모바일용 블록암호 알고리듬 HIGHT의 하드웨어 구현

박해원 · 신경욱
금오공과대학교 전자공학부

An implementation of block cipher algorithm HIGHT for mobile applications

Hae-won Park · Kyung-wook Shin
School of Electronic Eng., Kumoh National Institute of Technology
E-mail : goekd1216@kumoh.ac.kr

요 약

본 논문에서는 한국기술표준원(KATS)과 국제표준화기구(ISO/IEC)에 의해 표준으로 채택된 블록암호 알고리즘 HIGHT의 효율적인 하드웨어를 구현하였다. HIGHT 알고리듬은 USN과 RFID와 같은 유비쿼터스 환경에 적합하도록 개발되었으며, 128 비트 마스터 키를 사용하여 64 비트 평문을 64 비트 암호문으로, 또는 그 역으로 변환한다. 저면적과 저전력 구현을 위해 암호화 및 복호화를 위한 라운드 변환 블록과 키 스케줄러의 하드웨어 자원이 공유되도록 설계 최적화를 하였다. 0.35-㎛ CMOS 표준 셀 라이브러리를 이용한 합성결과, HIGHT64 코어는 3,226 게이트로 구현되었으며, 80-㎒@2.5-V로 동작하여 150-Mbps의 성능을 갖는 것으로 평가되었다.

ABSTRACT

This paper describes an efficient hardware implementation of HIGHT block cipher algorithm, which was approved as standard of cryptographic algorithm by KATS(Korean Agency for Technology and Standards) and ISO/IEC. The HIGHT algorithm, which is suitable for ubiquitous computing devices such as a sensor in USN or a RFID tag, encrypts a 64-bit data block with a 128-bit cipher key to make a 64-bit cipher text, and vice versa. For area-efficient and low-power implementation, we optimize round transform block and key scheduler to share hardware resources for encryption and decryption. The HIGHT64 core synthesized using a 0.35-㎛ CMOS cell library consists of 3,226 gates, and the estimated throughput is 150-Mbps with 80-MHz@2.5-V clock.

키워드

HIGHT algorithm, block cipher, encryption

Ⅰ. 서 론

암호화는 컴퓨터에 저장되어 있거나 유·무선 네트워크를 통해 전달되는 정보를 제삼자가 가로채어 그 내용을 노출시키거나 의도적으로 내용을 조작·변경하는 등의 보안공격으로부터 정보를 보호하기 위한 수단으로 사용되며, 인터넷과 스마트 폰을 중심으로 한 정보화 사회가 보편화됨에 따라 그 중요성이 점점 증대되고 있는 핵심기술이다[1].

암호 시스템은 암호화 키(key)와 복호화 키가 동일한가에 따라 대칭형(비밀키 방식이라고도 함) 암호방식과 비대칭형 (공개키 방식이라고도 함) 암호방식으로 구분된다. 대칭형 암호방식은 국제적으로 널리 사용되고 있는 AES(Advanced Encryption Standard)[2]와 우리나라 표준인 SEED[3] 등이 있다. HIGHT(HIGh security and light weigHT)[4]는 무선인식 전자태그(RFID), 유비쿼터스 센서 네트워크(USN), 스마트카드(smartcard), 스마트폰 등과 같이 저전력, 경량화가 요구되는 휴대형 기기 및 모바일 환경에 맞게 암호 연산에 필요한 하드웨어 자원과 전력소비가 최소화되도록 개발된 64 비트 블록암호 알고리듬이며, 2006년 12월에 국내 표준(TTAS.KO-12.0040)으로 제정되어 2008년 12월에 개정 표준(TTAS.KO-12.0040/R1)이 발표되었으며, 2010년 6월에 ISO/IEC 국제표준으로 승인되었다.

