論文
10-4-8

대칭구조 SHACAL-1 블록 암호 알고리즘
(Symmetric structured SHACAL-1 block cipher algorithm)

김길호[1], 박창수[2]
김종남[3], 조경연[4]

요 약

본 논문에서는 간단한 대칭단을 사용하여 암호와 복호를 동일하게 개선한 SHACAL-1을 제안한다. SHACAL-1의 한 라운드는 20스텝으로 되어 있고 모두 4라운드로 구성되어 있으며 복호는 암호의 역함수로 되어있다. 본 논문에서 제안하는 대칭구조 SHACAL-1은 전반부, 대칭단 그리고 후반부의 3개 부분으로 구성한다. 전반부는 SHACAL-1의 암호 알고리즘으로 한 라운드는 10스텝으로 하여 4라운드로 구성한다. 후반부는 SHACAL-1의 복호 알고리즘으로 동일하게 구조를 가진다. 그 중간에 대칭단을 삽입하여 암호와 복호 알고리즘을 동일하게 구성한다. 제안한 대칭구조 SHACAL-1은 SHACAL-1과 수행시간 테스트 결과 거의 영향을 미치지 않은 결과를 보였고, 안전성 또한 대칭단의 적용으로 선형, 차분분석과 같은 높은 확률 패스를 이용한 공격에도 분석을 어렵게 하고 있다. 제안한 알고리즘은 암호와 복호가 다른 블록 암호에도 쉽게 적용 가능하며, 다른 블록 암호 설계에도 유용하다.

ABSTRACT

In this paper, we propose an improved SHACAL-1 of the same encryption and decryption with a simple symmetric layer. SHACAL-1 has 4 rounds, and each round has 20 steps. Decryption is becoming inverse function of encryption. In this paper, we proposed SHACAL-1 are composed of the first half, symmetry layer and the last half. The first half with SHACAL-1 encryption algorithm 1 round does with 10 steps and composes of 4 round. The last half identically with SHACAL-1 decryption algorithm, has a structure. On the center inserts a symmetry layer, encryption and decryption algorithm identically, composes. In the experiments, the proposed SHACAL-1 algorithm showed similar execution time to that of the SHACAL-1. Thanks to the symmetric layer, the proposed algorithm makes it difficult for the attacks which take advantages of high probability path such as the linear cryptanalysis, differential cryptanalysis. The proposed algorithm can be applicable to the other block cipher algorithms which have different encryption and decryption and useful for designing a new block cipher algorithm.

Key words: SHACAL-1, Feistel, Symmetric layer(대칭 단), CBC(Cipher Block Chaining)

1) 부경대학교 컴퓨터공학과 박사수료(정회원)
2) 부경대학교 컴퓨터공학과 박사
3) 부경대학교 컴퓨터공학과 교수
4) 부경대학교 컴퓨터공학과 교수(교신저자)

논문접수 : 2009. 08. 29.
심사완료 : 2009. 09. 22.

I. 서론

통신 기술의 발전과 사회의 전반적인 활동이 무선 통신망과 인터넷과 같은 범용 통신망을 이용한 지식기반 정보화 사회로 빠르게 진행함에 따라 정보보호에 대한 인식이 점차 높아지고 있으며, 정보보호를 위한 기술적 대응 조치로 암호화 기술이 일반적이다. 그리고 그동안 표준 암호로 널리 사용되어 온 DES(Data Encryption Standard)[1]의 안전성에 문제가 있어 미국을 비롯한 유럽 및 선진국들은 자국의 표준 블록 암호 개발에 주력하고 있다. 특히 미국의 경우 2000년에 AES(Advanced Encryption Standard)[2] 프로젝트를 통한 Rijndael[3] 알고리즘을 표준 블록 암호 알고리즘으로 선정하였으며, 2002년 유럽은 NESSIE(New European Schemes for Signatures. Integrity. and Encryption)[4] 프로젝트를 통해 여러 가지 암호 기반 기술의 표준을 선정하였다. 그리고 우리나라의 경우 자체 개발한 SEED[5]와 ARIA[6]를 국가 표준으로 제정하였다.

블록 암호를 설계하는 방식에는 Feistel[7] 구조와 SPN(Substitution Permutation Network)[8] 구조가 있으며, Feistel 구조는 암호와 복호 알고리즘이 동일한 것이 장점이고, SPN 구조는 암호와 복호 알고리즘이 서로 다른 특징이 있다. 특히 하드웨어로 블록 암호를 구현할 때 SPN 구조나 암호와 복호 알고리즘이 다른 경우에는 암호와 복호 알고리즘이 같은 경우보다 하드웨어 면적이 2배 정도 증가하게 되는 단점이 된다. 그리고 초기 블록 암호 개발은 주로 소프트웨어로 구현되었지만 현재는 빠른 정보처리를 위한 하드웨어 구현에도 많은 연구를 같이 병행하고 있다[2][4].

