論文
10-3-7

변형 피스탈 네트워크 블록 암호 알고리즘
Modified Feistel Network Block Cipher Algorithm

조 경 연(Gyeong-Yeon Cho)[1], 송 홍 복(Hong-Bok Song)[2]

요 약

본 논문에서는 변형된 피스탈 네트워크 구조의 128 비트 블록 암호 알고리즘을 제안한다. 제안한 알고리즘은 128, 196 또는 256 비트 키를 가지며, 입력 값 전체에서 선택된 32 비트씩 처리한다. 이러한 구조적 특성은 기존은 블록 암호 알고리즘들과 큰 차별이 되고 있다. 제안한 블록 암호 알고리즘은 국제 표준 블록 암호 알고리즘인 AES와 국내 표준 블록 암호 알고리즘인 SEED 및 ARIA와의 소프트웨어 수행 속도 면에서 많이 개선된 것을 보이고 있다. 이러한 특성을 이용하면 제한된 환경에서 수행해야 하는 스마트카드와 같은 분야에 많이 활용될 수 있을 것이다.

ABSTRACT

In this paper a modified Feistel network 128 bit block cipher algorithm is proposed. The proposed algorithm has a 128, 196 or 256 bit key and it updates a selected 32 bit word from input value whole by deformed Feistel Network structure. Existing of such structural special quality is getting into block cipher algorithms and big distinction. The proposed block cipher algorithm shows much improved software speed compared with international standard block cipher algorithm AES and domestic standard block cipher algorithm SEED and ARIA. It may be utilized much in same field coming smart card that must perform in limited environment if use these special quality.

논문 접수 : 2009. 6. 7.
심사 완료 : 2009. 6. 20.

1) 정회원 : 부경대학교 전자컴퓨터정보통신 공학부 컴퓨터공학과 교수
2) 정회원 : 동의대학교 공과대학 전자공학과 교수

1. 서 론

지식기반의 정보 사회로 빠르게 진행함에 따라 정보보호에 대한 인식이 점차 높아지고 있으며, 정보보호를 위한 기술적 대응조치로 암호기술이 일반적이다. 암호의 사용이 활성화됨에 따라 선진국들은 자국의 기술을 대외에 과시하고 정보보호제품의 수출입을 규제하는 등의 목적으로 자국의 표준 암호를 갖고자 많은 노력을 기울이고 있다. 미국의 경우 NIST[1](National Institute of Standards and Technology)에서 새로운 블록 암호 알고리즘 AES(Advanced Encryption Standard)[2]에 대한 공모 사업을 주관해서 Rijndeal을 선정했으며, 유럽 연합(EU)에서도 회원국들을 위한 암호 알고리즘 공모사업인 NESSIE[3](New European Schemes for Signatures. Integrity. and Encryption)를 진행하여 암호 알고리즘의 표준화를 추진하고 있다. 한편 일본도 2003년부터 CRYPTREC[4](Cryptography Research & Evaluation Committee)를 진행 중에 있다. 우리나라도 1999년 9월 자체 개발한 SEED[5]를 국가 표준으로 제정하였다.

본 논문에서는 변형된 피스탈 네트워크 구조를 가지는 블록 암호 알고리즘인 가칭 MFC(Modified Feistel bock Cipher)을 제안한다. 제안한 MFC의 블록 크기는 128 비트이며 이 중에서 32 비트만을 변경하는 과정을 반복 수행한다. 이러한 구조는 다른 블록 암호 알고리즘과 큰 차별을 보이고 있다. 또한 라운드 간에 정확히 대칭을 이루게 하였다. 키는 128, 192, 256 비트 3 종류를 사용할 수 있게 구성한다. 암호/복호는 전체 2라운드로 구성되었으며, 암호과정에서 8 비트 S 박스 두 종류와 간단한 논리 연산만으로 구성되었으며, 라운드마다 16개의 32 비트 라운드 키를 사용한다.

본 논문에서 제안한 MFC는 기존의 AES, SEED, ARIA 등과 비교하여 소프트웨어 구현시 동작 속도가 빠르며, 하드웨어 구현시 S-박스의 수가 적기 때문에 제한적인 환경에서 동작해야 하는 모바일 제품 등에 유용하다.

