stream ciphers. In block cipher algorithms, the message is broken into the fixed size of blocks and each block is en/decrypted individually. Stream ciphers encrypt one bit or one byte at a time. They rely on the generation of a pseudorandom keystream and use XOR function for encryption.

AES [14] and PRESENT [15] are known block cipher algorithms. AES is basically designed for efficient software implementation. The best known lightweight AES requires 3200 gate equivalent (GE) for implantation. PRESENT is another lightweight encryption algorithm which is based on S-Box and is inspired by the techniques used in DES and AES. This algorithm takes 32 rounds working with either 80 bit or 128 bit keys. In each round, after adding the round key to the plaintext, s-box is applied on it for 16 times in parallel. Area optimized version of PRESENT is implemented by 1075 GE and in 563 clock cycles generates 64-bit ciphertext.

Grain-128 [16] and Trivium [17] are two known stream cipher algorithms. Grain uses 128-bit key and 96-bit initial vector (IV) and Trivium uses 80-bit key and an 80-bit IV. Trivium provides a flexible trade-off between speed and gate count in hardware. It consists of three shift registers of different lengths.

The Hummingbird-2 (HB-2) [18] is hybrid cipher with 128-bit key and 64-bit initial vector. It operates on 16-bit blocks and it is performed by a series of exclusive-or, addition modulo and a nonlinear mixing operations. This algorithm has an initialization phase before encryption like Grain. The first version of this algorithm was vulnerable to a chosen-IV and chosen-message attack [19].

Private Key encryption algorithms can provide authentication and integrity using message authentication code (MAC) algorithms such as HMAC [20]. In this approach, the plaintext and the authentication key are both fed into the MAC algorithm and then the generated MAC digest is appended to the plaintext before encryption. In the receiver side, the received message will be accepted if the MAC of the decrypted message is identical with the received MAC.

Grain [16] and HB-2 [18] algorithms both provide optional authentication. In these two algorithms, the authentication code is appended to the encrypted message. Both of these two algorithms do not impose considerable hardware overhead for providing the authentication as they share the same hardware for encryption and authentication. However, authentication has its own timing overhead as it cannot be done along with encryption phase. In Grain method, the authentication key is required to be changed after each usage. Otherwise, the attacker will find the authentication key with comparing two or three different ciphertexts. Since exchanging keys in symmetric algorithms is a big challenge, refreshing this key imposes undesirable overhead to the algorithm.

This paper proposes a new symmetric encryption method for RFID systems, called RBS; Redundant Bit Security which provides both authentication and confidentiality at the same time with low overhead in performance, area and power consumption. This method is based on inserting the redundant bits into the altered plaintext. The contribution of this work is using the output of light RFID MAC algorithms proposed in [21], as the redundant bits and merging them with the altered original bits. This way, the redundant bits provide authentication, integrity and confidentiality. The light MAC algorithm in [21] supports different digest sizes. As a result, the number of redundant bits and accordingly the security level could be adjusted in RBS without requiring changing the underlying MAC algorithm.

3. PRELIMINARIES

Typically, encryption algorithms are based on applying some mathematical operations on the plaintext and ciphertext. Unlike these conventional encryption methods, the proposed RBS algorithm in this paper does not use these mathematics computations for encryption and decryption. Instead, the message is intentionally manipulated by inserting redundant bits into original bits. In this algorithm, the location of the original bits changes in the ciphertext. As an example, suppose that the original message is "1010". Inserting one redundant bit at the third place changes the message to "10110". Knowing that just one bit is redundant, the attacker confronts with four possible plaintexts. Besides redundant bits' locations, their values are important as well. For instance, assume that the original bits are all zero so adding one '0' bit as a redundant bit will not have the same effect as adding a "1" redundant bit in this case. Therefore, i) the number of redundant bits, ii) their locations and iii) their values are all important in hiding the plaintext inside the ciphertext. In other words, there is a relation between the security level of RBS algorithms and each of these three parameters. The first parameter is discussed in the following sub-section and the other two parameters are studied in Section4.

3.1 Key space

Security level is defined by answering this question: How long it will take for an attacker to break the algorithm and what resources he needs in order to have a reasonable chance of succeeding? One of the well-known tools for measuring the security level of an algorithm is key space. The security level of an encryption algorithm has a direct relation with its key space. Key space is the set of all possible keys that can be used to initialize a cryptographic algorithm.

Let us explain the concept in RBS algorithm. Suppose that n is the number of original bits or plaintext and m is the number of redundant bits. The ciphertext is an $(n+m)$-bit data obtained by insertion of redundant bits among plaintext bits. The location of redundant bits inside the ciphertext defines the secret key. Therefore, the secret key is simply an $(n+m)$-bit string where "1" in this string represents the location of redundant bit and "0" represents the location of plaintext bit in the ciphertext.

The size of key space (s) or the number of possible locations of redundant bits in the ciphertext depends on n and m and is expressed by Equation (1).

$$s = \binom{n+m}{m} = \frac{(n+m)!}{m!\,n!} = \frac{\prod_{m=1}^{m}(n+m)}{m!} \qquad (1)$$

In Equation (1), m and n are interchangeable. In other words, increasing either the number of redundant bits or the number of plaintext bits has the same effect on the key space size. As a result, fixing one of them, the size of key space can be adjusted to the desired security level by

changing the other parameter. However, increasing one of the factors and decreasing the other one results in different way as depicted in Figure 1.

