Design of Message Authentication Code with AES and SHA-1 on FPGA

Kuo-Hsien Yeh, Yin-Zhen Liang
Institute of Applied Information, Leader University, Tainan City, 709, Taiwan
E-mail: khyeh@mail.leader.edu.tw
TEL: 886-6-2558291
FAX: 886-6-2550870

Abstract

Combining AES 128-bit and SHA-1, we construct a Message Authentication Code and implement it on Altera FPGA chip. We use the math of finite-field in AES algorithm to reduce the complexity of AES module. Implementation of our architecture needs 17153 logic cell elements on an FPGA chip. The performance achieves 12.4 MHz in frequency. Moreover, the proposed design architecture does not require any memory bits.

Key words：Message Authentication Code (MAC), Advanced Encryption Standard (AES), Secure Hash Algorithm (SHA-1), Field Programmable Gate Array (FPGA)

1. Introduction

Authentication, which certifies data integrity and data origin, is becoming an important technique because the transfer of valuable information needed for electronic funds transfer, business contracts, etc. must be made across computer networks. Data integrity ensures that the data has not been modified or destroyed during transferring. Data origin authentication is the verification that the source of data received is as claimed.

In general, Message Authentication Code (MAC) can be achieved in three ways, Cipher-MAC, Hash-MAC (HMAC) and Hash-Cipher-MACs [1,2]. Cipher-MAC uses a cipher with some encryption techniques to process a message and takes the final result as the corresponding MAC. HMACs are based on cryptographic hash functions. HMACs have two functionally distinct parameters, a message input and a secret key known only to the message originator and intended receiver. Hash-Cipher-MAC combines a hash function and a cipher to construct MAC [1,2]. In this paper, we are interested in the efficient Hash-Cipher-MACs which are secure based on the properties of hash functions $H(\ )$ and block ciphers $E_k(\ )$. $E_k(H(X))$ of Hash-Cipher-MACs is referred to develop a constructing MAC in FPGA based on Rijndael's AES 128-bit of the U.S. National Institute of Standards and Technology (NIST) [3] and SHA-1 of Federal Information Processing Standards Publications 180-1 (FIPS PUB 108-1) of NIST [4].

2. Algorithms

2.1 MAC generation

The generation of our proposed MAC will be computed with a given message M as:

MAC = E_k(*Partial-SHA-1*(*M*)) where *Partial-SHA-1* extracts the left-most 128 bits of SHA-1. In the meantime, the input key is used as the key of AES. Therefore, the output of AES is MAC. The architecture is shown in Figure 1. In verification of MAC, the receiver can verify the MAC when he gets Message *M* and MAC. The receiver decrypts the MAC to get the *Partial-SHA-1* value and checks the correctness by computing *Partial-SHA-1(M)*.

2.2 AES

In AES algorithm [3], the process of encryption consists of the following steps: An initial key addition transformation. The requisite number of rounds, with each round composed of four different transformations, byte substitution, row shifting, column mixing and key addition. A final round is composed of three transformations, byte substitution, row shifting and key addition. Figure 2 shows the AES algorithm encryption structure. The transformations are described in [3].

For avoiding the use of look-up table and the decrease of security, a finite field inverse module is designed. It describes this algorithm based on standard basis for computing multiplicative inverse in GF(2^8) [5,6]. For a finite field GF(2^m) element *A*, the inversion of *A*, A^{-1}, can be calculated by a series of power of *A*.

$$A \bullet A^{-1} = 1 = A^{2^m-1} = A \bullet A^{2^m-2} \quad (1)$$

This implies that the inverse of A can be expressed as

$$A^{-1} = (A^{2^m-1}) \bullet (A^{2^m-2}) \ldots (A^{2^1}) \quad (2)$$

Considering the AES algorithm, A^{254} can be represented by the multiplications of the square property of *A* and A^2 [5,6]. Let us use A^2, the block diagram of A^{-1} circuit is as shown in Figure 3. It has six multipliers; it is clear that it needs a large number of multipliers.

This algorithm can implement the ByteSub transformation in AES easily. The improved structure, which is proposed by [5], uses the square property of A^3 and A^4. A^{254} can be represented as

$$A^{-1} = A^{2^8-2} = A^{254} = ((A^3)^2 \bullet (A^3)^8) \bullet ((A^3) \bullet (A^4))^{32}. \quad (3)$$

The improved inverse circuit can be drawn and shown in Figure 4.

As shown in Figure 4, the number of multiplier in this improved finite field inverse circuit is reduced from 6 multipliers to 3 [5].

The improved finite field inverse circuit in the ByteSub transformation differs from [8]. The method of [8] had to save 256-byte memory and considerable amount of operations.