2장에서는 MFC 알고리즘의 전체 구조와 각각의 라운드 함수에 대해서 자세히 소개하고, 라운드 키 생성과정, 암호/복호 과정을 설명한다. 3장에서 안전성 분석, 4장에서 효율성 분석 및 구현, 5장에서는 결론을 기술한다.

2. MFC 알고리즘

2.1 MFC 블록 암호 알고리즘의 특징

블록 암호를 구현하는 방식은 크게 2가지가 있으며, 하나는 피스탈 방식과 SPN 방식이 있다. 두 알고리즘은 서로 장단점이 있으며, 특히 피스탈 방식은 암호/복호 알고리즘이 동일한 것이 큰 장점이고, SPN 방식은 암호 알고리즘의 안전성 검정[12]이 보다 쉽다는 장점이 있다.

AES[10]는 암호/복호 알고리즘이 다른 단점이 있으며, SEED는 128 비트 블록을 피스탈 방식으로 구현을 위해 64 비트 서브-블럭으로 나누어 처리하고 있으며, 이는 32 비트 CPU에서는 무리가 있다. 그래서 64 비트 서브-블럭을 다시 32 비트 서브-블럭으로 나누어 처리하는 피스탈 방식이다. 이것은 일본에서 개발한 MISTY[14]의 방식과 유사하다. 최근에 개발된 ARIA[13]는 암호/복호 알고리즘이 동일한 SPN방식이다.

AES, ARIA에서의 SPN 방식에서는 모든 평문을 치환(Substitution)하고 그 결과를 확산(Permutation)하는 방식인데, 차분 및 선형공격에서는 가장 취약한 패스를 선정하여 공격한다. 따라서 모든 평문을 S-박스로 치환시키는 것은 비효율적이다. 본 논문에서는 128 비트 평문을 4개의 32 비트로 나누어 치환과 확산을 시키는 방식을 취하고, 암호/복호 알고리즘을 동일하게 하기 위하여 대칭(mirroring)을 도입했다.

다음은 MFC 블록 암호 알고리즘의 1라운드 진행 과정이며, W 함수를 중심으로 정확히 대칭을 이루고 있다.

A → H → F → W → F → H → A

2.2 MFC 블록 암호 알고리즘의 표기법

MFC의 기본적인 사양은 다음과 같다.
* 변형된 피스탈 네트워크 구조
* 블록 크기 - 128 비트
* 사이즈 - 128, 192, 256 비트
* 2 라운드

본 논문에서 사용하는 기호는 다음과 같다.

|| : 연접
& : 논리곱 연산(AND)
| : 논리합 연산(OR)
^ : 배타적 논리합 연산(XOR)
~ : 인버트(NOT)
<< : 왼쪽 쉬프트
>> : 오른쪽 쉬프트
ROTL(a,b) : 블록 a를 b 만큼 좌로 회전연산

A : 키 덧셈(Add round key layer)
H : 초기화(Whitening layer)
F : 암호 / 복호화(Security layer)
W : 블록 단위 역순(Inverse layer)
G : F함수 내에서 초기화(Whitening layer)
S : F함수 내에서 바이트 치환(S-box)
P : F함수 내에서 확산(Diffusion layer)

K[i] : 32 비트 라운드 키
r : 라운드
X[i]/Y[i] : 32 비트 입력/출력
X/Y : 128 비트 입력/출력

X = X[3] || X[2] || X[1] || X[0]
Y = Y[3] || Y[2] || Y[1] || Y[0]

2.2.1 A 함수

라운드 키 덧셈연산을 수행하는 함수로 블록 단위 연산을 통해 32 비트 입력에 32 비트 라운드 키와 XOR 연산을 수행 후 32 비트 결과를 출력한다. 이를 4번 반복하여 최종적으로 128 비트 결과를 출력한다.

Y[i] = X[i] ^ K[i]

2.2.2 H 함수

초기화 과정을 수행하는 함수로 32 비트 3개 블록과 32 비트 라운드 키를 입력으로 받아 32 비트 결과를 출력한 후 선택되지 않은 1개의 32 비트 블록과 마지막으로 XOR 연산을 수행하고 이를 4번 수행 후 최종적으로 128 비트 결과를 출력한다.