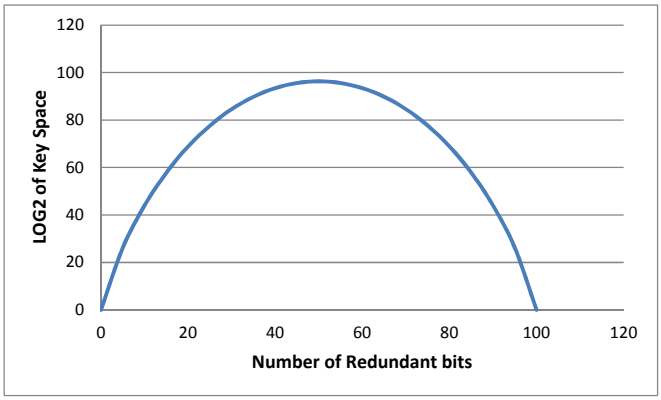


Figure 1. Size of key space vs. number of plaintext bits (n) and redundant bits (m) when $n + m$ is fixed

In this figure n and m are changing while the total number of bits is constant ($n + m = 100$). It shows that when these two factors are far from each other, the key space will reach to its minimum size. The maximum size of key space happens when the number of redundant bits is equal with the size of plaintext. Based on this graph, high security level for small block of plaintexts is not possible unless with large number of redundant bits. Likewise, it is not possible to provide high security level with low number of redundant bits.

Figure 2 exhibits how big the key space can be for different n, and m when ($n = m$) which helps in finding the desired number of plaintext bits and redundant bits.

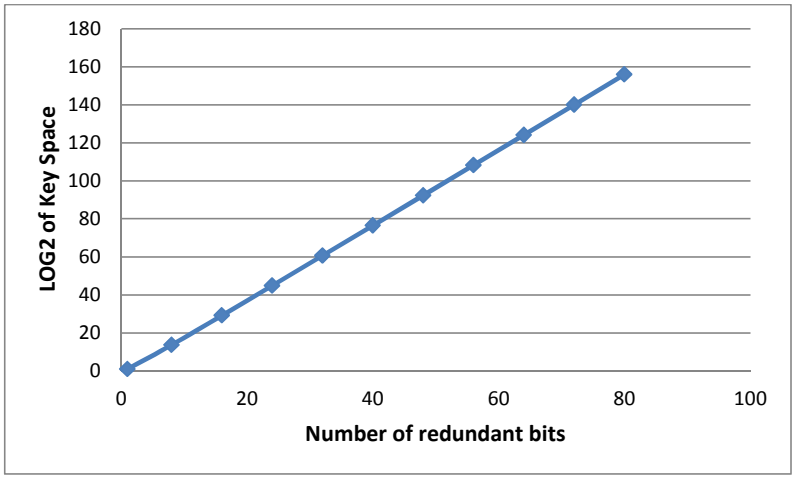


Figure 2. Size of key space vs. number of plaintext bits (n) and redundant bits (m) when $n=m$

As mentioned before, there is a relation between the size of the key space of an encryption algorithm and its security level against possible attacks. The question is how big the space key should be to guarantee the desired security. The Brute-Force attack has been studied for finding the boundary of the key space for RBS algorithm. In this attack the attacker performs a complete search through all possible keys of the key space to find the right key.

The 2^{128} key space size is computationally secure against Brute-Force attack [22]. Applying this number in Equation (1), there will be a variety of choices for *m* and *n*. Figure 3 demonstrates the relation between size of key space and *m* for (*n*=64). Increasing *m* from 0 to 128; the key space (*s*) will exponentially increase from 1 to 2^{172}.

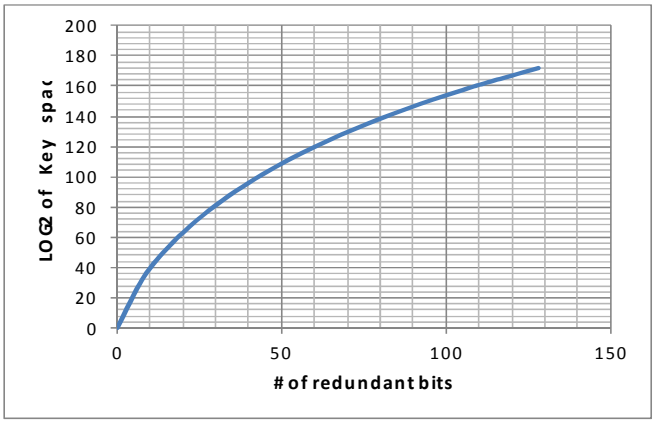


Figure 3. Size of key space vs. number of redundant bits (*m*) when plaintext size (*n*) is fixed (*n*=64)

Table-1 shows a possible set of *m* ad *n* for $s=2^{128}$. Considering the fact that the required energy for transmitting the message increases by the length of ciphertext, *m* and *n* should be chosen regarding this fact. Referring to Table 1, this happens when $(n,m) \in \{(64,68), (65,67), (66,66), (67,65), (68,64)\}$ which is highlighted in Table 1. The best choice is (64, 68) as data blocks are processed and stored normally in multiples of 8-bits.

