SEED 암호화 알고리즘의 설계

권 명 진, 김 도 완, 이 종 화, 조 상 복
울산대학교 전기전자정보시스템 공학과
전화 : 052-259-1629

A Design of SEED Cipher Algorithm

Myoung-Jin Kwon, Do-Wan Kim, Jong-Hwa Lee ,Sang-Bock Cho
School of Electrical Engineering, University of Ulsan
E-mail : old-fox76@hanmail.net

Abstract

A cipher algorithm with a 128-bit symmetric key block, SEED developed by KISA [1] (Korea Information Security Agency) is designed by using VHDL to implement hardware architecture. It has been adopted by most of the security systems in Korea. SEED is designed to utilize the S-boxes and permutations that balance with the current computing technology. It has the Feistel structure with 16 rounds. The same procedure for data encryption and decryption makes possible an easy and practical hardware implementation. The primary functions used in SEED are F function and G function.

This paper proposes an iterative architecture of F function, a modified architecture of G function and an iterative architecture of key scheduling algorithm. The designed SEED encrypts and decrypts exactly the test vectors. It is expected to extend to various application fields if the design of control blocks is added.

본 논문은 반도체 설계 교육센터(IDEC)의 지원에 의해 이루어졌습니다.

1. 서 론

암호화알고리즘은 암·복호화에 사용되는 키의 특성에 따라 암·복호화 키가 같은 대칭키(비밀키) 암호화 방식과 암·복호화 키가 서로 다른 비대칭키(공개키) 암호화방식으로 크게 구분할 수 있다. 비대칭키 암호화방식은 소수를 해로 가지는 큰 수의 인수분해 어려움에 근거를 두고 만들어진 암호화 방식으로 암·복호화 방식이다[2]. 대칭키 암호화방식에 비해 키 관리 차원에서 문제점은 없어졌지만, 복잡한 수학 공식으로 인해 처리속도가 느리고 하드웨어 구현이 어렵다. 그래서 주로 전자서명 등과 같은 저용량 데이터의 암호화에 사용된다. 반면 대칭키 암호화방식은 암·복호화시에 같은 키를 사용하기 때문에 키 관리 차원에서 주의를 요해야 한다는 단점이 있지만 하드웨어 구현이 용이하며 처리속도가 빠르기 때문에 많은 양의 데이터 암호화시에 효과적으로 사용된다[3]. 대칭키 암호알고리즘은 데이터 처리 형식에 따라 스트림 암호알고리즘과 블록 암호알고리즘으로 나눌 수 있다. SEED는 국내 표준으로 제정된 128비트 대칭키 암호화알고리즘으로, 블록 단위로 데이터를 처리하는 블록 암호화알고리즘이다. 블록 암호화알고리즘은 대부분 Feistel 구조로 설계된다. Feistel 구조란 각각 t비트인 L_0, R_0, 블록으로 이루어진 2t비트 평문 블록(L_0, R_0)이 r라운드(r≥1)를 거쳐 암호문(L_r, R_r)으로 변환되는 반복 구조를 말한다. 이러한 Feistel 구조는 라운드 함수에 관계없이 역변환이 가능하며(즉, 암·복호화 과정이 같음), 두 번의 수행으로 블록간의 완전한 diffusion이 이루어지며, 알고리즘의 수행속도가 빠르고, H/W 및 S/W로 구현이 용이하고, 아직 구조상의 문제점이 발견되고 있지 않다는 장점을 지니고 있다.

본 논문에서는 SEED 알고리즘을 VHDL로 설계하고 Simulation해 봄으로써 실제 칩으로 제작되기 전에 알고리즘의 신뢰성 및 동작특성을 확인하고자 한다.

2. SEED 블록 암호화 알고리즘

SEED 알고리즘의 전체 구조는 Feistel 구조로 이루어져 있으며, 128비트의 평문 블록단위당 128비트 키로부터 생성된 16개의 64비트 라운드 키를 입력으로 사용하여 총 16라운드를 거쳐 128비트 암호문을 출력한다.

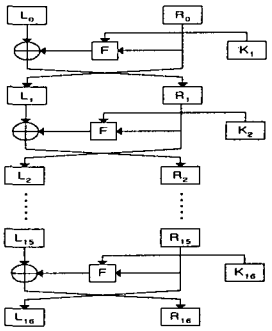

그림 4. SEED 구조도

그림 1은 SEED 전체 구조도를 나타내고 있는데, 128비트 평문블록을 2개의 64비트 블록(L_0(64), R_0(64))으로 나누어, 16개의 64비트 라운드키를 이용하여 16라운드를 수행한 후, 최종 128비트 암호문 블록(L_{16}(64), R16(64))을 출력한다. SEED 알고리즘은 크게 F함수, G함수, S-Box, 키 생성 블록으로 이루어져 있다. S-Box는 입력 데이터를 비선형 변화시켜 전체 알고리즘의 안전성을 보장하며, G함수는 S-Box의 출력 값들을 충분히 섞어주는 역할을 하며, F함수는 키 생성 블록에서 만들어진 라운드 키와 G함수의 출력 값을 입력으로 받아들여 전체 알고리즘의 특성을 결정지어준다[1].

