352-비트 암호 알고리즘의 하드웨어 설계*

박 영 호**

Hardware Design of 352-bit Cipher Algorithm

Park, Young Ho

〈Abstract〉

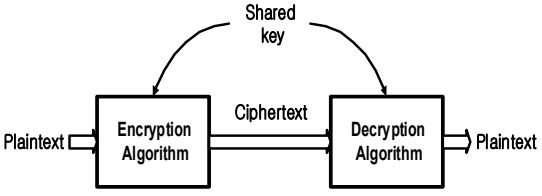
Conventional DES has been not only shown to have a vulnerable drawback to attack method called 'Meet in the Middle', but also to be hard to use that it is because software implementation has a number of problem in real time processing. This paper describes the design and implementation of the expanded DES algorithm using VHDL for resolving the above problems. The main reason for hardware design of an encryption algorithm is to ensure a security against cryptographic attack because there is no physical protection for the algorithm written in software. Total key length of 352 bits is used for the proposed DES. The result of simulation shows that the inputted plaintext in cryptosystem are equal to the outputted that in decryptosystem.

Key Words : VHDL, Meet in the Middle, DES Algorithm, Round Architecture

Ⅰ. 서론

암호시스템은 암호화 키(key)와 복호화 키가 동일한가의 여부에 따라 대칭형 암호시스템과 비대칭형 암호시스템으로 분류할 수 있다[1].

대칭형 암호시스템은 암호화 키와 복호화 키가 동일한 암호시스템을 말하고(예 : DES), 비대칭형 암호시스템은 암호화키와 복호화 키가 동일하지 않은 암호시스템을 말한다(예 : RSA)[2]). 암호시스템의 개념을 블록도로 표현하면 <그림 1>과 같다. <그림 1>의 암호시스템에서는 어떤 평문(Plaintext)을 키와 함께 암호기(Encryption Algorithm)에 입력하면 암호문(Ciphertext)이 생성된다. 이 암호문을 암호기에 사용했던 동일한 키와 함께 복호기(Decryption Algorithm)에 입력하면 원래의 평문이 생성된다.



<그림 1> 암호시스템

현재 가장 보편적으로 실용화되어 사용되는 암호 알

* 본 논문은 2008년 부천대학 학술연구비 지원에 의해 연구되었음.
** 부천대학 e-비즈니스과

고리듬은 IBM의 Lucifer 알고리듬을 기반으로 개발한 DES(Data Encryption Standard)이다[3,4]. 그러나 DES는 미국 상무성 표준으로 채택되어 사용된 이후 지속적으로 키 길이가 짧은 것이 문제점으로 지적되어 왔다.

1990년 Biham과 Shamir에 의해 평문의 입력과 암호문의 차이를 분석한 Differential Cryptanalysis 공격방법을 변형된 DES에 사용하는데 대한 위험성이 입증되었고[5], 1993년 Matsui는 선형해독법을 발표하여 56 비트의 DES 알고리듬이 기지의 평문과 암호문 쌍을 이용한 공격방법에 취약성을 보이고 있다[6].

이 공격방법은 키의 전수 검사보다 빠른 DES 공격방법으로 주목받고 있다. 그러나 키 길이가 짧아서 발생하는 취약성 이외에는 DES가 발표 된 지 20여 년이 지난 오늘까지 큰 문제점은 알려지지 않고 있다. 2중 DES는 'Meet-in-the-Middle' 공격에 취약하며[7], 3중 DES는 키 길이를 효과적으로 확장할 수 있으나 DES를 3회 반복하는데 필요한 3배의 속도를 수행해야하는 어려움이 있다[8].

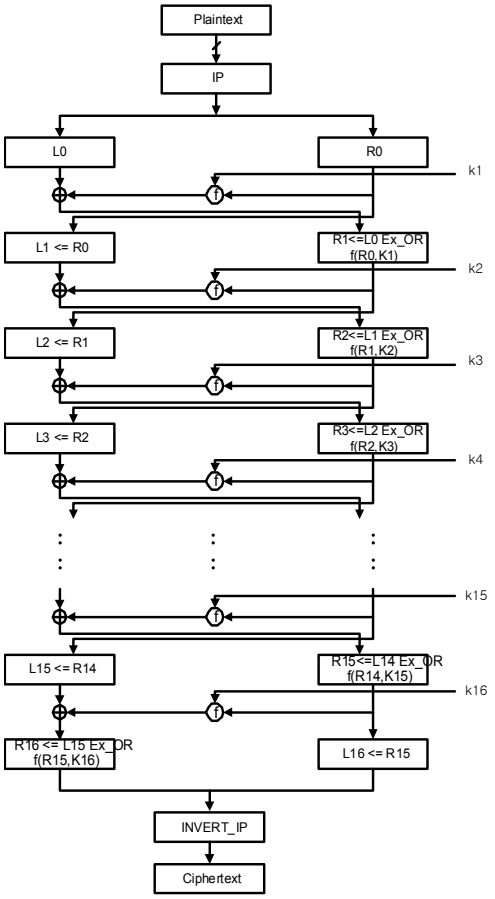
2장에서는 기존의 DES 암호 알고리즘에 대한 개념을 살펴보고, 3장에서는 본 논문에서 제안한 DES 알고리듬의 각 블록의 모델링 및 구현에 대해 설명한다. 4장에서는 제안한 DES를 시뮬레이션을 통해 암호시의 평문과 복호화시의 생성된 평문이 동일함을 보이고, 마지막으로 5장에서 결론을 맺는다.