Table 1. The number of bits required in ciphertext to have $s=2^{128}$

M	50	55	57	60	63	64	65	66	67	68	70	73	78	91
N	91	81	78	73	70	68	67	66	65	64	63	60	57	50
c*	140	136	135	133	133	132	132	132	132	132	133	133	135	140
c : # of bits in the ciphertext*														

4. RBS ALGORITHM

In proposed RBS algorithm, one 64-bit data block is encrypted into a 132-bit ciphertext using a 132-bit secret key. In other words, 68-bit redundant data is distributed among 64 bits of plaintext. Since the number of redundant bits, their locations and their values are all important in the security level of the algorithm; this section is dedicated to find the relation between these parameters and the security level.

4.1 Location of the redundant bits inside the ciphertext

The distribution of redundant bits inside the ciphertext should not follow any linear mathematic or non-linear mathematic function, otherwise i) the size of the key space will be reduced, ii) a dependency will be constructed among redundant bits and iii) redundant bits will be distributed uniformly among plaintext bits. Therefore, the position of every redundant bit must be independent of other bits' positions. This way, if one of the redundant bits being exposed it will

just shrink the key space while location of other redundant bits is still secret. The solution is random distribution of redundant bit inside the ciphertext.

4.2 The value of redundant bits inside the ciphertext

In addition to providing confidentiality of the sent data, the injected redundant bits can carry some additional information about the original data as well. For generating these redundant bits, there are three options:

- Choosing constant values. In this case, the redundant bits are the same for different plaintexts. Comparing different ciphertexts, the attacker can easily figure the location of the redundant bits.
- Choosing random values. In this case there would be several ciphertexts for one plaintext. This way the attacker can find the location of the redundant bits by comparing the different resulted ciphertexts for the same plaintext.
- Values of redundant bits are injective functions of the plaintext. So, there is a unique redundant data per each plaintext. As a result, plaintext and redundant data cannot be distinguished easily in the ciphertext.

Among these three approaches, the third one is chosen as it has potential to provide both attack prevention and authentication. This algorithm can be implemented by splitting the plaintext into small blocks and performing mathematical functions on each of blocks individually. At the end, all block are combined while it is encrypted by a secret key. The pseudo program of this algorithm is presented in Figure 4.

1. Split the plaintext into several small segments S_i.
2. for each S_i
   Shift/rotate/add/XOR (S_i , a constant number N_i)
3. Combine all segments S_i to a single segment S
4. Encrypt (S, secret key K) using a symmetric algorithm

//K will be used in the receiver side for authenticating the sender

Figure 4. Redundant data generation algorithm

One applicable implementation for the presented algorithm in Figure 4 is MAC algorithm as a very small change in the plaintext will produce a very different output. Using MAC algorithm for generating the redundant bits, integrity and authentication will be provided as well as confidentiality which will be discussed in the following subsection.

4.2.1 Message authentication and Data Integrity

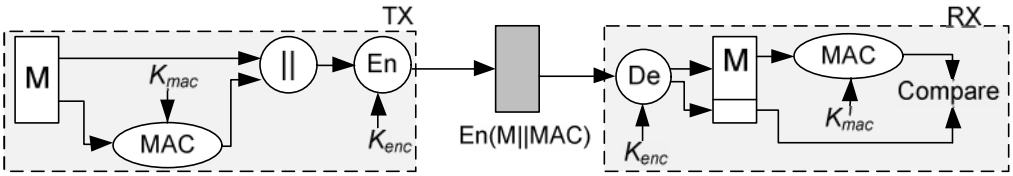
Data integrity is defined as maintaining correctness and consistency of a message. Since the message is sent via wireless network, the message is in danger of being altered in transmission by an adversary or environmental hazards, such as heat, dust, and electrical surges. Therefore, the receiver should validate the received data.

Message authentication guarantees that the received message has been sent by an eligible user. It is crucial for a party – tag or reader - which receives a message to be sure who has sent it.

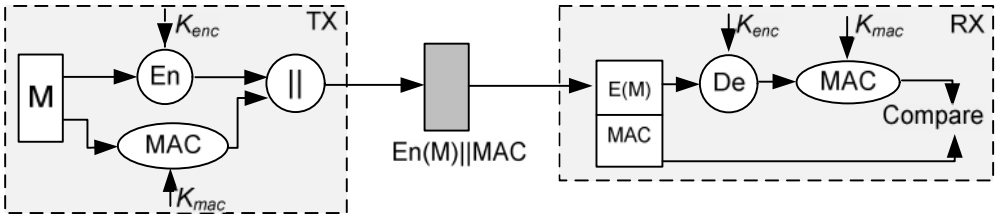
Message Authentication Code or MAC is a piece of information which is used for both data integrity and authentication purposes. It is generated by a MAC algorithm which has two inputs i) an arbitrary-length message and ii) a shared key between two parties. Typical MAC algorithms such as HMAC and MD-5 guarantee that no collusion will happen in their outputs for different input messages.

In general, there are three protocols for embedding the MAC inside the ciphertext.

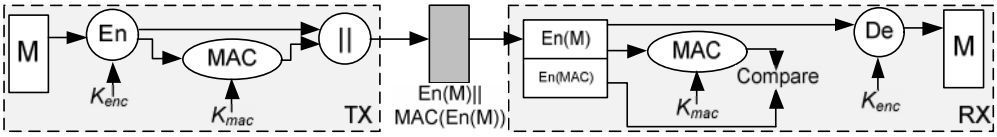
- After generating the MAC using the authentication key, K_{mac}, MAC will be attached to the original message and then the new message is encrypted by encryption key, K_{enc} (Figure 5.a) [20]. In the receiver side, after decrypting the received data, the MAC part will be regenerated and then compared with the received one. Generating the same MAC means that the message is intact and it is sent by an authorized user. Otherwise the received message will be discarded.
- The generated MAC will be attached to the end of the encrypted message before transmission (Figure 5.b). Grain [16] and HB-2 [18] use this protocol.
- Instead of the plaintext, the MAC of the encrypted plaintext is attached to encrypted message before transmission (Figure 5.c).



