Implementation of a Secure Near Field Communication System on a FPGA

Ahmet Çağrı Bağbaba, Berna Örs

Istanbul Technical University, Turkey
bagbaba@itu.edu.tr, siddika.ors@itu.edu.tr

Abstract

NFC (Near Field Communication) based applications are used in many areas. Widely usage of these systems requires utilizing of cryptography algorithms in order to solve security problems. NFC includes RFID (Radio Frequency Identification) tags in its structure. In this work, the main focus is to provide authentication between two RFID tags on FPGA. We adopt Identity Based Signature scheme based Authentication Protocol, and hardware/software co-design Authentication Protocol system is implemented on FPGA to use NFC systems securely. We show the implementation of digital signatures in RFID tags.

1. Introduction

In recent years, the rapidly evolving wireless communication methods have become indispensable in many areas. With the NFC (Near Field Communication) and RFID (Radio Frequency Identification) technologies developed for this purpose, payment and contactless transactions such as public transport tickets and data transfer between mobile phones can be easily done [1]. These systems which have entered in many applications have security vulnerabilities and these security vulnerabilities can be used by malicious people. Because of these vulnerabilities, the usage of NFC systems under secure conditions has become a necessity. The most rational solution to ensure secure conditions for NFC application is the using of cryptography algorithms.

RFID is widely used for identification of moving or stationary objects. Despite the fact that the classical barcod systems are easy to use, they have disadvantages about data storage and changing of data on barcods. As a solution for these problems, RFID systems, which have smart cards inside and use radio frequency for identification through wireless devices, are emerged [2]. Examples of application areas of RFID systems are product distribution chain applications, patient identification, library, museum, and monitoring of valuable items in transportations [3]. In terms of their functions, RFID tags can be divided into three groups as active, passive, and semi passive [3].

In this paper, we have proposed an implementation of an authentication protocol between RFID tags and a reader on an FPGA (Field Programmable Gate Array). The authentication protocol was proposed by Ith, Oyama, Inomata, and Okamota in [4]. For implementation of this protocol, firstly, a hash function and a modular multiplication [5] modules were designed as hardware blocks and then all of these blocks were controlled by software implemented on the MicroBlaze processor [6].

2. Authentication Algorithm

In Ith, Oyama, Inomata, and Okamota's work, main target is designing of a digital signature on RFID tags. For this purpose, Identity Based Signature method was used [4].

2.1. Protocol

Figure 1 shows authentication protocol used in our implementation [4]. This protocol formed on Identity Based Signature [4]. Therefore, it has signature generation and verification parts. In the signature generation part the following steps are executed: 1) Get private key from the key generator 2) Compute the hash of the message to be signed 3) Compute the signature by using private key. In the signature verification part, the following steps are executed: 1) Get the corresponding public key 2) Compute the hash of the message to be signed 3) Compute the pairing for verification.

Unlike Ith, Oyama, Inomata, and Okamota's work, the pairing was not used in this work for verification. Instead of pairing, comparing of hash results was used for verification. It is obvious that if the hash results are the same, verification is achieved.

2.2. Identity Based Signature

The protocol in Figure 1 is based on Identity Based Signature method, so it is important to clarify this method. Identity Based Signature is a different form of Identity Based Encrpytion [4]. Signing of a message and verification of a signature is explained in Figure 2. There are two user as signer and verifier. The signer Alice signs the message by her private key called as $sk_{IDAlice}$. The verifier Bob verifies the signature by his public key called as pk_{PKG} and Alice's ID. Bob takes his public key from PKG (Public Key Generator). Signing and verification process can be explained in four part as Setup, Private Key Extraction, Signing, and Verification [4].

Setup: Public key and private key are generated by PKG. The public keys remain constant during whole process and are sent to all the users.

Private Key Extraction: The signer takes her own private key, which is about her identity, from Private Key Generator.

Signing: Using her private key, Alice signs the message M and sends M and the signature pair to the verifier Bob.

Verification: Bob verifies the signature with Alice's public key and identity.

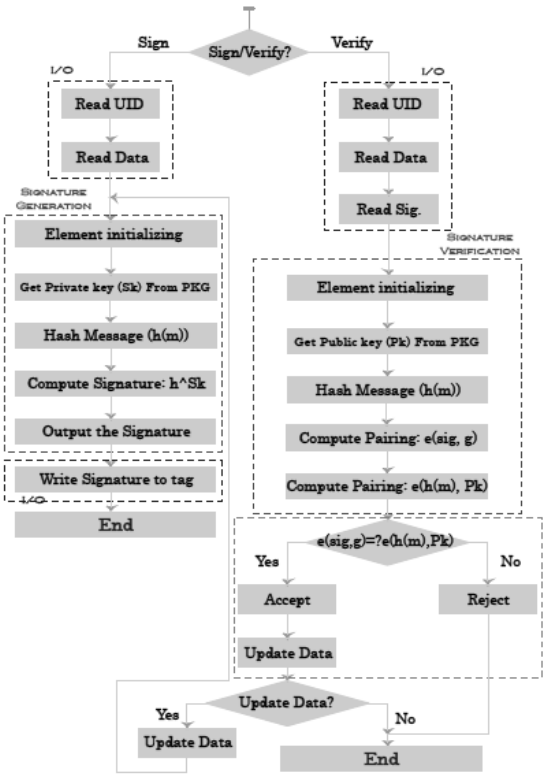

Fig. 1. Implemented protocol [4]

