대칭구조 RC6 블록 암호 알고리즘

김길호* · 박창수** · 김종남*** · 조경연****

Symmetry structured RC6 block cipher algorithm

Gil-Ho Kim* · Chang-Soo Park** · Jong-Nam Kim*** · Gyeong-Yeon Cho****

본 연구는 중소기업청의 산학연공동기술 개발지원사업(선도형), 한국산업 기술재단의 지역혁신 인력양성사업의 지원으로 수행 되었습니다.

요 약

암호/복호 알고리즘이 서로 다른 RC6을 간단한 논리 연산과 회전 연산을 사용한 대칭단의 삽입으로 암호/복호를 같게 하는 알고리즘을 제안한다. 즉 RC6의 전체 라운드의 반은 RC6의 암호 알고리즘을, 나머지 반은 RC6의 복호 알고리즘을 사용하고 암호와 복호 알고리즘 중간에 대칭 단을 넣어 암호/복호가 같은 개선된 RC6을 구현했다. 제안한 RC6 알고리즘은 기존의 RC6 알고리즘과 수행 속도에서는 거의 차이가 없고, 안전성은 대칭단의 삽입으로 차분 및 선형 분석에 필요한 높은 확률의 패스를 단절시켜 효과적인 분석을 어렵게 하고 있다. 제안한 대칭단 알고리즘은 암호/복호가 다른 블록 암호 알고리즘에 간단히 적용하여 암호/복호가 같게 만들 수 있으며, 새로운 블록 암호 알고리즘의 설계에도 좋은 아이디어로 사용할 수 있다.

ABSTRACT

RC6 which has different algorithm of encryption and decryption has been proposed to have the same algorithm between encryption and decryption through inserting symmetry layer using simple rotate and logical operation. That means the half of whole RC6 round uses encryption algorithm and the rest of it uses decryption one and symmetry layer has been put into the middle of encryption and decryption. The proposed RC6 algorithm has no difference with the original one in the speed of process. However it is quite safe because by inserting symmetry layer the path of high probability which is needed for differential and linear analysis is cut off so that it is hard to be analyzed. The proposed symmetry layer algorithm can be easily applied to the algorithm which has different encryption and decryption and make it same, and it can be good idea to be used to design a new block cipher algorithm.

키워드

RC6, Feistel, SPN, Symmetry layer, CBC mode

* 부경대학교 컴퓨터공학과 박사과정
** 부경대학교 컴퓨터공학과 계약교수
*** 부경대학교 컴퓨터공학과 교수
**** 부경대학교 컴퓨터공학과 교수(교신저자)

접수일자 2008. 10. 21

Ⅰ. 서 론

데이터 무결성(Integrity)을 보장하는 수단으로 데이터에 대한 대칭 키(Symmetric key) 암호화가 일반적이다. 대칭 키 암호는 블록 암호(Block cipher)와 스트림 암호(Stream cipher)로 나눌 수 있으며, 블록 암호는 많은 알고리즘들이 개발 되었고, 특히 미국의 AES(Advanced Encryption Standard)[1] 선정 프로젝트, 유럽의 NESSIE (New European Schemes for Signatures. Integrity. and Encryption)[2] 프로젝트, 일본의 CRYPTREC (Cryptography Research and Evaluation Committees)[3] 프로젝트 등 선진 각국들은 자국의 정보보호를 위한 암호 알고리즘을 공모를 통해 선정하고 있다. 우리나라는 자체 개발한 SEED[4]와 ARIA[5]를 표준 128비 블록 암호로 제정하였다. 초기 블록 암호는 소프트웨어로 구현되었지만 최근에는 빠른 정보처리를 위한 하드웨어 구현에 많은 연구가 진행 중이다.

블록 암호는 크게 Feistel[6] 구조와 SPN(Substitution Permutation Network)[7]구조로 분류할 수 있으며, Feistel 구조는 암호/복호 알고리즘이 같다는 것이 장점이고 SPN구조는 암호/복호 알고리즘이 다른 특징이 있다. 특히 하드웨어로 암호 알고리즘을 구현할 때 암호/복호 알고리즘이 다른 SPN 구조는 Feistel 구조보다 하드웨어 면적이 약 2배정도 증가하는 단점이 있다.

간단하고, 빠르며, 안전한 블록 암호 알고리즘인 RC6은 미국의 AES 선정 프로젝트, 유럽의 NESSIE 프로젝트, 일본의 CRYPTREC 프로젝트의 128 비트 및 가변길이 블록 암호 선정의 최종 후보 암호 알고리즘이었다. RC6은 변형된 Feistel 구조로 암호/복호 알고리즘이 다른 특징을 가지고 있고 이러한 특징은 암호 알고리즘을 하드웨어로 구현할 때 암호/복호 알고리즘이 같은 것보다 면적이 상당히 증가하는 단점이 된다.