II. 관련 연구

가장 대표적인 블록 암호화 알고리즘으로 알려진 DES는 64 비트의 데이터와 56 비트 길이의 키를 사용하여 64 비트의 암호화 결과를 생성한다.

DES는 암호화, 복호화 알고리즘이 대칭적이며 치환과 대치, 그리고 S_box로 구성된 블록 암호화 시스템이다.

DES 암호화에 대한 전체적인 구성은 <그림 2>와 같다. DES의 암호화 과정에는 암호화하고자 하는 평문과



<그림 2> DES의 암호화 과정

키(k_1 ... k_{16})의 두 가지 입력이 들어간다. 64 비트의 평문 블록은 초기치환(initial permutation : IP)후에 32 비트씩 좌(L_0), 우(R_0)부분으로 나뉘게 되며 그 다음 16라운드의 계산을 거치게 된다.

16라운드 후에는 오른쪽(R_{16})과 왼쪽(L_{16}) 부분이 합쳐져서 역 초기 치환(inverse initial permutation : IP^{-1} : INVERT_IP)을 거침으로써 암호문(Ciphertext)이 생성된다.

본질적으로 대치된 64 비트 입력은 16라운드를 거치며 매 라운드의 결과로 64 비트의 중간 값을 생성한다. 각 64 비트 중간 값의 좌우 절반은 분리된 32 비트 값으로 취급되며 L(왼쪽)과 R(오른쪽)로 분류된다. 각 라운드

의 전체적인 처리는 다음 공식으로 요약된다.

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$

여기서 ⊕은 비트 단위 XOR함수를 의미한다.

따라서 각 라운드의 좌측 출력(L_i)은 단순히 그 라운드의 우측 입력(R_{i-1})과 같다. 우측 출력(R_i)은 L_{i-1}을 R_{i-1}과 k_i의 복합함수 ⓕ에 XOR한 결과이다.

DES에서 복호화 과정은 암호화 알고리즘과 비슷하기 때문에 구현이 간단하다. 차이가 있다면 단지 반대의 순서로 계산을 행한다는 것뿐이다. 이것은 암호화를 위한 각 라운드의 키가 K_1, K_2, K_3, ... K_{16}이라면 복호화를 위한 키는 K_{16}, K_{15}, K_{14}, ... K_1이 되는 것이다.

기존의 DES 알고리즘은 위와 같은 과정으로 소프트웨어로 구현되어 하드웨어보다 실행시간이 늦다. 본 논문에서는 VHDL을 이용하여 하드웨어로 설계한다.

Ⅲ. 제안된 352 비트의 DES 알고리즘 설계

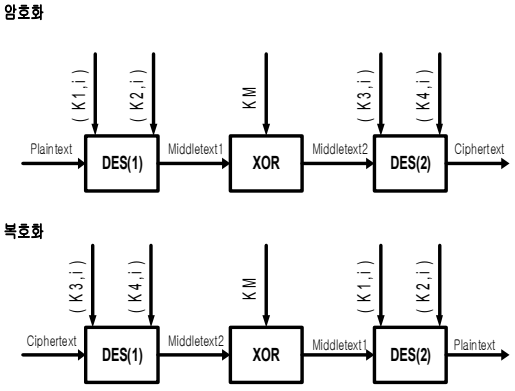
3.1. 352 비트 알고리즘 설계

제안한 352비트 DES 알고리즘은 다음과 같은 설계조건을 정한다.

(조건1) 암·복호 처리시 수행되는 모든 박스(box)중에서 S_box를 제외한 나머지 박스를 만들지 않고 출력 비트와 입력 비트를 직접 연결하였다. 이는 암·복호 처리시 기억장소를 줄임으로써 게이트(gate) 수를 강조하고 이에 처리 속도를 향상시키고자 함이다.

(조건2) 192 비트의 키(key)길이를 352 비트 길이로 확장하였다. 이는 암호 비도를 증가시키기 위함이다.

(조건3) 입력 데이터의 한 블록단위를 64 비트에서 96 비트로 읽어서 3개의서브 블록(L, M, R)이 16라운드를 반복 수행하도록 하였다.



<그림 3> 제안된 352비트 암호시스템

<그림 3>은 상기 조건하에 설계된 352비트 알고리즘 암호시스템의 블록도이다. 352비트 알고리즘은 평문 96 비트와 킷값 128 × 2($k_{1,i}$ $k_{2,i}$, $k_{3,i}$ $k_{4,i}$) + KM(96 비트) = 352 비트가 이용된다. 킷값은 $k_{1,i}$, $k_{2,i}$에 각각 64 비트씩 128 비트가 적용되고 KM에는 96 비트, $k_{3,i}$, $k_{4,i}$에 각각 64 비트씩 128 비트가 적용된다. 여기서 $k_{1,i}$, $k_{2,i}$와 $k_{3,i}$, $k_{4,i}$에는 패리티 비트를 제외한 56 비트씩 112 비트가 적용된다. KM은 패리티 비트를 제외하지 않고 96 비트 모두를 적용하는데 이는 DES(1)에서 생성된 중간문(Middletext1) 96 비트와 XOR 연산을 하기 위함이다. <그림 3>의 동작은 아래와 같다.