Y[i] = ROTL(X[(i+1) mod 4], 13) ^
ROTL(X[(i+2) mod 4], 19) ^
ROTL(X[(i+3) mod 4], 29) ^ K[4r+i] ^
((X[(i+1) mod 4] & X[(i+2) mod 4]) |
(X[(i+1) mod 4] & X[(i+3) mod 4]) |
(X[(i+2) mod 4] & X[(i+3) mod 4]))

X[(i+4) mod 4]

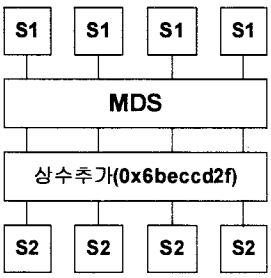
2.2.3 F 함수

본 논문에서 제안한 핵심 암호/복호 알고리즘으로 전체 3 부분으로 나누어져 있다. 초기화 부분인 G 함수, 비선형변환인 바이트 치환 S 함수, 선형변환인 확산 단계 P 함수로 구성되어 있다.
F 함수는 입력으로 3개의 32 비트 블록과 2개의 32 비트 라운드 키를 사용해서 32 비트 출력을 만들고, 이를 4번 수행 후 최종적으로 128 비트를 출력한다.

Y[i] = S2(P(S1(G(X[(i+1) mod 4], X[(i+2) mod 4], X[(i+3) mod 4], K[i])), K[i+1])) ^

X[(i+4) mod 4]

그림 1은 F 함수 내부의 병렬로 된 4개의 비선형변환 바이트 치환과 확산 단계인 MDS(maximum distance separable)를 통과 후 상수 0x6beccd2f 를 추가한 후 다시 병렬로 된 4개의 비선형변환 바이트 치환하는 과정으로 SPS 구조를 이루고 있다.



[그림 21] SPS 구조
[Fig. 1] SPS structure

2.2.3.1 G 함수

F함수 내에서 초기화를 수행하는 함수로 H 함수와 유사하며, 입력으로 32 비트 3개의 블록과 32 비트 1개의 라운드 키를 받아서 내부 연산을 수행 후 32 비트 결과를 출력한다.

```
Y[i] = ROTL(X[(i+1) mod 4], 13) ^
ROTL(X[(i+2) mod 4], 19) ^
            ROTL(X[(i+3) mod 4], 29) ^
K[4r+i] ^
      ((X[(i+1) mod 4] & X[(i+2) mod 4]) |
       (~X[(i+1) mod 4] & X[(i+3) mod 4])
)
```

2.2.3.2 S 함수

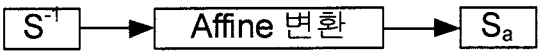
비선형 바이트 치환 S 함수는 4개의 병렬 8 비트 입/출력 S-박스 두 개로 구성된다. S-박스는 다음의 성질을 만족하도록 선정하였다.

* 입출력 크기: 8 비트
* 최대 차분/선형 확률: 2^{-6}
* 대수적 차수: 7
* 고정점 및 반고정점이 없을 것

이러한 성질을 만족하는 것들로는 유한체 GF(2^8)상의 함수 x^{-1}에 아핀 변환을 취한 형태가 널리 사용되고 있다.[7] x^{-1}과 특성이 같은 것으로 x^{-2n}인데 n = 0,...,7이 있다.

본 논문에서 사용한 S-박스는 유한체 GF(2^8) 상의 함수 x^{-1}에 아핀 변환을 사용한다. 이때 소수 다항식(irreducible polynomial)을 가변적으로 사용했다.

그림 2는 S-박스 생성 과정을 나타낸 것으로 첫 번째 통과 S-박스에서는 소수 다항식은 $P(x)=x^8+x^4+x^3+x^2+1$이고, 아핀 변환 다항식은 $P(x)=x^8+1$이다. $P(x)=x^8+1$은 소수 다항식은 아니나 아핀 변환의 규칙성이 좋으므로 이를 선정했다.



