

RBS: Redundant Bit Security algorithm for RFID systems

Zahra Jeddi, Esmaeil Amini and Magdy Bayoumi

The Center for Advanced Computer Studies

University of Louisiana at Lafayette

Lafayette, LA 70504, USA

{zxj4791, exa2685, mab}@cacs.louisiana.edu

Abstract— In this paper, we propose a symmetric encryption algorithm for RFID systems called RBS (Redundant Bit Security). RBS is based on inserting redundant bits into the original data bits. The location of redundant bits inside the transmitted data represents the secret key. The redundant bits are generated by Light Weight Mac algorithm in order to provide integrity and authentication as well. The implemented hardware architecture meets the resource constraints of RFID systems, and encryption and decryption time is shorter compared to existing algorithms.

I. INTRODUCTION

Radio Frequency Identification (RFID) provides automatic identification for any object like items, people, animals, etc. In general, each RFID based system consists of three parts. i) Transponder or tag that keeps data and is implemented on objects, ii) Transceiver or reader which provides electromagnetic field for activating tags and reading their data through radio frequency waves, and iii) A subsystem which receives data from reader, keeps data, and processes it.

Since the communication between tags and reader is done through an unsecure wireless channel, a mechanism is required to ensure the security of the communication. One common solution is encryption of the data before transmission. Encryption algorithms are divided into two main groups: Public key and private key algorithms.

Public key algorithms such as RSA and ECC are very strong in terms of security and provide reliability, confidentiality, integrity, availability and non-repudiation altogether. ECC-based systems offer similar security for smaller key sizes compared to RSA-based systems [1] and they are smaller, faster, and consume less power compared to RSA-based systems [2]. A lot of research has been done on hardware-efficient ECC implementations in the literature. In [3-4], the authors have proposed ECC algorithm which is adapted to RFID systems. They have reduced 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. Reducing the flexibility of ECC algorithm by limiting the parameters such as using only one special elliptic curve [5], selecting specific field sizes [6] or choosing specific prime numbers [7-9] are other ways to make ECC lighter. Although applying these limitations leads to meet the power constraints of RFID systems, but, any change in security parameters imposes replacing all tags with new ones.

Recently researches have been directed towards private key schemes as public key algorithms have still significant challenges for RFID systems' implementation. However, private key algorithms do not provide authentication and integrity alone. To overcome this problem, Message Authentication Code (MAC) algorithms such as HMAC have been proposed [10]. In this method, the message and a secret key_{mac} are fed into the MAC algorithm and the output digest is attached to the end of the message before encryption. The received message will be accepted if the MAC of the decrypted message is identical with the received MAC.

Alternative private key scheme has been proposed in [11] which is based on adding redundant bits to original data bits. These redundant bits are supposed to be random bits, and are generated by linear feedback shift register, LFSR. These bits then will be integrated with other symmetric encryption methods like AES to produce secure results. However, the authors have not mentioned how they have merged redundant bits with original bits and how their method provides confidentiality without bounding with other symmetric encryption algorithms. Besides, their method does not provide all security requirements alone and requires to be integrated with other symmetric algorithm. In this paper, we have tried to find answers for these questions.

In this paper, we propose 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 like [12], as the redundant bits and merging them with the altered original bits. This way, the redundant bits provide encryption, authentication and data integrity. This light MAC algorithm supports different digest sizes. Using this approach, the number of redundant bits and accordingly the security level could be adjusted in our system without requiring changing the underlying MAC algorithm.

The rest of this paper is organized as follows: The relation among key space, redundant bits and security is discussed in sections II. Then, we discuss the relationship between the locations of redundant bits in the ciphertext and their values in section III. In section IV the security analysis of the algorithm will be investigated. The proposed implementation and its evaluation are presented in sections V and VI respectively and section VII concludes the paper.

II. KEY SPACE

In RBS algorithm, we intentionally manipulate the message by inserting some redundant bits into original plaintext data. Suppose that the original data is “1010”. Adding one redundant “1” bit in the third place, we will have “10110”. Now the attacker confronts with four possible plaintexts while just one of them is the real one. In addition to the location of redundant bits, their values are important as well. For example, when the original data is a string of zeros, inserting one ‘0’ bit as redundant bit will not have the same effect as inserting a “1” redundant bit. Consequently, the number of redundant bits, their locations and their values are all important in hiding the plaintext inside the ciphertext. Altering some bits of the original bits in the ciphertext, the attacker will face with more possible plaintexts and finding the original plaintext becomes more challenging.