SHACAL-1은 국제 표준 해시 알고리즘인 SHA-1[9]의 압축 함수에 기반을 두었으며, 유럽의 NESSIE 프로젝트에 0~512비트 가변 키를 지원하는 160비트 블록 암호 알고리즘으로 제안되었지만, 키 스케줄링의 약점에 대한 연관 키 분석(Related Key Cryptanalysis)[10]으로 최종적으로 선정되지는 않았다. SHACAL-1은 간단한 논리연산과 덧셈에 대한 모드 연산만으로 구성된 4 라운드로 나누어진 총 80스텝(step)으로 구성된 암호와 복호가 다른 변형된 Feistel 구조이다. 각 라운드는 20스텝으로 구성되어 있다.

본 논문에서는 간단한 대칭단(Symmetric layer)을 사용하여 암호와 복호를 동일하게 개선한 대칭구조 SHACAL-1을 제안한다. 대칭구조 SHACAL-1은 한 라운드마다 10스텝으로 4 라운드를 적용하여 총 40스텝은 SHACAL-1의 암호 알고리즘을, 그리고 한 라운드마다 10스텝을 적용한 총 4라운드 나머지 40스텝은 SHACAL-1의 복호 알고리즘을 적용하고, 그 중간에 대칭단을 삽입함으로서 암호와 복호 알고리즘을 동일하게 개선했다.

제안한 대칭구조 SHACAL-1은 기존의 SHACAL-1과 수행시간 테스트 결과 거의 영향을 미치지 않은 결과를 보였고, 안전성 또한 대칭단의 적용으로 선형분석(Linear Cryptanalysis)[11], 차분분석(Differential Cryptanalysis)[12]과 같은 높은 확률 패스를 이용한 공격에도 분석을 어렵게 하고 있다. 그리고 연관 키 공격에 약한 SHACAL-1의 키 스케줄링 알고리즘을 수정하여 연관 키 공격에도 강하게 했다. 제안한 알고리즘은 암호와 복호가 다른 블록 암호에도 쉽게 적용 가능하며, 다른 블록 암호 설계에도 유용하다.

또한 암호 시스템 전체를 구성하는 경우 해시 함수가 필수적으로 포함되어야 한다. 제안한 대칭구조 SHACAL-1은 SHA-1해시 함수를 기반으로 설계된 블록 암호 알고리즘으로 하드웨어로 암호 시스템을 구현할 때 약간의 회로를 추가하여 블록 암호 시스템을 구현할 수 있으며, 이는 다른 블록 암호 알고리즘을 사용한 암

호 시스템에 비해 하드웨어를 간소화 할 수 있는 장점이 된다.

본 논문의 구성은 2장에서 SHACAL-1 알고리즘을 좀 더 상세히 설명하고, 3장에서는 대칭단에 대한 자세한 설명과 4장에서 제안한 대칭구조 SHACAL-1의 수행결과 분석 및 안전성을 검증하고 5장에서 결론으로 끝맺는다.

II. SHACAL-1

SHACAL-1은 160비트 블록 암호 알고리즘으로 사용된 연산자는 다음과 같다.

(가) ▪ + : 정수 덧셈 mod 2^{32}
(나) ▪ - : 정수 뺄셈 mod 2^{32}
(다) ▪ $ROT_i(X)$: X에 대해 i만큼 왼쪽회전연산
(라) ▪ ⊕ : 비트별 xor연산
(마) ▪ & : 비트별 and연산
(바) ▪ | : 비트별 or연산
(사) ▪ ~ : 비트별 not연산
(아)
(자) 초기 160비트 평문 $X(= X_1 \,||\, X_2 \,||\, X_3 \,||\, X_4 \,||\, X_5)$는 32비트 단위로 나누어진 5개의 임시저장소인 A_0, B_0, C_0, D_0, E_0에 각각 입력된다. A_0, B_0, C_0, D_0, E_0을 총 80스텝을 수행한 후 암호문 A_{80}, B_{80}, C_{80}, D_{80}, E_{80}을 만든다. 각 라운드의 암호화 과정은 다음과 같다. 여기서 i는 스텝을 말한다.
(차)

(타)
(파) $A_i = K_i + ROTL_5(A_{i-1}) + f(B_{i-1},C_{i-1},D_{i-1}) + E_{i-1}$
(하) $\quad + z_i$
(거) $B_i = A_{i-1}$
(너) $C_i = ROT_{30}(B_{i-1})$
(더) $D_i = C_{i-1}$
(러) $E_i = D_{i-1}, \quad (0 \le i \le 79)$
(머)

(카)
(버) 위 과정에서 K_i는 32비트 라운드 키이고, 라운드 함수인 f()함수와 라운드 상수 z_i는 다음과 같다.
(서)