[그림 2] S-박스 생성 과정
[Fig. 2] S-box creation process

식(1)은 그림 4의 F 함수에서 첫 번째 통과 S-박스의 생성 과정이다. 두 번째 통과 S-박스는 첫 번째와 다르게 $P(x)=x^8+x^7+x^5+x+1$, 아핀 변환 소수 다항식은 $P(x)=x^8+x^7+x^6+x^2+1$이다.

$$S_a = S^{-1} \bullet 0x1f \qquad (1)$$

식(2)는 두 번째 통과 S-박스 생성식이다.

$$S_a = S^{-1} \bullet 0x38 \oplus 0x94 \qquad (2)$$

2.2.3.3 P 함수

확산 계층인 P 함수는 4 X 4 행렬로 표시하고 입력 32 비트는 $GF(2^8)$ 내에서 행렬 곱셈을 적용하여 32 비트 출력을 만든다. $GF(2^8)$ 내에서 행렬곱셈에 사용된 소수 다항식은 $P(x) = x^8 + x^4 + x^3 + x^2 + 1$이다. P 함수를 그림 3에 보인다.

$$P : GF(2^8)^4 \rightarrow GF(2^8)^4$$

$$(X_0, \ldots, X_3) \rightarrow (Y_0, \ldots, Y_3)$$

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} 2e & 17 & 5c & b8 \\ b8 & 2e & 17 & 5c \\ 5c & b8 & 2e & 17 \\ 17 & 5c & b8 & 2e \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

[그림 3] P 함수
[Fig. 3] P function

확산 계층 설계에 있어서 중요한 고려사항으로 안전성과 여러 환경에서의 효율적인 구현 가능성이다. MFC 암호 알고리즘은 다음과 같은 설계 방향을 가지고 설계한다.

* AES[6]에서 제시된 강력한 분석 방법들에 대하여 충분한 내성을 가져야 한다.
* 8 비트 및 32 비트 소프트웨어 및 하드웨어 구현에 적합해야 한다.
* 동종의 확산 함수 중에서 안전성과 효율성을 고려할 때 우수해야 한다.

2.2.4 W 함수

라운드의 마지막 단계인 W 함수는 32 비트 4개의 블록순서를 역순으로 바꾸는 함수이다. 입력으로 128 비트를 받아 128 비트를 출력한다.

Y[0] = X[3],
Y[1] = X[2],
Y[2] = X[1],
Y[3] = X[0]

2.2.5 라운드 키 생성

라운드 키 생성 알고리즘은 다음과 같은 설계 기준으로 설계한다.

* 연속된 두 라운드에서 사용된 값은 적어도 하나는 달라야 한다.
* 같은 두 값을 사용하여 생성한 두 라운드 키의 일부분으로 두 값의 일부를 예측 할 수 없어야 한다.
* 암호/복호 시 라운드 키 생성에 소요되는 시간이 적어야 한다.

MFC 알고리즘은 2라운드이며 라운드 마다 16개의 라운드 키가 필요하므로 전체 암/복호화 과정에서 총 32개의 32 비트 라운드 키가 필요하다. 한편 키 길이는 128, 192 또는 256 비트 길이이다.
라운드 키 생성을 위해서 32개의 32 비트 라운드 키를 초기화한다. 라운드 키 초기화 과정은 다음과 같다.

```
for(i = KL; i < 32; i--)
  { K[i] += (K[i-KL] << 1) ^ ((K[i-KL] >>
31)& 0xb715167) ; }
```

키 길이가 128, 912 및 256 비트일 때 KL은 각각 4, 6 및 8이 된다. 입력된 초기화 과정 이후에 라운드 키 생성은 다음과 같이 한다.

```
for(i = KL; i < 32; ; i+=4)
 { j=i-KL ;
   K[i] ^= S2(K[j+1]) ;
   K[i+1] ^= ROTL(S2(K[j+2]), 11) ;
   K[i+2] ^= ROTL(S2(K[j+3]), 17) ;
```

```
    K[i+3] ^= ROTL(S2(K[j]), 29) ;
}
```

여기서 S2()는 32 비트 값을 8 비트 단위로 나누어 S-박스를 통한 비선형 바이트 치환을 수행하는 연산이다.