Suppose that n and m are the size of plaintext and redundant data respectively. The ciphertext is an $(n+m)$ bit data obtained by combining the original bits by redundant bits without any changes in the order of original bits. The location of redundant bits inside the ciphertext is the secret key. The key is an $(n+m)$ -bit string where 1's in this string present the locations of redundant bits and 0's present the locations of original bits in the ciphertext.

Key space is the set of all possible keys that can be used to initialize a cryptographic algorithm. The size of key space; the number of possible locations of redundant bits in the ciphertext, depends on n and m . Therefore, increasing m or n , the size of the key space s , will grow exponentially. The relationship between s , n and m is expressed by Equation (1).

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

In Equation (1), m and n are interchangeable; increasing either m or n has the same effect on key space size. Figure 1 shows the relation between s and m where the number of original bits is constant ($n=64$). Changing m from 0 to 128; key space s , will exponentially grow from 1 to 2^{172} .

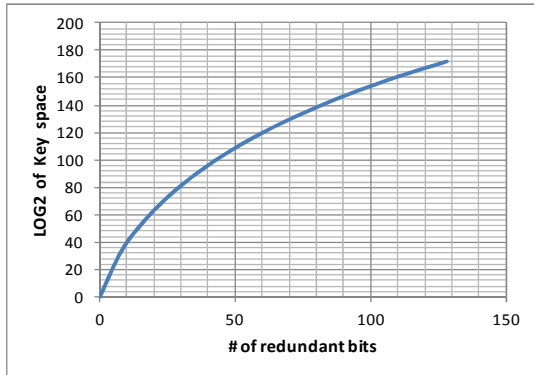


Figure 1. Key space growth while plaintext size is fixed (64 bits)

The Brute Force attack is one of the most effective attacks, where the attacker performs a complete search through all possible keys of the key space to find the right one. Therefore, the key space must be large enough otherwise the attacker can find the key by iterating through all possible keys. The 2^{128} key space size is computationally secure enough against brute-force attack [13]. Referring to equation

(1) n , m will be chosen such that the key space $s = 2^{128}$. Meanwhile, we want to minimize m ; the number of redundant bits. As Figure 2 shows, the minimum m , n happens when $m=n=66$ (132-bit ciphertext).

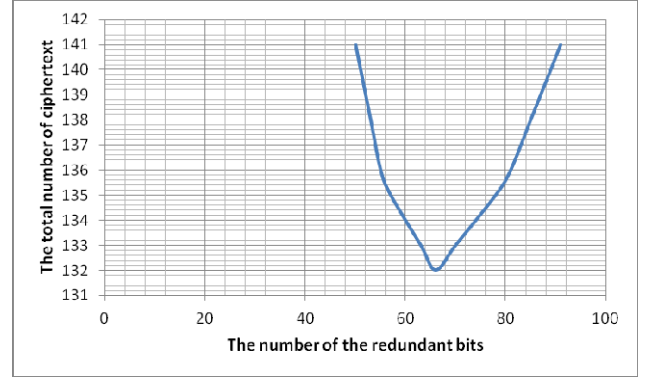


Figure 2. the relation between the number of redundant bits and ciphertext to provide 2^{128} key space

III. RBS METHOD

As mentioned in Section II, the number of redundant bits, their locations and their values are all important in hiding the plaintext inside the ciphertext which will be discussed in following subsections.

A. Redundant bits' location in the ciphertext

The locations of redundant bits inside the ciphertext define the secret key in our RBS algorithm. Therefore, the distribution of redundant bits inside the ciphertext should not follow any linear or non-linear mathematic function. Otherwise i) it will reduce the size of the key space, ii) it will make a dependency among redundant bits and iii) redundant bits will be distributed inside the ciphertext uniformly. This way, if the attacker figures out the location of one of the redundant bits, he can easily figure out other bits locations.

To increase the security level of RBS, each redundant bit's location must be independent of other redundant bits' locations. Hence, if the location of one of redundant bits or original bits becomes disclosed, it will not reveal other redundant bits' locations and just the key space will shrink.

