情報保護學會論文誌
第12券 第1號, 2002. 2

DC와 LC에 대해 안전성 증명 가능한 블록 암호 알고리즘 FRACTAL*

김 명 환**, 이 인 석**, 백 유 진**, 김 우 환**, 강 성 우***

New Block Encryption Algorithm FRACTAL with Provable Security against DC and LC

Myung-Hwan Kim**, In-Sok Lee**, Yoo-Jin Baek**,
Woo-Hwan Kim**, Sung-Woo Kang***

요 약

본 논문에서는 새로운 블록 암호 알고리즘인 FRACTAL 암호 알고리즘을 제안한다. FRACTAL은 128비트의 블록 길이와 128비트 키를 사용하는 8라운드 Feistel 구조의 암호 알고리즘이며, 블록 암호 알고리즘에 대한 가장 강력한 공격법인 차분 분석과 선형 분석에 대한 안전성이 증명 가능하다.

ABSTRACT

In this article, a new block encryption algorithm FRACTAL is introduced. FRACTAL adopts 8-round Feistel structure handling 128 bit inputs and keys. Furthermore, FRACTAL possesses the provable security against DC and LC, which are known to be the most powerful attacks on block ciphers.

keyword : *block cipher, provable security, DC, LC*

Ⅰ. 서 론

블록 암호 알고리즘은 고정된 크기의 키를 사용하여 고정된 크기의 블록 단위로 암·복호화 연산을 수행하며 암호화 키와 복호화 키가 같은 알고리즘을 말한다. 블록 암호 알고리즘은 그 특성상 상대적으로 짧은 키로 빠른 암·복호화를 구현하기 때문에 전자 상거래 등에서 정보의 기밀성과 정보 처리의 효율성을 동시에 충족시킬 수 있는 필수 기술이 되었다.

1976년 미국의 국가표준국(NIST)이 미국의 상업용 표준 블록 암호 알고리즘으로 채택한 DES는 그 이후 개발된 여러 블록 암호 알고리즘의 모델이 되었다. 유럽과 일본 역시 일찍부터 정부와 대기업을 중심으로 블록 암호 알고리즘 개발에 많은 노력을 기울여왔으며 우리나라에서도 1998년에 국내 전자상거래의 안전성 확보를 위한 기반 기술의 하나로 한국정보보호진흥원(KISA)의 주도 하에 블록 암호 알고리즘 SEED를 개발하여 상업용 표준 암호 알고리즘으로 채택하였다.

한편 블록 암호 알고리즘에 대한 다양한 공격방법이 연구되고 발전되었다. 특히 가장 성공적인 공격

* 본 연구는 한국정보보호진흥원 연구과제(2001-S-073) 지원으로 수행하였습니다.
** 서울대학교 자연과학대학 수리과학부({mhkim,islee,yjbaek,whkim}@math.snu.ac.kr)
*** 한국정보보호진흥원(swkang@kisa.or.kr)

방법으로 평가되는 차분 분석과 선형 분석에 의하여 DES의 안전성이 위협받기에 이르자 NIST는 최근 DES를 AES로 대체하였다.

이제는 이처럼 다양한 공격법에 대하여 안전하면서도 구현하기 쉽고 효율적인 블록 암호 알고리즘의 개발이 요구되고 있다. 본 논문의 목적은 차분 분석과 선형 분석 등 기존의 공격 방법에 대해서 안전하고, 수학적 이론에 근거하여 그 안전성을 증명할 수 있는 새로운 블록 암호 알고리즘을 제안하고자 하는 것이다.

본 논문의 구성은 다음과 같다. 먼저 2절에서는 블록 암호 알고리즘의 차분 분석(DC)과 선형 분석(LC)에 대한 안전성 증명에 필요한 중요한 개념과 용어 및 성질을 다루고, 3절에서 새로운 블록 암호 알고리즘인 FRACTAL의 구조와 키 생성 알고리즘을 소개한 후, 4절에서 그 안전성을 분석한다. "FRACTAL"은 fractal에서 따온 이름으로 64비트 → 32비트 → 16비트로 비트의 크기를 반씩 줄여도 계속 같은 형태의 구조가 반복되는 F-함수의 구조적 특성을 나타내었다. FRACTAL의 특징은 다음과 같다 :

- 128비트 대칭 키 블록 암호 알고리즘
- 128비트 키 크기 (더 큰 키도 사용가능)
- 8라운드 Feistel 구조
- 대부분의 연산을 바이트 단위로 수행
- F-함수 안에서 확산효과를 위한 선형 함수 $L1$, $L2$, $L3$ 사용
- DC에 대한 안전성 증명 가능
- LC에 대한 안전성 증명 가능

등이다.

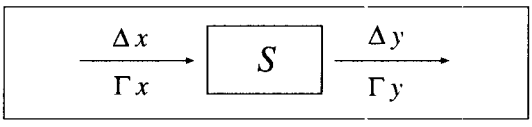
II. 안전성 증명

"차분특성(differential characteristic)"은 차분 분석법에서 블록 암호 알고리즘을 공격하는데 성공적으로 사용되었다. 즉, 확률이 충분히 큰 차분 특성이 존재하는 경우 차분 분석을 이용하여 비밀키에 대한 몇 비트의 정보를 복구해 낼 수 있다는 것이다. 한편 Lai, Massey, Murphy는 그 역이 성립하지 않다는 것을 보이고 차분 분석에 대한 강도(strength)를 반영하는 척도로 차분 특성을 대신하여 "차분(differential)"을 제안하였다.[6] 차분은 차분 특성들의 집합으로 이해할 수 있다. 어떤 암호의 최대 차분 특성 확률이 낮다고 해서 그 암호가 차분 분석에 대해 안전하다고 말할 수는 없지만, 최대 차분 확률이 충분히 낮으면 그 알고리즘은 일반적인 차분 분석에 대해 안전함이 알려져 있다. 한편 Nyberg, Knudsen은 처음으로 최대 차분 확률이 충분히 낮은 블록 암호 알고리즘의 예를 제시하였고[8] 그러한 성질을 차분 분석에 대한 "증명 가능한 안전성(provable security)"이라 불렀다.

선형 분석도 차분 분석과 마찬가지로 처음에는 블록 암호를 공격하는데 선형 분석의 "특성(characteristic)"을 이용하였으나 최근에 Nyberg에 의해 최소 선형 집합(linear hull)이라 불리는 특성들의 집합이 소개되었는데[8], 이는 선형 분석에 대한 강도의 엄밀한 척도로 여겨진다. [9]에서 제시된 예는 최소 선형 집합이 작아 선형 분석에 대해서도 증명 가능한 안전성을 지니고 있다. 그러나 계산의 효율성은 별로 좋지 않다는 단점이 있다.

2.1 정의 및 정리

먼저 키와 독립적인 S-함수(그림 2.1)에 대한 차분 분석 및 선형 분석의 확률에 대한 정의와 그에 따른 성질들을 살펴보기로 하자.