2.2.6 암호/ 복호화 과정

MFC 블록 암호 알고리즘은 ASIC 등의 하드웨어 구현에서 효율성을 위하여 작은 게이트 수를 사용하도록 설계했으며, 스마트카드 등의 소프트웨어 구현에 적합하도록 작은 메모리 요구량을 가지도록 설계하여, 다양한 플랫폼에 적용 가능하도록 설계되었다.
그림 4는 MFC 블록 암호 알고리즘의 1라운드 암호화 과정을 설명하는 것으로 입력 128 비트에 라운드 키 128 비트가 XOR 연산을 수행 후 2.2.2절에서 설명한 초기화 과정을 수행하는 H 함수를 통과한다. H 함수의 입력으로 선택된 32 비트 입력 3개와 라운드 키 32 비트로 총 128 비트가 되고, 출력은 32 비트로 출력 결과는 입력에서 선택되지 않은 32 비트와 XOR 연산을 수행하여 32 비트만이 변경된다. 이러한 과정을 4회 반복하여 최종적으로 128 비트 화이트닝(Whitening)을 수행한다. 다음으로 본 논문에서 제안한 핵심 알고리즘인 F 함수로서 F 함수 내부는 초기화 과정, 확산과 치환 과정이 있으며, 자세한 내용은 2.2.3.3절과 2.2.3.2절에서 설명했다. MFC 블록 암호 알고리즘은 전체 2 라운드이므로 그림 4의 과정이 정확하게 W 함수를 기준으로 대칭을 이루고 있다. 복호화 과정은 암호화 과정과 똑 같으며, 복호화 과정에서 라운드 키의 적용은 암호화 과정의 역순으로 하면 된다.

3. 안전성 분석

블록 암호 알고리즘을 공격하는 방법으로 가장 잘 알려지고 강력한 방법이 Biham and Shamir[8]에 의해 제안된 차분분석법과 Matsui[9]가 제안한 선형분석법이 있다. 차분 및 선형 공격에 대한 안전성은 차분특성 확률과 선형근사 확률 값으로 알 수 있다.
제안한 MFC 블록 암호 알고리즘은 내부의 핵심 암호/복호알고리즘은 2 라운드 SPN구조를 하고 있어 활성화된 S-박스를 쉽게 구할 수 있다. SPN 구조 암호 알고리즘의 안전성 평가는 Hong[11]등에 의해서 제안된 바 있다.

3.1 차분특성 확률과 선형근사 확률

최대 차분특성 확률을 dp^f 일 때 확률 값은 다음과 같다.

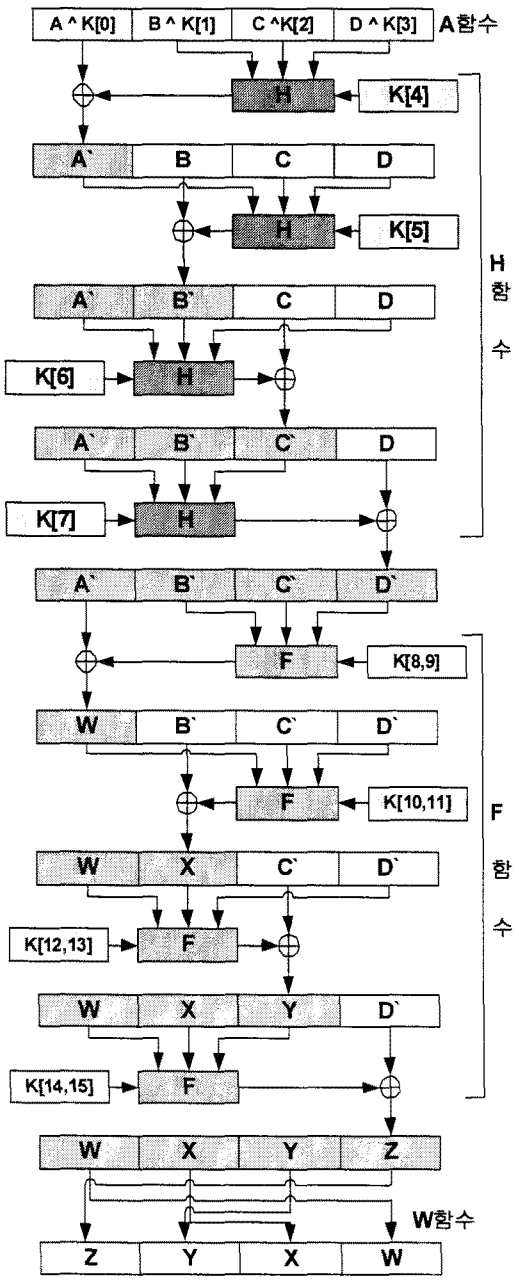
$$dp^f \equiv \max_{\Delta x \neq 0, \Delta y} \frac{\#\{x \mid f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{2^n}$$

