情報保護學會論文誌
第11券 第5號, 2001. 10

# 27라운드 SKIPJACK에 대한 포화 공격*

황 경 덕**, 이 원 일**, 이 성 재***, 이 상 진**, 임 종 인**

# Saturation Attacks on the 27-round SKIPJACK

Kyungdeok Hwang**, Wonil Lee**, Sungjae Lee***, Sangjin Lee**, Jongin Lim**

요 약

본 논문에서는 포화 공격(saturation attack)[7]을 SKIPJACK에 적용해 본다. 우리가 제시하는 포화 공격의 핵심은 SKIPJACK에 대한 16라운드 distinguisher의 구성 방법에 있으며 이것은 18라운드(5~22)와 23라운드(5~27) SKIPJACK에 대한 공격을 가능하게 한다. 또한 16라운드 distinguisher를 기반으로 하여 20라운드 distinguisher를 구성할 수 있는데 이것은 22라운드(1~22)와 27라운드(1~27) SKIPJACK에 대한 공격을 가능하게 한다. 27라운드 SKIPJACK에 대한 공격에 필요한 선택 평문은 $2^{50}$개이며 이 때의 공격 복잡도는 $3 \cdot 2^{75}$이다.

ABSTRACT

This paper describes saturation attacks on reduced-round versions of SKIPJACK. To begin with, we will show how to construct a 16-round distinguisher which distinguishes 16 rounds of SKIPJACK from a random permutation. The distinguisher is used to attack on 18(5~22) and 23(5~27) rounds of SKIPJACK. We can also construct a 20-round distinguisher based on the 16-round distinguisher. This distinguisher is used to attack on 22(1~22) and 27(1~27) rounds of SKIPJACK. The 80-bit user key of 27 rounds of SKIPJACK can be recovered with $2^{50}$ chosen plaintexts and $3 \cdot 2^{75}$ encryption times.

**keyword** : *Saturation attack, Skipjack, block cipher, cryptanalysis*

## Ⅰ. 서 론

SKIPJACK[14,15]는 64비트 블록 암호 알고리즘으로써 1993년에 개발되었지만 미국의 정책에 의하여 그 세부적인 구조는 공개되지 않고 있었다. 그 후 1998년, NSA에 의하여 세부적인 구조가 처음 발표되게 된다. SKIPJACK의 세부적인 구조가 발표된 이후 SKIPJACK을 분석하기 위한 몇 가지 접근들이 이루어졌다. SKIPJACK에 대한 첫 번째 분석 결과는 Biham 등[2]에 의하여 발표되었다. 그들은 먼저 G함수의 구체적인 특성과 G함수 내부의 S-box에 대한 성질들을 연구하였다. 그들은 또한 축소된 SKIPJACK에 대한 부정 차분 공격을 고려하였다[3,4]. 그 후 Biham 등[1]은 축소된 SKIPJACK에 대한 불능 차분 공격을 제안하였으며, 이 공격은 최대 31라운드로 구성된 SKIPJACK의 마스터 키를 전수조사보다 빠르게 복구할 수 있음을 보여주었다. 이 결과는 지금까지 알려진 SKIPJACK에 대한 공격 결과들 중 가장 좋은 결과가 되고 있다. Knudsen 등[6]도 SKIPJACK에 대한 몇 가지 형태의 공격을 제안하였다. 그들이 연구한 주된 공격 방법은 부정 차분 공격이었으며, 부메랑 공격의 효율성에 대해서도

* 본 연구는 정보통신부의 대학 S/W 연구센터 지원 사업의 연구 결과
** CIST({kdhwang, wonil, sangjin, jilim}@cist.korea.ac.kr), Korea University
*** 한국정보보호진흥원(sjlee@kisa.or.kr)

언급하였다. 그러나 이전에 발표된 Biham 등[1]의 31라운드 SKIPJACK에 대한 불능 차분 공격 결과를 향상시키지는 못하였다. 게다가 최근에 Granboulan[12]은 Knudsen 등[6]의 차분 공격 방법이 몇 가지 결점들을 내포하고 있음을 지적하였다.