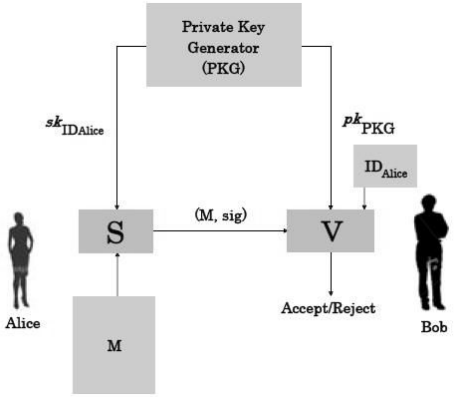

Fig. 2. Identity Based Signature Scheme [4]

3. Implementation of the Authentication Protocol

Implementation of the authentication protocol shown in Figure 1 is requires computation of the hashes of the messages and signature through modular exponentiation. Hence, main focus in this paper is to compute hash function and exponentiation.

The design has been implemented on Digilent Spartan-6 Design Platform [7]. There is a XCS6LX45 FPGA, produced by Xilinx on this board. The software design has been implemented on MicroBlaze, which is a soft-processor core designed by Xilinx for FPGAs [6].

3.1. Hash Function

Hash function is implemented by using *Encryption* block as shown in Figure 3. In every iteration a 64 bit block of the message that will be signed is fed to the hash block. The hash block includes *EXOR*, *Encryption* and *Register* blocks. After all the blocks of the message is finished the hash value is read from the *Register*.

EXOR, *Encryption* and *Register* blocks were designed in Xilinx ISE and Verilog. The main input of hash function is 64 bit message. The two outputs of the *EXOR* block have 32 bits, because of the structure of the *Encryption* block. Correspondingly, *Encryption* block has two output as 32 bits. *Register*'s output is connected to one of the *EXOR* inputs.

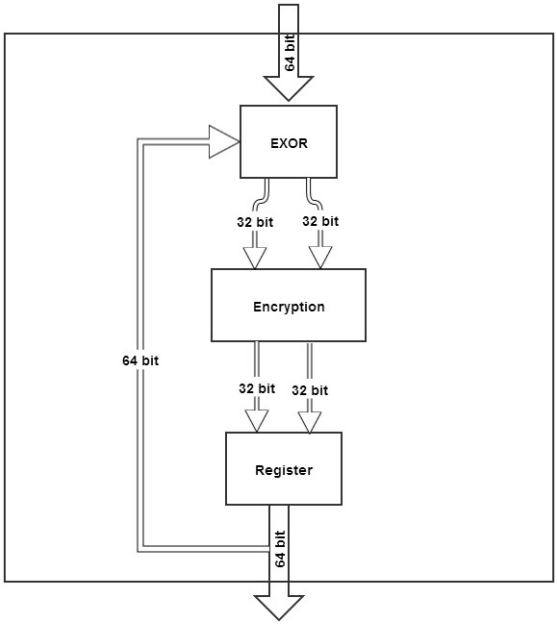

Fig. 3. Hash Function Structure

Tiny Encryption Algorithm (TEA), which was proposed by Wikram Reedy Andem in 2003, was used for *Encryption* block. The architecture of TEA is shown in Figure 4. TEA is suitable for embedded systems owing to high performance for embedded systems, ease of implementation, high speed, low energy consumption, low cost, and being lightweight [8]. The TEA design's main aim is to provide minimum memory space and have maximum speed. Moreover, it uses Feistel Encryption type. As a result of this, when plain text is changed 1 bit, this reflects to output, which name is chipper text, as 32 bit. Also, time performance of TEA is quite impressive for computers and workstations [9].

TEA includes 128 bit length encryption key, called as *K*. This key is divided into four parts as *K[0]*, *K[1]*, *K[2]*, and *K[3]* which are 32 bit length. In Figure 4, encryption routine of TEA can be seen. Encryption is comprised from 64 Feistel Cycle. The plain text is processed by dividing into two parts as 32 bit. Different keys are used for each cycle. End of the 64 cycle, chipper text is derived. Another constant delta is calculated from Equation 1 [9].