[그림 1] S-함수(S-box)

[정의 2.1]

S를 n비트의 입력, 출력 값을 가지는 고정된 함수라 하고, X Y는 각각 S의 가능한 2^n개의 입력, 출력 값의 집합이라 하자. 주어진 Δx, $\Gamma x \in X$와 Δy, $\Gamma y \in Y$에 대하여

$$dp^S(\Delta x \to \Delta y) := \frac{\#\{x \in X \mid S(x) \oplus S(x \oplus \Delta x) = \Delta y\}}{2^n}$$

$$lp^S(\Gamma x \to \Gamma y) := \left(\frac{\#\{x \in X \mid x \cdot \Gamma x = S(x) \cdot \Gamma y\}}{2^{n-1}} - 1 \right)^2$$

를 각각 S의 차분 확률, 선형 확률이라 정의한다. 여기서 $a \cdot b$는 Z_2상에서의 내적이고 $\#A$는 집합 A의 원소의 개수를 나타낸다.

[정의 2.2]

함수 S에 대하여

$$dp^S_{\max} := \max_{\Delta x \neq 0, \Delta y} dp^S(\Delta x \rightarrow \Delta y)$$
$$lp^S_{\max} := \max_{\Gamma x, \Gamma y \neq 0} lp^S(\Gamma x \rightarrow \Gamma y)$$

를 각각 S의 최대 차분 확률, 최대 선형 확률이라 정의한다. 차분 분석이나 선형 분석에 강한 S-box는 임의의 $\Delta x(\neq 0)$, $\Gamma x \in X$와 Δy, $\Gamma y(\neq 0) \in Y$에 대해서도 dp^S와 lp^S의 값이 작게 나타나야 한다. 따라서 최대 차분 확률과 최대 선형 확률은 어떤 S-box의 차분 분석이나 선형 분석에 대한 안전성을 증명하는데 필요한 중요한 개념이다.

다음에 소개하는 두 개의 보조정리는 여러 가지 형태의 블록 암호 알고리즘의 차분 확률과 선형 확률을 계산하는데 매우 유용하다.[7]

[보조정리 2.3]

n비트의 입력, 출력 값을 가지는 함수 S에 대하여

$$lp^S(\Gamma x \rightarrow \Gamma y) = \left(\frac{1}{2^n} \sum_{x \in X} (-1)^{(x \cdot \Gamma x) \oplus (S(x) \cdot \Gamma y)} \right)^2$$

이 성립한다.

[보조정리 2.4]

임의의 함수 S에 대하여

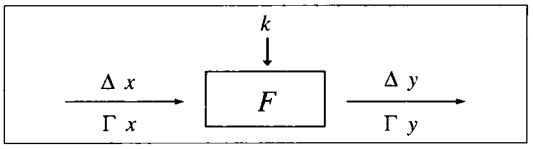
$$\sum_{\Delta y \in Y} dp^S(\Delta x \rightarrow \Delta y) = 1, \quad \sum_{\Gamma x \in X} lp^S(\Gamma x \rightarrow \Gamma y) = 1$$

이 성립한다. 특히 S가 전단사이면,

$$\sum_{\Delta x \in X} dp^S(\Delta x \rightarrow \Delta y) = 1, \quad \sum_{\Gamma y \in Y} lp^S(\Gamma x \rightarrow \Gamma y) = 1$$

이 성립한다.

이제 키에 의존하는 F-함수(그림 2.2)에 대한 차분 분석 및 선형 분석의 확률에 대한 정의와 그에 따른 성질들을 살펴보기로 하자. 먼저 K는 가능한 모든 키 값들의 집합이라 하고 $F(k)$를 고정된 키 k에 대응되는 함수라 할 때, 각각의 k에 대한 $F(k)$의 차분 확률 및 선형 확률의 평균으로 함수 F의 차분 확률 및 선형 확률을 정의한다.



〔그림 2〕 F-함수(F-function)

[정의 2.5]

키에 의존하는 F-함수에 대하여

$$dp^F(\Delta x \rightarrow \Delta y) := \frac{1}{\#K} \sum_{k \in K} dp^{F(k)}(\Delta x \rightarrow \Delta y),$$
$$lp^F(\Gamma x \rightarrow \Gamma y) := \frac{1}{\#K} \sum_{k \in K} lp^{F(k)}(\Gamma x \rightarrow \Gamma y)$$

를 각각 F의 차분 확률, 선형 확률이라 정의한다. 특히, F가 암호화 함수이고 임의의 $\Delta x(\neq 0)$, $\Gamma x \in X$와 Δy, $\Gamma y(\neq 0) \in Y$에 대해서도 dp^F와 lp^F의 값이 충분히 작으면 F가 차분 분석과 선형 분석에 대해 증명 가능한 안전성을 지닌다고 말한다.

[정의 2.6]

키에 의존하는 F-함수에 대하여

$$dp^F_{\max} := \max_{\Delta x \neq 0, \Delta y} dp^F(\Delta x \rightarrow \Delta y),$$
$$lp^F_{\max} := \max_{\Gamma x, \Gamma y \neq 0} lp^F(\Gamma x \rightarrow \Gamma y)$$

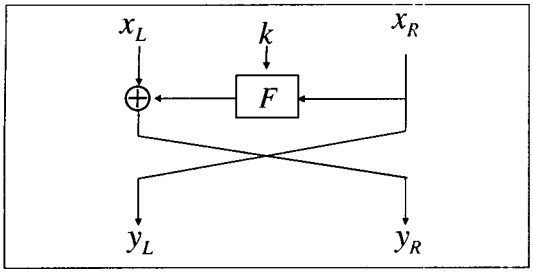
를 각각 F의 최대 차분 확률, 최대 선형 확률이라 정의한다.

2.2 Feistel 구조의 안전성

여기서는 앞 절에서 논의한 치환 함수를 이용한 Feistel 구조의 블록 암호 알고리즘 중에서 차분 분석과 선형 분석에 대해 증명 가능한 안전성을 지닌, 그 중에서 가장 기본적인 형태인 그림 3의 구조에 대한 여러 정리들을 소개하고자 한다.

[정의 2.7]

r-라운드 Feistel 구조를 가진 블록 암호 알고리즘에서 i번째 라운드의 입력 값과 출력 값을 각각 $x(i) = (x_L(i), x_R(i))$, $y(i) = (y_L(i), y_R(i))$라고 하자 $1 \leq i \leq r$). 따라서 $x(1)$은 평 (P_L, P_R)을, $y(r)$은 암호 (C_L, C_R)을 의미하며, $y(i) = x(i+1)$이 된다. 이때



(그림 3) Feistel 구조 1-라운드(one round Feistel)

$$DP(r,\ \Delta P\rightarrow\Delta C):$$
$$=\sum_{\Delta x(2),\cdots,\Delta x(r-1)}\prod_{i=1}^{r} dp^F(\Delta x_R(i)\rightarrow\Delta x_L(i)\oplus\Delta y_R(i)),$$

$$LP(r,\ \Gamma P\rightarrow\Gamma C):$$
$$=\sum_{\Gamma x(2),\cdots,\Gamma x(r-1)}\prod_{i=1}^{r} lp^F(\Gamma x_R(i)\rightarrow\Gamma x_L(i)\oplus\Gamma y_R(i))$$

를 각각 r-라운드 알고리즘의 차분 확률, 선형 확률이라 정의한다. 또한

$$DP(r) := \max_{\Delta P\neq 0,\Delta C} DP(r,\ \Delta P\rightarrow\Delta C)$$
$$LP(r) := \max_{\Gamma P,\Gamma C\neq 0} LP(r,\ \Gamma P\rightarrow\Gamma C)$$

를 각각 r-라운드 알고리즘의 최대 차분 확률, 최대 선형 확률이라 정의한다.