본 본문에서는 암호/복호가 다른 RC6을 간단한 논리 연산만으로 구성된 대칭단을 삽입하여 RC6의 암호/복호 알고리즘을 같게 개선하였다. 다시 말해서 RC6의 전체 20라운드 중 10라운드는 RC6의 암호 알고리즘을, 나머지 10라운드는 복호 알고리즘을 사용하고 중간에 대칭단을 삽입하여 RC6을 암호/복호를 같게 구성했다. 대칭단이 적용된 제안한 RC6 알고리즘은 기존의 RC6 알고리즘과의 수행속도 비교에서 거의 같은 수행속도를 보이고 있으며, 안전성 측면에서도 대칭단의 적용이 제안한 알고리즘의 암호 분석을 어렵게 하고 있다.

본 논문의 구성은 2장에서 RC6을 자세히 소개하고 3장에서 대칭단 및 대칭단이 적용된 RC6을 설명하고 4장에서 제안한 알고리즘의 수행결과 및 안전성을 검증하고 결론으로 마무리 한다.

Ⅱ. 관련연구

RC6은 변형된 Feistel 구조의 암호 알고리즘으로 RC6-$w/r/b$로 표기한다. 여기서 w는 워드의 크기로 32비트이며, r은 라운드 수로 블록의 크기가 128비트인 경우 20라운드 이고, b는 암호화 키의 바이트 수로 16바이트이다. 본 논문에서는 128비트 블록으로 설명한다.

RC6에서 사용된 연산자는 다음과 같다.

- $A+B$: 정수덧셈 mod 2^w
- $A-B$: 정수뺄셈 mod 2^w
- $A \oplus B$: 워드단위 비트 별 xor 연산
- $A * B$: 정수곱셈 mod 2^w
- $A \ll B$: B의 log w만큼 A를 왼쪽 회전연산
- $A \gg B$: B의 log w만큼 A를 오른쪽 회전연산
- $(A, B, C, D) = (B, C, D, A)$: 워드단위 할당연산

RC6의 암호/복호는 32비트 워드단위로 A, B, C, D, 4개의 저장 장소에 평문/암호문을 가지고 20라운드를 반복 수행 후 암호문/평문을 만들어 낸다. A, B, C, D, 4개의 워드는 라운드 함수 수행 후 모두 병렬로 워드 단위 왼쪽/오른쪽 회전 연산이 이루어지며, 라운드 함수 수행 전과 후에 화이트닝(whitening) 단계로 라운드 키와 덧셈/뺄셈을 수행한다. 라운드 함수 내의 핵심적인 안전성은 데이터 의존 회전 연산이고, 이 회전 연산 양은 $f(x) = x(2x+1)$의 2차 함수에 고정된 5비트 왼쪽 회전 연산으로 만들어진다.

RC6은 암호에 적용된 라운드 키는 복호할 때는 암호의 역순으로 적용한다. 그리고 RC6은 변형된 Feistel 구조로 암호/복호 알고리즘이 서로 다르다.

```
B = B + S[0];
D = D + S[1];
for(i=1; i<r; i++)
{
 t = (B*(2B + 1)) ≪ log w;
 u = (D*(2D + 1)) ≪ log w;
 A = ((A ⊕ t) ≪ u) + S[2i];
 C = ((C ⊕ u) ≪ t) + S[2i+1];
 (A, B, C, D) = (B, C, D, A);
}
A = A + S[2r+2];
C = C + S[2r+3];
```

그림 1. RC6 암호 알고리즘
Fig 1. RC6 encryption algorithm

```
C = C - S[2r+3];
A = A - S[2r+2];
for(i=r; i>=1; i--)
{
 (A, B, C, D) = (D, A, B, C);
 u = (D*(2D + 1)) ≪ log w;
 t = (B*(2B + 1)) ≪ log w;
 C = ((C - S[2i+1]) ≫ t) ⊕ u;
 A = ((A - S[2i]) ≫ u) ⊕ t;
}
D = D - S[1];
B = B - S[0];
```

그림 2. RC6 복호 알고리즘
Fig 2. RC6 decryption algorithm

Ⅲ. 제안사항

3.1 대칭단(Symmetry layer)구조

본 논문에서 제안하는 대칭단은 바이트단위 논리 연산과 고정된 회전 연산으로 구성되어 소프트웨어 및 하드웨어 수행 속도가 빠르다. RC6 전체 진행 라운드의 반은 암호 알고리즘을 수행하고, 나머지 반은 복호 알고리즘을 수행하면서 중간에 대칭단을 삽입하여 암호/복호가 다른 RC6을 암호/복호 알고리즘이 같은 개선된 RC6을 만든다.

대칭단의 목표는 몇 가지로 설정한다.

- 암호/복호가 동일한 RC6을 만들 것.
- 동일한 라운드의 적용에서 대칭단의 삽입으로 불규칙성을 통해서 RC6의 안전성을 향상시킬 것.
- 소프트웨어 및 하드웨어 구현이 쉬울 것.
- 기존의 RC6과의 소프트웨어 및 하드웨어 수행속도에서 큰 차이가 없을 것.