2.3 SHA-1

For a message of length less than 2^{64} bits, SHA-1 produces a 160-bit condensed representation of the message called a message digest. The message digest is used during generation of a signature for the message [4,7]. There are three execution steps in SHA-1 algorithm [4]: message padding, functions and constants used, and computing the message digest. To process *Mi*, we proceed as follows:

1. Divide M_i into 16 words W_0, W_1, … , W_{15}, where W_0 is the left-most word.
2. For t = 16 to 79 let $W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$. ($S^n$ is shift-left n bits)
3. Let $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$.
4. For t = 0 to 79 do

   $TEMP = S^5(A) + f_t(B,C,D) + E + W_t + K_t$;

   $E = D$; $D = C$; $C = S^{30}(B)$; $B = A$;

A = TEMP;

5. Let $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

Figure 5 shows the integral process of SHA-1.

In general, message schedule W_0, W_1, … , W_{79} is implemented as an array of eighty 32-bit words. However, we reduce the utility rate of the registers, so that we use the alternate method for computing a SHA-1 message digest [4]. It uses an array of sixteen 32-bit words, W_0, W_1, … , W_{15}, and it saves sixty-four 32-bit words of storage registers. They are designed separately in Alera FPGA chip device EP20K600EBC652, and we obtain a result, which the method of sixteen 32-bit words is less than the number of registers of eighty 32-bit words. The method of sixteen 32-bit words is a great register-saver. Table 1 shows the comparison of logic cell elements in Alera FPGA chip device EP20K600EBC652.

3. Design of MAC with AES and SHA-1 on FPGA

Implementation of our architecture needs 17153 logic cell elements and 388 pins on a FPGA chip. The proposed design uses 128 bits I/O and achieves 12.4 MHz in frequency. Moreover, our proposed design does not need memory bits. Table 2 illustrates the results of the integral architecture.

To test and verify our proposed MAC, we make a test to prove it. If the attacker grabs the data and modifies the source data, a receiver will obtain falsified data. We assume that a sender transforms a data, which is “00112233445566778899aabbccddeeff” in hexadecimal format. The MAC generation produces a correct value, which is “4EBC7A40BEBE5F78C91A592C527A4E9F” in hexadecimal format, and a receiver obtains it to decrypt the MAC. Unfortunately, an attacker not only grabs a data in the middle but also modifies it while a sender and a receiver are transforming each other. The receiver will obtain a falsified MAC, which is “4EBC7A40BEBE4078C91A592C527A4E9F” in hexadecimal format, and he decrypts it.

At last the receiver contrasts message digest, and he will detect his MAC, which is falsified. The integral diagram is shown in Figure 6. The result of simulation is shown in Figure 7.

4. Conclusions

Our proposed MAC, which combines AES 128-bit and SHA-1, utilizes the math of finite-field to improve AES algorithm and the alternate method for computing in SHA-1. The multiplication and inverse operations can reduce the complexity of AES module. Moreover, our proposed design does not require any memory bits.

There are two plans in our future work. First, in MAC hardware implementation, we may use full custom design to implement the MAC chip and improve the performance. Second, due to the continuous progress on System On Chip (SOC), MAC, AES, and SHA-1 modules will be applied Intellectual Property (IP).

5. References

[1] Yi-Shiung Yeh and Chan-Chi Wang, “Construct Message Authentication Code with One-Way Hash Functions and Block Ciphers”, IEICE Transactions on Fundamentals of

Electronics, Communications and Computer Sciences, Feb. 1999, pp.390-393.

[2] Ming-Hua Lee, “Construct Message Authentication Code with SHA-1 and AES”, National Chiao Tung University in partial Fulfillment of the Requirements for the Degree of Master, Hsinchu, Taiwan, June 2000.

[3] “Announcing the Advanced Encryption Standard (AES)”, Federal Information Processing Standards Publication 197, November 26, 2001.

[4] “Secure Hash Standard”, Federal Information Processing Standards Publication 180-1, April 17, 1995.

[5] Jeng-Yang Hwang, “The Design, Implementation and Application of Advanced Encryption Standard Algorithm”, I-Shou University in partial Fulfillment of the Requirements for the Degree of Master, Kaohsiung, Taiwan, June 2000.

[6] M.H. Jing, Y.H. Chen, Y.T. Chang, and C.H. Hsu, “The design of a fast inverse module in AES”, Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences, Vol.3, pp.298–303, 2001.

[7] A. J. Menezes, P.C.V. Oorschot, and S.A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1997.

[8] 王立洋, “CPLD Implementation of RijndaelCipher”, http://www.ccisa.org.tw/文章收錄/文件 /AES/CPLD%20Implementation%20of%20Rijndael%20Cipher王.ppt, November, 2000.