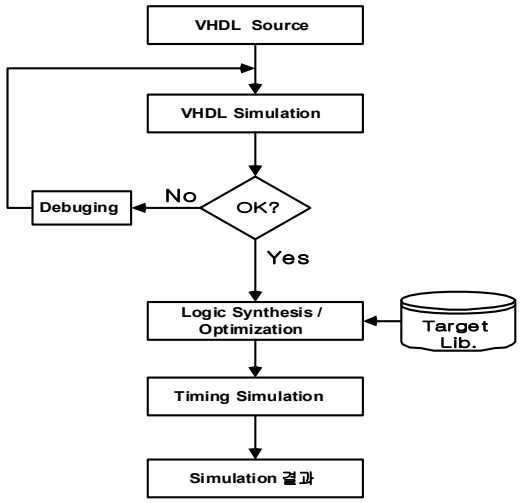
암호화시에 어떤 평문 96 비트와 킷값 $k_{1,i}$(56 비트), $k_{2,i}$(56 비트) 112 비트가 입력되면 첫 번째 암호화(DES(1))를 수행하여 Middletext1을 생성하고, 중간문과 킷값 KM(96 비트)을 XOR 연산 후 Middletext2를 생성한다.

다시 Middletext2(96 비트)와 킷값 $k_{3,i}$(56 비트), $k_{4,i}$(56 비트) 112 비트가 입력되면 두 번째 암호화(DES(2))를 수행하여 최종 암호문을 생성한다.

복호화시에는 암호화 N정과 비슷하나 킷값($k_{1,i}$ $k_{2,i}$, $k_{3,i}$ $k_{4,i}$)을 역순으로 수행하면 평문을 생성한다. 하드웨어로 구현하면 기존 DES의 64 비트의 데이터 처리보다 96 비트의 데이터를 처리하게 되므로 훨씬 수행속도가 빠르므로 실시간처리 환경에 매우 적합하다. 또한 352 비트

의 키 길이로 암호화되므로 암호 비도를 증가시킬 수 있다.

3.3 VHDL 모델링



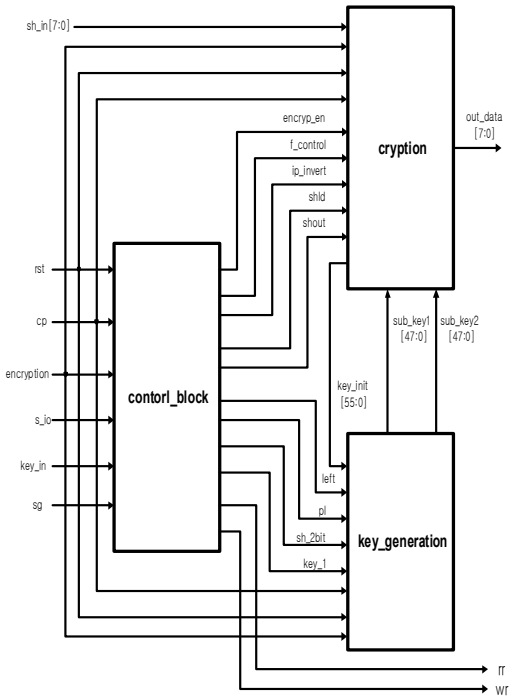
<그림 4> 설계 흐름도

<그림 4>는 352비트 DES의 설계 및 구현을 위한 설계 흐름도를 나타낸다.

352비트의 DES 알고리즘은 VHDL과 자동합성기법을 기본으로 탑다운(top-down)설계방식에 따라서 설계가 이루어진다.

352비트 알고리즘의 하드웨어 설계를 기능별로 나누면 암·복호처리부, 키 생성부, 그리고 제어부로 구성된다. 암·복호처리부는 암호화하고자 하는 블록 데이터 96 비트를 16라운드로 첫 번째 수행 (DES(1))후, 여기서 생성된 중간문(Middletext1)를 중간 킷값(KM) 96 비트와 XOR 연산을 거쳐 생성된(Middletext2)것과 다시 초기치환을 거쳐 16라운드를 수행(DES(2))하여 최종 암호문(Ciphertext)을 생성한다. 제안된 알고리즘에서는 입력되는 총 킷값은 352 비트이다. 이 킷값 중에 중간 키(KM : 96 비트)는 암·복호처리부에서 수행된다.

키 생성부는 256 비트를 키로 입력받아 첫 번째 (DES(1)) 암·복호 수행시 128 비트를 적용하며, 두 번째 (DES(2)) 암·복호 수행시 나머지 128 비트를 킷값으로 수행하도록 하였다. 제어부는 암·복호처리부(DES(1), MID_XOR, DES(2))와 키 생성부의 두 번($k_{1,i}$ $k_{2,i}$, $k_{3,i}$ $k_{4,i}$)의 동작을 위한 제어신호를 생성하는 곳으로 외부 입출력에 관계한 레지스터 조작과 데이터 흐름에 대한 제어코드를 생성하도록 한다.



<그림 5> 암·복호기의 전체 블록도

<그림 5>는 암·복호기의 전체 블록도를 도시하였다. VHDL로 구현한 352비트 알고리즘은 크게 세 개의 블록(block)으로 구성되며 데이터 입출력과 암·복호화 처리를 수행한다.

3.3.1 암·복호처리부

control block에서 보내는 신호에 따라 데이터를 8비트씩 입출력하는 과정의 수행 및 암·복호화 처리를 수행하게 된다.

데이터 입출력은 킷값과 평문을 포함하여 데이터의 입력과 출력이 interleave하게 이루어진다. 암·복호화 수행시에는 key_generation으로부터 352비트 알고리즘의 각 round에 대한 킷값을 입력받아 동작하게 된다.