블록암호화 방식에는 크게 ECB(Electronic Code Book)방식과 CBC(Cipher Block Cha- nining)방식이 있으며, 본 논문에서는 ECB방식을 채택하였다. 이는 CBC방식에 비해 하드웨어 구현이 용이하다는 장점이 있다.

3. F 함수 설계

Feistel 구조의 블록암호알고리즘은 F 함수의 특성에 따라 구분할 수 있다. SEED의 F함수는 수정된 64비트 Feistel 암호알고리즘으로 구성된다. F함수는 각 32비트 2개의 블록(C, D)을 입력으로 받아, 32비트 2개의 블록(C′, D′)을 출력한다. 즉, 암호화 과정에서 64비트(C, D)와 64비트 키 Ki(Ki,0;Ki,1)를 F함수의 입력으로 처리하여 64비트(C′, D′)을 출력한다[1]. 그림 2는 F함수의 구조를 나타내고 있다. F함수에는 32비트 XOR 3개, 32비트 가산기 3개 그리고 G함수 3개가 사용되었다. 3개의32비트 XOR 부분을 하나의 블록으로 설계하였으며[5], 32비트 가산기는 G함

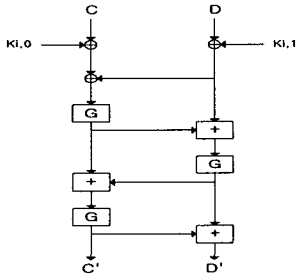

그림 5. F 함수 구조도

수에서와 같이 32비트 CLA(Carry-Look Ahead) 가산기를 설계하여 사용하였다.

4. G 함수 설계

G함수는 매우 좋은 특성을 갖는 두 개의 8비트 S-Box를 이용하여 입력의 각 바이트를 비선형 변환 후, 그 결과 32비트를 적절한 permutation을 통해 충분히 섞어 준다. 이는 결과적으로 F함수의 안전도를 높여 전체 SEED 구조의 안전도를 높이게 된다.

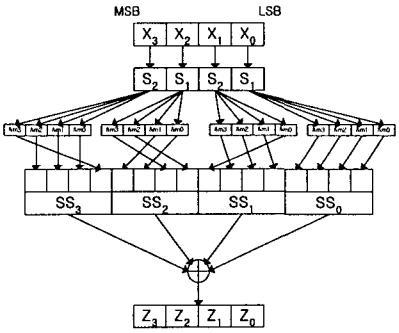

그림 6. G 함수 구조도

G 함수의 구조도는 그림 3과 같으며, 이를 식으로 나타내면 다음과 같다.

$$y_0 = S_1(x_0), y_1 = S_2(x_1), y_2 = S_1(x_2), y_3 = S_2(x_3)$$
$$z_0 = (y_0 \& m_0) \oplus (y_1 \& m_1) \oplus (y_2 \& m_2) \oplus (y_3 \& m_3)$$
$$z_1 = (y_0 \& m_1) \oplus (y_1 \& m_2) \oplus (y_2 \& m_3) \oplus (y_3 \& m_0)$$
$$z_2 = (y_0 \& m_2) \oplus (y_1 \& m_3) \oplus (y_2 \& m_0) \oplus (y_3 \& m_1)$$
$$z_3 = (y_0 \& m_3) \oplus (y_1 \& m_0) \oplus (y_2 \& m_1) \oplus (y_3 \& m_2)$$

여기서 m_i(masking byte)의 값은 m_0= 0x 03, m_1=0x0c, m_2=0x30, m_3=0xc0로 정의 된다[1].

그림 3과 같이 G 함수는 4개의 확장된 4바이트 SS-Box들의 xor로 구현할 수 있다. 본 논문에서 G 함수를 설계하는데 있어 단순히 S-Box1, S-Box2의 값을 사용하지 않고 그림 4와 같이 S-Box의 출력 값과 masking byte와의 bit-wise 연산을 모두 마친 결과 값을 ROM에 저장하여 최종적으로 SS-Box의 값을 산출하였다. 결과적으로 8개의 ROM Table이 사용되었지만, 설계 시 불필요한 연산과정을 줄이고 빠른 처리속도를 가져올 수 있었다.