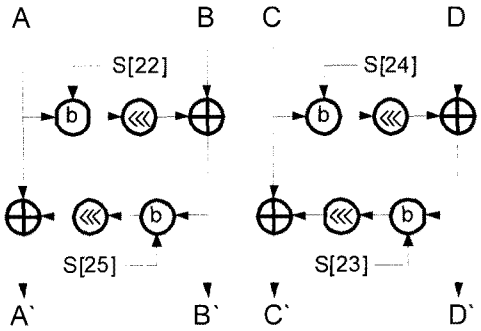

그림 3. 대칭단 흐름도
Fig 3. Symmetry layer flow diagram

그림 3은 대칭단을 그림으로 표현한 것으로 전체 128비트를 32비트 A, B, C, D로 나누고 A, B 블록과 C, D 블록으로 나누어 진행한다. 먼저 32비트 A와 라운드 키 $S[22]$를 가지고 수행한다. $b()$ 함수는 32비트 블록을 바이트 단위로 다시 나누어 and와 or 연산을 번갈아 수행한다. $b()$ 함수 수행 후 12비트 왼쪽 회전 연산을 수행하고 32비트 B와 xor 연산을 수행 후 출력 B'로 보낸다. 그리고 B'와 라운드 키 $S[25]$를 $b()$ 함수 실행 후 29비트 왼쪽 회전 연산을 수행한 다음 A와 xor 연산을 적용해서 출력 A'를 만든다. 같은 방법으로 블록 C, D를 수행한 후 출력 C', D'를 만든다.

그림 3의 대칭단 구조를 C 언어로 구현했으며, 구현한 함수는 Symmetry() 함수이다. Symmetry() 함수의 입력으로 받는 인자는 암호와 복호 과정에서 생성된 중간 결과 값인 32비트 4개의 포인터 변수 pt이고, Symmetry() 함수 내에서 다시 $b()$ 함수를 호출 한다.

$b()$ 함수는 32비트 입력 a와 라운드 키 인덱스 번호 $rkey$를 입력 받는다. $b()$ 함수는 32비트 입력을 4개의 바이트로 나누어 라운드 키와 and와 or 연산을 번갈아 수행한다. $Symmetry()$ 함수내의 $ROTL()$ 함수는 왼쪽 회전 연산을 수행하는 매크로 함수이다.

3.2 RC6에 대칭단 구현

대칭단을 RC6 알고리즘에 적용할 때 기존의 라운드 함수 내의 연산은 변경 없이 그대로 사용한다.

```
word32 b(word32 a, int rkey)
{
 union {
  word32 word;
  word8  byte[4];
 }tmp;
 tmp.word = a;
 tmp.byte[0] &= KEY.byte[rkey][0];
 tmp.byte[1] |= KEY.byte[rkey][1];
 tmp.byte[2] &= KEY.byte[rkey][2];
 tmp.byte[3] |= KEY.byte[rkey][3];
 return tmp.word;
}

word32 Symmetry(word32 *pt)
{
 word32 A, B, C, D, tmp;
 A = pt[0];
 B = pt[1];
 C = pt[2];
 D = pt[3];
 B = B ^ ROTL(tmp = b(A, 22), 12);
 D = D ^ ROTL(tmp = b(C, 24), 29);
 C = C ^ ROTL(tmp = b(D, 23), 12);
 A = A ^ ROTL(tmp = b(B, 25), 29);
}
```

그림 4 대칭단 알고리즘
Fig 4. Symmetry layer algorithm

그러나 전체 진행 라운드의 반은 암호 알고리즘을 나머지 반은 복호 알고리즘을 적용하고, 중간에 대칭단을 삽입한다.

그림 5는 제안한 알고리즘의 전체 진행과정을 그림으로 표현한 것으로 먼저 암호화 과정은 라운드 함수 진행 전에 화이트닝 단계로 라운드 키와 덧셈 연산을 수행한 후 10라운드 암호화 라운드를 수행한다. 각 라운드 연산에서 32비트 라운드 키를 2개씩 덧셈 연산에 사용한다. 다음으로 대칭 단의 적용은 3.1 대칭단 구조에서 설명한 대로 실행하며, 32비트 라운드 키를 4개 사용한다. 나머지 10라운드는 RC6의 복호 알고리즘을 적용하고 각 라운드마다 32비트 라운드 키 2개씩 뺄셈 연산을 수행 후 최종적으로 마지막 화이트닝 과정을 거친 후 128비트 암호문을 생성한다. 제안한 알고리즘의 복호는 그림 5의 과정을 그대로 수행하며, 라운드 키의 적용을 암호화 과정의 역순으로 적용한다. 그리고 화이트닝 단계에서 수행한 덧셈 연산을 뺄셈 연산으로 전환하고, 대칭단의 수행과정을 암호의 역순으로 적용한다.