3.3.2 키 생성부

각 round에 필요한 킷값을 생성하는 부분으로 제어신호에 따라 암호화와 복호화 사이의 key shift operation을 수행하게 된다.

3.3.3 제어부

이 블록의 역할은 데이터 입출력, 암·복호화 처리에 관한 제어신호를 생성하여 cryption(암·복호처리부)과 key_generation(키 생성부)에 제공하는 역할을 수행한다. 데이터 입력시 외부로부터 s_io, key_in, sg 신호를 받아 cryption과 key_generation block에 제어신호를 제공하고, 외부의 처리결과에 대한 신호를 보낸다. 또 암·복호화 수행시에는 외부와 단절된 상태에서 352비트 알고리즘의 암·복호화 수행과정인 ip, ronud, ip_invert, key_xor 연산, 키 생성 처리에 관한 신호를 생성하여 cryption block, key_generation block으로 보낸다.

cryption block에서는 ip, round, ip_invert, key_xor 연산 처리를 수행하면, key_generation에서는 주 round에서 필요한 킷값을 생성하게 된다.

3.4 암 · 복호처리부(cryption)

제안한 352비트 알고리즘의 암·복호처리부에서는 ip_box, f_function(F 함수부), 그리고 ip_invert, key_xor 연산부로 구성되어 있다. sh_in(7:0)에 의해 암·복호가 수행될 데이터와 암·복호화 수행시 필요한 킷값을 입력하며, 입력된 값은 암호부 레지스터(sh_reg)에 임시 저장된다.

세 개의 키($k_{1,i}$ $k_{2,i}$, KM, $k_{3,i}$ $k_{4,i}$)중 첫 번째 키($k_{1,i}$ $k_{2,i}$)와 세 번째 키($k_{3,i}$ $k_{4,i}$)값은 8 비트씩 입력되어 제어신호에 의해 8번 반복(64 비트) 되면 킷값으로 ep_box(패리티 비트 제거)를 거쳐 키 생성부로 전달된다.

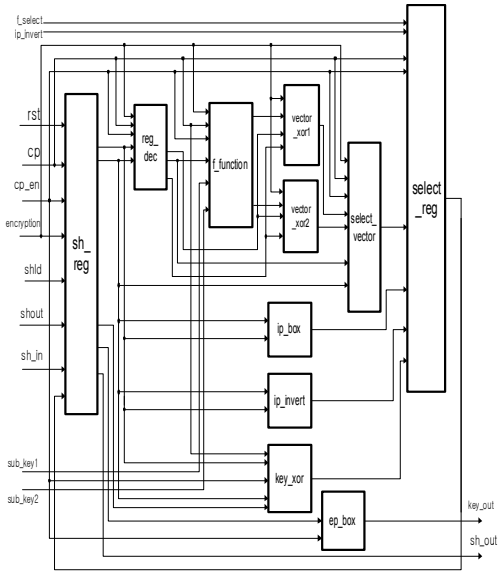
이를 4번 반복하여 224비트를 키 생성부에 전달되고 중간 킷값(KM)은 8 비트씩 12번 반복(96 비트)되어 내부 블록인 key_xor block의 레지스터에 저장된다. 킷값이 모두 키 생성부와 key_xor block에 전달된 후 8비트를 제어신호에 의해 12번 반복(96비트) 되면 데이터 값으로 임시 레지스터(sh_reg)에 저장된다.

Ip_box에 의해 초기치환을 수행한 후 암호부 레지스터에 저장된다. 암호화 수행에 있어 암호부 레지스터의 출력이 세부분(reg_l[31:0], reg_m[31:0], reg_r[31:0])으로 나뉘게 된다. 암호부 레지스터 출력 중 한 부분(reg_m[31:0])은 F 함수부에서 확장치환(E_box)을 거친 후 키 생성부로부터 입력받은 서브키(sub_key1[47:0], sub_key2[47:0])와 XOR 연산된 후 S_box를 거친 후, 연산된 값을 비선형치환(p_box)를 거쳐 각각의 출력(f_out1[31:0], f_out2[31:0])을 내보낸다. 동시에 다음 라운드의 암·복호화 수행을 위해 암호부 레지스터에 임시로 저장된다.

암호부 레지스터 출력 중 나머지 두 부분(reg_l[31:0], reg_r[31:0])은 F 함수부의 출력(f_out1[31:0], f_out2[31:0])과 XOR 연산되며, 다음 라운드의 암·복호 수행을 위해 암호부 레지스터에 임시로 저장된다. 352비트 알고리즘에서는 이와 같은 동작이 16번의 반복 수행된다.

모든 라운드의 수행이 된 후 제어 신호에 따라서 ip_invert에 의해 최종치환을 한 후 key_xor block의 연산을 거쳐 다시 한번 암호화(DES(2))를 수행하여 352 비트 암호문를 생성한다. 암·복호화처리부는 암·복호화하고자 하는 블록 데이터의 입출력을 수행하고 암·복호화 과정에 있어서 매 라운드의 수행결과를 래치하고, 암호의 비도와 밀접한 관계가 있는 F 함수부, 그리고 각각의 치환 및 key_xor로 구성된다.

위 설명을 <그림 6> 암·복호처리부의 내부 블록도를 도시하였다.