HIGHT는 128 비트 마스터 키를 사용하여 64 비트 평문을 64 비트 암호문으로 변환하며, 제한적 하드웨어 자원을 갖는 환경에서 구현될 수 있도록 바이트 단위의 기본 산술연산인 XOR, 덧셈, 순환이동만으로 구현되며, S-Box(Substitution-Box: 블록암호에서 암호문과 키 사이의 비선형성을 증가시키기 위해 사용하는 치환 연산

의 일종)를 사용하지 않아 SEED, AES 등 다른 블록암호 알고리즘 보다 하드웨어 구현이 간단하다는 장점을 갖는다.

본 논문에서는 모바일 환경에 적합하도록 최적화된 HIGHT 블록암호 코어를 설계하였으며, FPGA 구현을 통해 하드웨어 동작을 검증하였다.

II. HIGHT 블록암호 알고리듬

HIGHT는 64 비트의 평문(암호문) 블록을 128 비트의 마스터키로 암호화(복호화)하여 64 비트의 암호문(평문)을 생성하는 대칭키 방식의 블록암호 알고리듬이다. 변형 Feistel 구조를 기반으로 총 34 라운드의 연산을 통해 암호화(복호화)가 이루어지며, 128 비트의 마스터키로부터 생성되는 서브키가 라운드 변환에 사용된다. 라운드 변환과 서브키 생성이 바이트 단위의 순환이동(cyclic shift), mod-2 덧셈, mod-2^8 덧셈 등의 단순한 연산만으로 구현되어 모바일 및 유비쿼터스 환경에 적합한 작은 하드웨어와 저전력 구현이 가능하다는 특징을 갖는다.

HIGHT 알고리듬의 전체구조는 그림 1과 같이 초기변환, 32회의 라운드 변환, 최종변환의 총 34회 라운드 변환으로 구성되며, 초기변환과 최종변환은 순환이동 없이 화이트닝 키 가산만 이루어진다. 암호화과정과 복호화과정은 역순(reverse order)으로 이루어지며, 암호화 과정의 모듈로 덧셈은 복호화 과정에서 모듈로 뺄셈으로 구현되고, 순환이동 방향도 반대로 이루어진다. 또한, 라운드 서브키도 역순으로 사용된다.

2.1 암호화 과정

HIGHT의 암호화 과정은 64 비트의 평문 P에 화이트닝(whitening) 키 WK 및 서브 키 SK를 가산하고 바이트 단위 순환이동 시키는 초기변환, 32회의 라운드 변환(R1~R32) 그리고 최종변환을 통해 암호문 C를 생성하며, 그림 2와 같은 pseudo code로 표현된다. 화이트닝 키와 서브 키는 키 스케줄러 블록을 통해 생성된다.

초기변환은 4 바이트의 화이트닝 키 WK_0, WK_1, WK_2, WK_3을 이용하여 평문 P를 첫번째 라운드 변환의 입력 $X_0 = X_{0,7} \| X_{0,6} \| \cdots \| X_{0,0}$로 변환한다. 32회의 라운드 변환 중, R1~R31 라운드 변환은 8 바이트의 데이터 $X_{i-1} = X_{i-1,7} \| \cdots \| X_{i-1,0}$ (단, $i = 1, \cdots, 31$)를 받아 mod-2^8 및 mod-2 덧셈과 바이트 단위의 순환이동을 통해 $X_i = X_{i,7} \| \cdots \| X_{i,0}$로 변환한다. 한편, 32번째 라운드(R32) 변환에서는 바이트 순환이동 없이 mod-2^8 및 mod-2 덧셈만 이루어진다. 최종변환은 R32의 출력 $X_{32} = X_{32,7} \| \cdots \| X_{32,0}$에 4 바이트의 화이트닝 키 WK_4 WK_5, WK_6, WK_7을 가산하여 변환한다. 라운드 변환은 식 (1)과 같이 순환이동과 XOR 연산으로 구성되는 두 개의 F 함수를 갖는다.