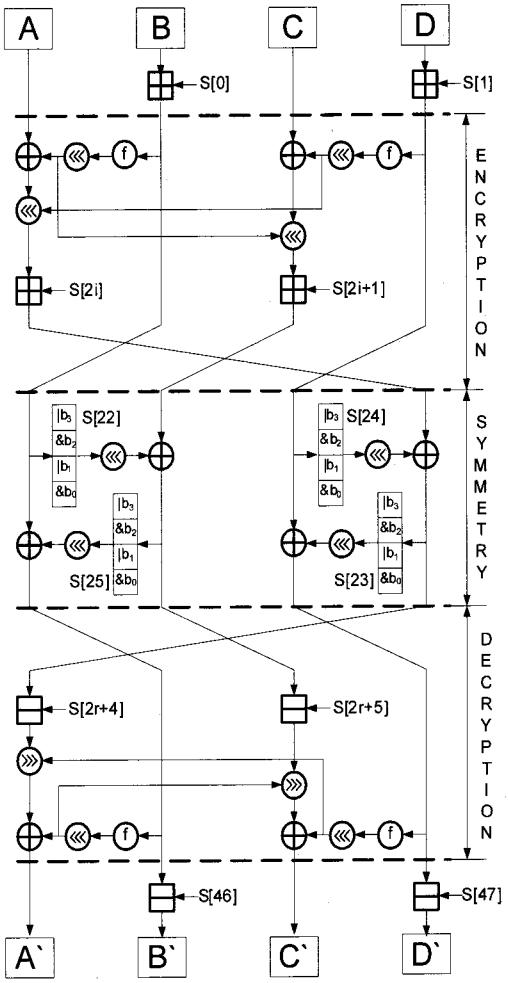

그림 5. 제안한 알고리즘 흐름도
Fig 5. Flow diagram of proposed algorithm

본 논문에서 제안한 알고리즘의 키 스케줄링은 RC6의 키 스케줄링을 그대로 사용하며, 단지 대칭 단에서 사용된 32비트 4개의 키를 더 생성해서 총 32비트 48개의 키를 사용한다.

Ⅳ. 연구결과 및 분석

4.1 수행 테스트 결과

본 논문에서 제안한 대칭단 구조를 적용한 RC6 알고리즘은 암호 운영모드는 CBC(Cipher Block Chaining) 모드를 적용하여 Visual Studio 2005 C 컴파일러를 사용하여 암호/복호가 정상적으로 수행되는 것을 확인했으며, 약 30MB 정도의 그림, 표, 특수문자 등이 있는 일반적인 한글 문서파일로 Windows XP, 셀러론 2.8Ghz, 700M RAM의 환경에서 기존의 RC6 알고리즘과 제안한 알고리즘의 수행 시간을 테스트했다. 결과는 표 1과 같으며, 제안한 알고리즘의 수행 시간이 약 3% 정도 증가하는 것으로 나타났다. 이는 대칭 단에서 적용한 간단한 논리 연산이 전체적인 암호/복호 알고리즘 수행에 거의 영향을 미치는 않는 것으로 판단된다.

표 1. 수행시간 테스트 결과(단위: 초)
Table 1. Test results of running time

시간 / 알고리즘	암호	복호
RC6	10.171 (100.00%)	10.156 (100.00%)
제안한 알고리즘	10.531 (103.54%)	10.438 (102.78%)

4.2 안전성 검증

RC6의 안전성 평가는 [8]에 잘 나타나 있으며 본 논문에서는 차분분석(Differential Cryptanalysis)[9], 선형분석(Linear Cryptanalysis)[10], 대칭단, 다른 공격 방법으로 안전성을 평가해서 제안한 알고리즘의 안전성을 평가한다.

그리고 다음과 같은 표기 방법을 이후 계속해서 사용한다. $x[i]$는 특정 입력으로 32비트 x의 i번 째 비트를 나타내며, 출력 $y[i]$, 라운드 키 $S[i]$도 $x[i]$와 같이 표기한다. x, y, S만 표시할 때는 32비트 워드이다. 그리고 x_0, x_1, x_2, x_3은 RC6의 32비트 4개의 입력이고, Δx는 입력 차분이고, Δy는 출력 차분이다. 마지막으로 e_i는 32비트 워드 e에서 i번째 비트만 1이고 나머지는 모두 0인 32비트 값이다.

4.2.1 차분 및 선형 분석에 의한 안전성 평가

첫 째로 RC6에서 사용된 모든 연산과 차분분석과의 관계를 알아보겠다. 먼저 화이트닝 단계의 라운드 키와 덧셈연산의 차분값의 차이를 생각해보면, 입력 x와 x'가 아주 조금의 차이가 있으면 $x+S$와 $x'+S$의 차이도 아주 조금만 발생한다. 왜냐하면 입력의 차이가 x와 x'만 있고 라운드 키와의 덧셈은 같은 값을 더하므로 라운드 키는 상수로서의 역할을 하게 된다. 특히 x와 x'가 MSB(most significant bit)만 다를 경우 $x+S$와 $x'+S$ 또한 MSB만 다르다. 같은 방법으로 $x[i]$와 $x'[i]$가 다를 경우($i<31$) $x[i]+S$와 $x'[i]+S$가 다를 확률은 1/2이다.