<그림 6> 암·복호처리부의 내부 블록도

3.4.1 sh_reg block

control block으로부터 제어신호를 받아 킷값과 데이터를 처리하여 각 블록에게 넘겨준다. shld 신호에 의해, shld가 '1' 일 때 8 비트씩 킷값과 데이터를 받아 레지스터에 저장한다.

shout 신호에 의해, shout가 '1' 일 때 sh_out으로 8 비트씩 출력한다. shld와 shout이 동시에 '1' 일 때 레지스터에 저장되어 있는 킷값과 데이터를 출력한 후 encryption 신호에 의해 암호화와 복호화를 수행한다.

cp_en 신호는 실제 암복호 처리를 수행하고, cp_cnt를 생성하여 16라운드 처리를 제어하고 352 비트 암·복호 처리를 위한 입출력 제어와 레지스터 처리를 한다.

3.4.2 reg_dec block

cp_en신호에 의해 동작하며, sh_reg block로부터 데이터를 입력받아 암호화 처리를 위해 96 비트 데이터를 세 개의 데이터 블록(L : 32 비트, M : 32 비트, R : 32 비트)으로 나누고, encryption 신호에 의해 복호화 처리시 세 개의 데이터 블록을 제어 처리한다.

3.4.3 key_xor block

352비트 방식에서 중간 킷값 96 비트를 저장하고 있다가 첫 번째 암호화(DES(1)) 후에 중간 암호문 데이터(Middletext1)과 XOR 연산을 수행하는 블록이다.

3.4.4 select_reg block

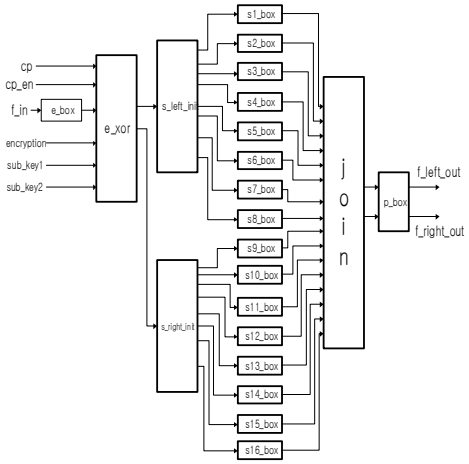
ip_box, ip_invert, key_xor, select_vector block의 출력을 받아 f_select와 ip_invert 신호에 의해 1라운드를 출력하고 다시 sh_reg block에 입력된다.

3.4.5 ip_box 와 ip_invert block

ip_box 와 ip_invert 는 암호화 알고리즘에서 처음과 끝에 동작하는 부분으로써 96비트 입력에 96비트의 출력을 가지고 있는 일대일 치환 기능을 가진다.

3.4.6 f_function(F 함수) block

F 함수부분은 DES 알고리즘에서 비도를 결정하는 중요한 부분이다. 따라서 VHDL를 이용한 실제 F 함수 내부를 블록화하여 <그림 7>을 통해 도시하였다.

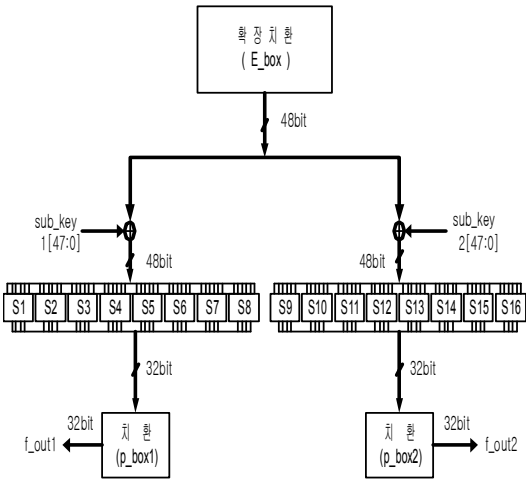


<그림 7> VHDL을 이용한 F 함수의 내부 블록도

reg_dec block으로부터 32 비트(M)의 데이터를 입력받고 키 생성부로부터 받은 sub_key1과 sub_key2로 암·복호의 F 함수부분을 수행한다.

F 함수는 DES에서 비도를 결정하는 중요한 부분으로써 S_box, E_box, p_box 와 XOR 연산으로 구성되어 있다. 먼저 암호화의 블록 데이터 중 32 비트를 F함수의 입력(reg_m[31:0] : M)으로 정하고 이 입력은 확장치환(E_box)을 거쳐서 32비트의 데이터를 48비트로 변환하다.

변환된 신호 48비트는 키 생성부에서 보내온 서브키(sub_key1[47:0], sub_key2[47:0])와 XOR 연산되어 S_box의 입력으로 보내진다. S_box을 통과한 신호를 단순한 비트 재배열(p_box)을 거쳐 출력된다. <그림 8>은 F 함수부 내부를 도시 하였다.



<그림 8> F 함수의 내부 블록도

(1) E_box와 p_box

E_box는 32비트 입력을 받아서 48비트로 출력하는 확장치환을 행하는 부분이다. 우선 비트의 배열을 보면 첫 번째 비트와 네 번째 비트를 각각 두 비트씩 출력과 연결된다. 두 번째와 세 번째는 단지 1비트의 출력과 연결된다.

p_box는 32비트의 입력에 32비트의 출력을 갖는 1:1 치환이다. S_box의 32 비트 출력은 p_box에서 치환된다. 이 치환은 무시되거나 이중으로 쓰이는 비트가 없어서 straight permutation 또는 그냥 permutation이라 부른다. 예를 들어 16번째 입력이 1번째 출력 비트가 되고, 7번째 입력이 2번째 출력 비트가 되는 등이다.

