Revini S. Shende, Mrs. Anagha Y. Deshpande / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 2, March -April 2013, pp.742-746

VLSI Design Of Secure Cryptographic Algorithm

Revini S. Shende, Mrs. Anagha Y. Deshpande
(Department of Electronics and Telecommunication, Smt. Kashibai Navale College of Engg., University of Pune, Pune)
(Department of Electronics and Telecommunication, Smt. Kashibai Navale College of Engg., University of Pune, Pune)

ABSTRACT

Lightweight cryptography (LWC) is an emerging research area which has to deal with the trade-off among security, cost, and performance. In this paper we present the idea and list some types of LWC algorithms. Hummingbird is a novel ultra lightweight cryptographic algorithm targeted for resource constrained devices like RFID tags, smart cards and wireless sensor nodes. The hybrid model of Hummingbird is explained keeping the constraint devices in mind and thus resulting in an easier software implementation. The paper presents the algorithms for the encryption as well as decryption process and shows some simulation results performed on Xilinx.

Keywords - **Lightweight cryptography, resource constrained devices, Hummingbird**

1.Introduction

Low-cost smart devices like RFID tags and smart cards are rapidly becoming pervasive in our daily life. Well known applications include electronic passports, contactless payments, product tracking, access control and supply chain management just to name a few. But the small programmable chips that passively respond to every reader have raised concerns among researchers about privacy and security breaches. A considerable body of research has been focused on providing RFID tags with cryptographic functionality, while scarce computational and storage capabilities of low cost RFID tags make the problem challenging. This emerging research area is usually referred as LWC which has to deal with the trade-off among security, cost, and performance.

LWC is a branch of modern cryptography which covers cryptographic algorithms intended for use in devices with low or extremely low resources. LWC does not determine strict criteria for classifying a cryptographic algorithms as lightweight, but the common features of lightweight algorithms are extremely low requirements to essential resources of target devices[1]

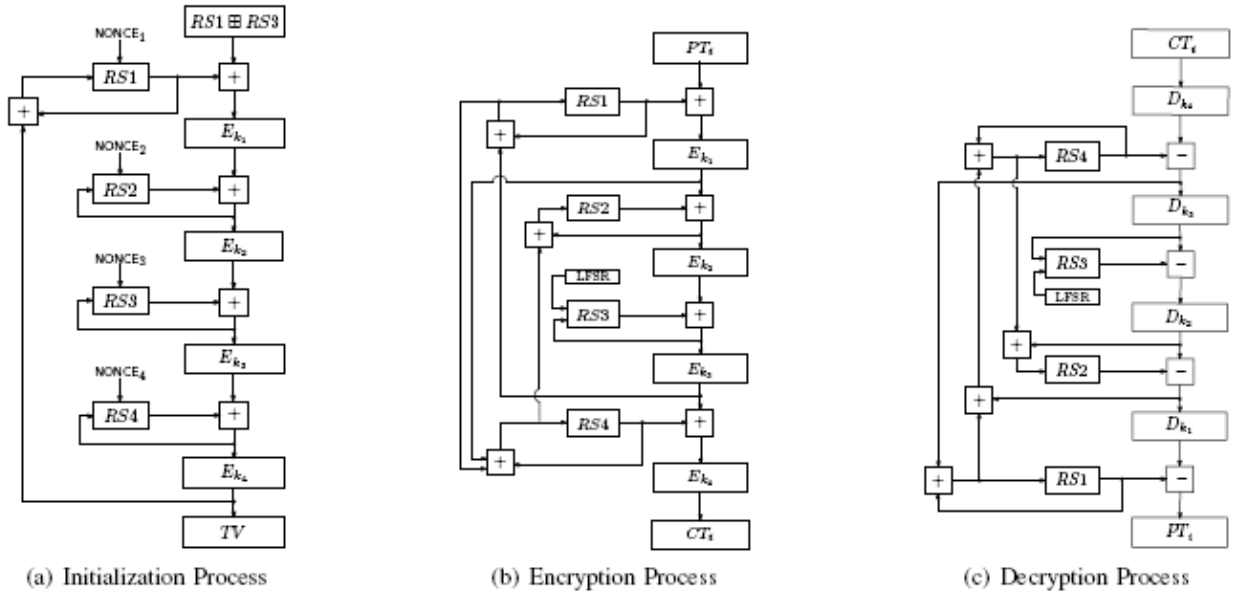
Some features of LWC are cryptography tailored to (extremely) constrained devices, strong cryptos, not intended for all- powerful adversaries, not intended to replace traditional cryptography. But LWC should influence new algorithms. lightweight algorithms should have a short internal state(to lower area), allow serialization (to lower power), have a short processing time(to lower energy), have a short output(to lower communication cost)[8]

1.1 Types of LWC Algorithm

The key issue of designing LWC algorithm is to deal with trade- off among security, cost and performance and find an optimal cost performance ratio. Quite a few lightweight symmetric ciphers that particularly target resource- constrained devices have been published in the past few years and those ciphers can be utilized as building blocks to design security mechanisms for embedded applications. All the previous proposals can be roughly divided into the following 3 categories.