$$F_0(X) = X^{\ll 1} \oplus X^{\ll 2} \oplus X^{\ll 7} \quad (1\text{-a})$$

$$F_1(X) = X^{\ll 3} \oplus X^{\ll 4} \oplus X^{\ll 6} \quad (1\text{-b})$$

복호화 과정은 초기변환(암호화 최종변환의 역변환), 32회의 라운드 변환, 최종변환(암호화 초기변환의 역변환)으로 이루어진다. 복호화 과정의 초기변환에는 4개의 화이트닝 키 WK_4, WK_5, WK_6, WK_7가 사용되며 mod-2^8 뺄셈과 XOR 연산으로 구성된다. 복호화의 R1~R32 변환에는 암호화의 역순으로 서브키가 사용되며, mod-2^8 뺄셈과 XOR 연산으로 구성된다. 최종변환에는 4개의 화이트닝 키 WK_0, WK_1, WK_2, WK_3가 사용된다.

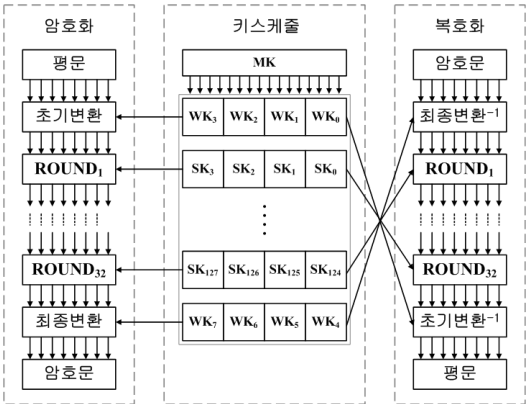

그림 1. HIGHT 알고리듬의 구조

```
HightEncryption(P,MK) {
  KeySchedule(MK,WK,SK);
  HightEncryption(P,WK,SK) {
    InitialTransformation( ) {
      X0,0←P0□WK0; X0,2←P2⊕WK1;
      X0,4←P4□WK2; X0,6←P6⊕WK3;
      X0,1←P1; X0,3←P3; X0,5←P5; X0,7←P7; }
    FOR i=1 TO 31 {
      RoundTransformation( ) {
      Xi,0←Xi-1,7⊕[F0(Xi-1,6)□SK4i-1];
      Xi,2←Xi-1,1□[F1(Xi-1,0)⊕SK4i-4];
      Xi,4←Xi-1,3⊕[F0(Xi-1,2)□SK4i-3];
      Xi,6←Xi-1,5□[F1(Xi-1,4)⊕SK4i-2];
      Xi,1←Xi-1,0; Xi,3←Xi-1,2;
      Xi,5←Xi-1,4; Xi,7←Xi-1,6; }
    }
    R32Transformation( ) {
     X32,1←X31,1□[F1(X31,0)⊕SK124];
     X32,3←X31,3⊕[F0(X31,2)□SK125];
     X32,5←X31,5□[F1(X31,4)⊕SK126];
     X32,7←X31,7⊕[F0(X31,6)□SK127];
     X32,0←X31,0; X32,2←X31,2;
     X32,4←X31,4; X32,6←X31,6; }
    FinalTransformation( ) {
     C0←X32,0□WK4; C2←X32,2⊕WK5;
     C4←X32,4□WK6; C6←X32,6⊕WK7;
     C1←X32,1; C3←X32,3;
     C5←X32,5; C7←X32,7; }
  }
}
```

그림 2. HIGHT 암호 알고리듬의 pseudo code

```
KeySchedule(MK,WK,SK) {
  WhiteningKeyGeneration( ) {
    FOR i=0 TO 7 {
      IF (0 ≤ i ≤ 3) THEN WK_i ← MK_{i+12};
      ELSE WK_i ← MK_{i-4}; }
  }
  SubKeyGeneration( ) {
    ConstantGeneration;
    FOR i=0 TO 7 {
      FOR j=0 To 7 {
        SK_{16i+j} ← MK_{j-i mod 8} □ δ_{16i+j}; }
      FOR j=0 To 7 {
        SK_{16i+j+8} ← MK_{(j-i mod 8)+8} □ δ_{16i+j+8}; }
  }
}
```