$$Z[i]=x[i]+S[i]+w[i-1],\ \ 0\le i\le 31$$
$$w[i]=x[i]S[i]+x[i]w[i-1]+S[i]w[i-1] \qquad (1)$$

$Z[i]$는 입력과 라운드 키와의 합(sum)이고, $w[i]$는 캐리(carry)이다. 특히 $w[-1]$는 0이다. 수식 (1)을 사용하여 $x+S$와 $x'+S$가 정확히 연속적으로 2비트 다를 확률은 1/4이다. 2개의 32비트 워드 x와 x'가 연속적으로 2비트 다를 경우 $x+S$와 $x'+S$가 1비트 다를 확률은 1/4이고, 연속적일 필요는 없지만 2비트 다를 확률은 3/8이다. 그래서 x와 x'가 연속적으로 2비트 다르면, $x+S$와 $x'+S$가 2비트 다를 확률은 5/8가 된다. 수식 (1)을 사용해서 좀 더 다양한 차분값을 설정 한 후 라운드 키와의 덧셈과의 관계를 설명할 수 있다.

다음은 데이터의존 회전 연산과 차분분석과의 관계를 알아보겠다. x와 x'가 j번째 비트만 다를 경우 두 워드가 같은 수의 회전 연산을 수행한 결과는 정확히 j번째 비트만 다르다. 만약 두 워드가 다른 수의 회전 연산 수행 결과로서 생긴 차분값은 예측하기 매우 어렵다. 이는 두 워드가 반드시 같은 값의 회전 연산을 수행하는 조건에서만 유용한 차분분석이 가능하다는 것을 의미한다.

다음은 2차 함수 $f(x)=x(2x+1)$의 차분분석을 생각해 보자. 확률이 1인 2차 함수 $f(x)=x(2x+1)$의 미세한 차이는 쉽게 계산이 된다. 다시 말해 32비트 입력 x_0과 x_1이 MSB만 다를 경우 $f(x_0)$과 $f(x_1)$ 역시 MSB만 다르다. 이와 같은 경우가 확률 1이 된다. 다음은 x_0과 x_1이 두 번째 MSB만 다를 경우 $f(x_0)$과 $f(x_1)$은 MSB와 두 번째 MSB만 다르게 된다. 이번에는 3 번째 MSB만

다를 경우 $f(x)$함수 수행 후 MSB, 두 번째 MSB, 세 번째 MSB가 모두 다르게 될 것이다. 같은 방법으로 두 개의 워드가 i번째 LSB(least significant bit)를 같게 유지하고 $f(x)$함수를 수행하면 적어도 i번째 LSB는 같게 유지된다. 이와 같이 높은 확률을 가지는 차분 패스(path)를 유지하면서 RC6 내의 모든 라운드 함수에 적용할 수 있다. 그러나 문제는 높은 차분 확률의 패스를 찾는 것이 가능하냐가 문제이다.

그리고 RC6의 개발자들은 차분분석을 xor 연산이 아니라 뺄셈 연산으로 분석하였다. 다시 말해 입력 x_0과 x_1의 차분값은 '$x_0 - x_1 \bmod 2^{32}$'가 된다. 먼저 두 워드 x_0과 x_1의 차이가 α인 경우 2차 함수 $f(x)$에 적용해 보면

$$\begin{aligned} f(x_0)-f(x_1) &= x_0(2x_0+1)-x_1(2x_1+1) \\ &= (x_1+\alpha)(2(x_1+\alpha)+1)-x_1(2x_1+1) \\ &= 4\alpha x_1+2\alpha^2+\alpha \end{aligned} \quad (2)$$

식 (2)의 결과로 α와 x_1 값에 의존적인 것을 알 수 있다. 이와 같은 방법을 1차 차분분석이라 하고 유사한 방법으로 4개의 워드 입력을 적용하는 2차 차분분석을 적용해 보면 다음과 같다. x_0과 x_1, x_2와 x_3의 차이는 α이고, x_1과 x_3의 차이가 β인 경우 이 4개의 입력 워드를 $f(x)$ 함수에 적용해 보면

$$\begin{aligned} & f(x_0)-f(x_1)-(f(x_2)-f(x_3)) \\ &= 4\alpha x_1+2\alpha^2+\alpha-(4\alpha x_3+2\alpha^2+\alpha) \\ &= 4\alpha(x_1-x_3) \\ &= 4\alpha\beta \end{aligned} \quad (3)$$

식 (3)의 결과는 입력 워드와 관계없이 입력 워드 간의 차이값 만으로 유용한 차분분석을 할 수 있음을 보이는 것이다. 그러나 이와 같은 2차 차분분석으로 같은 수의 회전 연산을 구성할 수 있는 패스를 찾기는 매우 어려우며, RC6 개발자들이 분석한 경험을 바탕으로 본 논문에서 6라운드, 2라운드 비반복(non-iterative) 차분(Differential)을 이용하여 8라운드 차분특성을 구성한 10라운드 차분분석을 했다.