$DP(r)$과 $LP(r)$에 대해서 다음이 성립함이 알려져 있다.[4]

[보조정리 2.8]
임의의 $r(\geq 1)$에 대하여

$$DP(r+1)\leq DP(r),\ LP(r+1)\leq LP(r)$$

이 성립한다. 즉, Feistel 구조의 블록 암호 알고리즘에서 라운드 수를 증가시키는 경우, 최대 차분 확률과 최대 선형 확률은 커지지 않는다.

[보조정리 2.9]
임의의 $r(\geq 1)$에 대하여

$$\sum_{\Delta C} DP(r,\ \Delta P\rightarrow\Delta C)=1,\ \sum_{\Gamma P} LP(r,\ \Gamma P\rightarrow\Gamma C)=1$$

이 성립한다.

최대 차분 확률에 관한 다음의 정리는 차분 분석과 선형 분석에 강한 함수를 사용함으로써 이들에 대하여 안전한 Feistel구조의 블록 암호 알고리즘을 만드는 것이 가능함을 보여주고 있다.[7,9]

[정리 2.10]
임의의 $r(\geq 4)$에 대하여

$$DP(r)\leq 2(dp^F_{\max})^2,\ \ LP(r)\leq 2(lp^F_{\max})^2$$

이 성립한다.

[정리 2.11]
F-함수가 치환이면, 임의의 $r(\geq 3)$에 대하여

$$DP(r)\leq (dp^F_{\max})^2,\ \ LP(r)\leq (lp^F_{\max})^2$$

이 성립한다

Ⅲ. FRACTAL 블록 암호 알고리즘

3.1 유한체 $GF(2^n)$과 그 위에서의 연산

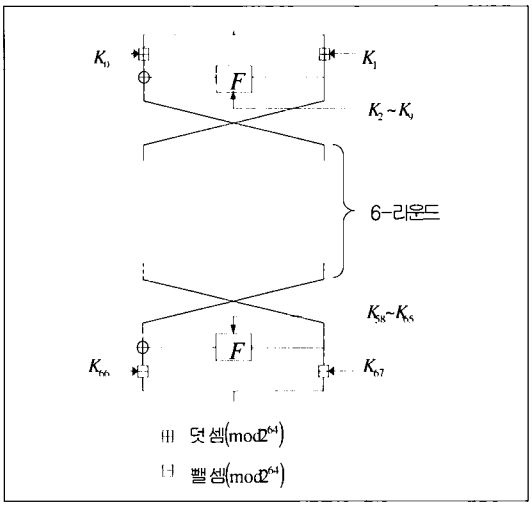
FRACTAL 블록 암호 알고리즘에서는 여러 연산이 바이트 단위로 수행이 되며 이 경우의 바이트란 유한 체 $GF(2^8)$의 한 원소를 의미한다. 유한 체 $GF(2^8)$은 다음과 같이 정의된다.

$f(x)$를 $GF(2)$위에서의 8차 기약다항식이라 하자. 그러면 유한 체 $GF(2^8)$의 모든 원소는 0 또는 1의 계수를 갖는 7차 이하의 다항식으로 표현될 수 있으며, 이 때 $GF(2^8)$위에서의 연산은 다음과 같이 정의된다.

- 덧셈(+) : 다항식 덧셈 (mod 2)
- 곱셈(*) : 다항식 곱셈 (mod $f(x)$)
- 곱셈에 대한 역원 : $GF(2^8)$의 원소 $g(x)$가 있을 때, 유클리드 알고리즘을 이용하면 $g(x)h(x)+f(x)k(x)=1$을 만족하는 다항식 $h(x)$, $k(x)$를 $GF(2)[x]$ 안에서 찾을 수 있는데, 이때 $h(x)$ (mod $f(x)$)가 $g(x)$의 곱셈에 대한 역원이 된다.

3.2 FRACTAL 알고리즘의 구조

FRACTAL의 전체구조는 8-라운드 Feistel 구조



〔그림 4〕 FRACTAL의 전체 구조(The Structure of FRACTAL)

〔그림 4〕이며 거기에 덧붙여 처음에 두 개의 64비트 부분 키 K_0, K_1을 평문에 더하고, 마지막에 역시 두 개의 64비트 부분 키 K_{66}, K_{67}을 빼 준다. 이 경우 덧셈과 뺄셈 (mod 2^{64})로 한다. Kilian, Rogaway[5] 등에 의해서 제안되었듯이 알고리즘의 맨 처음과 끝에 key xoring을 해 주면, 키 조사 공격에 대한 안전성이 강해지는 것으로 알려져 있다. modulo addition은 xoring 보다 효율성은 떨어지지만 알고리즘에서 주로 사용되는 xoring과 가환성(commutativity)을 지니지 않기 때문에 더 안전하다고 할 수 있다.

3.3 F-함수

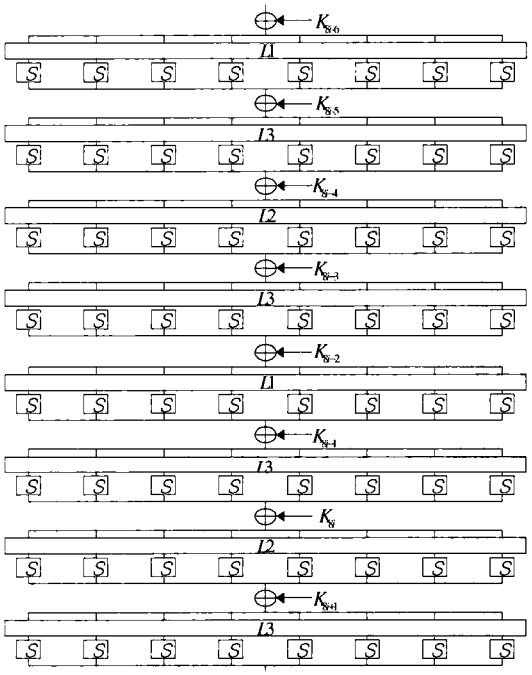
F-함수는 $L1$, $L2$, $L3$이라는 세 개의 선형 함수와 S-box, 그리고 부분 키 Xoring이 사용된다(〔그림 5〕). 즉, F-함수는 〔64비트 부분 키 Xoring → 선형 함수 → S-box〕의 과정이 총 8회 반복된다. 따라서 F-함수에는 총 8개의 64비트 부분 키가 쓰인다. 선형 함수는 매회 $L1$, $L2$, $L3$ 중 하나가 쓰이며, 1회부터 8회까지 $L1$-$L3$-$L2$-$L3$-$L1$-$L3$-$L2$-$L3$의 순서로 사용된다. 선형 함수 $L1$, $L2$, $L3$는 64비트 입력 값을 유한 체

$$GF_1 := GF(2)[x]/(x^8+x^6+x^5+x^3+1) \cong GF(2^8)$$

위에서의 길이가 8인 벡터로 이해하며, S-box는 8비트 입력 값을 유한 체

$$GF_2 := GF(2)[x]/(x^8+x^6+x^5+x^4+1) \cong GF(2^8)$$

의 한 원소로 이해한다. 이처럼 선형 함수와 S-box는 바이트 단위의 연산을 수행하게 함으로써 계산의 효율성을 높이고자 하였다.