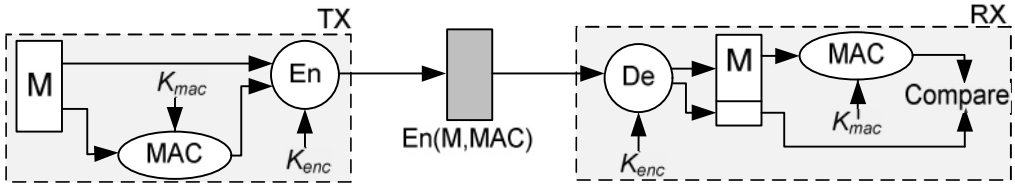
(a) First authentication protocol



(b) second authentication protocol



(c) third authentication protocol



(d) proposed authentication protocol

Figure 5. Embedding tag inside the ciphertext in different protocols

In the second and third protocols, the boundary between the MAC and message is clear. Hence, the MAC algorithms used in these protocols are required to be very secure against the substitution attack. In this attack, the adversary tries to replace the legitimate message with his own plaintext and MAC assuming it will be accepted by the receiver. In the other side, the first protocol is more secure against this attack because MAC is encrypted along with the plaintext and there is no direct access to it.

4.2.2 Message authentication and Redundant Bits

The second and third MAC generation protocols shown in Figure 5 are not used in RBS algorithm as we want to distribute the MAC output or redundant data among plaintext bits. Instead, we have used the first protocol with a slight modification (Figure 5.d). Here, the MAC part is inserted between message bits instead of being appended to the message. In other words, merging the MAC with plaintext is itself a part of encryption process. The distribution pattern of the MAC part inside the ciphertext is based on the encryption key. At the receiver side, the received data is broken into two parts based on the encryption key: the altered plaintext and redundant bits or MAC. Regenerating the MAC at the receiver side and comparing it with the received MAC, the receiver decides to keep the data or discard it.

4.3 Plaintext manipulation

How the plaintext appears inside the ciphertext is directly related to the confidentiality of the algorithm. Three possible scenarios for this issue will be discussed in the following subsections.

4.3.1 Direct appearance inside the ciphertext

In this approach, the original plaintext bits will be merged with redundant bits. Likewise, in decryption process, the plaintext can easily be extracted from the ciphertext by removing the redundant bits. Despite the simplicity of this method, it shrinks the key space and makes the algorithm vulnerable to some attacks like known plaintext attack and chosen plaintext attack. In these attacks, since the attacker knows the plaintext, those bits of ciphertext which have the same value of the plaintext will be potential locations for plaintext in the secret key. For example, if the plaintext is a string of zeros, all corresponding zeros in the ciphertext might be zero in the secret key too.

There are some ways to increase the key space size such as increasing the number of redundant bits or having separate encryption keys based on the plaintext pattern. Increasing the number of redundant bits introduces more overhead of MAC implementation and so more power for transmitting the ciphertext. Generating a new key based on the plaintext pattern and exchanging it with receiver are also challenging tasks in symmetric encryption algorithms.

4.3.2 Bitwise addition with constant-value keystream

In this approach, some fixed bits of the plaintext will always be altered in the ciphertext regardless of the pattern of the plaintext. It makes the algorithm secure against known plaintext attack because the attacker does not know which bits of plaintext are altered in the ciphertext. However, it is still vulnerable to chosen plaintext attack. For example, if the attacker changes only one bit of the plaintext, all bits of the plaintext except the changed one will appear in the ciphertext. Comparing two ciphertexts of two almost same embedded plaintexts shrinks the key space and makes it easier to find approximate location of the changed bit in the plaintext.

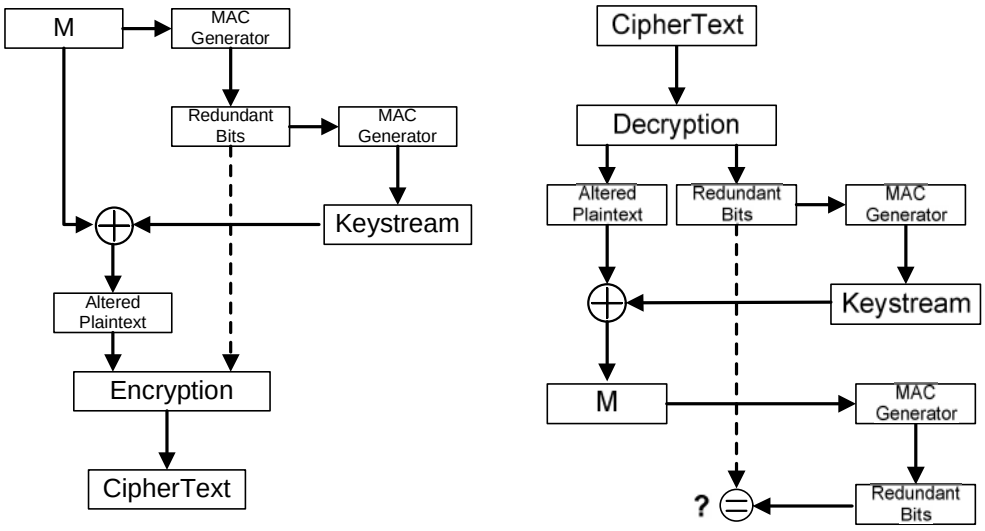
4.3.3 Bitwise addition with variable-value keystream

In this approach, the plaintext bits are XOR-ed with a keystream which is a function of the plaintext. In other words, there is a unique keystream per each plaintext. This approach is somehow similar to one-time pad where the plaintext is altered with a variable keystream. However in one-time pad method, the keystream is a random generated number and it is independent of the plaintext so for two same plaintexts it may generate different keystreams.