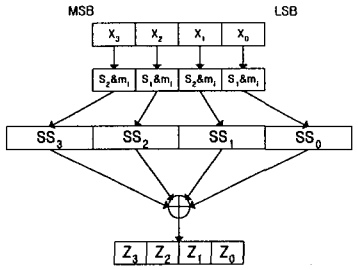

그림 7. 변형된 G 함수

4. 키 생성 블록 설계

SEED의 키 생성 알고리즘은 그림 5와 같다. 128비트의 암호키를 64비트씩 좌우로 나누어 이들을 교대로 8비트씩 좌/우로 회전이동 후, 결과의 4워드들에 대한 간단한 산술연산과 G 함수를 적용하여 라운드 키를 생성한다. 라운드 상수 KC_i는 황금비의 소수 부분으로 다음과 같이 생성한다[1].

$$KC_1 = int(\frac{\sqrt{5}-1}{2} \times 2^{32}) = 0x9e3779b9$$

$$KC_i = KC_{i-1} \lll 1 \; (2 \le i \le 16)$$

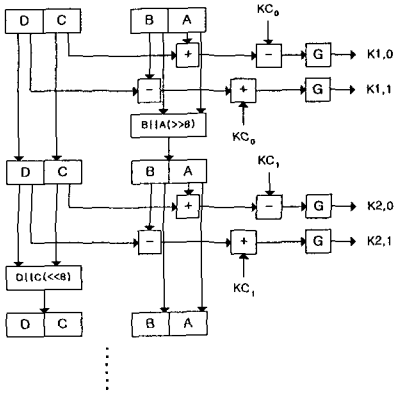

그림 8. 키 생성 블록

라운드 상수 값은 일단 KC_i값이 정해지면 나머지 라운드의 상수 값은 KC_i을 1비트 left shift에 의해서 구할 수 있으므로 실제적으로는 1비트 left shifter를 사용해서 초기 값만 입력하여 각의 라운드 상수 값을 입력하였다. 키 생성 블록은 병렬구조를 적용하여 입력 키 값에 대해 한번에 라운드 키가 구해지도록 설계하였다. 키 생성 블록에서의 주요 연산은 32비트 가·감산 연산이다. 32비트 가산기는 처리 속도면에 장점을 가지고 있는 32비트 CLA (Carry Look Ahead) 덧셈기를 사용하였으며, 감산기는 CLA를 사용하여 2의 보수연산 방법에 의해 설계되었다[4].

5. S-Box 설계

전단사함수의 지수승을 구하는 과정에서 DC, LC 특성이 가장 우수한 두 개의 지수를 이용해 지수함수를 선행변환 함으로써 얻어진 두 개의 행렬 값이다. S-Box는 S-Box1과 S-Box2 두 개로 이루어지며 그림 6과 같이 8비트 입력(0~255)을 받아 8비트 출력(0~255)을 내보낸다[1].

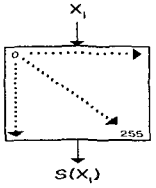

그림 9. S-Box

S-Box는 입력 데이터를 선형변환 시키기 위해 그림 3에서처럼 G 함수에서 사용된다. 앞에서 언급했듯이 G 함수에 사용된 S-Box는 S-Box의 출력 값과 m_i(masking byte)와의 bit-wise AND 연산결과를 미리 연산해서 저장한 확장된 형태의 S-Box를 설계하여 사용했으며, 수식으로 나타내면 다음과 같다.

$$SS_3 = S_2(X_3)\&m_2 \| S_2(X_3)\&m_1 \| S_2(X_3)\&m_0 \| S_2(X_3)\&m_3$$
$$SS_2 = S_1(X_2)\&m_1 \| S_1(X_2)\&m_0 \| S_1(X_2)\&m_3 \| S_1(X_2)\&m_2$$
$$SS_1 = S_2(X_1)\&m_0 \| S_2(X_1)\&m_3 \| S_2(X_1)\&m_2 \| S_2(X_1)\&m_1$$
$$SS_0 = S_1(X_0)\&m_3 \| S_1(X_0)\&m_2 \| S_1(X_0)\&m_1 \| S_1(X_0)\&m_0$$

6. 전체 구조 설계

그림 1에 나타나 있듯이 SEED는 Feistel구조로 16라운드를 반복적으로 수행한다. 본 논문에서는 전체 구조를 16 round flat 방식으로 16 round를 각각 별도로 하나씩 구현하여 데이터 처리속도를 향상시킨 ECB 방식을 채택하였다. ECB 방식은 블록 암호화 알고리즘의 기본 동작이며, 하드웨어 구현이 용이하고 병렬 구조가 가능하다.