〔그림 5〕 FRACTAL의 F-함수(F-function of FRACTAL)

3.4 선형 함수

F-함수에 사용되는 선형 함수들은 64비트 입력 값을 유한 체 GF_1상에서의 길이가 8인 벡터로 이해하며, 확산효과를 위하여 도입되었다. 또한 바이트 단위의 연산을 수행한다.

3.4.1 선형 함수 $L1$

$L1$-함수는 다음과 같은 행렬로 주어진다.

$$L1 : (GF_1)^8 \rightarrow (GF_1)^8$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & y & 1 & y & 1 & y & 1 & y \\ 1 & 1 & y & y & 1 & 1 & y & y \\ 1 & y & y & y^2 & 1 & y & y & y^2 \\ 1 & 1 & 1 & 1 & y & y & y & y \\ 1 & y & 1 & y & y & y^2 & y & y^2 \\ 1 & 1 & y & y & y & y & y^2 & y^2 \\ 1 & y & y & y^2 & y & y^2 & y^2 & y^3 \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix}$$

여기서 $a_i \in GF_1$이고 $y := 1 + x \in GF_1$이다 $L1$-함수의 C pseudo-코드는 다음과 같다.

입력 : $a_1, a_2, \ldots, a_8$ ↦ 출력 : $a_1, a_2, \ldots,$
$a_1 = a_1 + a_5$, $a_2 = a_2 + a_6$, $a_3 = a_3 + a_7$, a_4
$a_8 = a_4 + a_8$;
$a_5 = a_1 + x*a_5$, $a_6 = a_2 + x*a_6$, $a_7 = a_3 + x*a_7$,
$a_8 = a_4 + x*a_8$;
$a_1 = a_1 + a_3$, $a_2 = a_2 + a_4$, $a_5 = a_5 + a_7$,
$a_6 = a_6 + a_8$;
$a_3 = a_1 + x*a_3$, $a_4 = a_2 + x*a_4$, $a_7 = a_5 + x*a_7$,
$a_8 = a_6 + x*a_8$;
$a_1 = a_1 + a_2$, $a_3 = a_3 + a_4$, $a_5 = a_5 + a_6$,
$a_7 = a_7 + a_8$;
$a_2 = a_1 + x*a_2$, $a_4 = a_3 + x*a_4$, $a_6 = a_5 + x*a_6$,
$a_8 = a_7 + x*a_8$.

3.4.2 선형 함수 $L2$

$L2$-함수는 다음과 같은 행렬로 주어진다.

$$L2 : (GF_1)^8 \to (GF_1)^8$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & y & 1 & y & 0 & 0 & 0 & 0 \\ 1 & 1 & y & y & 0 & 0 & 0 & 0 \\ 1 & y & y & y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & y & 1 & y \\ 0 & 0 & 0 & 0 & 1 & 1 & y & y \\ 0 & 0 & 0 & 0 & 1 & y & y & y^2 \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix}$$

$L2$-함수의 C pseudo-코드는 다음과 같다.

입력 : $a_1, a_2, \ldots, a_8$ ↦ 출력 : $a_1, a_2, \ldots, a_8$
$a_1 = a_1 + a_3$, $a_2 = a_2 + a_4$, $a_5 = a_5 + a_7$,
$a_6 = a_6 + a_8$;
$a_3 = a_1 + x*a_3$, $a_4 = a_2 + x*a_4$, $a_7 = a_5 + x*a_7$,
$a_8 = a_6 + x*a_8$;
$a_1 = a_1 + a_2$, $a_3 = a_3 + a_4$, $a_5 = a_5 + a_6$,
$a_7 = a_7 + a_8$;
$a_2 = a_1 + x*a_2$, $a_4 = a_3 + x*a_4$, $a_6 = a_5 + x*a_6$,
$a_8 = a_7 + x*a_8$.

3.4.3 선형 함수 $L3$

$L3$-함수는 다음과 같은 행렬로 주어진다.

$$L3 : (GF_1)^8 \to (GF_1)^8$$

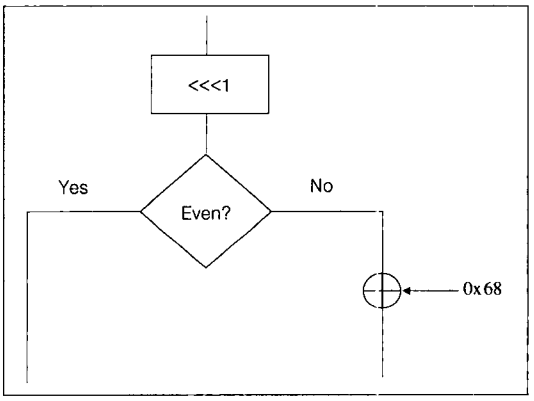
$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & y & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & y & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & y \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix}$$

$L3$-함수의 C pseudo-코드는 다음과 같다.

입력 : $a_1, a_2, \ldots, a_8$ ↦ 출력 : $a_1, a_2, \ldots, a_8$
$a_1 = a_1 + a_2$, $a_3 = a_3 + a_4$, $a_5 = a_5 - a_6$,
$a_7 = a_7 + a_8$;
$a_2 = a_1 + x*a_2$, $a_4 = a_3 + x*a_4$, $a_6 = a_5 + x*a_6$,
$a_8 = a_7 + x*a_8$.

3.4.4 x곱-함수

선형 함수의 정의에서 x곱-함수 $x*a$)가 자주 등장하는데 이 함수는 아래의 [그림 6]과 같이 간단하게 구현할 수 있다. 즉, 입력 값 8비트를 1비트 왼쪽으로 회전(rotation)시킨 후 최하위 비트가 1이면 0x68를 Xoring 해주고 최하위 비트가 0이면 그대로 둔다.



[그림 6] x곱-함수(multiplication by x)

3.5 S-함수(S-box)

S-함수(S-box)는 8비트 입력 값을 유한 체 GF_2 위에서의 한 원소로 이해하며 다음과 같은 함수로 주어진다. $S : GF_2 \to GF_2$, $g \mapsto (g + 0\text{xA})^{-1}$

여기서 $0\text{xA} = x^7 + x^5 + x^3 + x + 1 \in GF_2$이고 $0^{-1} = 0$

으로 이해한다. 0xAB를 더하여 역을 취한 것은 0과 1이 고정점이 되는 것을 방지하기 위함이다. 한편, GF_2를 정의하는데 사용된 기약 다항식 $x^8+x^6+x^5+x^4+1$은 $GF(2)$ 상의 원시 다항식(primitive polynomial)이다. 따라서 이 기약다항식은 $GF_2 = GF(2)(\alpha)$가 되는 α를 근으로 가진다.

3.6 키 생성 알고리즘

키 생성 알고리즘은 랜덤하게 주어지는 128비트 키를 이용하여 68개의 64비트 부분 키를 생성하는 알고리즘이다. 이 알고리즘에는 다음과 같은 함수와 상수들이 사용된다.

- S^8: FRACTAL 암호 알고리즘에 사용되는 S-box를 8번 병렬로 사용하는 함수
- $L1$, $L2$, $L3$: FRACTAL 암호 알고리즘에 사용되는 세 가지 선형함수
- $e = \lfloor (e-2)_{(2)} * 2^{64} \rfloor$: 자연대수 e를 2진법으로 표시했을 때의 소수 부분 64비트로, 실제 값은 다음과 같다. 단 $\lfloor \quad \rfloor$는 최대정수함수를 나타내는 기호이다.