B. Redundant bits' values

The values of the redundant bits are also important as they will make the attacker's job easier if they are chosen inappropriately. On the other hand, these bits can carry some information about the original data. For generating these redundant bits, we have three options:

- Choosing constant values: In this case the redundant bits are the same for different plaintexts. This way the attacker can easily figure out the location of the redundant bits by just comparing different ciphertexts.
- Choosing random values: In this case there would be several ciphertexts for one plaintext and the attacker can easily figure out the location of redundant bits by just comparing these ciphertexts in different times.
- Values of redundant bits are function of the plaintext: Therefore, there would be a unique redundant bit pattern per each plaintext. This way, the attacker cannot easily figure out redundant bits' locations.

We choose the last approach for generating redundant bits in our algorithm. One applicable implementation is using MAC algorithms where a very small change in the plaintext will produce a very different output. Meanwhile, MAC algorithms provide data integrity and authentication too.

At the sender side, the MAC generated redundant bits will be integrated with altered original bits based on encryption key. At the receiver side, the receiver breaks the received data into two parts: the altered original bits and redundant bits. By regenerating the redundant bits at the receiver side and comparing them with the received redundant bits, the receiver will figure out if the received data is intact or not. If these two are not equal, then data has either been changed by an attacker or by error during transmission and data will be discarded. This ensures integrity of the original data. Figure 3 demonstrates the process of encryption in transmitter side (TX) and decryption in receiver side (RX). K_{mac} is the key of MAC algorithm and K_{enc} is the encryption key used to merge the redundant bits with the manipulated original bits.

The key used for MAC algorithm (K_{mac}) is used for authenticating the sender as well. If the receiver decrypts the redundant bits string by the same key (K_{mac}) and the result differs from the expected string, it means that this transmitter is not eligible and the receiver data would be discarded.

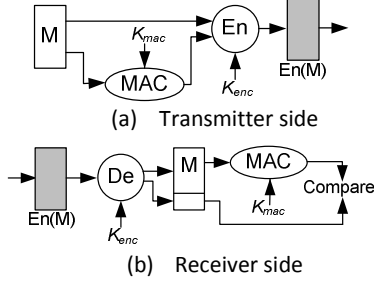


Figure 3. encryption in TX side and decryption in RX side

In typical MAC algorithm, the size of the output of MAC algorithms (digest) is generally fixed even for different sizes of inputs; each MAC algorithm has its own output sizes. For example, SHA-0 has 160-bit digest while MD5 has 128-bit digest. Since the digest of MAC represents redundant bits in our algorithm, the number of redundant bits must be chosen according to MAC algorithm. To overcome this limitation, we use light MAC algorithm presented in [12] as it supports different digest sizes.

C. plaintext Manipulation

To put the plaintext in the ciphertext we have four approaches.

- 1- Plaintext appears in the ciphertext directly which is vulnerable to both known plaintext and chosen plaintext attacks because it shrinks the key space.
- 2- Some bits of plaintext are altered before merging with redundant bits based on the encryption key. Regardless of plaintext pattern, some fixed bits of the plaintext will always be altered in the ciphertext. Suppose that K_{enc} is a binary sequence such that $K_{enc} = \{k_0, k_1, \dots, k_{n+m}\}$. This key will be broken into two binary sequences K_{enc1} and K_{enc2} such that $K_{enc1} = \{k_0, k_1, \dots, k_i\}$ and $K_{enc2} = \{k_{i+1}, k_{i+2}, \dots, k_{n+m}\}$ while $i = (n+m)/2$. Then, the plaintext will be

XOR-ed by $K'_{enc} = K_{enc1} \text{ XOR } K_{enc2}$. This way, the some bits of the plaintext will be altered depending on the value of K_{enc} and the number of altered bits is varying between zero bit (when $K_{enc1} = K_{enc2}$) to n (when $K_{enc1} = \sim K_{enc2}$). Since the attacker does not know the encryption key, he does not know how many bits of the plaintext and which of them have been altered. This way, the attacker confronts with 2^m different possible manipulated plaintexts. It makes the algorithm secure against known plaintext attack but it is still vulnerable to chosen plaintext attack. If the attacker change just one bit of the plaintext part, most of redundant bits in the ciphertext will change. Consequently, he can figure out the locations of original bits by comparing two generated ciphertexts of two almost same plaintexts.