본 논문에서는 현재까지 SKIPJACK에 대한 분석에 이용된 공격 법들과는 다른 '포화공격'이라 불리우는 공격법을 몇 가지 축소된 라운드를 가지는 SKIPJACK에 적용해보고자 한다. SKIPJACK은 총 32라운드로 구성되어 있으며 각 라운드 당 한번의 16비트 G함수 연산을 사용한다. 이러한 G함수 연산 과정은 32비트의 부분키에 의하여 결정되므로 대부분의 공격자는 각 라운드의 G함수를 알 수 없다. 그러나 여기서 주목할 것은 G함수가 전단사 함수라는 사실이다. 이 같은 사실로부터 G함수의 정의역 또는 공역에 속하는 모든 서로 다른 $2^{16}$개의 원소들을 XOR하면 0이 된다는 성질을 얻을 수 있다. 본 논문이 제시하는 공격 아이디어의 시발점은 바로 이러한 사실이 될 것이다. 즉 우리가 연구한 바에 의하면 G함수가 전단사 함수라는 사실은 최고 27라운드 SKIPJACK에 대한 포화 공격의 실마리를 제공하게 된다.

포화 공격은 선택 평문 공격이다. 우리가 제시하는 포화 공격의 핵심은 SKIPJACK에 대한 16라운드 distinguisher의 구성 방법에 있으며 이것은 18라운드(5~22)와 23라운드(5~27) SKIPJACK에 대한 공격을 가능하게 한다. 여기서 주의할 것은 우리가 제일 먼저 구성하고자 하는 16라운드 distinguisher는 5라운드부터 20라운드까지의 distinguisher를 의미한다는 것이다(즉 앞의 4개의 라운드와 뒤의 12개 라운드를 제외한 SKIPJACK에 대한 distinguisher를 의미한다). 그러므로 이러한 16라운드 distinguisher를 이용하여 공격할 대상인 18라운드와 23라운드 SKIPJACK은 각각 5~22라운드, 5~27라운드 SKIPJACK을 의미하는 것이다. 이러한 18라운드(5~22)와 23라운드(5~27) SKIPJACK에 대한 포화 공격에 요구되는 선택 평문의 개수는 각각 $2^{17}$, $2^{18}$개이며, 이 때 공격 복잡도의 단위를 한번의 SKIPJACK 암호과정으로 정의하면 공격 복잡도는 각각 $2^{44}$, $3\cdot 2^{75}$이 된다.

또한 16라운드 distinguisher를 기반으로 하여 20라운드 distinguisher를 구성할 수 있는데 이것은 22라운드(1~22)와 27라운드(1~27) SKIPJACK에 대한 공격을 가능하게 한다. 이 경우 20라운드 distinguisher는 16라운드 distinguisher의 윗부분으로 4라운드 확장하는 방법을 통하여 얻어질 수 있다. 그러므로 이러한 20라운드 distinguisher는 1라운드부터 20라운드까지의 distinguisher를 의미한다. 따라서 이러한 20라운드 distinguisher를 이용하여 공격할 대상인 22라운드와 27라운드 SKIPJACK은 각각 1~22라운드, 1~27라운드 SKIPJACK을 의미하게 된다. 이러한 22라운드와 27라운드 SKIPJACK에 대한 포화 공격에 요구되는 선택 평문의 개수는 각각 $2^{49}$, $2^{50}$개이며, 이 때의 공격 복잡도는 각각 $2^{44}$, $3\cdot 2^{75}$이 된다.

'포화 공격'이라는 용어는 전·후 whitening과정을 포함한 7라운드 TWOFISH와 앞의 whitening 과정만을 포함한 8라운드 TWOFISH에 대한 공격을 Lucks[7]에 의해서 처음 등장하였다. 그러나 Lucks가 제시한 '포화 공격'이라 불리우는 공격 형태는 이미 'Square 공격'이라는 용어로써 알려져 있었다. Square 공격[5]은 SQUARE 블록 암호 개발자들에 의하여 자신들이 개발한 알고리즘에 대한 분석방법의 하나로 처음 소개되었다. 물론 Lucks가 Square-like 구조에만 적용되어 왔던 'Square 공격'을 좀 더 일반화된 정의를 이용하여 처음으로 이와 다른 Feistel-like 구조를 가지는 TWOFISH에 대한 분석을 시도하고 있다는 점에서 의미가 있으나 공격 아이디어는 근본적으로 이와 동일하다. 그러므로 '포화 공격'이라는 용어는 'Square 공격'이라는 용어의 일반화로 볼 수 있으며 의미 전달에 있어서 전자가 효율적이므로 앞으로 이러한 공격 형태는 '포화 공격'이라는 용어를 사용하여 표현하기로 약속한다. 그러므로 AES 후보들 가운데 Square-like구조를 가진 CRYPTON[8,9]과 RIJNDAEL[10]에 적용된 공격 방법들[11,13] 역시 '포화 공격'이라는 범주에 속하게 된다. 또한 본 논문에서 제시하는 SKIPJACK에 대한 공격도 이러한 형태에 속하므로 '포화 공격'이라는 용어를 사용하기로 한다.