최대 선형근사 확률을 lp^f 일 때 확률 값은 다음과 같다.

$$lp^f \equiv \max_{\Gamma x, \Gamma y \neq 0} \left| 2 \bullet \frac{\#\{x \mid x \bullet \Gamma x = f(x) \bullet \Gamma y\}}{2^n} - 1 \right|^2$$

$$dp^f = lp^f = 2^{-6}$$

차분특성 확률과 선형근사 확률은 S-박스에서 정의될 뿐 아니라 블록 암호 전체에 대해서도 정의가 가능하다. 그러나 S-박스의 입출력이 8 비트로 작기 때문에 차분/선형 확률을 쉽게 계산 할 수 있으나, 블록 암호 전체는 보통 128 비트 입출력 크기로 계산이 쉽지 않다.



[그림 4] 1라운드 암호/복호 과정

[Fig. 4] 1 round encryption/decryption progress

차분특성 확률과 선형근사 확률은 S-박스에서 정의될 뿐 아니라 블록 암호 전체에 대해서도 정의가 가능하다. 그러나 S-박스의 입출력이 8비트로 작기 때문에 차분/선형 확률을 쉽게 계산 할 수 있으나, 블록 암호 전체는 보통 128비트 입출력 크기로 계산이 쉽지 않다. 차분공격에 대한 S-박스의 안전성은 최대 차분특성 확률인 dp^f에 의해서 결정된다. 최대 차분확률이 작을수록 S-박스는 차분공격에 안전하다. 같은 방식으로 선형공격에 대해서도 최대 선형근사 확률 lp^f이 작으면 선형공격에 안전하다.

어떤 S-박스에 0이 아닌 입력 차분이나, 0이 아닌 출력 마스크 값이 주어졌을 때 이 S-박스를 차분/선형 공격 관점에서의 활성 S-박스(Active S-box)라고 한다. 한편 SPN 구조 암호 알고리즘에서 차분/선형 공격 관점에서 활성 S-박스의 개수와 밀접한 관계가 있다. 차분/선형 공격에서 활성 S-박스가 많으면 차분/선형 확률이 적을 가능성이 크며, 반대로 활성 S-박스가 작으면 차분/선형 확률이 클 가능성이 있다. 이런 이유로 2 라운드 SPN 구조에서 브란치 수(branch number)라는 개념[12]을 정의한다.

확산단계 P 함수에서 브란치 수 β 는 다음과 같다.

$$\beta = \min_{x \neq 0} \{wt(x) + wt(P(x))\}$$

여기서 $wt(x)$는 0이 아닌 컴포넌트의 개수이다.

3.2 활성화된 S-박스의 수

Hong등에 의해서 제안된 SPN구조의 암호 알고리즘의 안전성 증명조건으로 2라운드(SPS구

조)에 n개의 병렬로 연결된 S-박스가 MDS 확산계층으로 연결될 때 다음과 같은 조건을 만족해야 한다.

* 라운드 키는 독립적이며, 어떠한 편차도 없어야 한다.
* 확산 계층의 브란치 수는 n+1(MDS)이다.
* 최대 차분/선형 확률은 dp^f, lp^f 이다.

MFC 알고리즘의 증명 가능한 안전성은 다음과 같다. F함수 내부의 S함수는 4개의 8 비트 S-박스가 병렬로 연결되었으며, 각각의 최대 차분/선형 확률 값은 2^{-6}이며, 확산계층 P함수는 MDS 코드로 최소 브란치 수는 5이다.

$$\left(dp^f\right)^5 = \left(lp^f\right)^5 = \left(2^{-6}\right)^5 = 2^{-30}$$

이를 4회 반복 수행하므로 1라운드 안전성은 2^{-120}이며, 2라운드에서 2^{-240}으로 충분한 안성성이 있는 것으로 판단된다.