- 3- Changing most of plaintext bits by XOR-ing them with redundant bits to make the algorithm secure against both known plaintext and chosen plaintext attacks. Since by changing one bit of plaintext, most redundant bits change, XOR-ing the plaintext and redundant bits changes most bits of plaintext as well. This solution is vulnerable when the plaintext is a string of zero. Since the result of XOR-ing the string of zeros with redundant bit is redundant bit, it will make the redundant bits be repeated in the ciphertext. Therefore, the attacker can easily find some of redundant bits' locations in the ciphertext in this special case because non-repetitive bits represent the location of redundant bits in ciphertext.
- 4- To overcome the problem in the third approach, we combine the second and third approaches together. In this new approach, the plaintext is XOR-ed with the K'_{enc} to obtain a different plaintext when the input plaintext is a zero string. Then, it is XOR-ed with the last bit of transmitted redundant bit before transmission. In our algorithm we have used this approach.

IV. SECURITY ANALYSIS

In this section, we investigate the security strength of RBS against Know Plaintext and Chosen Plaintext attacks. RBS algorithm generates a secure ciphertext for the attacker for the following reasons:

- Even if the attacker knows the plaintext, he does not know the value of redundant bits.
- Since the attacker does not know the key, for every bits of ciphertext, he does not know if this bit belongs to plaintext or redundant part.
- Even if the attacker knows that a particular bit belongs to plaintext, he does not know if this bit has been altered or not.
- Even if the attacker figures out that one particular bit of the ciphertext belongs to the plaintext he would not find out the position of that bit in the original plaintext.

Known plaintext attack: One of the most known methods for cracking the cryptographic algorithms is finding key by having a pair of plaintext and ciphertext. Most of known algorithms are vulnerable if the attacker knows the plaintext of a given ciphertext. The question is how long it takes for that algorithm to be cracked by this information. In our method, we investigate this situation by considering the

worst case; when the plaintext is a string of zero bits. In the ciphertext of this plaintext, i) some bits of plaintext have already been altered and they appeared as ‘1’ in the ciphertext, ii) values of redundant bits are unknown for attacker even if he knows the plaintext. To find the plaintext in this ciphertext, the attacker needs to find out for each bit in the ciphertext, if it belongs to plaintext or redundant part.

Chosen plaintext attack: in this attack, the attacker has access to the encryption device and tries to find the encryption key by comparing different plaintexts/ciphertexts. In our algorithm, by changing just one bit of plaintext, redundant bits will change. Besides, the plaintext is XOR-ed with redundant bits. Therefore most bits of new ciphertext will be changed. This way, by just changing one bit in the plaintext, the new ciphertext will completely be changed which makes it very secure against this attack.

V. IMPLEMENTATION

The hardware architecture of RBS consists of three parts: redundant bit generator, encryption part, and decryption part.

A. Redundant bit generator

The area complexity of redundant bit generator depends on MAC digest size. For example, for a 64-bit digest, the area complexity of light MAC presented in [12] is 1800 Gate Equivalent (GE). Figure 4 shows a typical implementation of this part.

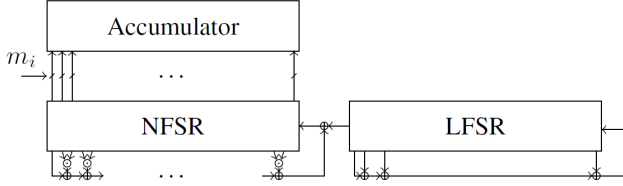


Figure 4. the hardware suggested for MAC in [12]

B. Encryption

Figure 5 shows a typical implementation of the encryption part in transmission side. This part merges n -bit plaintext with m -bit redundant bits based on the value of $n+m$ -bit key. As it mentioned before, each bit in the key presents the location of original bit and redundant bit in ciphertext. “0” means that this place is reserved for original bit and “1” means that it is reserved for redundant bit.

At first, the counter will be initialized to $n+m$. Next, the counter will countdown once and key register will be shifted to right once in each clock cycle. The least significant bit of key register defines whether the plaintext bit or the redundant bit must be sent to the modulator. If it is zero, the plaintext register will be shifted to right and its least significant bit is XOR-ed with the last transmitted bit of redundant register and K'_{enc} . The result will be sent to modulator. Otherwise, the redundant register will be shifted to right and its least significant bit will be sent to modulator. This procedure will be repeated for $n+m$ cycles until the counter becomes zero. At this time, both plaintext register and redundant-bits register have sent all their data to modulator. This part is composed of three XOR gates, one 2×1 bit MUX, one 1×2 bit DEMUX and one $\text{LOG}_2(n+m)$ -bit counter.