그림 6에서 B_6은 시작 6라운드로서 초기 값으로 첫 번째 32비트 워드의 차이는 MSB만 1로 하고 두 번째 워드의 차이는 26번째 비트만 1로 셋팅 후 6라운드를 진행한다. 32비트 값이 0인 것은 차이가 없는 것을 뜻한다. 그 다음 E_2는 마지막 2라운드 진행으로 B_6의 6라운드를 그대로 이어 받아 진행 하게 된다. 10라운드 전체 확률은 다음과 같이 계산된다.

$$\begin{aligned} & \rho^6 * q_t * P_s * P_v * P_u * q_{u-5} * q_{u-10} * \rho^1 * q_{t-10} \\ &= \rho^6 * q_{26} * P_{26} * P_{26} * P_{26} * q_{21} * q_{16} * \rho^1 * q_{16} \\ &\approx 2^{-30} * 2^{26-34} * 2^{-4} * 2^{-4} * 2^{26-31} * \\ &\quad 2^{21-31} * 2^{16-31} * 2^{-5} * 2^{16-34} \\ &\approx 2^{-30} * 2^{-8} * 2^{-4} * 2^{-4} * 2^{-5} * \\ &\quad 2^{-10} * 2^{-15} * 2^{-5} * 2^{-18} \\ &\approx 2^{-99} \end{aligned} \quad (4)$$

식 (4)에서 $t = s = v = u = 26$이고, P_s, P_v는 참고논문[8]의 보조정리 6에 의해 계산된 것이며, q_t와 P_u는 보조정리 8에 잘 정리 되어 있고, ρ^6은 RC6의 회전 연산은 LSB의 5비트까지만 유효하므로 (2^{-5})이 여섯 번 수행하므로 $(2^{-5})^6 = 2^{-30}$이 된다.

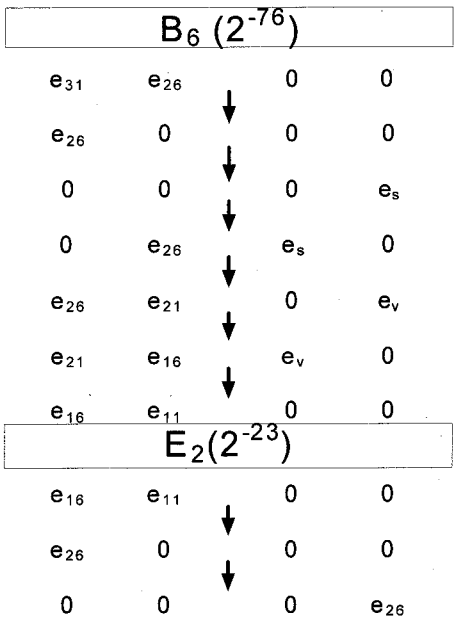

그림 6. 10라운드 차분분석
Fig 6. Ten round of differential cryptanalysis

두 번째는 선형분석이다. 먼저 입력 x가 2차 함수 $f(x)=x(2x+1)$을 수행 한 후 LSB는 항상 같아지는 확률 1의 결과를 만들어 낸다. 이는 MSB가 중요한 차분 분석과 유사하며, 간단한 2라운드 반복(iterative) 선형 근사값(Approximation)으로 $r-2$라운드 선형 분석을 할 수 있다. 물론 차분분석과 마찬가지로 높은 확률 패스를 찾는 것이 쉽지 않다. 그리고 선형분석은 크게 2가지 방법이 있다. 회전 연산의 근사값을 구하는 식 '$A=B\ll C$'에서 근사값을 구하는데 32비트 A, B, C의 각각의 1비트만으로 구하는 방법과 A, B의 1비트만으로 구하는 방법이다. 이 중 A, B의 1비트만으로 구하는 방식이 좀 더 유용하다.

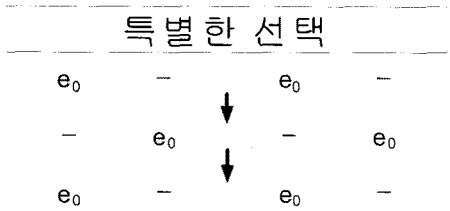

그림 7. 2라운드 반복 선형분석
Fig 7. Two round iterative of linear cryptanalysis

그림 7은 2라운드 반복적 선형분석으로 본 논문에서는 2라운드 반복적 분석을 5번 적용해서 10라운드 선형분석을 했다. 선형분석 과정은 첫 번째 세 번째 워드에서 LSB만 취하여 2라운드를 진행하면 다음과 같은 식을 얻을 수 있다.

$$a_u * \rho * a_v * \rho * 2^3 \tag{5}$$

식 (5)에서 a_u, a_v는 u와 v 위치(position)에서 덧셈 연산의 단일 선형 근사값의 편차(bias)이고, ρ는 회전 연산의 선형 근사값의 편차이다. 위 식 (5)를 참고문헌[8]의 보조정리 10을 적용하여 계산한 결과는 2^{-11}이다. 이 결과를 10라운드에 적용하는 식은 다음과 같다.

$$(2^{-11})^{r/2} * 2^{(r/2)-1} \tag{6}$$

식 (6)에서 r은 라운드 수이다. 최종적으로 10라운드의 결과는 2^{-51}이다.