The first category consists of highly optimized and compact hardware implementations for standardized block ciphers such as AES, IDEA and XTEA, whereas the proposals in the second category involve slight modifications of a classical block cipher like DES for lightweight applications. Finally, the third category features new low cost designs, including lightweight block ciphers SEA, PRESENT and KATAN and KTANTAN as well as lightweight stream ciphers Grain, Trivium and MICKEY. A good survey covering recently published LWC implementation can be found in [8].

Hummingbird is a recently proposed ultra LWC targeted for low-cost smart devices. It has a hybrid *structure* of block cipher and stream cipher and is developed with both lightweight software and lightweight hardware implementations for constrained devices in mind. The hybrid model can provide the designed security with a small block size and is therefore expected to meet the stringent response time and power consumption requirements for a variety of embedded applications.



(a) Initialization Process (b) Encryption Process (c) Decryption Process

Fig. 1. A Top-Level Description of the Hummingbird Cryptographic Algorithm

Hummingbird is resistant to the most common attacks to block ciphers and stream ciphers including birthday attack, differential and linear cryptanalysis , structure attacks, algebraic attacks, cube attacks, etc

1.2 Hummingbird Cryptographic Algorithm

Hummingbird is neither a block cipher nor a stream cipher, but a rotor machine equipped with novel rotor-stepping rules. The design of Hummingbird is based on an elegant combination of block cipher and stream cipher with 16-bit block size, 256-bit key size, and 80-bit internal state. The size of the key and the internal state of Hummingbird provides a security level which is adequate for many embedded Applications.

For clarity, the notation listed in Table No.1 are used in the algorithm description. A top-level structure of the Hummingbird cryptographic algorithm is shown in Figure 1, which consists of four 16-bit block ciphers E_{ki} or D_{ki} (i = 1; 2; 3; 4), four 16-bit internal state registers RSi (i = 1; 2; 3; 4), and a 16-stage Linear Shift Feedback Register (LFSR). Moreover, the 256-bit secret key K is divided into four 64-bit subkeys k_1; k_2; k_3 and k_4 which are used in the four block ciphers, respectively.

1.2.1Initialisation Process

The overall structure of the Hummingbird initialization algorithm is shown in Figure 1(a). When using Hummingbird in practice, four 16-bit random nonces $NONCE_i$ are first chosen to initialize the four internal state registers RSi (i = 1; 2; 3; 4),

PT_i	the i-th 16-bit plaintext block, $i = 1, 2, \ldots, n$
CT_i	the i-th 16-bit ciphertext block, $i = 1, 2, \ldots, n$
K	the 256-bit secret key
$\mathbf{E}_K(\cdot)$	the encryption function of Hummingbird with 256-bit secret key K
$\mathbf{D}_K(\cdot)$	the decryption function of Hummingbird with 256-bit secret key K
k_i	the 64-bit subkey used in the i-th block cipher, $i = 1, 2, 3, 4$, such that $K = k_1 \| k_2 \| k_3 \| k_4$
$E_{k_i}(\cdot)$	a block cipher encryption algorithm with 16-bit input, 64-bit key k_i, and 16-bit output, i.e., $E_{k_i} : \{0,1\}^{16} \times \{0,1\}^{64} \rightarrow \{0,1\}^{16}, i = 1, 2, 3, 4$
$D_{k_i}(\cdot)$	a block cipher decryption algorithm with 16-bit input, 64-bit key k_i, and 16-bit output, i.e., $D_{k_i} : \{0,1\}^{16} \times \{0,1\}^{64} \rightarrow \{0,1\}^{16}, i = 1, 2, 3, 4$
RSi	the i-th 16-bit internal state register, $i = 1, 2, 3, 4$
LFSR	a 16-stage Linear Feedback Shift Register with the characteristic polynomial $f(x) = x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^3 + 1$
$\boxplus$	modulo 2^{16} addition operator
$\boxminus$	modulo 2^{16} subtraction operator
$\oplus$	exclusive-OR (XOR) operator
$m \lll l$	left circular shift operator, which rotates all bits of m to the left by l bits, as if the left and the right ends of m were joined.
$K_j^{(i)}$	the j-th 16-bit key used in the i-th block cipher, $j = 1, 2, 3, 4$, such that $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$
$S_i(x)$	the i-th 4-bit to 4-bit S-box used in the block cipher, $S_i(x) : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4, i = 1, 2, 3, 4$
NONCE_i	the i-th nonce which is a 16-bit random number, $i = 1, 2, 3, 4$
IV	the 64-bit initial vector, such that $IV = \mathsf{NONCE}_1 \| \mathsf{NONCE}_2 \| \mathsf{NONCE}_3 \| \mathsf{NONCE}_4$