$f_i(B,C,D) = (B\&C)|(\sim B\&D), \quad (0 \le i \le 19)$
$f_i(B,C,D) = B \oplus C \oplus D,$
$(20 \le i \le 39,\ 60 \le i \le 79)$
$f_i(B,C,D) = (B\&C)|(B\&D)|(C\&D), \quad (40 \le i \le 59)$

(어)
(저) 라운드 함수 f()는 각각 $f_{if}()$, $f_{xor}()$, $f_{maj}()$로 표시한다.
(처)

(타)
(퍼) $z_i = 0x5a82799, \quad (0 \le i \le 19)$
(허) $z_i = 0x6ed9eba1, \quad (20 \le i \le 39)$
(고) $z_i = 0x8f1bbcdc, \quad (40 \le i \le 59)$
(노) $z_i = 0xca62c1d6, \quad (60 \le i \le 79)$
(도)

(커)
SHACAL-1의 라운드 키를 생성하는 키 스케줄링 알고리즘은 다음과 같다.

$K = K_0 \,||\, K_1 \,||\, \ldots \,||\, K_{15}$

$K_i = ROT_1(K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}),$
$\quad (16 \le i \le 79)$

SHACAL-1은 20스텝 단위로 서로 다른 f() 함수와 라운드 상수 z_i를 수행하는 특성이 있고, 라운드마다 32비트 라운드 키를 한 개씩 사용한다. SHACAL-1의 복호화 과정은 위의 라운드 함수 각각의 역함수가 존재하며 덧셈연산은 뺄셈연산으로 수행해야하며, 라운드 키의 적용은 암호화의 역순으로 수행한다. SHACAL-1은 비선형 변환인 S-box가 없으며, 라운드 마다 다른 간단한 논리연산으로 구성된 비선형 변환 f() 함수를 사용하는 특징을 가지고 있다[9].

III. 대칭 단 구조

3.1 대칭단 구조

본 논문의 제안 사항인 대칭단은 간단한 논리 연산과 자리바꿈으로 구성 되어 있고, 대칭단의 목표는 다음과 같다.

(로) ▪ 암호와 복호가 동일한 SHACAL-1을 만들 것.

(모) ▪ 대칭단의 삽입으로 불규칙성을 통해서 SHACAL-1 의 안전성을 향상시킬 것.

(보) ▪ 소프트웨어 및 하드웨어 구현이 쉬울 것.

(소) ▪ 기존의 SHACAL-1과의 소프트웨어 및 하드웨어 수행속도에서 큰 차이가 없을 것.

(오)

(조)

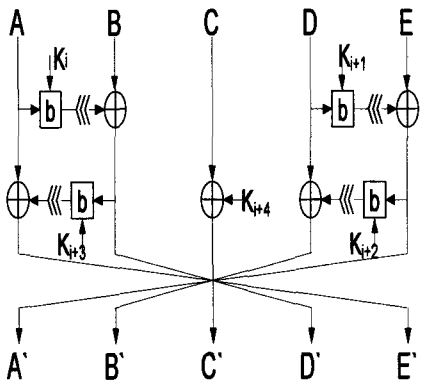

그림 1. 대칭단 흐름도

그림 1.은 대칭단의 진행과정을 그림으로 표현한 것으로 전체 160비트를 32비트 A, B, C, D, E로 나누고 A, B 블록과 D, E 블록으로 나누어 진행한다. 먼저 32비트 A와 라운드 키 K_i를 가지고 b()함수를 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 D'로 보낸다. 그리고 D'와 라운드 키 K_{i+3}을 b()함수 실행 후 29비트 왼쪽 회전 연산을 수행한 다음 A와 xor 연산을 적용해서 출력 B'를 만든다. 유사한 방법으로 블록 D, E를 수행한 후 출력 A', E'를 만든다. 마지막으로 중간에 있는 32비트 C는 라운드 키 K_{i+4}와 xor 연산을 수행 한 후 C'로 보낸다.