(2) S_box

제안한 352비트 알고리즘에서는 S_box가 16개 사용된다. F함수에서 확장치환(E_box)을 거쳐 48 비트로 나온 값들은 키 생성부로 출력된 서브값(sub_key1[47:0], sub_key2[47:0])과 XOR 연산을 한다. 이 결과 48 비트는 6 비트씩 나누어져 처음 6 비트는 S_box S_1과 S_9에 다음 6비트는 S_2와 S_{10}에, …순으로 입력되고 각 S_1, S_2, …,S_{16}들은 각각 4비트를 출력하게 된다.

S_box들의 역할은 각각 6비트를 받아들여서 4 비트로 출력하는 것이다. 각 S_box들은 비선형 특성, 입력의 1 비트 값이 변화했을 때 출력이 적어도 2 비트가 변화하는 성질 등 여러 특성을 가지고 있다.

예를 들어 입력되는 6 비트를 a_1, a_2, a_3, a_4, a_5, a_6 라 하자. 최상위 a_6과 최하위 a_1를 병렬로 결합하여 생긴 이진수를 십진수로 변환하면 0, 1, 2, 3 중의 하나가 될 것이고, 이것으로 S_box의 행을 결정한다. 마찬가지로 a_2, a_3, a_4, a_5를 병렬로 결합하여 만든 이진수를 십진수로 변환시키면 0, 1, 2, 3,…,15중의 하나가 되고 이것을 S_box의 열로 결정한다.

위의 행과 열이 만나는 숫자를 읽으면 되는데 모두 0부터 15사이의 숫자이므로 이진수로 바꾸면 4 비트가 된다. 즉, 각각의 S_box를 지나면서 6 비트의 입력이 4 비트 출력으로 바뀐다.

이 S_box들이 352비트 알고리즘의 핵심이 된다. 암호화 알고리즘 내에서 이 치환 계산이 가장 중요한 과정이다. 나머지 모든 연산은 선형적이고 분석하기도 쉽다. 그러나 S_box는 비선형적이며 암호 보안성에서 핵심을 이루는 부분이다.

3.5 키 생성부(key_generation)

이 키는 암호화 키이며 동시에 복호화 키이다. 복호화 과정은 16라운드의 역 과정을 밟아 가는 것이며 동일한 키를 사용하기 때문에 키의 안전한 관리가 필수적이다. 8

개의 십진수를 각각 8 비트의 이진수로 변환하는데, 마지막 8 비트는 키의 생성이나 보관 과정에서 일어날 수 있는 오류를 파악하기 위함이다.

352비트 알고리즘의 키 생성부는 256 비트 중 패리티 비트를 제거한(ep_box 의해) 암・복호처리부로부터 224 비트를 키 값으로 입력받아서 28 비트씩 8블록으로 나누어진 시프트 레지스터에 저장하며, 나누어진 8블록은 암・복호 수행 중 key_1의 제어신호에 따라 네 개의 블록씩(key_reg1, key_reg2, key_reg3, key_reg4와 key_reg5, key_reg6, key_reg7, key_reg8) 매 라운드마다 정해진 1 비트 또는 2 비트씩 시프트 하여 그 출력을 압축 치환하여 매 라운드마다 키 값을 적용하여 ($k_{1,i}$, $k_{2,i}$)와 ($k_{3,i}$, $k_{4,i}$) 값이 암・복호 처리부로 전달된다.

암・복호처리부에서 패리티 비트를 제거한 키를 입력받아 키 생성부의 레지스터(key_register) 입력 K_in[55:0]이 된다. 이러한 입력을 네 번 반복하여 킷값으로 224 비트가 저장되고 저장된 값은 28 비트씩 8부분의 시프트 레지스터로 나누어지며, 암・복호화 수행시 key_1의 신호로써 key1($k_{1,i}$, $k_{2,i}$)과 key2($k_{3,i}$, $k_{4,i}$)중 하나를 선택하게 되고, k_left와 k_sh_2bit로써 각 라운드에 필요한 key($k_{i,j}$: i는 1 또는 2, j는 1, 2, 3, ..., 16)를 생성한다.

key_register 부분이 수행되면 EP1_box와 EP2_box를 거쳐 두 개의 서브키 sub_key1[47:0], sub_key2[47:0]를 암・복호처리부로 전달한다.

3.6 제어부(control_block)

제어부에서는 암・복호처리부와 키 생성부의 동작을 위한 제어신호를 생성하는 곳으로 352비트 알고리즘의 제어신호는 데이터 입출력에 관한 제어신호(data_input), 키 입출력에 관한 제어신호(key_input), 암・복호 수행에 관한 제어신호(encryp_process)로 나눌 수 있다. 제어부는 다섯 개의 블록으로 구성되어있다.

3.6.1 main block

이 블록은 외부로부터 s_io, key_in 신호를 받아 상호 배타적으로 data_input block, encyp_process block, key_input block을 실행시키게 된다.

또한, main block은 각각 data_input, encyp_process, key_input block으로부터 작업완료 신호로써 data_job_end, key_job_end, data_job_end 신호를 받아 각 블록의 실행 시킨다.

3.6.2 key_input block

main block으로부터 실행되어지는 이 블록은 외부로부터의 키 입출력에 관한 제어신호를 생성한다. 이 블록은 실행 동시에 rr(read request)가 '1'로 하여 외부로부터의 key_input을 요청하게 되고 sg(signal grant)가 '1'로 set 되면 crypto_block의 레지스터에 8 비트 데이터를 저장하도록 한다.