4. 구현 및 비교 분석

4.1 소프트웨어 구현

MFC 블록 암호 알고리즘 변형된 피스텔 네트워크 구조를 가지며 암호/복호가 동일하게 설계되었다. 이러한 구조의 장점은 구현 시 복호화 부분을 따로 구현하지 않아도 되므로 소프트웨어 구현에서는 코드의 크기를 하드웨어 구현에서는 게이트의 수를 줄일 수 있는 장점을 가지고 있다. MFC 알고리즘은 컴파일러로 VC++ 6.0으로 32 비트 소프트웨어로 구현했다. 암호/복호 알고리즘의 핵심인 F 함수는 S-박스 치환과 확산과정으로 라운드 키 생성 알고리즘과 일부 중복 된다. 그래서 암호/복호 실행 전 라운드 키 생성 과정을 수행하는데, 이때 S-박스 치환과 확산과정 P 함수를 한 번만 미리 계산해 메모리에 저장해서 사용한다.
표1은 MFC 알고리즘을 Pentium-4 Window/XP에서 GCC로 프로그램하여 수행 속도를 비교한 결과이다. 표 1에서 MFC의 수행 시간을 1.0으로 정규화하였다.

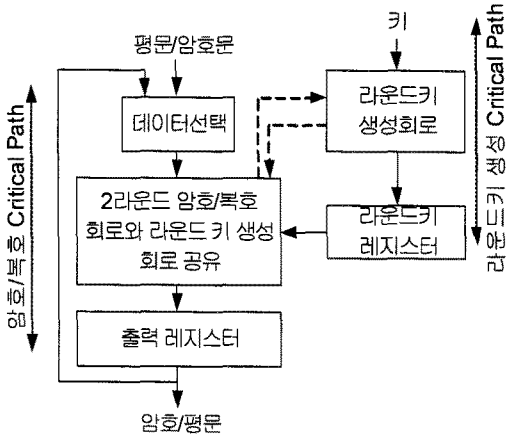
<표 1> 소프트웨어 수행속도 비교
<Table 1> Software execution speed comparison

Algorithm	Time
MFC	1.00
AES	2.77
SEED	1.99
ARIA	4.22

표-1로부터 본 논문에서 제안한 MFC 알고리즘의 소프트웨어 동작 속도는 국제 표준인 AES에 비하여 2.5배 이상 빠르며, 국내 표준인 SEED와 비교해도 2배 가까이 빠른 것을 알 수 있다.

4.2 하드웨어 설계

MFC 알고리즘의 하드웨어 블록도를 그림 5에 보인다. 암호/복호 회로의 구성을 최소화를 목표로 ASIC, FPGA로 구현할 수 있다.



[그림 5] 하드웨어 구현 블록도

[Fig. 5] Hardware implementation block diagram

하드웨어 구성에서 가장 큰 실리콘 면적을 하지하는 것은 S-박스이다. 표-2에 AES 및 SEED와 MFC을 하드웨어로 구현 시에 요구되는 S-박스의 수를 보인다.

<표 2> S-박스의 수

<Table 2> Required number of S-Box

Algorithm	No. of S-Box
MFC	64 ea
AES	320 ea
SEED	192 ea
ARIA	192 ea

표-2로부터 본 논문에서 제안한 MFC를 하드웨어로 구현 시에 S-박스의 수가 적게 소요되므로 기존 방식에 비하여 하드웨어가 간단한 장점을 가진다. 한편 하드웨어 구현 시의 동작 속도는 S-박스 통과 횟수에 주로 영향을 받는데 AES는 10번, ARIA 는 12번, SEED는 48번인데, MFC 알고리즘은 2라운드에서 16번이다. 따라서 MFC는 SEED보다는 하드웨어 동작 속도가 빠르나 AES 및 ARIA보다는 느린 단점을 가진다.

5. 결론

본 논문에서는 변형된 피스탈 구조를 가지는 블록 암호 알고리즘인 가칭 MFC(Modified Feistel block Cipher)를 제안했다. MFC 암호 알고리즘은 암호화 과정에서 32 비트 단위로 값이 변화하는 변형된 피스탈 네트워크 구조이다.