0xB7E151628AED2A6A

- $\tau = \lfloor (\tau-1)_{(2)} * 2^{64} \rfloor$: 황금비 τ를 2진법으로 표시했을 때의 소수 부분 64비트로, 실제 값은 다음과 같다.

0x9E3779B97F4A7C15

키 생성 알고리즘은 다음과 같다.

- 입력 : 128비트 키 K K_L: 키 K의 왼쪽 64비트, K_R: 키 K의 오른쪽 64비트
- 출력 : 68개의 64비트 부분 K_0, K_1, ... , K_{67}.

$K_{-1} := K_L \oplus K_R$; $K_0 := K_R$; $K_1 := K_L$;

각 i $(1 \le i \le 11)$에 대하여,

$K_{6i-4} := S^8(K_{6i-7} \oplus e \oplus (6i-4))$;

$K_{6i-3} := S^8(K_{6i-6} \oplus \tau \oplus (6i-3))$;

$K_{6i-2} := S^8(K_{6i-5} \oplus e \oplus \tau \oplus (6i-2))$;

$K_{6i-1} := L_1(K_{6i-3}) \oplus K_{6i-2} \oplus (6i-1)$;

$K_{6i} := K_{6i-4} \oplus L_2(K_{6i-2}) \oplus 6i$;

$K_{6i+1} := L_3(K_{6i-4}) \oplus K_{6i-3} \oplus (6i+1)$.

상수 e, τ 와 함수 S^8, $L1$, $L2$, $L3$ 등을 사용하여 부분 키들이 비선형적으로 생성되도록 함으로써 연관 키 공격이나 기타 키를 이용한 공격이 힘들도록 설계되었다. 또한, F-함수에서 사용한 것과 같은 함수를 사용함으로써 설계가 용이하고 계산의 효율성을 높도록 하였다. 키 K의 크기가 128비트보다 큰 경우에도 키 생성 알고리즘을 약간 변형하여 필요한 부분 키들을 생성할 수 있다.

3.7 복호화 알고리즘

FRACTAL 알고리즘은 Feistel 구조이기 때문에 복호화 알고리즘은 부분 키 순서를 제외하고는 암호화 알고리즘과 동일하다.

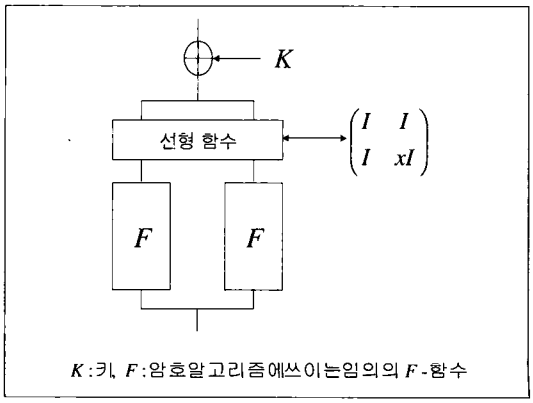
Ⅳ. FRACTAL 암호 알고리즘의 안전성 증명과 효율성

4.1 DC 및 LC에 대한 안전성

이 절에서는 FRACTAL의 DC 및 LC에 대한 안전성을 증명하기 위하여 안전성 증명이 용이한 FRACTAL-TH 암호 알고리즘을 도입하여 FRACTAL-TH 암호 알고리즘의 안전성을 증명하고, 이 알고리즘과 FRACTAL이 동등함을 보임으로써 FRACTAL 암호 알고리즘의 안전성을 증명한다.

[정리 4.1]

아래 그림과 같은 구조를 r번 사용하는 알고리즘을 생각하자.



K:키, F:암호알고리즘에쓰이는임의의 F-함수

만약 $r \ge 2$이고 부분 키들이 독립적이고 랜덤하면 다음이 성립한다. $DP(r) \le (dp^F)^2$, $LP(r) \le (lp^S)^2$.

(증명)

보조정리 2.8에 의하면 $r=2$인 경우에만 증명해도 충분하다. 먼저 dp에 대해서 증명해 보자.

$(\alpha_0, \beta_0) \neq (0,0)$를 입력 차라 하고, (α_1, β_1)를 1-라운드 후의 출력 차, (α_2, β_2)를 2-라운드 후의 출력 차라 하자. 그러면 다음과 같은 식이 성립한다.

$$\begin{aligned}
&\Pr((\alpha_0, \beta_0) \rightarrow (\alpha_2, \beta_2)) \\
&= \sum_{\alpha_1, \beta_1} \Pr((\alpha_0, \beta_0) \rightarrow (\alpha_1, \beta_1)) \Pr((\alpha_1, \beta_1) \\
&\quad \rightarrow (\alpha_2, \beta_2)) \\
&= \sum_{\alpha_1, \beta_1} \Pr((\alpha_0 + \beta_0, \alpha_0 + x*\beta_0) \rightarrow (\alpha_1, \beta_1)) \\
&\quad \Pr((\alpha_1 + \beta_1, \alpha_1 + x*\beta_1) \rightarrow (\alpha_2, \beta_2))
\end{aligned}$$

(Case 1)

$\alpha_0 + \beta_0 = 0$인 경우 : 이 경우에는 $\alpha_1 = 0$이고 $\beta_1 \neq 0$이다. 따라서

$\Pr((\alpha_1 + \beta_1, \alpha_1 + x*\beta_1) \rightarrow (\alpha_2, \beta_2)) \leq (dp^F)^2$이고

$\Pr((\alpha_0, \beta_0) \rightarrow (\alpha_2, \beta_2))$

$\leq (dp^F)^2 \sum_{\alpha_1, \beta_1} \Pr((\alpha_0 + \beta_0, \alpha_0 + x*\beta_0) \rightarrow (\alpha_1, \beta_1))$

$\leq (dp^F)^2$

이 성립한다.

(Case 2)

$\alpha_0 + x*\beta_0 = 0$인 경우의 증명은 Case 1과 동일하다.

(Case 3)

$\alpha_0 + \beta_0 \neq 0$, $\alpha_0 + x*\beta_0 \neq 0$인 경우 :

$\Pr((\alpha_0 + \beta_0, \alpha_0 + x*\beta_0) \rightarrow (\alpha_1, \beta_1)) \leq (dp^F)^2$에서

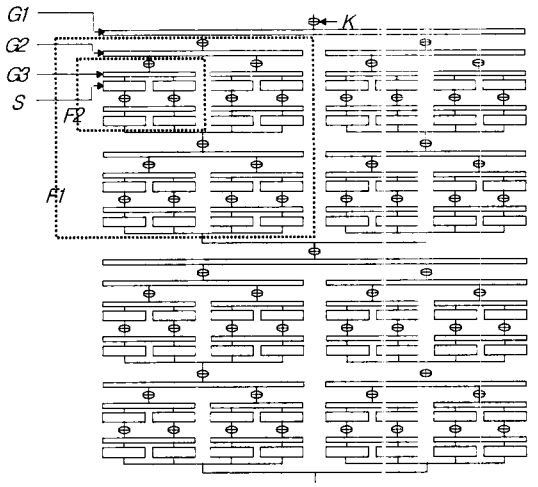
$\Pr((\alpha_0, \beta_0) \rightarrow (\alpha_2, \beta_2))$

$\leq (dp^F)^2 \sum_{\alpha_1, \beta_1} \Pr((\alpha_1 + \beta_1, \alpha_1 + x*\beta_1) \rightarrow (\alpha_2, \beta_2))$

$\leq (dp^F)^2$

를 얻는다.