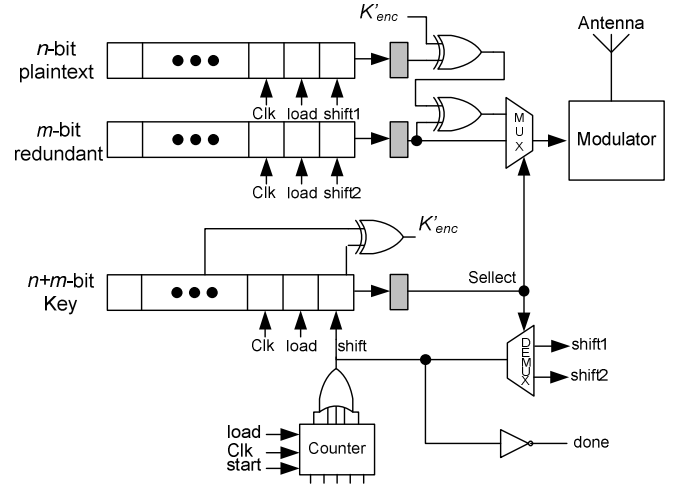


Figure 5. Encryption in transmission part

C. Decryption

The implementation of this part is very similar to encryption part. In this part, after receiving data from antenna and demodulating it, the received bit will be shifted to one of the plaintext register or the redundant register based on the least significant bit of the key register. This part is composed of two 1×2 bit DEMUX, two XOR gates and one $\text{LOG}_2(n+m)$ -bit counter.

To implement both encryption and decryption parts, we do not need to have the exact size of registers for the redundant and key registers. Since the counter keeps the number of loops and also key register keeps the number of redundant bits, the number of shifting for all registers will be done automatically in their own order. Limiting the maximum number of redundant bits to n , we can use n -bit redundant register and $2n$ -bit key register. It provides the flexibility to use this hardware for any number of redundant bits without any changes in the hardware.

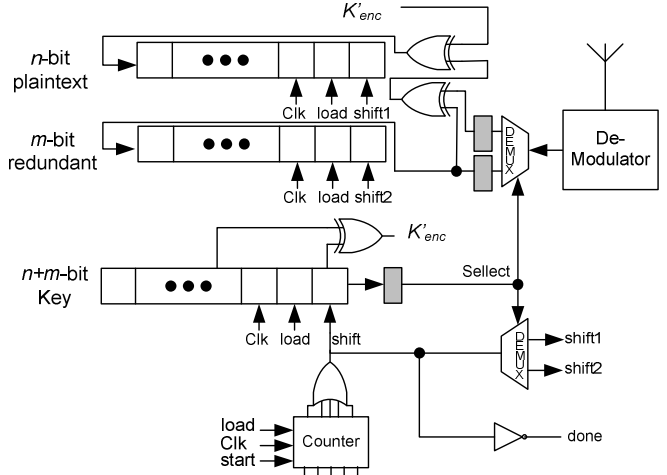


Figure 6. Decryption in receiver part

The area, timing and consequently the power overhead of both encryption and decryption parts is negligible since we have just shifting and selection modules. Inserting redundant bits (encryption) and removing them (decryption) will be done at the time of sending and receiving data. The only

overhead part is light MAC function which is used for authentication and generation of redundant bits which will be discussed in next section.

VI. EVALUATIONS

Overheads of RBS method can be divided into two parts: MAC part and encryption/decryption part. The overhead caused by MAC part depends on the number of redundant bits (digest size).

Table 1 - Comparing proposed method with other methods without considering MAC part

	RBS	AES [14]	PRESENT [15]	Grain [16]	HB2/ HW16[17]
# of gates	~120 GE (+1800 GE for MAC)	3200 GE	2332 GE	1857 GE	2332 GE
Clock cycle	66	1032	64	128	128
Block size	132 bits ($n=66, m=66$)	128 bits	64 bits	1 bit	16 bits

The area and performance overheads of our proposed system, AES, and PRESENT, Grain and HB2 are presented in Table 1. The encryption/decryption part of RBS system is composed of three 2×1 DEMUX, one 2×1 MUX, 5 Xor gates, one $\text{LOG}_2(n+m)$ -input OR gate and one $\text{LOG}_2(n+m)$ -bit counter. The counter is shared by encryption and decryption part as they do not occur at the same time. For $n=m=66$, the encryption/decryption part is implemented by almost 120 2-input NAND gates. This area overhead and consequently its power consumption overhead are negligible compared to other systems. Even considering the MAC part (1800 GE), the area complexity of our design is less than other designs. It should be noted that all algorithms in Table 1, except RBS, the authentication is optional which has its own area overhead. Besides, it imposes extra 64-bit data in ciphertext.