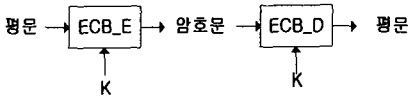

그림 8. ECB 동작 모드

그림 8은 간단한 ECB 동작모드를 나타내고 있으며 암호화 과정의 흐름도는 그림 9과 같다. 그림 10은 암호화 과정의 전체 구조도를 블록 별로 나타내고 있다[6].

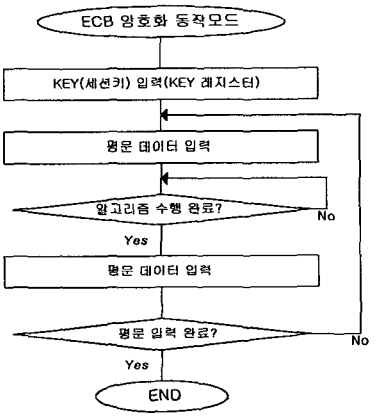

그림 11. ECB 동작모드의 흐름도

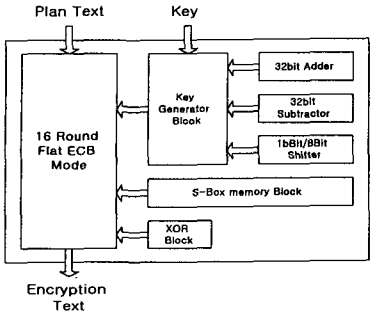

그림 12. SEED Block Diagram

7. Simulation 결과

VHDL Compile과 Simulation은 ModelSim을 통해 이루어 졌으며, test-bench는 KISA에서 주어진 Test Vectors를 참고로 하여 의해 만들어 졌다. 16라운드 수행 결과는 다음과 같다.

평문 입력 데이터를 "010203040506070809 0A0B0C0D0E0F"로 하고 Key 입력 값은 "0 00000000000000000000000000000000"이다. 암호화 출력 값 "213FE169FF413714599D2 4D4936328B0"를 복호화 했을 때, 그림 10 및 11과 같이 입력 데이터가 정확하게 다시 회복되는 것을 볼 수 있다.

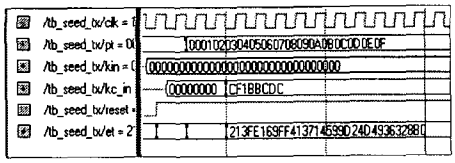

그림 13. 16 라운드 암호화 Simulation

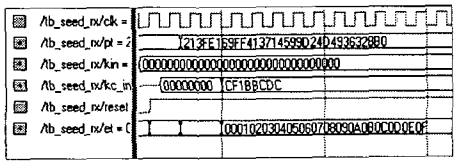

그림 14. 16 라운드 복호화 Simulation

8. 결론

주어진 알고리즘에 의해 128비트 블록 암호화 방식인 SEED를 설계하였다. 전체 수행 시간은 468 ns가 걸렸으며, 암호화 처리속 는 20 Mbps가 나왔다. Simulation 결과에서 알 수 있듯이 SEED 알고리즘을 제안된 방식에 따라 VHDL을 통해 설계하였을 때, Test Vectors와 정확히 일치하는 결과를 출력하였으며, 하드웨어로의 설계가 가능하다는 것을 알 수 있었다.

설계된 SEED 알고리즘이 효과적으로 적용되기 위해서는 입력데이터를 블록단위로 끊어주고 그에 따른 키 값을 제어해줄 수 있는 제어블록의 설계가 병행되어야 그 활용분야를 넓힐 수 있고, 보다 안정적으로 동작할 것이다.

9. 참고문헌

[1] "A Design and Analysis of SEED", Korea Information Security Association, DEC. 1998.

[2] Bruce Schneier, "Applied Cryptography", John Wiley & Sons, Inc., pp.461-474.

[3] Alfred J.Menezes, Paul C.van Oorchot, Scott A.Vanstone, "Handbook of App-lied Cryptography", CRC Press, pp.15- 21.

[4] 김영철, 정연모, 조중휘, 홍윤식, "디지털 시스템 설계를 위한 VHDL", 홍릉과학출판사, pp.236-249

[5] Jong-Ho Shin, Jun-Woo Kang, "ASEED Encryption Chip Design", IDEC Conference 2000, pp.105-108.

[6] A.Schubert, W.Anheier, "Efficient VL-SI Implementation of Modern Symmetric Block Ciphers", IEEE, pp.757-760