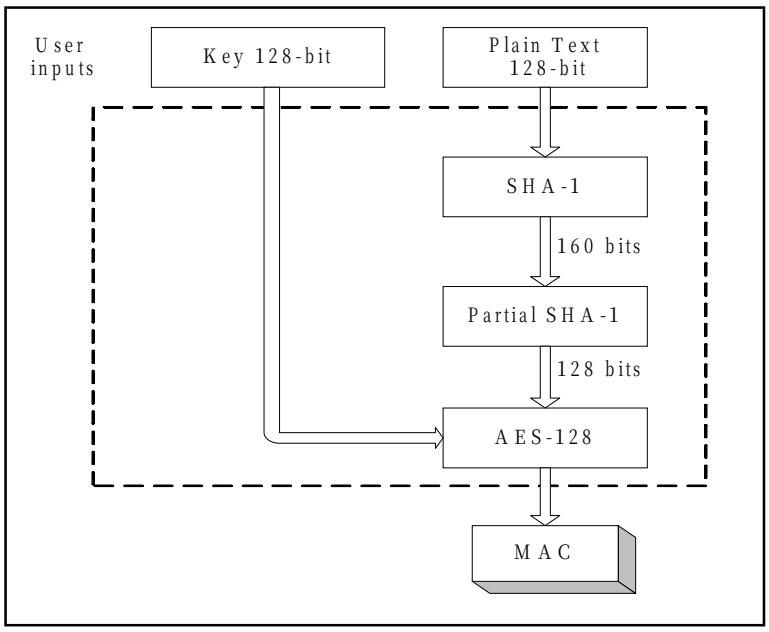

Figure 1. The integral architecture of MAC

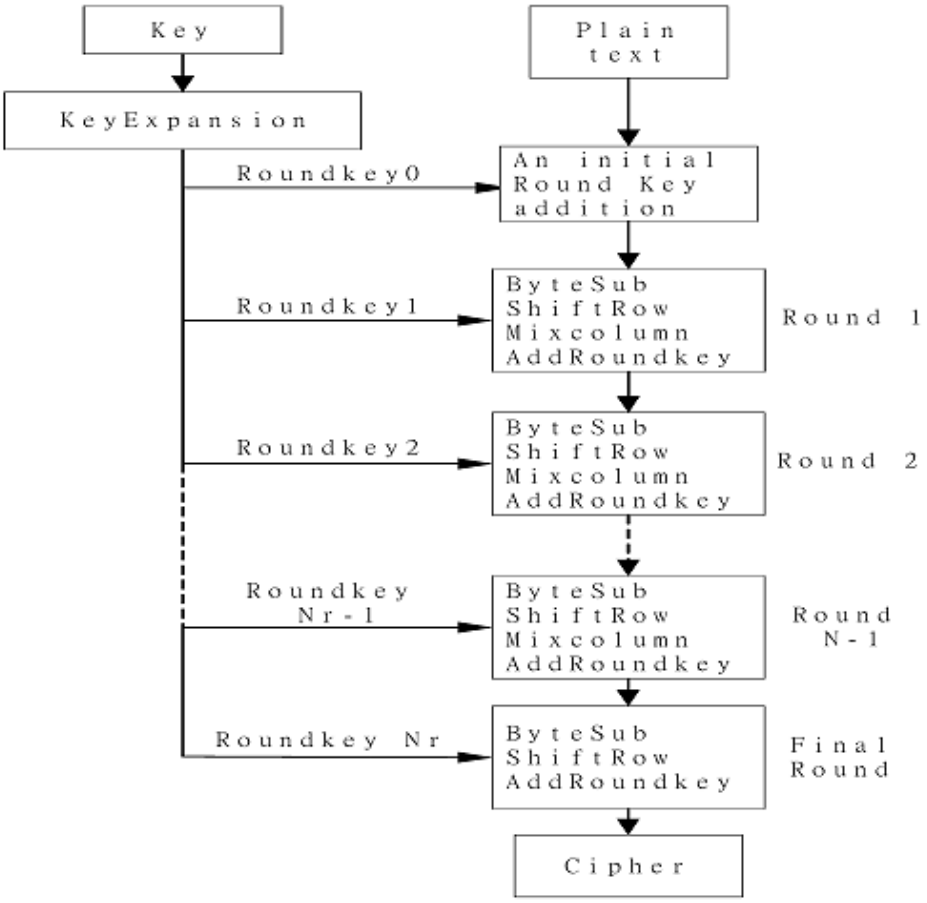

Figure 2. The AES algorithm encryption structure

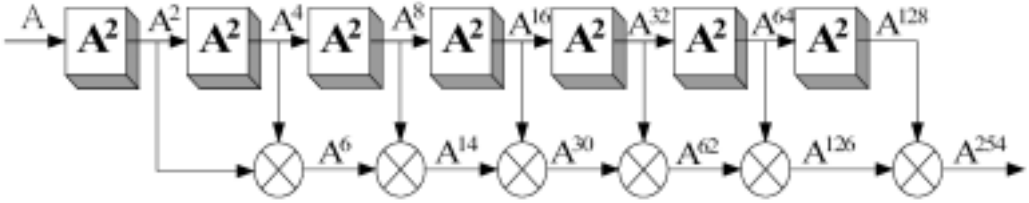

Figure 3. The finite field inverse circuit using square property

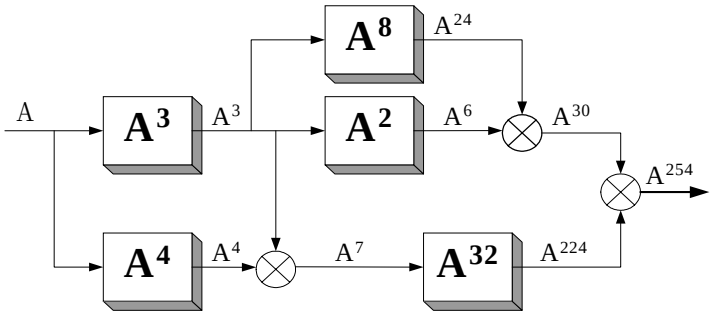

Figure 4. The improved finite field inverse circuit

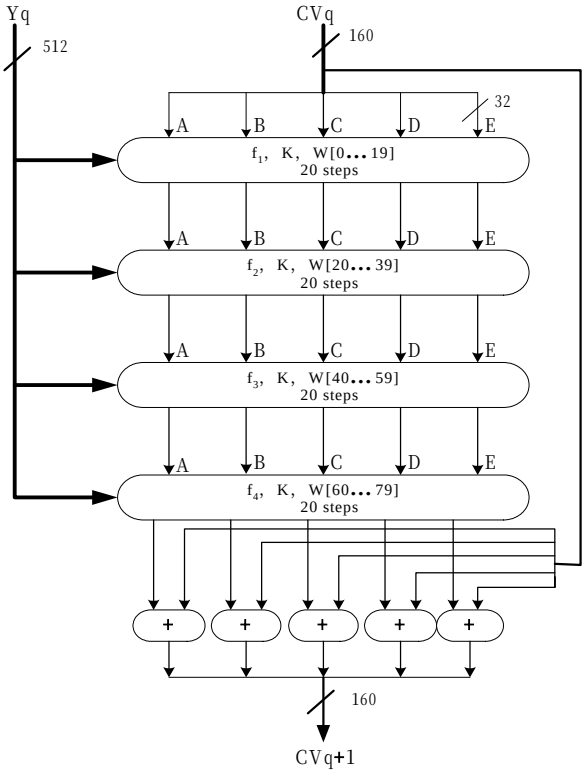

Figure 5. The integral process of SHA-1

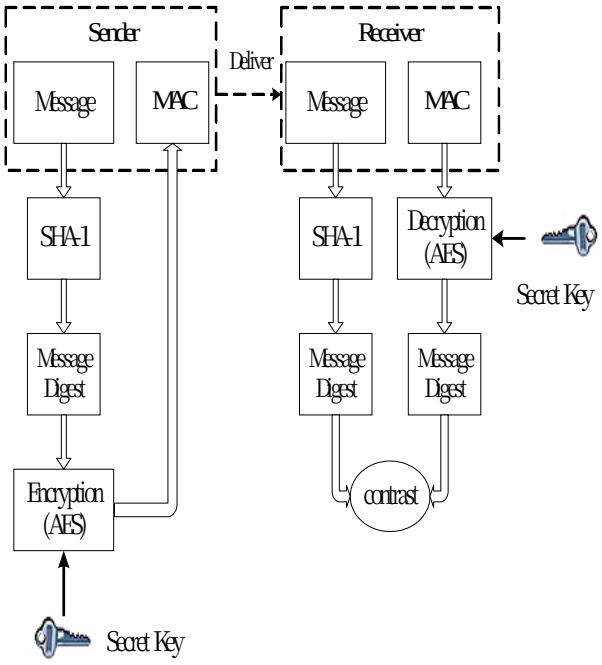

Figure 6. The integral diagram

Figure 7. The result of simulation of falsified MAC

Table 1. The comparison of logic cell elements

	Eighty 32-bit words	Sixteen 32-bit words
Logic Cell Elements	5198	2291

Table 2. The results of the integral architecture

	MAC	AES 128-bit Encryption/Decryption	SHA-1
I/O pins	388	(cell)	(cell)
Memory (bits)	0	0	0
Logic Cell Elements	17153	14444	2291
Frequency	12.4 MHz	13.4 MHz	33.8 MHz
FPGA Chip	EP20K600EBC652	EP20K600EBC652	EP20K600EBC652