respectively, followed by four consecutive encryptions on the message RS1$\boxplus$ RS3 by Hummingbird running in initialization mode (see Figure 1(a)). The final 16-bit ciphertext TV is used to initialize the LFSR. Moreover, the 13th bit of the LFSR is always set to prevent a zero register. The LFSR is also stepped once before it is used to update the internal state register *RS*3. The algorithm for Hummingbird initialization process is as follows:

Algorithm 1 Hummingbird Initialization

Input: Four 16-bit random nonce NONCE_i $(i = 1, 2, 3, 4)$
Output: Initialized four rotors RSi_4 $(i = 1, 2, 3, 4)$ and LFSR

1: $RS1_0 = \mathsf{NONCE}_1$ [Nonce Initialization]
2: $RS2_0 = \mathsf{NONCE}_2$
3: $RS3_0 = \mathsf{NONCE}_3$
4: $RS4_0 = \mathsf{NONCE}_4$
5: **for** $t = 0$ to 3 **do**
6: $V12_t = E_{k_1}((RS1_t \boxplus RS3_t) \boxplus RS1_t)$
7: $V23_t = E_{k_2}(V12_t \boxplus RS2_t)$
8: $V34_t = E_{k_3}(V23_t \boxplus RS3_t)$
9: $TV_t = E_{k_4}(V34_t \boxplus RS4_t)$
10: $RS1_{t+1} = RS1_t \boxplus TV_t$
11: $RS2_{t+1} = RS2_t \boxplus V12_t$
12: $RS3_{t+1} = RS3_t \boxplus V23_t$
13: $RS4_{t+1} = RS4_t \boxplus V34_t$
14: **end for**
15: LFSR = TV_3 | 0x1000 [LFSR Initialization]
16: **return** RSi_4 $(i = 1, 2, 3, 4)$ and LFSR

1.2.2Encryption Process

The overall structure of the Hummingbird encryption algorithm is depicted in Fig. 1(b). After a system initialization process, a 16-bit plaintext block PT_i is encrypted by first executing a modulo 2^{16} addition of PT_i and the content of the first internal state register *RS*1. The result of the addition is then encrypted by the first block cipher E_{k1} .

This procedure is repeated in a similar manner for another three times and the output of E_{k4} is the corresponding ciphertext CT_i. Furthermore, the states of the four internal state registers will also be updated in an unpredictable way based on their current states, the outputs of the first three block ciphers, and the state of the LFSR. Algorithm 2 describes the detailed procedure of Hummingbird encryption:

1.2.3Decryption Process

The overall structure of the Hummingbird decryption algorithm is illustrated in Figure 1(c). The decryption process follows the similar pattern as the encryption and a detailed description is shown in the following Algorithm 3.

Algorithm 2 Hummingbird Encryption

Input: A 16-bit plaintext PT_i and four rotors RSi_t $(i = 1, 2, 3, 4)$
Output: A 16-bit ciphertext CT_i

1: $V12_t = E_{k_1}(PT_i \boxplus RS1_t)$ [Block Encryption]
2: $V23_t = E_{k_2}(V12_t \boxplus RS2_t)$
3: $V34_t = E_{k_3}(V23_t \boxplus RS3_t)$
4: $CT_i = E_{k_4}(V34_t \boxplus RS4_t)$
5: $\mathsf{LFSR}_{t+1} \leftarrow \mathsf{LFSR}_t$ [Internal State Updating]
6: $RS1_{t+1} = RS1_t \boxplus V34_t$
7: $RS3_{t+1} = RS3_t \boxplus V23_t \boxplus \mathsf{LFSR}_{t+1}$
8: $RS4_{t+1} = RS4_t \boxplus V12_t \boxplus RS1_{t+1}$
9: $RS2_{t+1} = RS2_t \boxplus V12_t \boxplus RS4_{t+1}$
10: **return** CT_i