Case 1,2,3에 의해서 $DP(2) \leq (dp^F)^2$이 성립한다. LC에 대한 증명은 DC에 대한 증명과 거의 유사하므로 생략한다. ■



〔그림 7〕F-TH-함수(F-TH-function)

FRACTAL의 DC 및 LC에 대한 안전성을 증명하기 위하여 안전성 증명이 용이한 FRACTAL-TH 암호 알고리즘을 생각하자. FRACTAL-TH는 FRACTAL과 같은 Feistel 구조로 된 암호 알고리즘으로서, FRACTAL-TH의 F-함수에 해당되는 F-TH-함수는 〔그림 7〕과 같은 구조를 가지고 있다.

여기서 S 함수는 FRACTAL 알고리즘의 S-box와 동일한 함수이고, $G1$, $G2$, $G3$-함수는 다음과 같은 GF_1상의 행렬로 주어진다.

$$G1 := \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & x & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & x & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & x & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & x
\end{pmatrix}$$

$$G2 := \begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & x & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & x & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & x & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & x
\end{pmatrix}$$

$$G3 := \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & x & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & x & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & x & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & x
\end{pmatrix}$$

FRACTAL의 키 생성 알고리즘을 적당히 변형하면 $L1$, $L2$, $L3$ 대신에 $G1$, $G2$, $G3$를 사용), FRACTAL-TH의 키 생성 알고리즘을 얻을 수 있을 뿐만 아니라, 두 알고리즘이 동등한 구조를 갖도록 할 수 있다. 따라서 FRACTAL 알고리즘과 FRACTAL-TH 알고리즘은 똑같은 안전성을 가지게 된다.

[정리 4.2]

F-TH를 F-함수로 사용하고 3-라운드 이상의 Feistel구조를 갖는 FRACTAL-TH에 대해서 부분 키들이 독립적이고 랜덤하면 다음이 성립한다.

$$DP \le \frac{1}{2^{96}}, \quad LP \le \frac{1}{2^{96}}.$$

(증명)

정리 4.1을 반복해서 사용하면 다음과 같은 식을 얻는다.

$$dp^{F2} \le (dp^{S})^2 = (\frac{1}{2^6})^2 = \frac{1}{2^{12}},$$
$$lp^{F2} \le (lp^{S})^2 = (\frac{1}{2^6})^2 = \frac{1}{2^{12}},$$
$$dp^{F1} \le (dp^{F2})^2 = (\frac{1}{2^{12}})^2 = \frac{1}{2^{24}},$$
$$lp^{F1} \le (lp^{F2})^2 = (\frac{1}{2^{12}})^2 = \frac{1}{2^{24}},$$
$$dp^{F-TH} \le (dp^{F1})^2 = (\frac{1}{2^{24}})^2 = \frac{1}{2^{48}},$$
$$lp^{F-TH} \le (lp^{F1})^2 = (\frac{1}{2^{24}})^2 = \frac{1}{2^{48}}.$$

마지막으로 정리 2.11을 사용하면 원하는 결과를 얻는다. ■

[정리 4.3]

FRACTAL 암호 알고리즘과 FRACTAL-TH 암호 알고리즘은 DC 및 LC에 대하여 똑같은 안전성을 가진다.

(증명)

FRACTAL과 FRACTAL-TH의 선형함수 $L1$, $L2$, $L3$, $G1$, $G2$, $G3$에 대하여,

$L1 = G1 \circ G2 \circ G3$, $L2 = G2 \circ G3$, $L3 = G3$이므로 FRACTAL과 FRACTAL-TH는 동등한 구조를 갖게 되고 따라서 정리가 성립한다. ■

4.2 다른 공격법에 대한 안전성

불능 차분 공격, 연관 키 공격, 약한 키 공격, 대수적 구조를 이용한 공격 등, DC와 LC 이외의 다른 공격에 대한 FRACTAL 암호 알고리즘의 안전성을 간략히 살펴보자.

4.2.1 불능 차분 공격

Feistel구조는 F-함수의 선택에 상관없이 다음과 같은 불능 차분이 존재한다.

$$(a, 0) \not\to (a, 0)$$

그러나 FRACTAL 알고리즘은 8-라운드 구조이며 각 라운드에 사용되는 부분 키의 크기가 크기 때문에 불능 차분 공격이 불가능하다.

4.2.2 연관 키 공격과 약한 키 공격

FRACTAL 암호 알고리즘의 키 생성 알고리즘에서는 S-box라는 비선형 함수를 사용하며 각 라운드의 부분 키를 생성할 때 라운드에 해당되는 상수를 Xoring하는 과정을 수행한다. 따라서 약한 키(weak key)가 존재할 확률이 매우 낮고, 연관 키(related key) 공격이 어려울 것으로 기대된다.

4.2.3 대수적 구조를 이용한 공격

최근 수학적인 구조가 안전성에 영향을 미칠 수 있음이 지적되었는데, Ferguson[3] 등은 Rijndael을 대수식(algebraic formula)으로 나타낼 수 있으며 Rijndael의 안전성이 특정 형태의 대수 방정식을 푸는 문제의 어려움에 기인함을 보였다.

이러한 대수적 구조를 이용한 공격에도 잘 견딜 수 있도록 하기 위하여, FRACTAL 암호 알고리즘에서는 비선형 위수가 가장 큰 것으로 알려진 유한 체 위에서의 역수 함수를 S-box로 사용하였다. 또한 선형 함수를 정의할 때 사용한 유한 체 GF_1과 S-box를 정의할 때 사용한 유한 체 GF_2를 서로 다른 기약 다항식으로부터 유도하였다.

4.3 FRACTAL의 구현 및 효율성

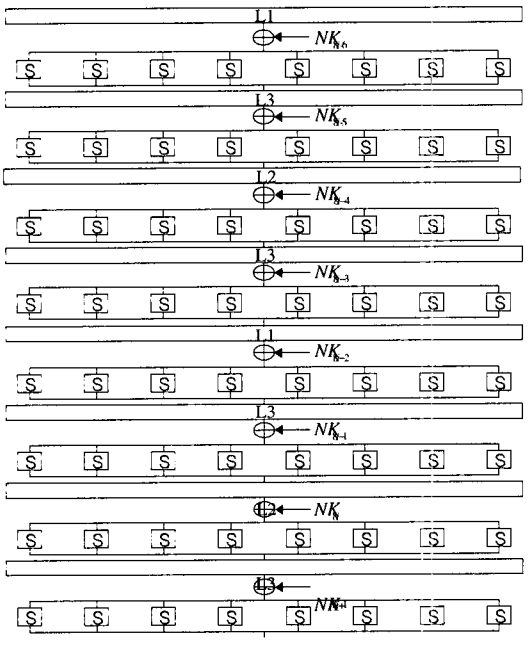
FRACTAL 블록 암호 알고리즘의 설계에 있어서 가장 중요시 된 점은 안전성의 증명에 있었다. 따라서 그 구현 속도는 다른 알고리즘, 특히 AES 후보

알고리즘에 비해서 약간 저하된 것으로 나타나고 있다. 그러나 그 내부 연산으로 바이트 단위의 연산을 사용한다는 점과 F-함수의 각 입력 바이트들이 거의 동등하게 사용되기 때문에 병렬화가 가능하다는 점 등은 FRACTAL이 8비트 연산을 사용하는 환경(예를 들어 스마트 카드)에서는 빠른 구현이 가능하다는 것을 보여주고 있다. 또한 AES로 채택이 된 Rijndael[2]에서 사용한 32비트 테이블 참조법(table look-up)을 사용하여 32비트 연산 환경(예를 들어 개인용 컴퓨터 등)에서도 빠른 구현이 가능하다.