그림 1.의 대칭 단 구조를 C 언어로 구현했으며, 구현한 함수는 symmetry() 함수이다. symmetry() 함수의 입력으로 받는 인자는 a, key, c0, c1이 있다. a는 암호와 복호 과정에서 생성된 중간 결과 32비트 포인터 변수이고, key는 키 스케줄링 알고리즘에 의해 생성된 라운드 키의 인덱스 번호이며, c0은 암호일 경우는 0이고 복호일 경우는 2로 셋팅 되고, c1은 암호일 경우는 2이고 복호일 경우는 0으로 셋팅 된다. symmetry() 함수는 다시 32비트 또는 8비트 단위로 처리를 위해 b() 함수를 호출하고, b() 함수내부처리는 union 구조를 하고 있다. symmetry() 함수내의 지역변수로 word32 temp, rt는 논리연산을 수행 후 고정된 회전 연산을 위한 32비트 임시 변수이다. ROTL() 함수는 데이터의존 왼쪽 회전(Data Dependent Rotation) 연산을 수행하는 매크로 함수이다. symmetry() 함수와 b() 함수의 소스는 아래와 같다.

```
word32 b(word32 a, word32 rkey)
{
  union
  {
    word32 word;
    word8  byte[4];
  }tmp, s;
```

```
  tmp.word = a;
  s.word = rkey;

  tmp.byte[0] &= s.byte[0];
  tmp.byte[1] |= s.byte[1];
  tmp.byte[2] &= s.byte[2];
  tmp.byte[3] |= s.byte[3];

  return tmp.word;
}

void symmetry(word32 *a, int key, int c0,
int c1)
{
  word32 temp[2], rt;

  temp[0] = a[1] ^ ROTL(rt = b(a[0],
ss[key + c0]),
           12);
  temp[1] = a[4] ^ ROTL(rt = b(a[3],
ss[key + c0 + 1]),
           29);
  a[1] = a[3] ^ ROTL(rt = b(temp[1],
ss[key + c1]),
         12);
  a[4] = a[0] ^ ROTL(rt = b(temp[0],
ss[key + c1 + 1]),
         29);
  a[3] = temp[0];
  a[0] = temp[1];
  a[2] ^= ss[key + 4];
}
```

3.2 SHACAL-1에 대칭단 구현

대칭단을 SHACAL-1 알고리즘에 적용할 때 기존의 4개의 라운드 함수 내의 연산은 변경 없이 그대로 사용한다. 그러나 각 라운드마다 20스텝을 10스텝으로 줄여서 4라운드를 SHACAL-1의 암호 알고리즘을 적용하고, 3.1 대칭단 구조에서 설명한 대칭단을 적용한 다음 10스텝씩 4라운드를 SHACAL-1의 복호 알고리즘을 적용한다.

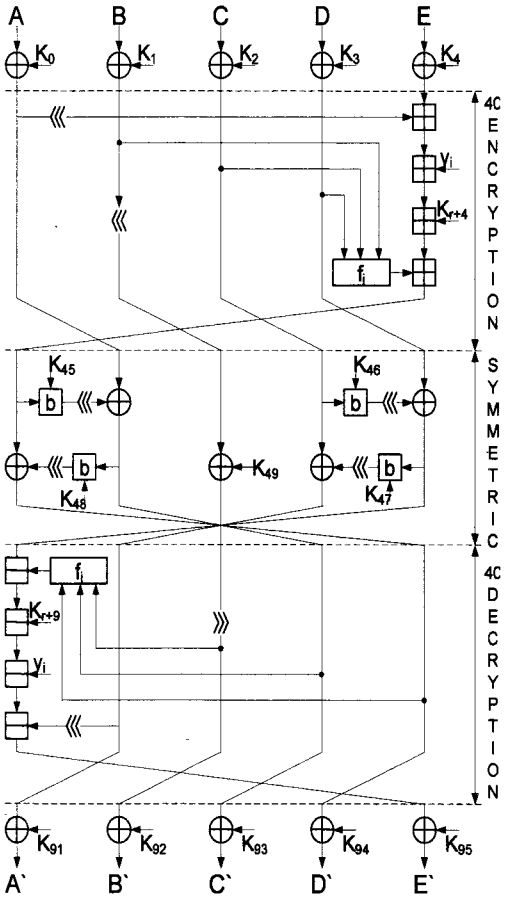

그림 2. 제안한 알고리즘의 전체 진행과정

그림 2.는 제안한 알고리즘의 전체 진행과정을 그림으로 표현한 것으로 먼저 암호화 과정은 라운드 함수 진행 전에 화이트닝(whitening) 단계로 라운드 키와 xor 연산을 수행한 후 4라운드 40스텝 SHACAL-1암호화 라운드를 수행한다. 각 라운드 연산에서 32비트 라운드 키를 1개씩 덧셈 연산에 사용한다.

다음으로 대칭 단의 적용은 3.1 대칭단 구조에서 설명한 대로 실행하며, 32비트 라운드 키를 5개 사용한다. 나머지 4라운드 40스텝은 SHACAL-1의 복호 알고리즘을 적용하고 각 라

운드마다 32비트 라운드 키 1개씩 뺄셈 연산을 수행하고 최종적으로 마지막 화이트닝 과정을 거친 후 160비트 암호문을 생성한다.