여기서 지적하고 싶은 것은, 본 논문에서 제시되는 포화공격이 Knudsen 등[6]과 마찬가지로 Biham 등[1]이 발표한 31라운드 SKIPJACK에 대한 불능 차분 공격 결과를 향상시키지는 못하였다는 점이다. 그러나 포화 공격 관점에서의 SKIPJACK에 대한 분석 결과를 최초로 제시했다 점에서 본 논문의 의미를 찾을 수 있다.

본 논문의 구성은 다음과 같다. 우선 2절에서는 이론 전개를 위해 필요한 용어들을 정의하고 이 용어

들의 활용 방향을 제시한다. 3절에서는 SKIPJACK에 대한 간략한 소개를 한다. 4절에서는 distinguisher의 구성 방법에 대하여 언급한다. 여기서는 먼저 16라운드 distinguisher의 구성 과정을 보인 후 이를 기반으로 하여 20라운드 distinguisher의 구성 방법을 설명한다. 5절에서는 먼저 4절에서 제시한 16라운드 distinguisher를 이용하여 18라운드(5~22)와 23라운드(5~27) SKIPJACK에 대한 공격이 가능함을 보인다. 또한 4절에서 제시한 20라운드 distinguisher를 이용하여 22라운드(1~22)와 27라운드(1~27) SKIPJACK에 대한 공격이 가능함을 보인다. 끝으로 6절에서 SKIPJACK에 대한 기존의 공격 결과와 포화 공격 결과를 비교한 후 결론을 언급한다.

## Ⅱ. 표기법

- 16비트 데이터 채널(16-bit data channel)

SKIPJACK은 64비트 블록을 16비트 단위씩 4개로 나누어 암호화를 진행시키므로 총 4개의 16비트 평문 블록이 각각 4개의 16비트 데이터 채널을 통화여 암호화된다고 생각할 수 있다.

- 워드(word)

위의 16비트 데이터 채널을 통과하는 값들을 '워드'라고 부르기로 하자. 즉 하나의 워드는 16비트 데이터를 의미하게 된다.

- 기호 1

$I_n$은 모든 $n$비트 수열들의 집합을 의미한다.

- 기호 2

$(\alpha^i, \beta^i, \gamma^i, \delta^i)$는 4개의 16비트 데이터 채널을 지나는 64비트 데이터를 의미한다. 여기서 윗 첨자를 가진 그리스 소문자들은 고정된 16비트 상수를 의미한다. 이 때 윗 첨자 $i$는 이러한 64비트 데이터가 $i$라운드 SKIPJACK의 입력임을 뜻한다. 따라서 우리의 표기법에 의하면 $i$라운드 SKIPJACK 출력은 $(\alpha^{i+1}, \beta^{i+1}, \gamma^{i+1}, \delta^{i+1})$이 된다. 그러므로 $n$라운드 SKIPJACK에 대한 하나의 입력 평문 $(\alpha^1, \beta^1, \gamma^1, \delta^1)$은 $(\alpha^i, \beta^i, \gamma^i, \delta^i)$, $[2 \le i \le n+1]$와 같이 여러 형태로 변화하며 4개의 16비트 데이터 채널을 지난 후 암호문을 형성하게 된다.