3.6.3 data_input block

암・복호할 데이터의 입출력을 제어하는 신호를 생성하는 블록으로써 데이터를 보내는 외부 장치와 sg, rr, wr(write request) 신호로써 교신하게 된다. 기본적으로 데이터의 입출력은 8 비트 단위로 이루어져, 총 96 비트의 데이터를 내보내고 받아들이는 동작을 교차적으로 수행하게 된다.

먼저 wr을 '1'로 활성화시키고 sg가 '1'로 활성화될 때까지 대기하면, '1'로 활성화되었을 때, 처리가 끝난 데이터 8 비트를 받아드린다. 이러한 주기를 12회 실행하여 96 비트 데이터를 내보내고 새로운 데이터 96 비트를 받아들여지게 된다.

3.6.4 encryp_process

외부로부터 clock을 입력받아 crypto block과 key_generation block의 동기화 및 round 처리와 key_generation에 관한 제어신호를 생성한다. crypto block의 프로세스의 과정을 크게 ip, f_function, ip_invert, key_xor(중간문과 KM의 연산부분) 과정으로 나누어 f_function, ip_invert, 제어신호로써 세 가지

processing을 선택하게 된다. 또 key_generation block에 제어를 위해서 left, sh_2bit 신호를 사용한다.

3.6.5 mux block

key_input, data_input, encryp_process block에서 생성된 제어신호를 out port로 선택적으로 내보내는 역할을 수행한다.

Ⅳ. 시뮬레이션 결과

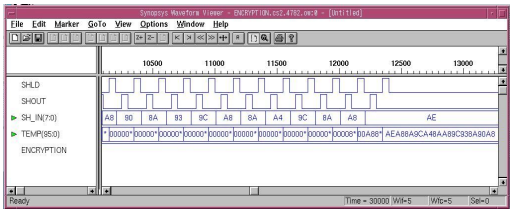
본 논문에서 제안한 352비트 알고리즘은 workstation에서 synopsys사의 Logic Synthesis 툴을 사용하여 논리합성 및 시뮬레이션을 수행하였다. 다음에 나타낼 <그림 9>는 임의의 킷값을 입력으로 하여 시뮬레이션을 수행한 것이다.

킷값(ASCII) :

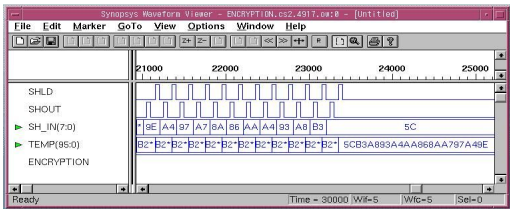
donggukuniversitycomputerscienceinfdepartment

평문(Plaintext) 값(ASCII) :

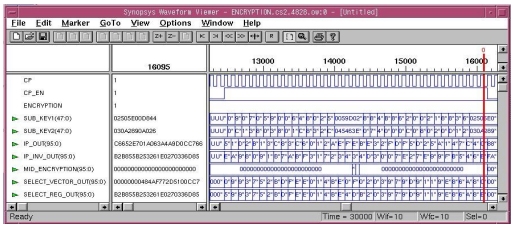
theinternetworksecurity.을 적용하였다.



