Twofish 알고리즘을 이용한 저용량 암호화 Chip의 모듈설계

김 영 득°, 장 영 조
한국기술교육대학교 정보기술공학부
{skystory°, yjjang}@kut.ac.kr

Module Design of Low Volume Cryptography Chip using Twofish Algorithm

Young-Deuk Kim°, Young-Jo Jang
School of Information Technology, Korea University of Technology and Education

요 약

Twofish 알고리즘은 작은 부피의 로직, Triple-DES보다 강력한 암호화 레벨, 암호화 속도 등을 갖추어 모듈 설계 알고리즘으로 선정하였다. Twofish 알고리즘은 bitwired-XOR, Permutation, S-box, MDS, PHT를 걸치는 H함수를 각기 다른 키로 반복 라운드를 함으로써 대상 데이타를 암호화 한다. 64~256bit의 키 크기와 라운딩 횟수를 조정하여 모듈의 부피나 처리속도를 유동성 있게 조절할 수 있는 장점이 있다. 하드웨어 기기와 응용에 사용하기 위하여 VHDL 모듈로 알고리즘을 설계하고 그 동작을 검증하였다. 구현된 회로는 기존의 방법에 비하여 파이프라인 단계를 적용하므로써 약 23%의 속도 향상을 얻을 수 있었다.

1. 서론

오늘날 인터넷뿐만 아니라 하드웨어에서도 개인 정보의 비밀유지 요구가 증가하고 있다. 암호화 알고리즘의 복잡한 수식에 의해 소프트웨어로 동작할 때 제한 받는 저속 문제를 해결하고자 암호화 칩의 요구 또한 증가하는 추세이다. 그중 소형하드웨어 형태로만 탑재가 가능한 IC-Card, USB 메모리 스틱, 핸드폰 등에 사용하기 위한 빠른 속도와 저용량의 암호화 칩이 필요하다. 이를 위하여 Triple-DES 알고리즘을 보강한 Twofish 알고리즘을 설계 및 구현하였다. Twofish 알고리즘은 Schneier 등이 개발한 AES 2차 후보 알고리즘으로 key에 의존하는 S-box을 갖는 16라운드의 Feistel 구조의 블록 암호 알고리즘으로 응용에 따라서 유연성 있게 Key 스케쥴링을 할 수 있는 장점이 있다.

2. TWOFISH 알고리즘

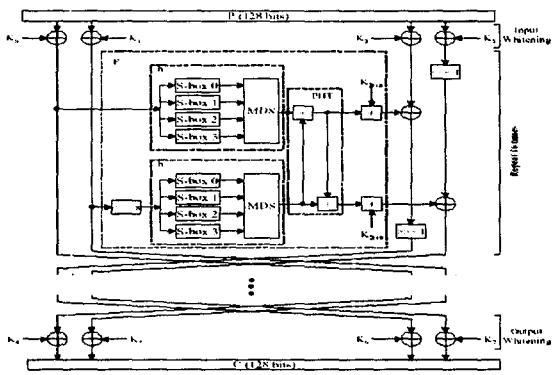

그림 1. Twofish 전체 블록도

128비트의 평문(planetext)이 들어오면 K0~K3키로 32비트씩 XOR 연산을 하는 whitening을 한다. 다음 permutation, S-box, MDS, PHT와 라운드키(K8~K39)를 ADD시켜주는 F 블럭을 이용하여 16번의 라운딩을 하게 된다. 이때 각 라운딩 후 64비트씩 상하위를 바꾸어 주게 되는데 각 라운딩에 두개의 키가 소요된다. 마지막으로 K4~K7키로 32비트씩 XOR연산을 하는 whitening을 수행하면 암호화된 cipher text가 나오게 된다. 여기서 사용되는 XOR 연산은 비트와이어드(bitwired) 연산이다. 쉬프팅 연산은 순환(rotation) 쉬프팅을 사용한다. ADD는 32비트에서의 modulo 2^{32}이 사용된다. 서브키인 S0, S1과 K0~K39는 Key 스케쥴에 의해 생성된다.

S-box는 q0, q1의 permutation 블럭을 실행한 후 global key에서 key 스케쥴링에 의해 파생된 S0, S1와 중간에 XOR 연산을 하는 모듈로서 그림2에 S-box의 블록도를 보인다.