- 기호 3

$(A^i, \beta^i, \gamma^i, \delta^i)$은 4개의 16비트 데이터 채널을 지나는 64비트 데이터들의 집합을 의미한다. 여기서 윗 첨자를 가진 그리스 소문자들은 고정된 16비트 상수를 의미한다. 또한 굵은 알파벳 대문자는 $I_{16}$의 부분집합을 의미한다. 또한 윗 첨자 $i$는 이러한 64비트 데이터 집합이 $i$라운드 SKIPJACK의 입력 집합임을 뜻한다. 즉 $(A^i, \beta^i, \gamma^i, \delta^i) = \{(\alpha^i, \beta^i, \gamma^i, \delta^i) \in I_{16}^4 \mid \alpha^i \in A^i\}$이 성립한다. 이 때 집합 $(A^i, \beta^i, \gamma^i, \delta^i)$의 원소의 개수는 $A^i$의 원소의 개수와 같게 된다. 또한 $(A^i, B^i, \gamma^i, \delta^i)$도 비슷하게 정의 할 수 있으며 $|A^i| = k$, $|B^i| = l$ 라고 하면 이 집합의 원소의 개수는 $kl$과 같게 된다. $(A^i, B^i, C^i, D^i)$등도 위와 비슷하게 정의 할 수 있다. 따라서 $n$라운드 SKIPJACK에 대한 입력 평문들의 집합 $(A^1, \beta^1, \gamma^1, \delta^1)$은 $(A^i, B^i, C^i, D^i)$, $[2 \le i \le n+1]$와 같이 여러 형태의 집합으로 변화하며 4개의 16비트 데이터 채널을 지난 후 암호문 집합을 형성하게 된다. 물론 여기서 굵은 알파벳 대문자로 표현된 $I_{16}$의 부분집합들은 원소의 개수가 하나일 수도 있고 그 이상일 수도 있으며 $I_{16}$자체가 될 수도 있다. 다른 형태의 입력 평문 집합에 대한 처리 과정도 비슷하게 설명될 수 있다.

- 다중 집합(multiset)

본 논문에서는 distinguisher의 구성방법을 16비트 데이터 채널의 데이터 처리과정을 통해 효율적으로 설명하기 위하여 '다중집합'이라는 용어를 이용하기로 한다. '다중집합(multiset)'이란 일반적인 집합의 정의와 달리 그 집합 속에 반복되는 원소를 같은 원소로 취급하지 않고 다른 원소로 인정하는 집합을 의미한다.

- 포화 집합(saturated set)

집합 $M$을 $I_w$의 원소들로 구성되어 있으면서 $k \cdot 2^w$개의 원소를 가지는 다중집합이라고 하자. 만일 $I_w$의 모든 원소들이 각각 정확하게 $k$번씩 다중집합 $M$에 나타난다면 이 때 $M$을 '$k$-포화집합'이라고 정의한다. 또한 이 때 집합 M이 '$k$-포화되었다'고 말한다. 본 논문에서는 $w=16$인 경우만을 고려한다. 그러므로 $k=1$인 경우에는 1-포화집합은 $I_{16}$과 같은 집합이 된다. 앞으로 1-포화집합은 앞의 1을 생략하여 포화집합이라고 표기하기로 하고 이 때 $M$이 포화되었다고 말한다.

• 균일 집합(balanced set)

집합 $N$을 $I_w$의 원소들로 구성되어 있는 집합 ($N \subset I_w$)이라고 하자. 만일 $N$의 모든 원소들을 XOR한 값이 0이 된다면($\oplus_i x_i = 0, x_i \in N$) 이 때 $N$을 '균일집합'이라고 정의한다. 본 논문에서는 $w=16$인 경우만을 고려할 것이다. 이 정의에 의하면 임의의 $k$에 대하여 어떤 집합 $M$이 $k$-포화집합이면 $M$은 균일집합이 된다는 사실을 쉽게 알 수 있다.