The encryption/decryption process in our proposed method, RBS is done during the transmission/reception. The performance of RBS is limited by generating MAC output which takes 66 clock cycles.

VII. CONCLUSION

We have proposed a new symmetric encryption method for RFID systems which is based on inserting redundant bits into original data bits. The proposed method provides authentication, integrity and confidentiality, all together. The security level of the proposed system can be adjusted without changing the underlying MAC algorithm. The timing, area and power consumption overhead of the proposed method is negligible which makes it applicable even for very resource limited RFID systems.

REFERENCES

- [1] H. Eberle, N. Gura, S.C. Shantz, V. Gupta, L. Rarick, "A Public-key Cryptographic Processor for RSA and ECC", Application-Specific Systems, Architectures and Processors, 2004. Page(s): 98 - 110
- [2] M. I. Faisal, Z. Jeddi, E. Amini and M. Bayoumi, "An Architecture for Variable Dimensional Finite Field $\text{GF}(2^m)$ Arithmetic Operations for Elliptic Curve Cryptography", Journal of Low Power Electronics, Volume 7, Number 3, August 2011, pp. 314-327(14)
- [3] 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.

- [4] Y. K. Lee, K. Sakiyama, L. Batina, I. Verbauwhede, "Elliptic-Curve-Based Security Processor for RFID", Computers, IEEE Transactions on Volume: 57, Issue: 11, 2008, Page(s): 1514 - 1527
- [5] S. 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.
- [6] 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'07, pp. 1831-1834, May 2007.
- [7] J. P. Kaps, "Cryptography for Ultra-Low Power Devices," Ph.D.dissertation, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 2006.
- [8] E. Öztürk, B. Sunar, and E. Savaş, "Low-Power Elliptic Curve Cryptography Using Scaled Modular Arithmetic," In Proceedings of the sixth International Workshop on Cryptographic Hardware in Embedded Systems (CHES), volume 3156 of Lecture Notes in Computer Science, pp. 92-106. Springer-Verlag, Aug 2004.
- [9] G. Gaubatz, J. P. Kaps, E. Öztürk, and 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, March 2005.
- [10] D. Hankerson, A. Menezes, S. Vanstone, "Guide to Elliptic curve cryptography", book, springer-verlag 2004
- [11] M.-L. Hsia, O. T.-C. Chen, "Low-Complexity Encryption Using Redundant Bits and Adaptive Frequency Rates in RFID", ISCAS 2007.
- [12] M. Agren, M. Hell, T. Johansson, "On Hardware-Oriented Message Authentication with Applications towards RFID", Lightweight Security & Privacy (LightSec), 2011.
- [13] http://en.wikipedia.org/wiki/Brute-force_attack
- [14] Feldhofer, M.; Wolkerstorfer, J.; Rijmen, V.; "AES implementation on a grain of sand", Information Security, IEE Proceedings 2005
- [15] A. Poschmann, "Lightweight Cryptography: Cryptographic Engineering for a Pervasive World", PhD Thesis
- [16] T. Good and M. Benaissa, "Hardware Results for Selected Stream Cipher Candidates", eSTREAM <http://www.ecrypt.eu.org/stream/papersdir/2007/023.pdf>
- [17] D. Engels, M. O. Saarinen and E.M. Smith, "The Hummingbird-2 Lightweight Authenticated Encryption Algorithm." RFIDSec 2011

APPENDIX

Here we demonstrate the RBS encryption by a simple example. The length of plaintext data and redundant part is 16 bits. Ciphertext1 is generated based on approach one (Section III-C) where the plaintext appears in the ciphertext directly. Ciphertext2 belongs to approach three where the plaintext is XOR-ed with the last transmitted redundant bit and ciphertext3 is generated by XOR-ing the plaintext with K'_{enc} and Redundant bit (Approach 4). The black typed bits represent redundant bits and the underlined ones represent plaintext data in the ciphertext.

```

Plaintext:      0000000000000000
Redundant part: 1100101001111001
Kenc:           01110110001010011101001010100101
K'enc:          1010010010001100
Ciphertext1:    011001000001000010100101000001
Ciphertext2:    111001000001000011111110000011
Ciphertext3:    011001000001000010111111010011

```

In ciphertext2, each one 1-0 and 0-1 transition presents the locations of redundant bits. This security problem is solved in ciphertext3 where transitions may happen either at redundant part or at plaintext part.