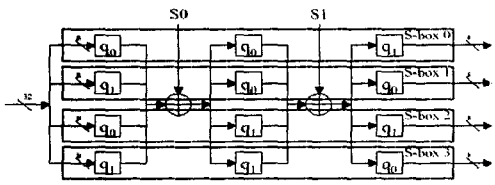

그림 2. S-boxes 블록도

S-box에서 사용된 permutation q0및 q1은 입력되는 8비트의 데이타를 니블 단위로 나누어 처리하게 된다. 4비트의 a, b는 쉬프팅, XOR연산을 걸쳐 t0~t3의 블럭으로 들어간다. t0~t3는 그림3과 같이 각각에 대응되는 값으로 변하게 된다. Permutation q0, q1는 블럭의 구조는 같으나 t0~t3을 달리하며 상이한 출력값을 가지며 S-box에서 번갈아 가며 사용된다.

MDS 블럭은 갈루아 이론의 GF(2^8)을 이용하여 나온 maximum distance separable matrix를 이용하여 입력된 데이타 값을 변환해준다.

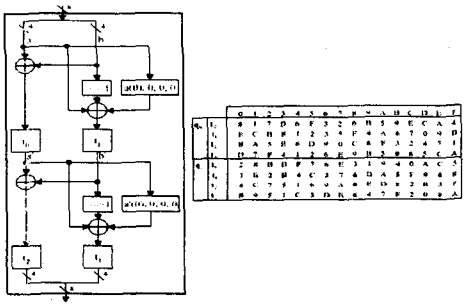

그림 3. Permutation q

여기서 y0~y3은 각각 8비트씩 4개의 값이고 출력인 z0~z3 또한 8비트씩 4개의 값으로 이루어져 32비트의 출력이 나온다. 식(1)을 보면 3개의 승산으로 이루어진 것을 알 수 있다.

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \quad \cdots \quad \text{식 (1)}$$

PHT(Pseudo-Hadamard Transform) 블럭은 두개의 모듈러 가산으로 이루어진 간단한 함수이다. 32비트의 두개의 입력값 a, b를 이용하여 다음과 같은 식으로 a'= a + b, b' = a + 2 * b의 출력값을 가진다.

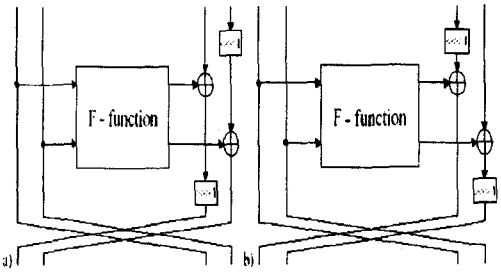

그림 4. 암호화와 복호화의 틀린점 a)암호화b)복호화

암호화(encrytion)과 복호화(decryption) 방법을 그림4와 같이 쉬프트의 위치로 간단히 제어 할 수 있다. 대부분의 FPGA구현에서는 같은 F-function을 사용하고 MUX를 사용하여 쉬프트 값을 제어함으로써 암호화와 복호화를 수행한다.

S0,S1과 K0~K39의 32비트로 구성된 42개의 서브키를 생산하기 위해서 key 스케쥴링을 하게 된다. 우선 global key를 그림4와 같은 RS matrix에 넣어 서브키인 S0, S1을 만든다. Global key가 128비트이면 S0~S1이 생성되고 192비트이면 S0, S1, S2가 생성, 256비트이면 S0, S1, S2, S3이 생성된다.

$$\begin{pmatrix} S_{i,0} \\ S_{i,1} \\ S_{i,2} \\ S_{i,3} \end{pmatrix} = \begin{pmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{pmatrix} \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix} \cdots \text{식(2)}$$

식(2)의 RS매트릭스 역시 갈루아 필드 GF(2^8)을 이용한 승산을 한다. 또 다른 서브키인 K0~K39는 그림5와 같이 8과 9의 쉬프트가 더 추가된 F_function을 이용하여 만든다. 이때 M0~M3은 128비트의 global key를 4등분한 것이다. (M3 상위32비트, M0 하위32비트) 입력값인 2i 및 2i+1에서 i는 해당키의 순번이 $2^{24} + 2^{16} + 2^8 + 2^0$과 곱해져서 들어가게 된다. 만일 K12를 생성할 때엔 입력값은 16진 값으로 0C0C0C0C이 들어가게 된다.