4.2.2 대칭단 안전성 평가

대칭단 내에는 라운드 키와 and와 or 연산, 그리고 고정된 길이의 왼쪽 회전 연산이 있다. 라운드 키와 and와 or 연산의 차분 및 선형 분석의 영향을 표 2, 표 3에 나타냈다.

표 2에서 $S[i]=0$일 경우 and 연산으로 입력 차분 $\Delta x[i]=1$은 출력 차분 $\Delta y[i]=0$이 되므로 차분분석에 유용하며, $S[i]=1$일 경우 or 연산으로 입력 차분 $\Delta x[i]=1$은 출력 차분 $\Delta y[i]=0$이 되므로 차분분석을 할 수 있다. 표 3은 $S[i]=0$일 때 입력 $x[i]$와 and 연산은 출력 $y[i]$는 무조건 0이 되므로 선형분석에 영향이 있으며, $S[i]=1$일 때 입력 $x[i]$와 or 연산은 출력 $y[i]$는 무조건 1이 되므로 선형분석을 할 수 있다.

표 2. and 와 or 의 차분분석
Table 2. AND / OR of differential cryptanalysis

and	$S[i]$	$\Delta x[i]$, $\Delta y[i]$	영향
	0	$\Delta x[i]=1 \rightarrow \Delta y[i]=0$	yes
	1	$\Delta x[i]=1 \rightarrow \Delta y[i]=1$	no
or	$S[i]$	$\Delta x[i]$, $\Delta y[i]$	영 향
	0	$\Delta x[i]=1 \rightarrow \Delta y[i]=1$	no
	1	$\Delta x[i]=1 \rightarrow \Delta y[i]=0$	yes

표 3. and 와 or 의 선형분석
Table 3. AND / OR of linear cryptanalysis

and	$S[i]$	$x[i]$, $y[i]$ 의 값	영 향
	0	$x[i]$와 독립으로 $y[i]=0$	yes
	1	$y[i] = x[i]$	no
or	$S[i]$	$x[i]$, $y[i]$ 의 값	영 향
	0	$y[i] = x[i]$	no
	1	$x[i]$와 독립으로 $y[i]=1$	yes

대칭단에서 사용된 라운드 키 $S[i]$는 1/2의 확률을 가진다. 입력차분 $\Delta x[i]$나 입력 값 $x[i]$는 $S[i]$와 and와 or 연산을 통해 출력차분 $\Delta y[i]$나 출력 값 $y[i]$를 예측할 수 없는 방해가 일어난다. 특히 Δx의 Hamming weight가 h일 경우 $\Delta y=0$일 때 적용된 라운드 키의 확률은 2^{-h}이다. 예를 들어 32비트 길이에서 $\Delta x \neq 0$ 이며, $\Delta y=0$일 때 사용된 라운드 키를 얻을 수 있을 확률은

$\frac{1}{2^{32}}\sum_{i=1}^{32}\binom{32}{i}2^{-i}$ 이다.

제안한 알고리즘의 차분, 선형분석은 RC6 암호 알고리즘을 적용한 10라운드에서 확률이 높은 차분, 선형 패스가 대칭단에서 라운드 키와의 논리 연산의 적용 후 변화 또는 단절이 일어나 대칭단 이후 유효한 차분, 선형 패스의 구성을 어렵게 하고 있다.

4.2.3 다른 공격방법에 의한 안전성 평가

Square[11] 공격과 같은 바이트 패턴이 각각의 라운드 사이에서 전파되는 특성을 이용한 공격도 대칭단의 논리 연산과 회전 연산을 통해 바이트 단절이 일어나 Square 공격에도 내성이 있다.

부정차분분석(Truncated Differentials Cryptanalysis)[12] 바이트 단위로 연산이 적용되는 암호에서 바이트 패턴이 서로 다른 경우 1, 같은 경우 0으로 정의된 차이값을 가지고 분석하는 것으로 입력차분과 출력차분의 전부를 고려해야하는 차분분석보다 쉽고 정확하게 확률을 계산할 수 있다. 본 논문에서 제안한 대칭단은 RC6의 암호/복호 알고리즘이 반반씩 적용되는 중간에 삽입이 되며, RC6의 암호/복호 라운드와는 다른 독립적인 불규칙라운드이다. 대칭단이 전체 라운드 중간에 삽입 될 때 암호 전체의 안전성을 나타내주는 확률뿐만 아니라 차분 패스도 변하게 된다. 이는 대칭단에서 적용되는 라운드 키와 논리연산이 차분 패스 분석을 더욱 어렵게 하고 있으며, 효과적인 차분 패스를 찾기 위한 대칭단에서 사용된 라운드 키를 추론하는 것은 그 라운드 키에 의존한다는 것을 의미한다. 결론적으로 대칭단에서 사용하고 있는 3가지 논리연산이 위와 같은 효과로 인해 대칭 단 이후의 유용한 부정차분분석의 구성을 어렵게 하고 있다.