3.3 키 스케줄링

본 논문에서 제안한 알고리즘의 키 스케줄링은 SHACAL-1의 연관 키 공격으로부터 내성이 있게 기존의 스케줄링을 개선했다. 그리고 대칭단에서 사용된 라운드 키 5개와 암호화 시작 전 화이트닝단계에서 5개의 라운드 키와 복호화를 수행한 후 5개의 라운드 키를 사용하므로 전체 라운드 키는 총 32비트 95개를 사용한다.

다음은 개선된 키 스케줄링 알고리즘이다.

$K = K_0 \,||\, K_1 \,||\, \ldots \,||\, K_{15}$

$K_i = f_{if}(K_{i-3}, K_{i-8}, K_{i-14}) \oplus zi, \quad (16 \le i \le 54)$

$K_i = f_{maj}(K_{i-8}, K_{i-14}, K_{i-16}) \oplus zi, \quad (55 \le i \le 94)$

512비트 초기 키 값은 $K_0 \sim K_{15}$이고, $K_{16} \sim K_{95}$는 SHACAL-1의 비선형 변환 함수 중 $f_{if}()$, $f_{maj}()$를 각각 16 ~ 54스텝, 55 ~ 94스텝까지 적용하고, 라운드 상수인 zi는 순서대로 0x6ed9eba1, 0xca62c1d6을 적용한다.

IV. 대칭 구조의 구현 및 검증

4.1 대칭구조 SHACAL-1 암호 알고리즘 구현

본 논문에서 제안한 대칭구조 SHACAL-1 알고리즘은 CBC(Cipher Block Chaining) 운영모드를 적용하여 Visual Studio 2005 C 컴파일러를 사용하여 암호와 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 SHACAL-1 알고리즘과 제안한 알고리즘의 수행 시간을 테스트했다. 결과는 표 1과 같으며, 제안한 알고리즘의 수행 시간이 1.74% 정도 증가하는 것으로 나타났다.

이는 대칭 단에서 적용한 간단한 논리 연산이 전체적인 암호와 복호 알고리즘 수행에 거의 영향을 미치는 않는 것으로 판단된다.

표 1. 수행 테스트 결과(단위: 초)

Time / Algorithm	Encryption	Decryption
SHACAL-1	10.765 (100.00%)	10.828 (100.00%)
Proposed Algorithm	10.953 (101.74%)	11.016 (101.74%)

4.2 안전성 검증

SHACAL-1에 대한 안전성 분석[13]은 알고리즘 개발자에 의한 차분분석, 선형분석을 했으며, 개발자와 같은 방법으로 1라운드에 10스텝을 적용한 4라운드 40스텝의 차분, 선형분석으로 제안한 대칭구조 SHACAL-1의 안전성을 분석한다. 그리고 연관 키 공격에 취약한 SHACAL-1의 키 스케줄링 알고리즘으로 슬라이드 공격(slide attack)[14] 과정을 설명하고 개선된 키 스케줄링 알고리즘을 제안한다.

이후 본 논문에서 사용할 표기법은 다음과 같다. $x[i]$는 특정 입력으로 32비트 x의 i번 째 비트를 나타내며, 출력 $y[i]$, 라운드 키 $K[i]$도 $x[i]$와 같이 표기하고, Δx는 입력 차분이고, Δy는 출력 차분이다. 그리고 e_i는 32비트 워드 e에서 i번째 비트만 1이고 나머지는 모두 0인 32비트 값이다.

4.2.1 차분, 선형공격

대칭구조 SHACAL-1의 차분분석은 두 가지로 나눌 수 있는데, 첫 번째 xor 연산과 정수 덧셈을 적용하는 것과 두 번째로 SHACAL-1의 라운드 함수 중 비선형 변환 함수 $f_{if}()$, $f_{xor}()$, $f_{maj}()$의 적용이다.

첫 번째로 입력 x와 x'가 아주 조금의 차이가 있으면 $x+K$와 $x'+K$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 $x+K$와 $x'+K$ 또한 MSB만 다르다. 같은 방법으로 $x[i]$와 $x'[i]$가 다를 경우($i<31$) $x[i]+K$와 $x'[i]+K$가 다를 확률은 1/2이다.

$$S[i]=x[i]+K[i]+c[i-1], \quad 0\le i\le 31$$
$$c[i]=x[i]K[i]+x[i]c[i-1]+K[i]c[i-1] \quad (1)$$

$S[i]$는 입력과 라운드 키와의 합(sum)이고, $c[i]$는 캐리(carry)이다. 특히 $c[-1]$는 0이다. 수식 (1)을 사용하여 $x+K$와 $x`+K$가 정확히 연속적으로 2비트 다를 확률은 1/4이다. 2개의 32비트 워드 $x$와 $x`$가 연속적으로 2비트 다를 경우 $x+K$와 $x`+K$가 1비트 다를 확률은 1/4이고, 연속적일 필요는 없지만 2비트 다를 확률은 3/8이다. 그래서 $x$와 $x`$가 연속적으로 2비트 다르면, $x+K$와 $x`+K$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.