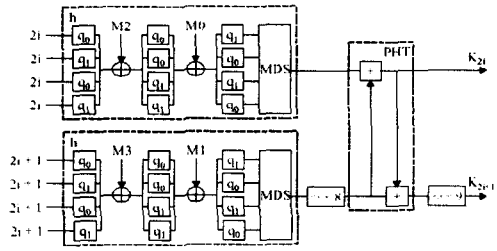

그림 5. 키스케쥴링

3. VHDL 설계 및 시뮬레이션

기본적인 모듈인 Permutation_q0,q1을 VHDL로 구현하고 S-Box, MDS, PHT, S-Box과 MDS로 이루어진 g_blk, RSmatrix의 구현을 위해 각 라운딩과 서브키 생성을 위해 수정된 F_function, 레지스터, ROR(right rotation shift), XOR, modadd32, 이들을 제어하는 제어회로로 구성된다. RAM이나 ROM을 사용하지 않고 10개의 32비트 레지스터만을 사용하여 암호화를 수행하였다. 가장 낮은 계층 단위의 component에서의 실행시간이 15ns~30ns에서 처리되는것과 동일하게 각각의 컴포넌트들의 동작 실행시간을 맞추었다. 결과적으로 많은 클럭이 소요되지만 코어(core)가 빠른 클럭에서 동작 될 수 있도록 하였다.

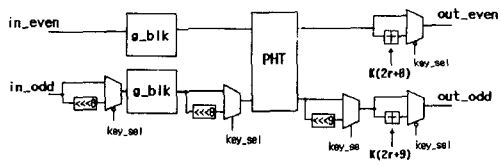

그림 6. 수정된 F_function (modif_F) 블록도

그림7은 F함수 블록의 시뮬레이션이다. f_M0~f_M3 입력은 키스케쥴링을 수행 시에는 M0~M3값이 입력되고 암호화 처리시 S0, S1의 값이 입력된다. 입력에서 출력까지의 딜레이는 평균 80ns 정도가 소요된다.

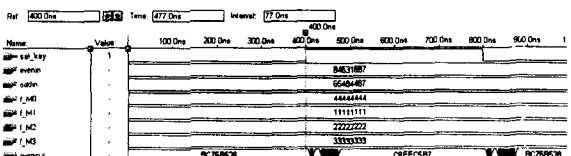

그림 7.수정된 F_function 시뮬레이션

각 라운드시 사용되는 F function과 Key 스케쥴의 공통적인 부분을 참고하여 32비트 MUX를 사용하여 하나의 블럭에서 사용할 수 있도록 F_function을 수정하였다. key_sel이 0이면 라운드 펑션을 수행하고 1이면 키 스케쥴링을 실행한다.

표 1.

Idle	K0,K1(상위whitening)		K2,K3(하위whitening)	
K8,K9	1 Round	. . .	K38,K39	16 Round
K4,K5(상위whitening)		K6,K7(하위whitening)		

컨트롤 블럭에서는 코어의 전체적인 흐름을 제어하고 처리순서를 관장하게 된다. 다음은 컨트롤 시퀀스로 표1과 같은 순서로 코어를 동작시킨다. 총 37단계의 상태를 가진다. 처음 idle은 동작 전 상태를 나타낸다. 그 다음의 두번째 세번째에서는 초기 킷값인 K0~K3의 생성과 키 값과 초기 입력 값과의 XOR연산을 수행한다. 16라운드에서 각 라운드에 사용되는 두개의 서브키를 생성하고 n Round에서 암호화를 수행하게 된다. 이렇게 16라운드 처리가 끝나면 K4~K7 생성하고 마지막 출력 값과 XOR연산을 하고 끝난다. 처리순서는 한 클럭에 동작 하도록 설계 하였다. 즉 한 암호화 나 복호화의 처리는 37클럭으로 처리된다. 컨트롤 블록에서는 처리순서에 적합한 각 블록의 컨트롤 신호를 만들어 전체 흐름을 관장한다. 그림8은 컨트롤 블록의 시뮬레이션 결과 파형이다.