MFC 알고리즘은 128 비트 블록 단위로 키는 128, 192, 256 비트를 사용할 수 있도록 구성되었다. 암호/복호는 전체 2라운드로 구성했으며, 8 비트 S-박스와 논리연산만으로 간결하게 구성했다. 라운드마다 16개의 32 비트 라운드 키를 사용하여 전체 라운드 키는 32개이다. 라운드와 라운드 사이는 대칭을 이루고 있다.

제안한 MFC 알고리즘은 소프트웨어로 구현하였으며, 비교 결과 국제 표준 알고리즘인 AES 및 국내 표준 알고리즘인 SEED보다 수행 속도가 빠른 것으로 나타났다. 그리고 하드웨어 구현 시에 소요되는 S-박스의 수가 AES, SEED 및 ARIA보다 상당히 적으므로 하드웨어 구조가 간단하다. 반면에 하드웨어 동작 속도는 SEED보다는 빠르지만 AES 및 ARIA보다는 느리다.

이러한 특성으로 인하여 본 논문에서 제안한 MFC 알고리즘은 제한적인 환경에서 수행되는 스마트카드와 같은 분야에 활용될 수 있을 것으로 전망된다.

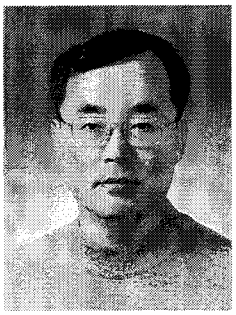
참 고 문 헌

[1] NIST, http://www.nist.gov/aes/

[2] "Report on the Development of the Advanced Encryption Standard(AES)",

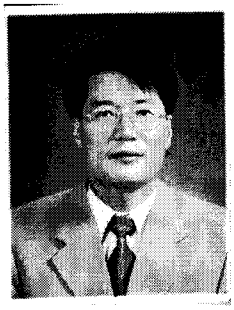
http://www.csrc.nist.gov/encryption/aes/round2/r2report.pdf.
[3] NESSIE project, New European Schemes for Signatures. Integrity. and Encryption, http://cryptonessie.org/
[4] CRYPTREC REPORT 2000, http://www.ipa.go.jp/security/fy12/report/cryptrec-report2k.pdf
[5] SEED, http://www.kisa.or.kr/seed/
[6] NIST FIPS PUB 197: Advanced Encryption Standard, November 2001.
[7] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita, Camellia: A 128-bit block cipher suitable for multiple platforms-design and analysis, LNCS 2012, pages 39-56. Sprihger, 2000.
[8] Eli Biham and Adi Shamir, "Differential cryptanalysis of DES-like cryptosystems." Journal of Cryptology 4(1): 3-72, 1991
[9] Mitsuru Matsui, "Linear cryptanalysis method for DES cipher." In Tor Helleseth, editor, Advances in Cryptology-Eurocrypt '93, volume 765 of Lectuer Notes in Computer Science, pages 386-397: Verlag, Berlin, 1994.
[10] Joan Deamen and Vincent Rijmen, "The Design of Rijndeal" Springer, 2001.
[11] S. Hong, S. Lee, J. Lim, J. Sung, and D. Cheon. "Provable security against differe ntial and linear cryptanalysis for the SPN structure" In Fast Software Encryption 2000, LNCS 1978. Springer-Verlag 2000.

조 경 연



1983-1990 : 삼보컴퓨터 기술연구소 부장
1990년 : 인하대학교 공학박사
1991년-현재 : 부경대학교 전자컴퓨터정보통신 공학부 교수
1998년-현재 : (주)에이디칩스 기술 고문
관심 분야 : 전산기 구조, 반도체 회로 설계, 정보보호

송 홍 복



1985-1990년 :동의과학대학 조교수
1990년 : 동아대학교 공학박사
1989-1990년 : 일본구주공대 객원연구원
1991년-현재 : 동의대학교 공과대학 전자공학과 교수
1994-1995년 : 일본 미야쟈키대학 전기.전자공학부 (POST-DOC)
관심분야 : 다치논리이론 및 다치논리시스템 설계, VLSI설계, 마이크로프로세서 응용