V. 결 론

본 논문에서 RC6의 암호와 복호 알고리즘이 다른 것을 간단한 논리 연산과 고정된 회전 연산만으로 이루어진 대칭단을 삽입하여 RC6의 암호와 복호를 같게 개선했다. 즉 RC6의 10라운드는 암호 알고리즘을, 나머지 10라운드는 복호 알고리즘을 사용하고 중간에 대칭단을 삽입해서 암호와 복호를 같게 구현했다. 제안한 알고리즘을 기존의 RC6과 수행 속도를 비교해서 별 차이가 없었으며, 안전성에서도 대칭 단의 적용이 암호 알고리즘을 분석하는데 유효한 차분 및 선형 패스를 단절 또는 변화를 통해 방해하여 어렵게 하고 있다. 따라서 대칭단을 포함한 전체 20라운드의 안전성 분석은 유효한 차분, 선형 패스를 찾기가 어려우므로 보다 깊은 연구가 필요하며, 이는 다음 연구과제로 남긴다.

제안한 알고리즘은 제한적인 하드웨어 환경인 스마트카드나 전자 칩이 내장된 RFID의 능동형 태그 등에 구현 시 하드웨어의 면적을 상당 부분 줄일 수 있다. 그리고 기존의 블록 암호 알고리즘 중 암호와 복호가 다른 알고리즘도 간단한 대칭단의 삽입으로 암호와 복호를 같게 할 수 있으며, 새로운 블록 암호 알고리즘의 설계에도 좋은 아이디어로 사용할 수 있다.

참고문헌

[1] "Report on the Development of the Advanced Encryption Standard(AES)," http://www.csrc.nist.gov/encryption/aes/.

[2] "New European Schemes for Signatures. Integrity. and Encryption(NESSIE)," http://cryptonessie.org/.

[3] "Cryptography Research and Evaluation Committees (CRYPTREC)," http://www.cryptrec.go.jp/.

[4] SEED, http://www.kisa.or.kr/seed/.

[5] ARIA, http://www.nsri.re.kr/ARIA/.

[6] H. Feistel, "Cryptography and Computer Privacy," Scientific American, Vol.228, No.5, pp. 15-23, 1973.

[7] C. E. Shannon, "Communication theory of secrecy system," *Bell System Technical Journal*, Vol.28, No.4, pp. 656-715, 1949.

[8] S. Contini, R.L. Rivest, M.J.B. Robshaw, and Y.L. Yin, "The security of RC6," http://www.rsasecurity.com/rsalabs/aes.

[9] E. Biham and A. Shamir, "Differential cryptanalysis for DES-like cryptosystem," Journal of Cryptlogy 4(1): 3-17, 1991.

[10] M. Matsui, "Linear cryptanalysis method for DES cipher," In Tor Helleseth, editor, Advances in Cryptology-Eurocrypt '93, LNCS Vol.765, pp. 386-397, 1994.

[11] J. Daemen, L. Knudsan, and V. Rijmen, "The block cipher square," Proceeding of FSE'97 LNCS Vol.1267, pp. 149-165, 1997.

[12] L. R. Knudsen, "Truncated and higher order differential," Fast Software Encryption-Second International Workshop, LNCS Vol.1008, pp. 196-211, 1995.

저자소개

김길호(Gil-Ho Kim)

2000년 한국방송통신대학교 전자계산학과 학사
2002년 부경대학교 컴퓨터공학과 석사
2007년 부경대학교 컴퓨터공 학과 박사수료
※관심분야 : 반도체회로설계, 암호 알고리즘, 컴퓨터 구조

박창수(Chang-Soo Park)

1995년 인제대학교 전자공학과 학사
2001년 부경대학교 컴퓨터공학과 석사
2007년 부경대학교 컴퓨터공학과 박사
2008년 - 현재 부경대학교 누리사업단 계약교수
※관심분야 : 반도체회로설계, 암호 알고리즘, 컴퓨터 구조

김종남(Jong-Nam Kim)

1995년 2월 금오공과대학교 전자공학과 학사
1997년 2월 광주과학기술원 정보통신공학과 석사
2001년 8월 광주과학기술원 기전공학과 박사
2001년 8월 - 2004년 2월 KBS 기술연구소 선임연구원
2004년 4월 - 현재 부경대학교 전자컴퓨터정보통신 공학부 교수
2003년 3월 - 현재 (주)홈캐스트 사외이사
※관심분야 : 영상신호처리, 멀티미디어 보안 등

조경연(Gyeong-Yeon Cho)

1990년 인하대학교 공과대학 전자 공학과 정보공학전공 박사
1983-1991년 삼보컴퓨터 기술연구소 책임연구원
1991년 - 현재 부경대학교 공과대학 전자컴퓨터정보통신공학부 교수
1991-2001년 삼보컴퓨터 기술연구소 비상임기술고문
1998년 - 현재 에이디칩스 사외이사 겸 비상임기술 고문
※관심분야 : 전산기구조, 반도체회로설계, 암호 알고리즘