그림 3. 키 스케줄러의 pseudo code

2.2 라운드 키 생성

HIGHT의 암호화와 복호화에 사용되는 8 바이트의 화이트닝 키와 128 바이트의 서브 키는 키 스케줄 알고리듬에 의해 생성되며, pseudo code는 그림 3과 같다. 초기변환과 최종변환에 사용되는 화이트닝 키는 마스터 키의 상위 4바이트와 하위 4바이트가 직접 얻어지며, 라운드 서브 키는 7 비트 초기값(101_1010)을 갖는 LFSR (linear feedback shift register)의 32 비트 출력과 마스터 키의 mod-2^8 가산을 통해 생성된다. 랜덤 값 δ를 생성하는 LFSR의 원시다항식은 식(2)와 같이 주어진다.

$$Z_2[x] = x^7 + x^3 + 1 \quad (2)$$

III. HIGTH64 코어 설계

HIGHT64 암호/복호 코어의 전체 구조는 라운드 처리부, 키 스케줄러 그리고 제어블록으로 구성된다. 라운드 처리부는 34번의 라운드 연산을 통해 암호/복호 연산을 수행하며, 각 라운드의 연산은 단일 클록주기에 처리된다. 키 스케줄러는 각 라운드 연산에 사용되는 32 비트의 서브 키를 on-the-fly 방식으로 생성한다.

3.1 라운드 변환 블록

라운드 변환 블록은 그림 4와 같이 구성되며, 64 비트의 평문(암호문) 입력과 32 비트의 라운드 서브 키를 받아 초기변환, R1~R32, 그리고 최종변환으로 구성되는 34번의 라운드 연산을 반복적으로 처리하여 암호(복호) 연산을 수행한다. 입력된 64 비트의 평문(암호문)은 바이트(8-비트) 단위로 분할되어 짝수(0, 2, 4, 6)번째 바이트는 f0-함수, f1-함수 연산과 라운드 서브 키 가산을 거쳐 각각 홀수(1, 3, 5, 7)번째 평문(암호문) 바이트와 가산(감산) 연산이 수행된다. 그림 6에서 기호 ⊞는 mod-2^8 가산을 나타내고, 기호 □는 mod-2 가산을 나타낸다. f0, f1 블록은 식(1)이 나타내는 f-함수를 처리하며, 시프트 동작과 XOR 연산으로 구현된다.

HIGHT 알고리듬의 암호과정과 복호과정은 라운드 변환의 순서와 서브 키의 사용이 역순으로 처리되며, 그

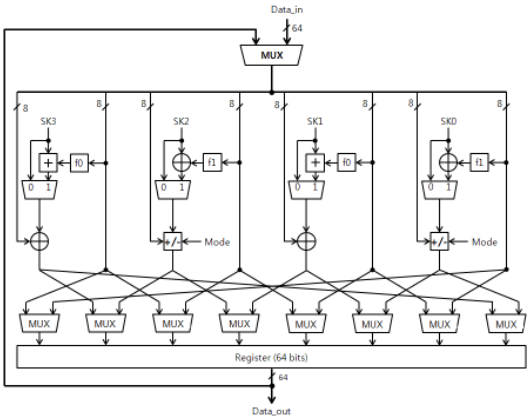

그림 4. 라운드 변환 블록의 구조

리고 바이트 단위의 순환이동 방향도 반대가 된다. 본 논문에서는 하드웨어 공유를 통해 암호연산과 복호연산을 통합하여 구현하였으며, mode 신호에 의해 암호연산(mode=0)과 복호연산(mode=1)이 구분된다. 초기변환과 최종변환의 짝수번째 바이트는 f-함수, 라운드 키 가산, 바이트 단위 시프트 등의 연산이 수행되지 않으며, 홀수번째 바이트는 바이트 단위 시프트 없이 화이트닝 키 가산만 이루어진다. 본 논문에서는 초기변환 및 최종변환과 R1~R32의 연산 및 시프트가 동일한 회로구조로 구현되도록 설계하였다.