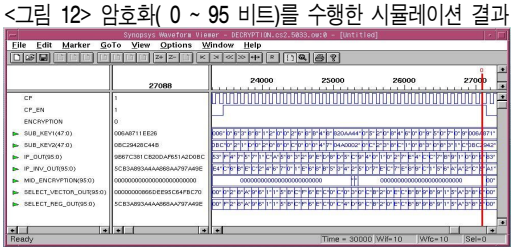
<그림 10> 평문(0 ~ 95 비트) 입력에 대한 시뮬레이션 결과



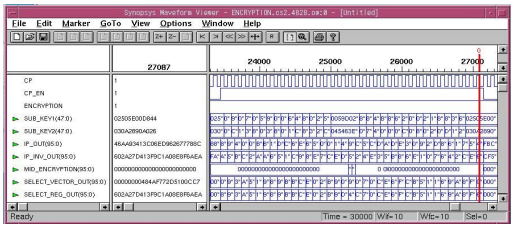
<그림 11> 평문(96 ~ 191 비트) 입력에 대한 시뮬레이션 결과



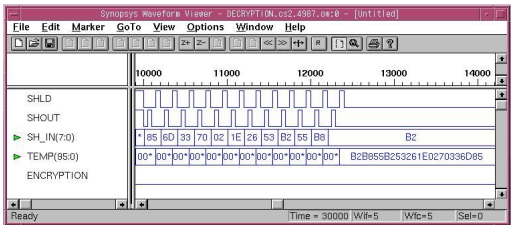
<그림 12> 암호화(0 ~ 95 비트)를 수행한 시뮬레이션 결과



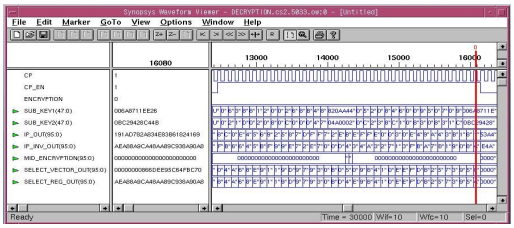
<그림 13> 암호화(96 ~ 191 비트)를 수행한 시뮬레이션 결과



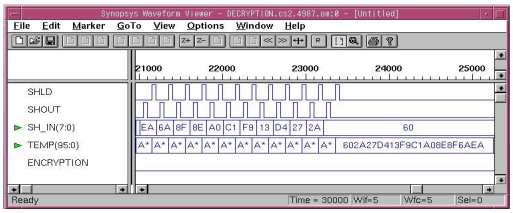
<그림 14> 암호문(0 ~ 95 비트) 입력에 대한 시뮬레이션 결과



<그림 15> 암호문(96 ~ 191 비트) 입력에 대한 시뮬레이션 결과



<그림 16> 복호화(0 ~ 95 비트)를 수행한 시뮬레이션 결과



<그림 17> 복호화(96 ~ 191 비트)를 수행한 시뮬레이션 결과

<그림 10>은 입력되는 평문 값 중 0비트에서 95비트까지의 입력에 대한 시뮬레이션을 수행한 결과이고, <그림 11>은 96비트에서 191비트까지의 입력에 대한 시뮬레이션을 수행한 결과이다. <그림 12>, <그림 13>은 킷값과 평문을 가지고 암호화를 수행한 시뮬레이션 결과이다. <그림 14>, <그림 15>는 암호화 처리를 한 후 생성된 암호문을 가지고 복호화에 입력하는 시뮬레이션 결과이다. <그림 16>, <그림 17>은 암호문을 복호화 수행하여 평문으로 생성되는 시뮬레이션이다. 이때 입력된 평문과 출력된 평문은 동일함을 알 수 있다.

제안된 알고리즘은 C언어를 이용하여 모델링한 후 VHDL로 설계하였고 검증 기법은 top-down 과 bottom-up 방식을 적절히 결합하여 사용하였으며, DES의 테스터 벡터[9]와 일치하는지를 검증하였다. 설계한 회로를 UMC 0.25μm CMOS 라이브러리와 Synopsys Design-Compiler를 합성하였다. <표 1>은 암호 프로세서의 성능 분석에 대해 정리한 것이다.

<표 1> 암호 프로세서 성능 분석

암호 알고리즘	DES, 352비트 DES
Technology	UMC 0.25μm
성 능	DES(64) : 3.4ms DES(352) : 10.6ms
Timing 검증	100MHz

성능에서는 기존의 DES보다 352 DES가 발생되는 게이트 수가 증가함에 다소 부담되는 것으로 보이나 암호화 하는 비도에 대해서는 기존 DES보다 월등히 향상됨을 예측할 수 있다.

Ⅴ. 결론 및 향후과제

현재 DES 알고리즘들은 간단히 소프트웨어로 구현할 수도 있으나 처리 속도가 느려진다는 단점이 있다. 이러한 처리 속도 상의 단점을 극복하기 위해 알고리즘의 하드웨어 구현이 필요하며 이 경우 처리 속도는 소프트웨어에 비해 수십 배 이상 빠르다.

본 논문에서는 기존 DES의 암호 비도를 증가시키기 위해 암호분석의 평가지표에 따라 352비트 DES 알고리즘을 제안하였다. VHDL을 이용한 시뮬레이션을 통해 암호화시의 평문과 복호화시의 생성된 평문이 동일함을 보였다.

제안된 시스템의 응용분야로는 전자 상거래에 이용되는 신용카드 번호 및 현금자동인출기 암호화 등에 널리 사용될 수 있다.

앞으로, 암호 알고리즘의 해킹의 비도에 대한 연구와 증명이 계속 진행되어야 할 것이며, 고속의 컴퓨팅이나 고속통신에 있어 고속의 암호화는 필수적으로 해결되어야 할 과제이다.

참고문헌

[1] W. Diffie, M.E. Hellman, “New directions in cryptography”, IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp. 644-654.

[2] IEEE P1363/D13, “Standard Specifications for Public Key Cryptography”, 1999.

[3] NBS, “*Data Encryption Standard*”, FIPS Pub, 46, U.S, National Bureau of Standard, Washington DC, Jan. 1977.

[4] Data Encryption Algorithm, American National Standard X3, 92. ANSI, NY. 1981.

[5] Eli Biham and Adi Shamir, “Differential Cryptanalysis of DES-like Cryptosystems”, Journal

of Cryptology, No. 4, 1991, pp. 3-72.

[6] M.Matsui, “Linear Crytanalysis of DES Cipher(I)”, Symposium on Cryptography and Information Security ’93, 1993.

[7] “2x Isolated Double-DES: Another Weak Two-Level DES Structure”, http://www3.10pht.com/pub/blackcrwl/encrypt/2XISOLAT.TXT.

[8] Ralph C. Merkle and Martin E. Hellman, “On the Security of Multiple Encryption”, Communications of the ACM, Vol. 24, No. 7, July 1981, pp. 465-467.

[9] 한국 정보 보호 센터, 128 비트 블록 암호 알고리즘(SEED)개발 및 분석보고서, 1998, 12.

■ 저자소개 ■

박 영 호
Park, Young Ho

1992년 3월~현재
부천대학 e-비즈니스과 교수
2000년 2월 동국대학교 컴퓨터공학과 (공학박사)
1986년~1991년 8월
통계청 전산사무관
1985년 2월 동국대학교 전자계산학과 (공학석사)
1983년 2월 동국대학교 전자계산학과 (경영학사)

관심분야 : 컴퓨터보안, 응용소프트웨어
E-mail : yhpark@bc.ac.kr

논문접수일 : 2008년 11월 9일
수 정 일 : 2009년 1월 3일
게재확정일 : 2009년 1월 10일