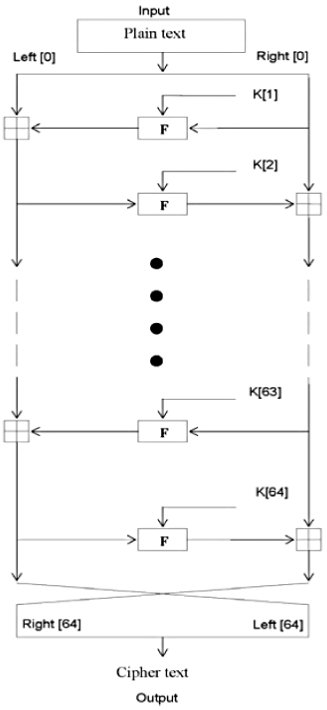

Fig. 4. TEA Encryption Routine [9]

$$Delta = (\sqrt{5} - 1) * 2^{31} = 9E3779B9_h \quad (1)$$

64 Feistel Cycle is equal to 32 cycle. This is explained in Figure 5. In other word, Figure 5 includes 64 Feistel Cycle or 32 cycle. Moreover, TEA has *Logical Shift*, *EXOR* and *Add* hardware, which are designed in Verilog. All these blocks are effective for each cycle.

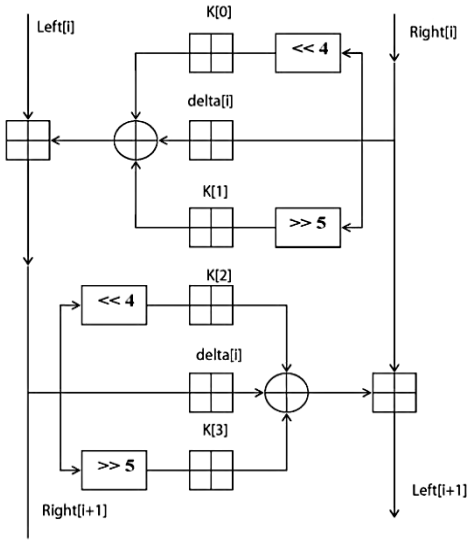

Fig. 5. TEA i. cycle [9]

3.2. Modular Exponentiation

In order to implement the authentication protocol, it is necessary to compute signature h^{Sk} by using hash and private key. Therefore, second design is exponentiation. For computing exponentiation, Arış's hardware design was used [5]. This design is Montgomery Modular Multiplication [10] and in VHDL. The illustration of the usage of modular multiplication hardware and MicroBlaze is in Figure 6. To obtain exponentiation by using this modular multiplication, Algorithm 1 [11] was used on MicroBlaze. Equation 2 shows that modular multiplication result, which we want to calculate, and Equation 3 shows that the Montgomery Modular Multiplication result obtained from MicroBlaze.

$$s = a.b \bmod p \quad (2)$$

$$Mont(a,b)=s^* = a.b.2^{-n} \bmod p \quad (3)$$

In Equation 2 and 3, s is the modular multiplication result, s^* is the Montgomery Modular Multiplication result, a is the multiplier, b is the multiplicand, p is the modulus, and n is the bit length of p. All bit lengths are 64 bit in this paper. Modulus p is chosen as 64 bit odd and prime number.

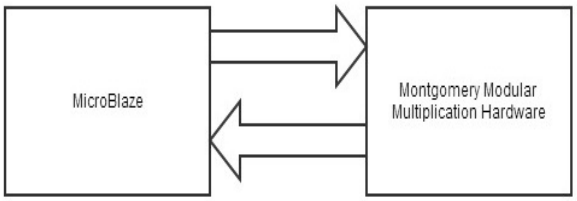

Fig. 6. Communication between microprocessor and hardware

Algorithm 1: Exponentiation [11]
INPUT: $p = (p_{n-1}...p_0)_b$, $R = b^n$, $p' = -p^{-1} \bmod b$, $e = (e_t...e_0)_2$ with $e_t = 1$, and an integer x, $1 \leq x < p$.
OUTPUT: $x^e \bmod p$.

1. X ← Mont(x, $R^2 \bmod p$), $A \leftarrow R \bmod p$.
2. For i from t down to 0 do the following:
   2.1 $A \leftarrow$ Mont(A, A).
   2.2 If $e_i = 1$ then $A \leftarrow$ Mont(A, X).
3. $A \leftarrow$ Mont (A,1).
4. Return (A).

In Algorithm 1, function *Mont* means Equation 3. Because of the fact that every Mont function brings R^{-1} multiplier, *Mont(x, R^2 mod p)* function is calculated at the beginning to provide only one R multiplier for each step.