3.2 키 스케줄러 블록

키 스케줄러 블록은 128 비트의 마스터 키를 입력받아 8바이트의 화이트닝 키와 128 바이트의 서브 키를 생성한다. 화이트닝 키는 초기변환과 최종변환에 각각 4 바이트씩 사용되며, 서브 키는 R1~R32에 4 바이트씩 사용된다. 라운드 키 생성 블록은 그림 5와 같이 바이트 단위의 좌·우 순환이동을 갖는 128 비트 레지스터 CLRBS_Reg, 32비트의 랜덤 값을 생성하는 delta_gen 블록, mod-2^8 가산기 그리고 MUX로 구성된다. 초기변환과 최종변환에 사용되는 화이트닝 키는 16 바이트의 마스터 키로부터 상위 4바이트와 하위 4바이트가 직접 출력되며, 라운드 서브 키는 delta_gen 블록의 32-비트 출력과 마스터키의 연산을 통해 생성된다. 화이트닝 키와 라운드의 서브 키의 생성을 동일한 회로에 적용하기 위해 화이트닝 키는 detla_gen의 출력 대신 0을 가산하

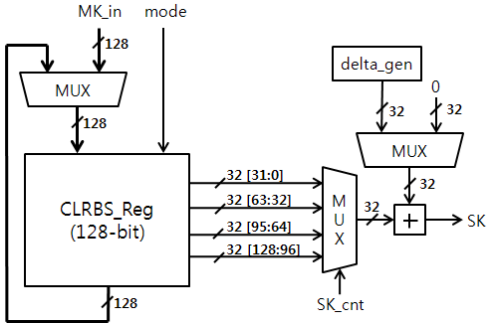

그림 5. 키 스케줄러 블록

여 생성된다. CLRBS_Reg 블록을 통해 4라운 주기로 생성되는 128비트의 라운드 키는 MUX를 통해 32-비트씩 선택되어 라운드 변환 블록으로 입력된다.

복호화 과정에 사용되는 화이트닝 키와 라운드 키는 암호화 과정의 역순으로 생성되며, 마스터 키의 순환이동도 반대 방향으로 이루어진다. 본 논문에서는 이를 효율적으로 구현하기 위하여 mode 신호에 따라 왼쪽 순환이동과 오른쪽 순환이동이 선택적으로 이루어지도록 CLRBS_Reg 블록을 설계하였다. delta_gen 블록은 7비트의 초기 값(101_1010)을 이용하여 28 비트의 랜덤 값을 생성하는 블록이며, 양방향 순환이동을 갖는 LFSR로 구현된다.

Ⅳ. 기능검증 및 성능평가

설계된 HIGHT64 코어는 Verilog HDL로 모델링 되었으며, 그림 6은 ModelSim을 이용한 기능검증 결과이다. 64비트의 평문 "*0123456789abcdef*"와 128비트의 암호키 "*000102030405060708090a0b0c0d0e0f*"를 입력벡터로 사용하였다. 그림 6에서 암호화 연산의 결과로 64비트의 암호문 "*7a6fb2a28d23f466*"이 출력되었고, 이를 다시 복호한 결과는 암호과정에 입력으로 사용된 평문 "*0123456789abcdef*"이 출력됨을 확인함으로써 논리기능이 정상적으로 동작함을 확인하였다.