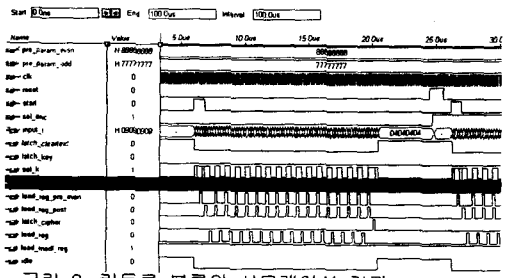

그림 8. 컨트롤 블록의 시뮬레이션 결과

최종 코어의 입출력핀의 구성은 입력으로 clk, start, reset, en_decryt, input[127..0], inkey[127..0]이 있고 출력은 output[127..0], idle로 구성된다. en_decryt은 암호화 또는 복호화 과정을 결정하고, input에는 128비트의 평문이나 암호문이 입력되고, inkey는 128비트의 키값이 입력된다.

4. 실험결과 및 결론

Key 값을 다음과 같이 입력하고 PT와 같은 평문 입력 시 암호화된 문장, CT값과 같이 출력되었다. 이 암호화된 key와 CT를 입력하여 복호화된 과정을 거치게 되면 PT와 같은 값의 평문값이 나왔다. 이 과정을 논문들에 나와 있는 다른 값을 가진 key와 PT를 입력하여 여러번의 시뮬레이션을 거친 결과도 암호화 및 복호화 과정을 거친 결과 또한 올바른 평문을 얻을 수 있었다. 그림9은 표2의 입력값에 대한 시뮬레이션 결과이다. KEY는 고정되어 입력되고 암호화시 PT가 입력으로 넣고 암호화를 수행하면 CT값이 출력되고, 복호화시 CT를 입력하면 PT의 값이 출력된다.

표 2. IP 입력값

KEY = 5E383D5DF1F87A99BA55EF3B1199A2E8
PT = 932271E09F9BDFC0CE0D6107A49E7C68
CT = 9140E0838F689303BF79A38EFFD5AD8E

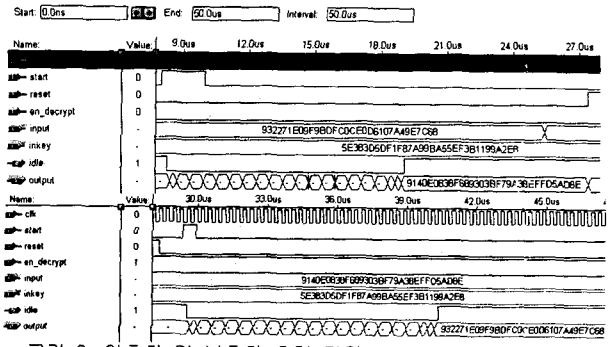

그림 9. 암호화 및 복호화 출력 파형

표3은 본 논문에서 구현된 칩과 공개된 Twofish IP의 성능을 비교한 것이다. 표1에서 보이는 것과 같이 en/decrytion수행 완료시 걸리는 최소시간은 한번의 암호화나 복호화 수행시 걸리는 시간을 말한다. 전체소요클럭은 한번의 암호화나 복호화에 걸리는 클럭의 수를 나타낸다.

표3. 다른 IP와의 비교

	본 논문 IP	공개 IP
전체소요클럭	37clock	73clock
클럭 폭	150ns	100ns
en/decrytion수행 완료시 걸리는 최소시간	11100ns	14600ns
LCs	4440	3168
전체입력핀	260	261
전체출력핀	129	129

최종 23%정도의 속도의 향상을 얻을 수 있었다. 이를 위하여 Altera FLEX 10K FPGA에 사용되는 되는 LC 클럭의 개수는 1272개 정도 더 사용된다. 전체 입출력 핀의 개수는 두개의 IP가 비슷하게 나타난다. 용량보다는 처리속도에 중점을 두어 설계하였다.

차후의 연구의 방향은 해당 하드웨어와 응용에 적합하도록 VHDL 모듈을 변경설계하고 해쉬함수나 몇가지 다른 암호화 알고리즘과 접목하여 좀더 암호화 레벨이 높은 알고리즘을 설계하여 비용대 성능비를 높이고자 한다.

참고문헌

[1] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall "Twofish: A 128-Bit Block Cipher", Counterpane Systems, June 1998.

[2] Pawel Chodowiec, Kris Gaj, "Implementation of the Twofish Cipher Using FPGA Devices" , George Mason University, July 1999.

[3] William Stalliings, "Network Security Essentials", Prentice Hall, 2000.

[4] Niels Ferguson, John Kelseyy, Bruce Schneierz, Doug Whiting ,"A Twofish Retreat: Related-Key Attacks Against Reduced-Round Twofish", Twofish Technical Report, Feb, 2000.