두 번째로는 비선형 변환 함수 $f_{if}()$, $f_{xor}()$, $f_{maj}()$에 대한 차분분석이다. 위의 비선형 함수들은 3비트를 입력 받아 1비트를 출력하는 함수로 표 2는 이러한 함수들의 xor 차분 분포를 나타낸 것이다. 표 2에서 x, y, z는 비선형 함수들의 1비트 입력으로 3비트 조합은 8가지의 경우가 있으며 $f_{if}()$, $f_{xor}()$, $f_{maj}()$의 각각의 xor 차분 분포는 표 2의 마지막 3개의 열에 나타냈다. 여기서 차분 분포가 1 또는 0은 입력 차분에 대응하는 출력 차분이 항상 1 또는 0이 되는 것을 말하고 0/1은 입력 차분에 대응하는 출력 차분이 0 또는 1이 될 확률이 1/2이 되는 것을 의미한다.

표 2 비선형 함수의 차분 분포

x	y	z	$f_{xor}()$	$f_{if}()$	$f_{maj}()$
0	0	0	0	0	0
0	0	1	1	0/1	0/1
0	1	0	1	0/1	0/1
0	1	1	0	1	0/1
1	0	0	1	0/1	0/1
1	0	1	0	0/1	0/1
1	1	0	0	0/1	0/1
1	1	1	1	0/1	1

위에서 설명한 차분분석 방법과 SHACAL-1 개발자들에 의해 소개된 가장 좋은 차분분석 특성을 이용하여 10스텝씩 적용하여 4라운드 암호 알고리즘의 차분분석 확률은 1, 3라운드는 각각 2^{-13}, 2, 4라운드는 각각 2^{-26}이다. 그래서 40스텝의 암호 알고리즘의 최종적인 확률은 2^{-78}이다. 그리고 대칭단을 적용한 후 40스텝의 복호 알고리즘을 적용한다. SHACAL-1의 복호 알고리즘과 암호 알고리즘의 차이는 덧셈을 뺄셈으로 그리고 왼쪽 회전연산을 오른쪽 회전연산으로 전환하면 된다. 특히 정수 덧셈과 뺄셈은 같은 방법으로 차분분석을 적용할 수 있으며, 특정한 키에 대해서 암호와 복호는 정확히 일대일 대응되는 전단사 함수이며, SHACAL-1 40스텝의 복호 알고리즘의 차분분석은 암호 알고리즘과 같다. 암호와 복호 스텝이 대칭단에 의해서 서로 불리 되어 있지만 가장 보수적인 분석으로 제안한 SHACAL-1의 최종적인 차분 확률은 $(2^{-78})^2 = 2^{-156}$이다.

대칭구조 SHACAL-1의 선형분석은 [11]의

*Piling-up Lemma*와 간단한 정수 덧셈 $y[i] = K[i] + x[i]$에서 선형 근사값(Linear Approximation)을 생각할 수 있다. 여기서 $x[i]$는 평문의 i번째 비트이며, $K[i]$는 고정된 키의 i번째 비트이다. $i = 0$이면 근사값은 언제나 1/2로 수렴하고, $i = 1$이고 $K[0] = 0$이면 1/2로 수렴하지만 $K[0] = 1$이면 확률이 1/2로 편차(bias)는 0이 된다. $i \geq 1$이고 $(K \wedge (2^i - 1)) = 0$인 경우 선형 근사값의 최대 편차를 가질 수 있으며, 식 (1)을 바탕으로 i보다 낮은 위치에서 캐리가 없는 것을 고려한다. 그래서 $i > 1$인 어떤 경우에도 최대 편차는 2^{-2}과 같거나 작다. 여기서 최대 편차를 주는 약한(weaker) 키의 개수는 i의 위치에 의존적이며, 예를 들어 $i = 30$일 때 최대 편차를 주는 2^{30}개의 키 중에서 하나를 취할 수 있다. 여기서 편차가 0인 어떤 키의 값을 알 수 있는데, 그 키의 값은 $(K \wedge (2^i - 1)) = 2^{i-1}$이다.

이와 같은 방법으로 대칭구조 SHACAL-1의 10스텝을 적용한 4라운드 암호 알고리즘의 선형분석은 우선 xor 연산으로 이루어진 2, 4라운드의 경우 각각의 라운드의 최대 편차는 2^{-6}이며, 전체적인 선형 근사값의 편차는 $(2^{-6})^2 * 2 = 2^{-11}$이 된다. 그리고 SHACAL-1 개발자에 의해 제안된 1라운드 10스텝의 최대 편차는 2^{-6}이다. 마지막으로 3라운드 10스텝의 최대 편차는 2^{-5}이다. 그러므로 대칭구조 SHACAL-1의 4라운드 40스텝의 암호 알고리즘의 선형분석은 $2^{-11} * 2^{-6} * 2^{-5} * 2^2 = 2^{-20}$이다. 앞서 설명한 차분분석과 마찬가지로 선형분석 또한 암호와 복호 알고리즘을 같은 방법으로 분석 할 수 있으며, 대칭단을 적용한 후 가장 유리한 조건을 사용한 방법으로 40스텝의 복호 알고리즘을 적용한 최종적인 선형분석 결과는 $(2^{-20})^2 * 2 = 2^{-39}$이다.