3.3. Implementation of all the design on an FPGA

In order to implement the authentication protocol shown in Figure 1, all the design were combined in one MicroBlaze project. There are tag and reader, which are used for authentication. The design on the MicroBlaze project is shown in Figure 7.

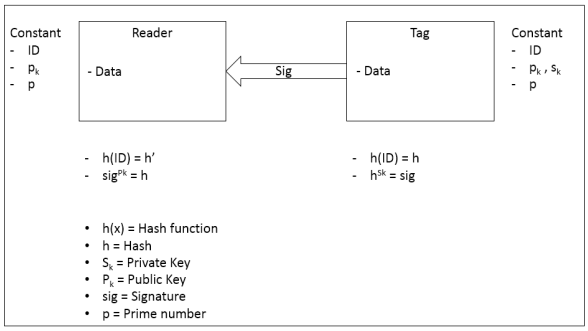

Fig. 7. The Authentication Protocol on the MicroBlaze Project

Verification is provided in reader in case of $h = h'$. This method was used instead of pairing which is in protocol. Firstly, Public Key (*pk*) was determined and then Private Key (*sk*) was obtained using *pk* and Equation 4. Equation 4 is calculated by aid of Mont function.

$$pk = sk^{-1} \bmod p \tag{4}$$

4. Implementation Results

Whole system's operating frequency is 100 MHz. Hash Function and Montgomery Modular Multiplication device utilization summary are shown respectively in Table 1 and Table 2. In Hash Function, TEA encrypts plain text in 290.563 ns.

Table 1. Hash Function Device Utilization Summary

Hash Function			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	135	54576	0%
Number of Slice LUTs	4169	27288	15%
Number of fully used LUT-FF pairs	135	4169	3%
Number of bonded IOBs	133	218	61%
Number of BUFG/BUFGCTRLs	1	16	6%

Table 2. Montgomery Modular Multiplication Device Utilization Summary

Montgomery Modular Multiplication			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	484	54576	0%
Number of Slice LUTs	700	27288	2%
Number of fully used LUT-FF pairs	404	780	51%
Number of bonded IOBs	260	218	119%
Number of BUFG/BUFGCTRLs	1	16	6%

5. Conclusion

This is the first practical implementation of the authentication protocol given in [4]. In this work, an authentication protocol was implemented on MicroBlaze for NFC systems. Hash Function and Modular Multiplication hardware were used and connected to the soft core processor MicroBlaze. On MicroBlaze, Exponentiation was calculated using Modular Multiplication hardware. Whole protocol was completed using Hash Function and Exponentiation outputs. Xilinx Spartan 6 FPGA was used for implementation and results were observed.

The main challenge during this work was controlling of Hash Function hardware via MicroBlaze because of the finite state machine structure of Hash Function. Designing *EXOR* block out of the finite state machine was the solution of this problem. However, this solution has slowed down the system.

6. References

[1] NFC Forum [Online]. Available: http://www.nfc-forum.org/aboutnfc/.

[2] Feldhofer, M., "An authentication protocol in a security layer for RFID smart tags," *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean* , vol.2, no., pp.759,762 Vol.2, 12-15 May 2004 doi: 10.1109/MELCON.2004.1347041

[3] Kavas, A. "Radyo Frekans Tanımlama Sistemleri", *Elektrik Mühendisliği, 80,* 430, pp. 74-80, 2007.

[4] Ith, P.; Oyama, Y.; Inomata, A.; Okamoto, E., "Implementation of ID-based signature in RFID system," *Communications, 2007. APCC 2007. Asia-Pacific Conference on* , vol., no., pp.233,236, 18-20 Oct. 2007 doi: 10.1109/APCC.2007.4433543

[5] Arış, A. "Design and Implementation of RSA Cryptosystem Using Partially Interleaved Modular Karatsuba-Ofman Multiplier", M.Sc. Thesis, Istanbul Technical University Computer Engineering, İstanbul, 2012.

[6] MicroBlaze Processor Reference Guide [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf

[7] Atlys™ Spartan-6 FPGA Development Board [Online]. Available: http://www.digilentinc.com.

[8] Abdelhalim, M. B.; El-Mahallawy, M.; Ayyad, M.; Elhennawy, A., "Implementation of a modified lightweight cryptographic TEA algorithm in RFID system", *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for* , vol., no., pp.509,513, 11-14 Dec. 2011

[9] Andem, V.R., 2003. A Cryptanalysis of the Tiny Encryption Algorithm, MSc. Thesis, The University of Alabama, ALABAMA.

[10] Montgomery, P.L, ”Modular Multiplication without trial division”, *Mathematics of Computation*, vol.44, no.170, pp 519-521, April 1985.
[11] Menezes, A., Van Oorschot, P. and Vanstone, S., “Handbook of Applied Cryptography”, CRC Press, 1996.