4.3.1 32 비트 테이블 참조법을 이용한 FRACTAL의 빠른 구현법

FRACTAL의 i라운드 F-함수에 사용되는 부분키 $K_{8i-6},\ldots,K_{8i+1}$에 대하여 새로운 부분키 $NK_{8i-6},\ldots,NK_{8i+1}$를 다음과 같이 정의하고 [그림 8]에 정의된 New-F 함수를 생각하자 :

$$NK_{8i-6}=L1(K_{8i-6}),\quad NK_{8i-5}=L3(K_{8i-5}),$$
$$NK_{8i-4}=L2(K_{8i-4}),\quad NK_{8i-3}=L3(K_{8i-3}),$$
$$NK_{8i-2}=L1(K_{8i-2}),\quad NK_{8i-1}=L3(K_{8i-1}),$$
$$NK_{8i}=L2(K_{8i})\quad NK_{8i+1}=L3(K_{8i+1})$$



[그림 8] FRACTAL의 New-F 함수

선형함수 L에 대하여 $L(A\oplus B)=L(A)\oplus L(B)$이 성립함을 이용하면 F와 New-F 함수는 같은 함수임을 쉽게 알수 있다. 즉 임의의 64비트 입력값 M에 대하여 다음이 성립한다:

$$F(M,K_{8i-6},\ldots,K_{8i+1})$$
$$=New-F(M,NK_{8i-6},\ldots,NK_{8i+1})$$

이제 64비트 부분키 K를 xoring 해주는 함수를 f_K라 하고 (즉 임의의 64비트 입력값 M에 대하여 $f_K(M)=M\oplus K$이다.) S-box를 8번 병렬로 사용하는 함수를 S^8라 하면 New-F 함수는 다음과 같이 쓰여질 수 있다.

$$New-F=S^8\circ f_{NK_{8i+1}}\circ g(8i,L3)\circ g(8i-1,L2)$$
$$\circ g(8i-2,L3)\circ g(8i-3,L1)\circ g(8i-4,L3)$$
$$\circ g(8i-5,L2)\circ g(8i-6,L3)\circ L1$$

(여기서 정수 i와 선형함수 L에 대하여 $g(i,L)=L\circ S^8\circ f_{NK_i}$ 이다.)

따라서 임의의 정수 i에 대하여 $g(i,L1)$, $g(i,L2)$, $g(i,L3)$를 빠르게 구현할 수 있으면 FRACTAL의 F-함수도 빠르게 구현이 됨을 알 수 있다.

가. $g(i,L1)$의 빠른 구현

$X=(x_1,\ldots,x_8)$, $Y=(y_1,\ldots,y_8)$을 각각 $g(i,L1)$의 입력값과 출력값이라 하고, $Y_1=(y_1,y_2,y_3,y_4)$, $Y_2=(y_5,y_6,y_7,y_8)$, $NK_i=(k_1,\ldots,k_8)$이라 하자. 그러면 다음이 성립한다:

$$Y_1=S(x_1\oplus k_1)\begin{pmatrix}1\\1\\1\\1\end{pmatrix}\oplus S(x_2\oplus k_2)\begin{pmatrix}1\\y\\1\\y\end{pmatrix}$$
$$\oplus S(x_3\oplus k_3)\begin{pmatrix}1\\1\\y\\y\end{pmatrix}\oplus S(x_4\oplus k_4)\begin{pmatrix}1\\y\\y\\y^2\end{pmatrix}$$
$$\oplus S(x_5\oplus k_5)\begin{pmatrix}1\\1\\1\\1\end{pmatrix}\oplus S(x_6\oplus k_6)\begin{pmatrix}1\\y\\1\\y\end{pmatrix}$$
$$\oplus S(x_7\oplus k_7)\begin{pmatrix}1\\1\\y\\y\end{pmatrix}\oplus S(x_8\oplus k_8)\begin{pmatrix}1\\y\\y\\y^2\end{pmatrix}$$

$$Y_2 = S(x_1 \oplus k_1)\begin{pmatrix}1\\1\\1\\1\end{pmatrix} \oplus S(x_2 \oplus k_2)\begin{pmatrix}1\\y\\1\\y\end{pmatrix}$$
$$\oplus S(x_3 \oplus k_3)\begin{pmatrix}1\\1\\y\\y\end{pmatrix} \oplus S(x_4 \oplus k_4)\begin{pmatrix}1\\y\\y\\y^2\end{pmatrix}$$
$$\oplus S(x_5 \oplus k_5)\begin{pmatrix}y\\y\\y\\y\end{pmatrix} \oplus S(x_6 \oplus k_6)\begin{pmatrix}y\\y^2\\y\\y^2\end{pmatrix}$$
$$\oplus S(x_7 \oplus k_7)\begin{pmatrix}y\\y\\y^2\\y^2\end{pmatrix} \oplus S(x_8 \oplus k_8)\begin{pmatrix}y\\y^2\\y^2\\y^3\end{pmatrix}$$

이제 256개의 32비트 워드로 이루어진 4개의 table SS_1, SS_2, SS_3, SS_4를 다음과 같이 정의하자 : 8비트 입력값 a에 대하여

$$SS_1(a) = \begin{pmatrix}S(a)\\S(a)\\S(a)\\S(a)\end{pmatrix}, \quad SS_2(a) = \begin{pmatrix}S(a)\\y*S(a)\\S(a)\\y*S(a)\end{pmatrix},$$
$$SS_3(a) = \begin{pmatrix}S(a)\\S(a)\\y*S(a)\\y*S(a)\end{pmatrix} \quad SS_4(a) = \begin{pmatrix}S(a)\\y*S(a)\\y*S(a)\\y^2*S(a)\end{pmatrix}$$

그러면 $z_i = x_i \oplus k_i$, $i=1,\ldots,8$에 대하여 다음이 성립함을 알 수 있다.

$$Y_1 = SS_1(z_1) \oplus SS_2(z_2) \oplus SS_3(z_3) \oplus SS_4(z_4)$$
$$\oplus SS_1(z_5) \oplus SS_2(z_6) \oplus SS_3(z_7) \oplus SS_4(z_8)$$

$$Y_2 = SS_1(z_1) \oplus SS_2(z_2) \oplus SS_3(z_3) \oplus SS_4(z_4) \oplus$$
$$y*(SS_1(z_5) \oplus SS_2(z_6) \oplus SS_3(z_7) \oplus SS_4(z_8))$$

나. $g(i, L2)$의 빠른 구현

$X=(x_1,\ldots,x_8)$ $Y=(y_1,\ldots,y_8)$을 각각 $g(i, L1)$의 입력값과 출력값이라 하고, $Y_1=(y_1, y_2, y_3, y_4)$, $Y_2=(y_5, y_6, y_7, y_8)$ $NK_i=(k_1,\ldots,k_8)$, $z_i = x_i \oplus k_i$, $i=1,\ldots,8$이라 하자. 그러면 다음이 성립한다:

$$Y_1 = SS_1(z_1) \oplus SS_2(z_2) \oplus SS_3(z_3) \oplus SS_4(z_4),$$
$$Y_2 = SS_1(z_5) \oplus SS_2(z_6) \oplus SS_3(z_7) \oplus SS_4(z_8).$$