Algorithm 3 Hummingbird Decryption

Input: A 16-bit ciphertext CT_i and four rotors RSi_t $(i = 1,2,3,4)$

Output: A 16-bit plaintext PT_i

$V34_t = D_{k_4}(CT_i) \boxminus RS4_t$ [Block Decryption]
$V23_t = D_{k_3}(V34_t) \boxminus RS3_t$
$V12_t = D_{k_2}(V23_t) \boxminus RS2_t$
$PT_i = D_{k_1}(V12_t) \boxminus RS1_t$
$\text{LFSR}_{t+1} \leftarrow \text{LFSR}_t$ [Internal State Updating]
$RS1_{t+1} = RS1_t \boxplus V34_t$
$RS3_{t+1} = RS3_t \boxplus V23_t \boxplus \text{LFSR}_{t+1}$
$RS4_{t+1} = RS4_t \boxplus V12_t \boxplus RS1_{t+1}$
$RS2_{t+1} = RS2_t \boxplus V12_t \boxplus RS4_{t+1}$
return PT_i

1.2.416-Bit Block Cipher

Hummingbird employs four identical block ciphers $E_{ki}(\cdot)(i = 1; 2; 3; 4)$ in a consecutive manner, each of which is a typical substitution-permutation (SP) network with 16-bit block size and 64-bit key as shown in the figure 2.

The block cipher consists of four regular rounds and a final round. The 64-bit subkey k_i is split into four 16-bit round keys $K_1^{(i)}$, $K_2^{(i)}$, $K_3^{(i)}$ and $K_4^{(i)}$ that are used in the four regular rounds, respectively.Moreover, the final round utilizes two keys $K_5^{(i)}$ and $K_6^{(i)}$ directly derived from the four round keys.

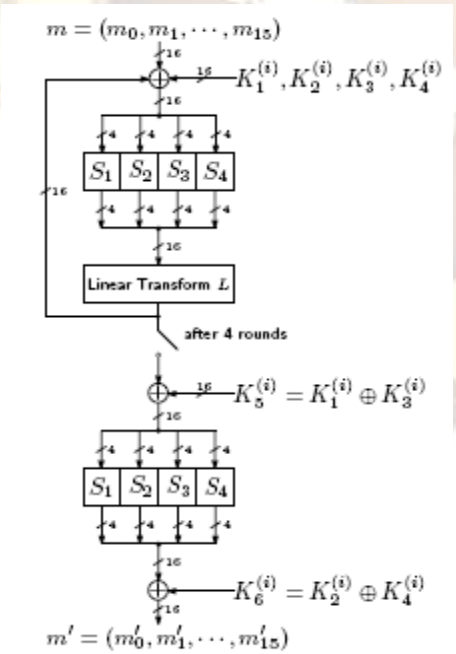


Fig. 2 The structure of block cipher in the Hummingbird cryptography algorithm.

While each regular round comprises of a key mixing step, a substitution layer, and a permutation layer, the final round only includes the key mixing and the S-box substitution steps. The key mixing step is implemented using a simple exclusive-OR operation, whereas the substitution layer is composed of four S-boxes with 4-bit inputs and 4-bit outputs as shown in Table No.2

Table No. 2 S-Boxes Used

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	8	6	5	F	1	C	A	9	E	B	2	4	7	0	D	3
$S_2(x)$	0	7	E	1	5	B	8	2	3	A	D	6	F	C	4	9
$S_3(x)$	2	E	F	5	C	1	9	A	B	4	6	8	0	7	3	D
$S_4(x)$	0	7	3	4	C	1	A	F	D	E	6	B	2	8	9	5

The selected four S-boxes, denoted by $S_i(x)$: $F_4^2 \rightarrow F_4^2$; $i = 1; 2; 3; 4$, are Serpent-type S-boxes ensure that the 16-bit block cipher is resistant to linear and differential attacks as well as interpolation attack. The permutation layer in the 16-bit block cipher is given by the following linear transform L: $\{0,1\}^{16} \rightarrow \{0,1\}^{16}$ defined as follows:

$$L(m) = m \oplus (m \lll 6) \oplus (m \lll 10),$$

where $m = (m_0;m_1;\ldots\ldots;m_{15})$ is a 16-bit data block. We give a detailed description for the encryption process of the 16-bit block cipher in the following Algorithm 4:

Algorithm 4 16-bit Block Cipher Encryption $E_{k_i}(\cdot)$

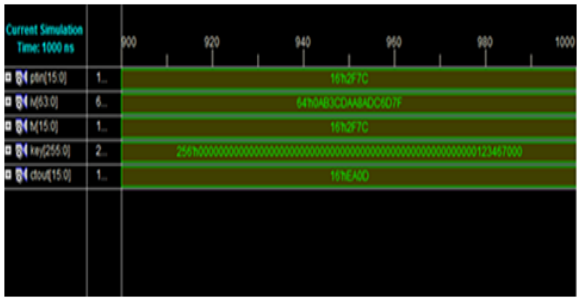
Input: A 16-bit data block $m = (m_0, m_1, \cdots, m_{15})$ and a 64-bit subkey k_i such that
subkey $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$

Output: A 16-bit date block $m' = (m'_0, m'_1, \cdots, m'_{15})$

1: **for** $j = 1$ to 4 **do**
2: $m \leftarrow m \oplus K_j^{(i)}$ [key mixing step]
3: $A = m_0 \| m_1 \| m_2 \| m_3$, $B = m_4 \| m_5 \| m_6 \| m_7$
$C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$
4: $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$
[substitution layer]
5: $m \leftarrow m \oplus (m \lll 6) \oplus (m \lll 10)$
[permutation layer]
6: **end for**
7: $m \leftarrow m \oplus K_1^{(i)} \oplus K_3^{(i)}$
8: $A = m_0 \| m_1 \| m_2 \| m_3$, $B = m_4 \| m_5 \| m_6 \| m_7$
$C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$
9: $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$
10: $m' \leftarrow m \oplus K_2^{(i)} \oplus K_4^{(i)}$
11: **return** $m' = (m'_0, m'_1, \cdots, m'_{15})$

The decryption process can be easily derived from the encryption and therefore is omitted here. This section describes the basic steps of the Hummingbird cryptographic algorithm along with their respective diagrams and algorithms that can be used to implement the algorithm efficiently.

2. Simulation Results



The above result is obtained on Xillinx ISE Modelsim by giving a 64-bit initialization vector 0ABECDAA8ADC6DTF which generates a plaintext of 16- bit 2F7C. Key of length 256 bit is used with value 123470000 and cipher text obtained is 16 bit EA0D. The simulation is kept for 1000ns.

3. Conclusion

This paper details about lightweight cryptography and its types and discusses the implementation of ultra lightweight cryptographic algorithm Hummingbird. The security and performance factor is very precisely achieved by the algorithm due to its prominent internal structure.

Compared to other lightweight FPGA implementations of block ciphers XTEA, ICEBERG, SEA, AES, Hummingbird can achieve larger throughput with the smaller area requirement. Consequently, Hummingbird can be considered as an ideal cryptographic primitive for resource constrained environment.

The efficient FPGA implementation of Hummingbird is possible using the given software algorithms so that it can achieve larger throughput with smaller area requirement. Also, Hummingbird can be used in high-security required devices as it is resistant to most cryptographic attacks.

REFERENCES

[1] Sergey Panasenko and Sergey Smagin, "Lightweight Cryptography: Underlying Principles and Approaches", *International Journal of Computer Theory and Engineering, Vol. 3, No. 4, August 2011.*

[2] X. Fan, G. Gong, K. Lauffenburger, and T. Hicks, "FPGA Implementations of the Hummingbird Cryptographic Algorithm", IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2010.

[3] K. Lauffenburger, X. Fan, G. Gong, T. Hicks, "Design Space Exploration of Hummingbird Implementations on FPGA's" Centre for Applied Cryptographic Research (CACR)Technical Reports, CACR-2010-27. http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-27.pdf

[4] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Ultra Lightweight Cryptography Resource- Constrained Devices", 14th International Conference on Financial Cryptography and Data Security- FC 2010

[5] Ismail San, Nuray At, "Enhanced FPGA Implementation of Hummingbird Cryptographic Algorithm", 14th International Conference on Financial Cryptography and Data Security - FC 2010.

[6] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Ultra Lightweight Cryptography for Low- Cost RFID Tags: Hummingbird Algorithm and Protocol," *Centre for Applied Cryptographic Research (CACR) Technical Reports*, CACR 2009

[7] T.Eisenbarth, S. Kumar, C. Paar, A.Poschman, and L. Uhsadel, " A Survey of Lightweight-Cryptography Implementations", IEEE Design & Test of Computers, vol. 24, no.6,pp.522-533,2007

[8] Xilinx Inc., "Spartan-3 FPGA Family Data Sheet", DS099, December 4, 2009, available at http://www.xilinx.com/support/documentation/datasheet/ds099.pdf