The dependency between the keystream and plaintext makes this approach more secure against known-plaintext and chosen-plaintext attacks as a small change in plaintext changed the number of altered bits and their locations as well. This feature motivates us to apply this approach in RBS. One applicable mean for implementing this approach is through MAC function. As mentioned before, the redundant bits are generated through MAC as well. For the sake of resource sharing, the same MAC hardware is used for generating both redundant bits and keystream.

Instead of using the MAC(*P*) as a keystream, the MAC(redundant bits) which is equivalent to MAC(MAC(*P*)) will be used for generating the keystream as illustrated in Figure 6(a). As this figure shows, the generated keystream is XOR-ed with the plaintext and eventually is merged with redundant data.

The decryption process is illustrated in Figure 6(b). Knowing the key, the receiver extracts the redundant part from the ciphertext. Afterwards, the keystream will be generated through MAC(redundant) which then will be used for recovering the original plaintext bits.



(a) RBS Encryption *(b) RBS Decryption*

Figure 6. Block diagram of encryption and decryption

5. IMPLEMENTATION

The hardware implementation of RBS is composed of three main parts: MAC generator, encryption part, and decryption part which will be discussed in detail in the following subsections.

5.1 MAC Generator

The applied MAC system in RBS is a special case as it must be flexible at the size of MAC which is typically fixed for most MAC algorithms. For example, SHA-0 and MD5 algorithms generate 160-bit and 128-bit MACs respectively. The MAC algorithm proposed in [21] is from universal hash function family and supports variable-size output MAC digest which motivated us to use it in RBS.

The MAC algorithm in [21] is composed of one linear shift register (LFSR), one non-linear shift register (NFSR) and one accumulator (Figure 7). LFSR and NFSR together build up a pseudo random number generator (PRNG). The output of this PRNG, s(x), is the result of performing bitwise addition of LFSR with the result of NFSR function which feeds back into the NFSR. The value of s(x) depends on the initialized value of these two registers.

The accumulator register bits are XOR-ed by NFSR's value if (m_i = 1). The input m_i, is the input message which is checked by accumulator bit by bit.

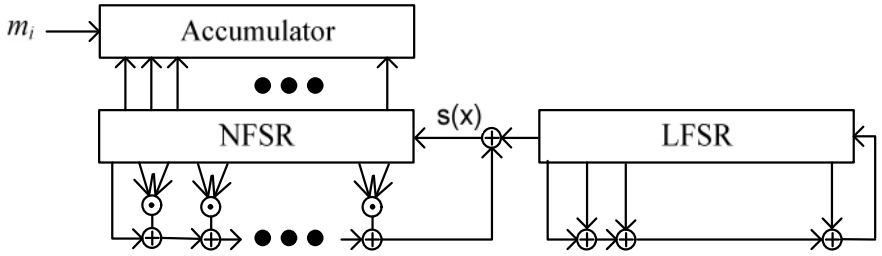


Figure 7. Hardware implementation of MAC algorithm in [21]

[LFSR,NFSR] registers are both initialized to authentication key and the accumulator register is initialized to zero. Then, the message enters bit by bit at each clock cycle. When all bits of input message are checked, the MAC will be ready in accumulator register.

Applying this hardware in RBS, the size of the accumulator and the NFSR registers must be equal to the length of the MAC or redundant data. In order to have the same key for both authentication and encryption, the size of LFSR key is set to n which forms a (n+m)-bit key when combined with m-bit NFSR key (Equation 2).

$$K_{authentication} = \{K_{LFSR}, K_{NFSR}\}$$
$$K_{LFSR}=\{K_0,\ldots,K_{n-1}\},\ K_{NFSR}=\{K_n,\ldots,K_{n+m-1}\} \qquad (2)$$

This MAC algorithm is inherently designed for stream ciphers and LFSR plays a major role in the process because its present state will be referred for refreshing the authentication key in the next communication step. Since RBS is a block cipher and it uses fixed authentication key for each communication step, so keeping LFSR register is not required anymore. However, the LFSR key is required in generating pseudo-random numbers. Therefore, the LFSR key enters to NFSR register bit by bit.

Similar to all universal hash functions, chosen MAC algorithm is guaranteed to be collision free as long as there is a unique key per each message [23]. So, there must be unique K_{NFSR} and K_{LFSR} per each message. To satisfy this condition in RBS, K_{NFSR} and K_{LFSR} are generated through performing bitwise addition on plaintext and initial key. The adapted MAC with RBS is shown in figure 8.