설계된 HIGHT64 코어는 FPGA 구현을 통해 검증하였다. 검증 시스템은 FPGA 보드, PC 및 UART 인터페이스 등으로 구성되며, Xilinx XC3S1000 FPGA 디바이스를 사용하였다. 그림 7은 검증 결과의 실행 화면이며, 평문을 암호화한 후 이를 다시 복호화하여 원래의 평문이 출력됨을 확인할 수 있다. 이와 같은 FPGA 구현 검증을 통해 설계된 HIGHT64 코어가 정상적으로 동작함을 확인하였다. 0.35-㎛ CMOS 셀 라이브러리를 이용한 합성결과, 라운드 처리부는 1,242 게이트, 키 스케줄러는 1,859 게이트, 그리고 제어부는 124 게이트로 구현되었으며, 전체 HIGHT64 코어는 3,226 게이트로 구현되었다. 시뮬레이션 결과, 설계된 회로의 최대 지연은 12.4-㎱로서 2.5-V 전원전압에서 80-MHz로 동작 가능할 것으로 평가되었으며, 따라서 약 150-Mbps의 성능을 갖는 것으로 평가되었다.

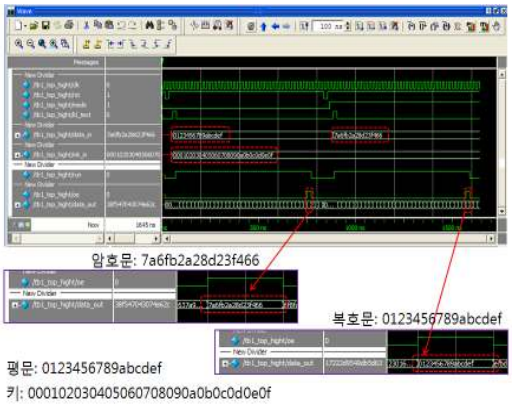

그림 6. 설계된 HIGHT64 코어의 기능검증 결과

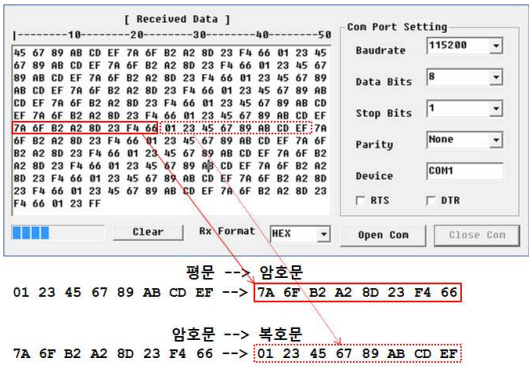

그림 7. HIGHT64 코어의 하드웨어 검증 결과

Ⅴ. 결 론

본 논문에서는 ISO/IEC 국제표준으로 최종 승인된 64비트 블록암호 알고리듬 HIGHT를 하드웨어로 구현하여 동작을 확인하였다. 설계된 HIGHT64 암호/복호 프로세서는 0.35-㎛ 표준 셀로 합성한 결과, 3,226 게이트로 구현되었으며, 무선인식 전자태그(RFID), 유비쿼터스 센서네트워크(USN), 스마트카드(smartcard) 등과 같이 저전력, 경량화가 요구되는 응용분야의 정보보호 코어로 활용이 가능하다.

참고문헌

[1] W. Stalling, *Cryptography and Network Security*, Prentice Hall, 1999.

[2] FIPS Publication 197, "Advanced Encryption Standard(AES)," available at: http://csrc.nist.gov/publication/fips/ fips197/fips-107.pdf

[3] 128비트 블록암호 알고리즘(SEED), TTAS.KO-12.0004 1999. 04

[4] HIGHT 블록암호 알고리즘 사양 및 세부 명세서, 한국인터넷진흥원, 2009. 07

감사의 글

※ 본 논문은 2010년도 금오공과대학교 연구비 지원에 의한 결과임

※ 반도체설계교육센터(IDEC)의 CAD Tool 지원에 감사드립니다.