4.2.2 연관 키 공격

연관 키 공격은 서로 다른 키이지만 선형변환처럼 간단한 키 스케줄링에 의해 생성된 라운드 키의 연관관계를 이용한 공격으로 같은 라운드의 반복적인 암호화 진행과정에서 서로 연관관계가 있는 라운드 키를 적용하는 것으로 다음과 같은 간단한 식으로 표현 할 수 있다.

$$KR_i^1 = KR_{i+1}^2$$

위 식 (2)에서 KR_i^1, KR_{i+1}^2는 초기 키 K^1, K^2에 의해 생성된 라운드 키이고, i는 라운드 수이다.

어떤 연관 키의 쌍(K^1, K^2)으로 $P_1 = f_{KR1}^2(P_2)$와 같은 관계를 만족하는 평문쌍(P_1, P_2)이 있고 이때 f_{KR1}^2는 초기 키 K^2로 만들어진 1라운드 키를 사용하여 암호화하는 과정이다. 이 평문에 대응되는 $C_1 = f_{KRr}^1(C_2)$를 만족하는 암호문쌍(C_1, C_2)의 f_{KRr}^1는 초기 키 K^1로 만들어진 r라운드 키를 사용하여 암호화하는 과정이다. 이와 같은 연관관계가 있는 키와 위에서 설명한 조건의 평문의 쌍이 주어지면 쉽게 라운드 키에 대한 정보를 얻을 수 있다. 위에서 설명한 분석방법은 간단한 연관관계가 있으면서 라운드 키에 독립적인 암호화 라운드 함수로 구성된 블록 암호 알고리즘의 공격에 사용하는 기법으로 슬라이드 공격이라 한다. SHACAL-1의 4라운드 80스텝 전부의 연관 키 공격[15]의 데이터 복잡도(data complexity)는 $2^{101.3}$의 결과를 보여주고 있다.

SHACAL-1의 키 스케줄링 알고리즘은 선형 궤환 쉬프트 레지스터(Linear Feedback Shift Register)로 동작한다. 이는 임의의 16개의 라운드 키만 알고 있으며, 나머지 모든 라운드 키의 차이 값을 알 수 있다. 이와 같은 선형적인 키 스케줄링 알고리즘으로 인해 SHACAL-1은 연관 키 공격에 매우 취약하다. 그래서 본 논문에서는 3.3절과 같이 SHACAL-1의 라운드 상수 zi를 추가하고, 라운드 함수에서 비선형 변환을 하는 $f_{if}()$와 $f_{maj}()$를 사용했다. 이는 키 스케줄

링에서 고정된 상수의 사용은 라운드 키의 모든 비트가 0 또는 1과 같이 매우 특이한 경우에도 라운드 키를 랜덤한 형태로 보이도록 할 수 있으며, 비선형 함수의 사용은 키의 변화에 따른 라운드 키의 변화를 예측하기 어렵게 하는 역할을 한다. 본 논문에서 제안한 키 스케줄링은 SHACAL-1의 연관 키 공격을 어렵게 하면서 소프트웨어 및 하드웨어 구현에 효율적으로 키 스케줄링 알고리즘으로 개선했다.

4.2.3 대칭단 분석

대칭단 내에는 라운드 키와 and, or, xor 연산, 그리고 고정된 길이의 왼쪽 회전 연산이 있다. 특히 라운드 키와 and와 or 연산의 차분 및 선형 분석[16]의 영향을 표 3, 표 4에 나타냈다.

표 3. and와 or의 차분분석

and	$K[i]$	$\Delta x[i]$, $\Delta y[i]$	effect
	0	$\Delta x[i]=1 \rightarrow \Delta y[i]=0$	yes
	1	$\Delta x[i]=1 \rightarrow \Delta y[i]=1$	no
or	$K[i]$	$\Delta x[i]$, $\Delta y[i]$	effect
	0	$\Delta x[i]=1 \rightarrow \Delta y[i]=1$	no
	1	$\Delta x[i]=1 \rightarrow \Delta y[i]=0$	yes

표 3에서 $K[i]=0$일 경우 and연산으로 입력 차분 $\Delta x[i]=1$은 출력 차분 $\Delta y[i]=0$이 되므로 차분분석에 유용하며, $K[i]=1$일 경우 or연산으로 입력 차분 $\Delta x[i]=1$은 출력 차분 $\Delta y[i]=0$이 되므로 차분분석을 할 수 있다.