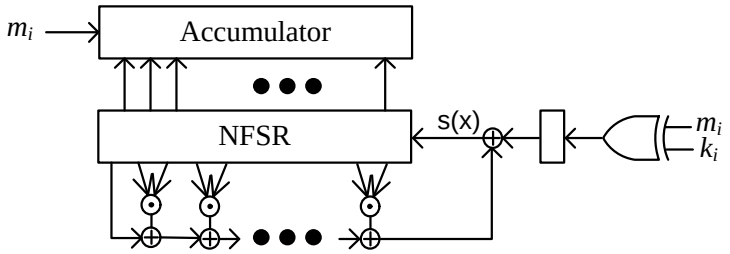


Figure 8- Adapted MAC with RBS

5.2 Encryption part

The encryption process completes in two phases. First, the plaintext bits are altered through bitwise addition with keystream which itself is the MAC(redundant data). In the second phase, the altered message is merged with redundant bits during data transmission. Figure 9 illustrates the process, where the altered plaintext bit (p_i), redundant bit (r_j), and encryption key bit (k_l) enter the cipher bit by bit and based on (k_l) value, either p_i or r_j will be transmitted.

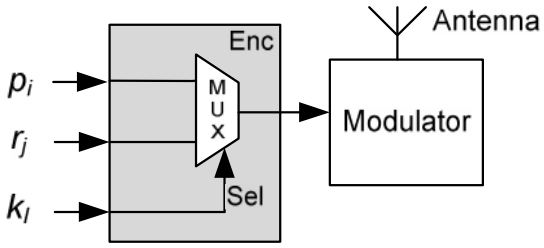


Figure 9. Encryption Part

5.3 Decryption part

At the first step of decryption process, redundant bits and altered plaintext bits will be extracted from the received ciphertext. The received bit will be considered as either altered plaintext bit or redundant bit based on the value of key. The keystream will be reconstructed using extracted redundant bits and the key. Performing bitwise addition on regenerated keystream and altered plaintext data, the original plaintext will be recovered. Eventually, the redundant data is regenerated by calculating MAC of the recovered plaintext and if it is different from received redundant data, the message will be discarded. The block diagram of the hardware implementation is presented in Figure 10.

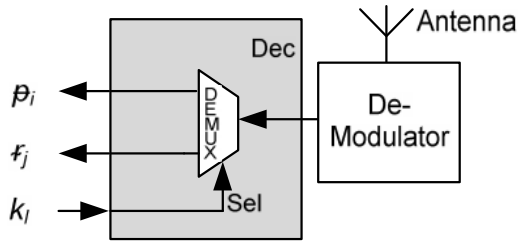


Figure 10. Decryption Part

5.4 Overall system

Figure 11 demonstrates the encryption and decryption parts together. Since the system is half-duplex; encryption and decryption do not happen at the same time; the En/De signal determines which process must be activated.

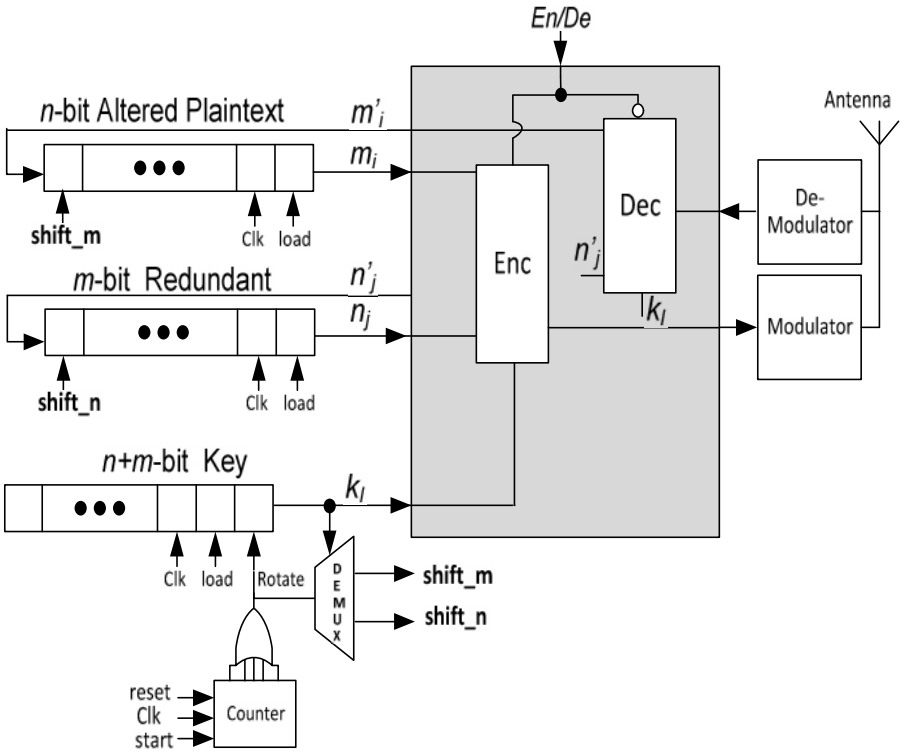


Figure 11. Cipher plus transmitter and receiver

#Encryption Algorithm	**#Decryption Algorithm**

counter = 0
for *i* in range (*n* + *m*)
{
if (key[*i*] = 0)
{
shift right plaintext register
send LSB(plaintext) to Enc module
}
else
{
shift right redundant register
send LSB(redundant) to Enc module
}
shift right key register
}
	counter = 0
	for *i* in range (*n* + *m*)
	{
	if (key[*i*] = 0)
	{
	Send data to MSB(plaintext)
	shift right plaintext register
	}
	else
	{
	send data to MSB(redundant)
	shift right redundant register
	}
	shift right key register
	}

m: plaintext data length *n*: redundant data length	

Figure 12. Encryption and Decryption algorithms

The encryption and decryption algorithms are shown in Figure 12. These algorithms are composed of successive steps of shifting and selection operations which allows encryption and decryption processes to be performed during sending and receiving data which makes the RBS implementation very efficient in terms of timing and power consumption. The only considerable overhead part is MAC function's implementation which is used for authentication, generation of redundant bits and keystream. The overhead details will be discussed in the next section.