다. $g(i, L3)$의 빠른 구현

$X=(x_1,\ldots,x_8)$, $Y=(y_1,\ldots,y_8)$을 각각 $g(i, L3)$의 입력값과 출력값이라 하고, $Y_1=(y_1, y_2, y_3, y_4)$, $Y_2=(y_5, y_6, y_7, y_8)$, $NK_i=(k_1,\ldots,k_8)$, $z_i = x_i \oplus k_i$, $i=1,\ldots,8$이라 하자. 또한 256개의 32비트 워드로 이루어진 4개의 table SS_5, SS_6, SS_7, SS_8을 다음과 같이 정의하자: 8비트 입력값 a에 대하여

$$SS_5(a) = \begin{pmatrix}S(a)\\S(a)\\0\\0\end{pmatrix} \quad SS_6(a) = \begin{pmatrix}S(a)\\y*S(a)\\0\\0\end{pmatrix},$$
$$SS_7(a) = \begin{pmatrix}0\\0\\S(a)\\S(a)\end{pmatrix} \quad SS_8(a) = \begin{pmatrix}0\\0\\S(a)\\y*S(a)\end{pmatrix}$$

그러면 다음이 성립한다:

$$Y_1 = SS_1(z_1) \oplus SS_2(z_2) \oplus SS_3(z_3) \oplus SS_4(z_4),$$
$$Y_2 = SS_1(z_5) \oplus SS_2(z_6) \oplus SS_3(z_7) \oplus SS_4(z_8)$$

따라서, 256개의 32비트 워드를 갖는 8개의 table을 이용하여 $g(i, L1)$, $g(i, L2)$, $g(i, L3)$의 구현을 빠르게 할 수 있으며 이를 이용하여 FRACTAL의 F-함수도 빠르게 구현할 수 있다.

4.3.2 구현 속도

FRACTAL 알고리즘을 비쥬얼 C++로 구현하여 Pentium Pro 500MHz에서 수행한 결과, 키 생성과 암호화에 각각 5347 cycle, 2718 cycle이 소요되었으며, 암호화 속도는 23.5 Mbps로 측정되었다. Feistel 구조의 2라운드를 SPN 구조의 1라운드와 계산 복잡도가 비슷하다고 가정하고 FRACTAL 알고리듬의 F함수가 8라운드로 이루어졌음을 고려하면 8라운드 FRACTAL 알고리듬은 32라운드 SPN 구조의 계산 복잡도를 가지는 것으로 생각할 수 있으며 따라서 FRACTAL의 속도는 AES 후보중 32라운드 SPN 구조를 가지는 Serpent의 속도와 비슷할 것으로 예측이 된다.

V. 결 론

이 논문에서 우리는 새로운 블록 암호 알고리즘인 FRACTAL를 제안하였다. FRACTAL은 128비트 대칭키 블록 암호 알고리즘으로서 128비트 키를 사용하는(또는 더 큰 키를 사용할 수 있는) 8-라운드

Feistel 구조로 이루어 져 있고, 차분 공격 및 선형 공격에 대한 안전성을 증명할 수 있도록 라운드 함수 F에 반복 구조를 사용하였으며, 확산 효과를 위하여 세 가지 종류의 선형 변환을 도입하였다. 또한 대부분의 연산을 바이트 단위로 수행하게 하여 계산의 효율성을 높였으며, 선형함수와 S-box는 유한 체 위에서의 연산을 주로 사용하도록 하였다.

FRACTAL 알고리즘은 차분 공격과 선형 공격 이외의 여러 가지 다른 공격에도 어느 정도 안전함을 보였다. 그러나 이미 알려진 다양한 공격 방법뿐만 아니라 아직 알려지지 않은 미지의 공격 방법에 대한 안전성은 더 연구되어야 할 것이다.

참 고 문 헌

[1] K. Aoki and K. Ohta, "Strict Evaluation of the Maximum Average of Differential Probability and the Maximum Average of Linear Probability", IEICE Trans. E80-A(1), pp. 2~8, 1997.

[2] J. Daemen, L.R. Knudsen, and V. Rijmen, "AES Proposal: Rijndael", 2000.

[3] N. Ferguson, R. Schroeppel, D. Whiting, "A simple algebraic representation of Rijndael", Proc. SAC '01, 2001.

[4] Y. Kaneko, F. Sano and K. Sakurai, "On Provable Security against Differential and Linear Cryptanalysis in Generalized Feistel Ciphers with Multiple Random Functions", Proc. SAC '97, 1997.

[5] J. Kilian, P. Rogaway, "How to protect DES against exhaustive key search", Advances in Cryptology - Crypto '96, LNCS 1109, pp. 252~267, Springer-Verlag, 1996.

[6] X. Lai, J .L. Massey and S. Murphy, "Markov Ciphers and Differential Cryptanalysis", In Advances in Cryptology - Eurocrypt '91, LNCS 547 pp. 17~38, Springer-Verlag, 1991.

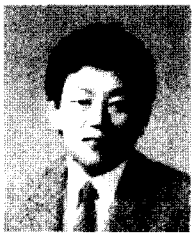
[7] M. Matsui, "New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis", In Fast Software Encryption '96, LNCS 1039, pp. 205~218, Springer-Verlag, 1996.

[8] K. Nyberg, "Linear Approximation of Block Ciphers", In Advances in Cryptology - Eurocrypt '94, LNCS 950, pp. 439~444, Springer-Verlag, 1995.

[9] K. Nyberg and L. Knudsen, "Provable Security against a Differential Attack", J. Cryptology 8(1), pp. 27~37, 1995.

〈著者紹介〉

김 명 환 (Myung-Hwan Kim) 정회원
1977년 2월 : 서울대학교 수학과 졸업
1985년 8월 : OHIO 주립대학교 대학원 수학과 박사
1986년 9월~1989년 1월 : 한국과학기술대학 수학과 조교수
1989년 2월~현재 : 서울대학교 수리과학부 교수
〈관심분야〉 정수론, 표현론, 암호학, 코딩이론



이 인 석 (In-Sok Lee)
1979년 2월 : 서울대학교 수학과 졸업
1986년 5월 : Yale 대학교 대학원 수학과 박사
1986년 9월~현재 : 서울대학교 수리과학부 교수
〈관심분야〉 표현론, 암호학, 코딩이론

백 유 진 (Yoo-Jin Baek)
1997년 2월 : 서울대학교 수학과 졸업
1999년 2월 : 서울대학교 대학원 수학과 석사
1993년 3월~현재 : 서울대학교 수리과학부 박사과정
〈관심분야〉 암호학

김 우 환 (Woo-Hwan Kim)
1998년 2월 : 서울대학교 수학과 졸업
2000년 2월 : 서울대학교 대학원 수학과 석사
2000년 3월~현재 : 서울대학교 수리과학부 박사과정
〈관심분야〉 암호학

강 성 우 (Sung-Woo Kang)
1996년 2월 : 중앙대학교 수학과 학사
2001년 8월 : 서울대학교 대학원 수학과 석사
2000년 12월~현재 : 한국정보보호진흥원 기술단/암호기술팀(연구원)
〈관심분야〉 암호학