표 4. and와 or의 선형분석

and	$K[i]$	$x[i]$, $y[i]$ 의 값	effect
	0	$x[i]$ independently $y[i]=0$	yes
	1	$y[i] = x[i]$	no
or	$K[i]$	$x[i]$, $y[i]$ 의 값	effect
	0	$y[i] = x[i]$	no
	1	$x[i]$ independently $y[i]=1$	yes

표 4는 $K[i]=0$일 때 입력 $x[i]$와 and연산은 출력 $y[i]$는 무조건 0이 되므로 선형분석에 영향이 있으며, $K[i]=1$일 때 입력 $x[i]$와 or 연산은 출력 $y[i]$는 무조건 1이 되므로 선형분석을 할 수 있다. 대칭단에서 사용된 라운드 키 $K[i]$는 1/2의 확률을 가진다. 입력차분 $\Delta x[i]$나 입력 값 $x[i]$는 $K[i]$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 선형, 차분분석은 SHACAL-1 암호 알고리즘을 적용한 40라운드에서 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.

V. 결론

SHACAL-1은 암호와 복호가 다른 변형된 Feistel 구조이다. 본 논문에서는 간단한 대칭단을 사용하여 암호와 복호를 동일하게 개선한 대칭구조 SHACAL-1을 제안한다. 대칭구조 SHACAL-1은 한 라운드마다 10스텝으로 4라운드를 적용하여 총 40스텝은 SHACAL-1의 암호 알고리즘을, 그리고 1라운드마다 10스텝을 적용한 총 4라운드 나머지 40스텝은 SHACAL-1의 복호 알고리즘을 적용하고, 그 중간에 대칭단을 삽입하여 암호와 복호 알고리즘을 동일하게 개선했다. 제안한 대칭구조 SHACAL-1은 기존의 SHACAL-1과 수행시간 테스트 결과 거의 영향을 미치지 않은 결과를

보였고, 안전성 또한 대칭단의 적용으로 선형, 차분분석과 같은 높은 확률 패스를 이용한 공격에도 분석을 어렵게 하고 있다. 그리고 제안한 알고리즘은 암호, 복호가 동일하기 때문에 하드웨어 구성이 간단해서 스마트카드 및 RFID 능동형 태그와 같은 제한된 하드웨어 및 소프트웨어 환경에서도 쉽게 구현 가능하고 또한 암호 시스템 전체를 구성할 경우는 해시함수가 반드시 필요하며, 대칭구조 SHACAL-1은 SHA-1의 표준 해시함수를 기반으로 만든 블록 암호 알고리즘이므로 간단하게 해시 함수를 구성하여 암호 시스템을 구현 할 수 있다. 그리고 연관 키 공격에 약한 SHACAL-1의 키 스케줄링 알고리즘을 연관 키 공격에 내성이 있도록 개선하였다. 본 논문의 제안 사항인 대칭단은 암호와 복호가 같은 새로운 블록 암호 설계에도 유용한 아이디어로 사용할 수 있다.

참고문헌

[1] National Bureau of Standards, Data Encryption Standard, FIPS-Pub. 46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., Jan. 1977.

[2] "Report on the Development of the Advanced Encryption Standard(AES)," http://www.csrc.nist.gov/encryption/aes/round2/r2report.pdf.

[3] J. Daemen, and V. Rijmen, "AES Proposal: Rijndael," http://www.csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf, 1999.

[4] NESSIE project, New European Schemes for Signatures Integrity and Encryption, http://cryptonessie.org/.

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

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

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

[8] C. E. Shannon, "Communication Theory of Secrecy System," *Bell System Technical Journal*, Vol. 28, No.4, pp. 656-715, 1949.

[9] NIST, *Secure hash standard*, FIPS-180-1, 1995.

[10] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys.", Journal of Cryptology, Vol. 7, No. 4, pp. 229-246, 1994.

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

[12] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems." Journal of Cryptology, Vol. 4, No. 1, pp. 3-72, 1991.

[13] H. Handschuh and D. Naccache, "SHACAL." In Proceedings of the First Open NESSIE Workshop, 2000.

[14] A. Biryukov and D. Wagner, "Slide Attacks." LNCS, Vol. 1636, pp.245-259, 1999.

[15] E. Biham, O. Dunkelman and N. Keller, "A Simple Related-Key Attack on the Full SHACAL-1." LNCS, Vol. 4377, pp.20-30, 2006.

[16] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Specification of Camellia - a 128bit Block Cipher," http://info.isl.ntt.co.jp/camellia/, 2000.