6. EXPERIMENTAL RESULTS

The different components of RBS implementation are synthesized by Synopsys Design Compiler in 90nm technology mode and the clock frequency is set to 10 MHz. The operating conditions are set to typical, the supply voltage is fixed at 1V, and the temperature is set to 25°C. Table 2 summarizes the reported area and power figures.

Table 2. Area and power reports of modules

	Area (GE)	Total Power (μW)
MAC Generation	1051	30.2
Enc/Dec Cipher	10	0.26
Total	1061	30.46

The RBS algorithm is compared with five other encryption algorithms in i) required key and initial vector size, ii) data block size, iii) the required number of clock cycles for completing the encryption process, iv) 2-input NAND GE equivalent area, v) and total power consumption for *clk*=10MHz. Table 3 summarizes the comparison results. All other compared methods' reports are for 130nm technology whereas RBS is synthesized in newer 90nm technology which has considerable effect on area and power. However, area reports are given in GE (Gate equivalent) which is independent of used technology. Considering the effect of technology scaling (α) on power consumption, static power and dynamic power will be decreased by α^2. Therefore, it can be estimated that the power report in RBS must be doubled in order to be comparable with other compared designs in 130nm which is still lower than other designs' power consumption.

RBS like Trivium, Grain, and HB-2 ciphers have used initial vectors (IV) for refreshing key which imposes extra clock cycles for initializing cipher process during the algorithm startup or whenever the key changes. These initial clock cycles are distinguished in parenthesis in Table 3.

Regarding the message authentication service, this service is optional for Grain and HB-2 algorithms while AES, PRESENT, and Trivium do not provide authentication service. In other words, they must be integrated with other algorithms to provide this service. The timing, area and power consumption reports in Table 3 for AES, PRESENT, Trivium, Grain, and HB-2 algorithms are calculated without considering implementation overhead of authentication part.

RBS algorithm provides authentication service as mentioned before. Besides, the timing, area, and power reports for RBS algorithm listed in Table 3 include the MAC generator part's area/power overheads which are still better than other compared designs.

Table 3. Comparing RBS with other encryption methods

	Mode	Key/IV (bits)	Block Size(bits)	Clock Cycles	Area (GE)	Total Power (µW)	Freq. (MHz)	Tech.
AES[14]	En	128/-	128	160	3200	300	10	130nm
PRESENT[15]	En	128/-	64	32	1884	7.34	0.1	180nm
Grain[16]	En/De	128/(128)	1	(512)* + 1	1857	167.73	10	130nm
Trivium[17]	En/De	80/(80)	1	(1333)*+ 1	2599	181.18	10	130nm
HB-2[18]	En	128/64	16	(80)* + 16	2332	156.8	10	130nm
RBS	En/De/ Authent	132/-	64	(68)* + 136	1061	30.46	10	90nm
* Cycles required for initialization								

Concerning the clock cycles, encryption/decryption in RBS algorithm is performed along with data send/receive and the performance of RBS is only limited by the time required for generating MAC output which is 65 clock cycles for generating redundant bits, and 68 clock cycles for generating keystream. Another one clock cycle is also required for bitwise addition of keystream with plaintext and 2 cycles for generating authentication keys. Altogether, 136 clock cycles is the timing overhead for encryption/decryption plus authentication. Similar to area and power overheads, RBS timing overhead is still comparable with other algorithms while their reported timing overhead just considers encryption/decryption process.

7. CONCLUSION

A new lightweight symmetric authenticated cipher for RFID systems is proposed in this paper. This cipher is based on inserting redundant bits among altered plaintext bits. Besides confidentiality, redundant bits provide authentication and integrity services as well. The implemented RBS algorithm requires less power and area compared to other known symmetric algorithms proposed for RFID systems. In addition to the location of bits, changing their order in the ciphertext provides more security which is a part of our future work.

REFERENCES

[1] S.E. Sarma, S.A. Weis, D.W. Engels, “RFID systems and security and privacy implications” in Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES), August 2002, pp. 454-470.

[2] H. Sitlia, H. Hamam, S.A. Selouani, “Technical Solutions for Privacy Protection in RFID”, European Journal of Scientific Research, Volume 38, Number 3, 2009, pp.500-508.

[3] H. Eberle, N. Gura, S.C. Shantz, V. Gupta, L. Rarick, S. Sundaram, “A Public-key Cryptographic Processor for RSA and ECC”, Application-Specific Systems, Architectures and Processors, 2004. Page(s): 98 – 110.

[4] M. I. Faisal, Z. Jeddi, E. Amini and M. Bayoumi, “An Architecture for Variable Dimensional Finite Field GF(2m) Arithmetic Operations for Elliptic Curve Cryptography”, Journal of Low Power Electronics, Volume 7, Number 3, August 2011 , pp. 314-327.

[5] E. Amini, Z. Jeddi, A. Khattb and M. Bayoumi, “A Low-Power Parallel Architecture for Finite Galois Field GF(2m) Arithmetic Operations for Elliptic Curve Cryptography”, Journal of Low Power Electronics, Volume 8, Number 4, August 2012 , pp. 440-451.

[6] P. Luo; X. Wang; J. Feng; Y. Xu ,“Low-power hardware implementation of ECC processor suitable for low-cost RFID tags”, Solid-State and Integrated-Circuit Technology, ICSICT 2008.

[7] Y. K. Lee, K. Sakiyama, L. Batina, I. Verbauwhede, "Elliptic-Curve-Based Security Processor for RFID", IEEE Transactions on Computers, Volume: 57, Issue: 11, 2008 , Page(s): 1514 - 1527 .
[8] E. S. Kumar and C. Paar, "Are standards compliant Elliptic Curve Cryptosystems feasible on RFID", In Proceedings of Workshop on RFID Security, page 19, Graz, Austria, July 2006.
[9] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede, "Public-Key Cryptography on the Top of a Needle", In Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS 2007, pp. 1831-1834, May 2007.
[10] G. Gaubatz, J.P. Kaps, E. Ozturk, B. Sunar, "State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks", PerCom Workshops 2005, pp. 146-150.
[11] E. Ozturk, B. Sunar, "Low-Power Elliptic Curve Cryptography Using Scaled Modular Arithmetic" In Proceedings of the 6th International Workshop on Cryptographic Hardware in Embedded Systems (CHES), volume 3156 of Lecture Notes in Computer Science, pp. 92-106. Springer-Verlag, Aug 2004.
[12] G. Gaubatz, J.P. Kaps, E. Ozturk, B. Sunar, "State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks", In Proceedings of the 3rd IEEE international conference on pervasive computing and communications workshops, PerCom 2005, Page(s): 146 – 150.
[13] S. Ray and G. P. Biswas, "Design of Mobile Public Key Infrastructure (M-PKI) Using Elliptic Curve Cryptography", International Journal on Cryptography and Information Security (IJCIS), Vol.3, No.1, March 2013.
[14] P. Hamalainen, T. Alho, M. Hannikainen, T.D. Hamalainen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core", 9th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, IEEE Computer Society, DSD 2006.
[15] C. Rolfes, A. Poschmann, G. Leander and C. Paar, "Ultra-lightweight implementations for smart devices security for 1000 gate equivalents", International Conference on Smart Card Research and Advanced Applications, CARDIS 2008, pp. 89-103.
[16] M. Agren, M. Hell, T. Johansson, W. Meier, "A New Version of Grain-128 with authentication", Symmetric Key Encryption Workshop, SKEW 2011.
[17] C. De Canniere. "Trivium: A stream cipher construction inspired by block cipher design principles", Information Security Conference, pages 171–186, ISC 2006.
[18] D. Engels, M.J.O. Saarinen, P. Schweitzer E.M. Smith. "The Hummingbird-2 Lightweight Authenticated Encryption Algorithm", RFIDSec 2011.
[19] M.J.O. Saarinen, "Cryptanalysis of Hummingbird-1", 18th International Workshop Fast Software Encryption, FSE 2011, pp. 328-341.
[20] D. Hankerson, A. Menezes, S. Vanstone, "Guide to Elliptic curve cryptography", book, springer-verlag 2004.
[21] M. Agren, M. Hell, T. Johansson, "On Hardware-Oriented Message Authentication with Applications towards RFID", Lightweight Security & Privacy (LightSec), 2011.
[22] http://en.wikipedia.org/wiki/Brute-force_attack.
[23] L.H. Nguyen, A.W. Roscoe, "New combinatorial bounds for universal hash functions", IACR Cryptology ePrint Archive, 2009, p.153

Authors

Zahra Jeddi received her BS degree in electrical engineering from Iran University of Science and Technology and her MS degree in Computer Engineering from Amirkabir University of Technology. She is currently a Ph. D. candidate at the Center for Advanced Computer Studies (CACS) at the University of Louisiana at Lafayette. Her research interests include low power design, computer architecture and security.

Esmaeil Amini received his BS degree from Sharif University of Technology and MS degree from Amirkabir University of Technology both in Computer Engineering. He is a Ph. D. graduate at the Center for Advanced Computer Studies (CACS) at the University of Louisiana at Lafayette. His research interests include computer architecture, security and low power design.

Magdy Bayoumi received the B.Sc. and M.Sc. degrees in electrical engineering from Cairo University, Egypt. He received the M.Sc. degree in computer engineering from Washington University, St. Louis, MO, and the Ph.D. degree in electrical engineering from the University of Windsor, ON, Canada.

He is currently Director of the Center for Advanced Computer Studies (CACS) and Department Head of the Computer Science Department, University of Louisiana, Lafayette. He is also the Edmiston Professor of Computer Engineering and Lamson Professor of Computer Science at the Center for Advanced Computer Studies, University of Louisiana at Lafayette, where he has been a faculty member since 1985. He is editor or coeditor of three books in the area of VLSI Signal Processing. His research interests include VLSI design methods and architectures, low-power circuits, and systems, digital signal processing architectures, parallel algorithm design, computer arithmetic, image and video signal processing neural networks, and wideband network architectures.