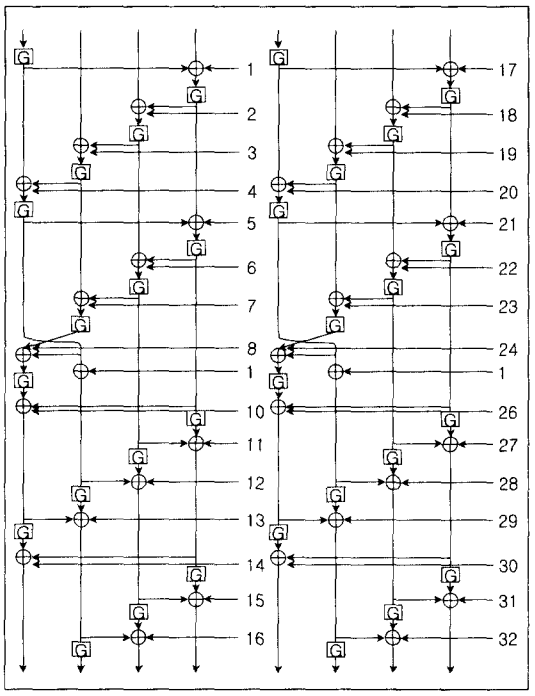
## Ⅲ. SKIPJACK 소개

64비트 알고리즘인 SKIPJACK은 A와 B라는 두 가지 형태의 변환규칙(라운드)으로 구성된 32라운드 블록 암호이다. 각각의 라운드는 비 선형 요소인 G 함수를 포함한 선형 피드백 쉬프트 레지스터(Linear feedback shift register)의 형태로 표현될 수 있다. SKIPJACK은 변환 규칙 A와 B를 8라운드씩 엇갈려 2번 반복 사용하여 32라운드를 구성한다. 이 때 이것을 일반적으로 $A^8B^8A^8B^8$와 같이 표현한다. 변환 규칙 A와 B는 〔표 1〕에 소개되었으며 이 때 윗 첨자 $k$는 라운드 수를 나타내며, 아랫 첨자가 사용된 $w_i$는 $i$번째 16비트 데이터 채널 안의 워드를 의미한다. *counter*는 라운드마다 고정된 상수(1~32)를 나타낸다.

G함수는 4라운드 Feistel 구조로 이루어진 전단사 함수이며 이 때의 라운드 함수로는 F함수를 사용한다. F함수는 8×8비트의 S-box(F table)에 의하여 구성되며 각 라운드마다 8비트의 부분키가 이용된다. 그러므로 G함수는 부분키로 총 32비트의 키를 필요로 하게된다. SKIPJACK에는 80비트 마스터키가 사용되며 키 스케줄 과정은 마스터키를 순환시키면서 32비트씩 잘라내어 각 라운드의 부분키를

〔표 1〕 변환 규칙 A와 B.

| | |
|---|---|
| 변환 규칙 A | $w_1^{k+1} = G^k(w_1^k) \oplus w_4^k \oplus counter^k$<br>$w_2^{k+1} = G^k(w_1^k)$<br>$w_3^{k+1} = w_2^k$<br>$w_4^{k+1} = w_3^k$ |
| 변환 규칙 B | $w_1^{k+1} = w_4^k$<br>$w_2^{k+1} = G^k(w_1^k)$<br>$w_3^{k+1} = w_1^k \oplus w_4^k \oplus counter^k$<br>$w_4^{k+1} = w_3^k$ |




〔그림 1〕 SKIPJACK 구조

생성한다. 그러므로 결국 5라운드마다 순환하여 마스터키가 영향을 미치게 되는 것이다[14].

〔표 1〕에 묘사된 라운드 함수들을 쉬프트 레지스터 안의 데이터를 그대로 유지하면서 서로 복잡하게 꼬여진 형태를 풀면 〔그림 1〕과 같이 간단하게 표현된다. 따라서 논문에서는 〔그림 1〕을 이용하여 이론을 전개할 것이다. 또 변환 규칙 A와 B에는 *counter*가 사용되지만 이것은 SKIPJACK에 대한 포화공격에 전혀 영향을 미치지 않기 때문에 실제로 본 논문에서는 *counter*를 생략하고 이론을 전개한다.

## Ⅳ. distingusiher 구성

안전한 블록 암호는 마스터 키를 모르는 공격자에게는 랜덤한 전단사 함수(random permutation)처럼 보여야 한다는 것은 잘 알려진 사실이다. SKIPJACK에 대한 distinguisher를 구성한다는 것은 SKIPJACK을 랜덤한 전단사 함수로부터 구별 가능하게 하는 도구를 구성한다는 의미와 같다. 본 논문에서는 어떤 조건을 만족시키는 평문 집합을 선택할 경우, 그에 대응하는 암호문 집합에는 랜덤한 전단사 함수인 경우에는 일어날 수 없는 성질들이 발생함을 보일 것이다. 이러한 목적을 위하여 구성

된 distinguisher는 최대 27라운드 SKIPJACK에 대한 포화공격을 실현시킨다. 여기서는 먼저 16라운드 distinguisher의 구성 과정을 보인 후 이를 기반으로 하여 20라운드 distinguisher의 구성 방법을 설명한다.

### 4.1 16 라운드 distinguisher

본 소절에서는 16라운드(5~20) distinguisher의 구성 방법을 설명한다. 이것은 다음절에서 18라운드(5~22)와 23라운드(5~27) SKIPJACK에 대한 공격을 가능하게 할 것이다. 여기서 우리는 16라운드 distinguisher의 첫 번째 라운드를 원래 구조의 순서를 따라 5라운드라고 언급하기로 약속한다. [그림 1]을 기본 구조로 하여 표현한 [그림 2]는 16라운드 distinguisher에 대한 이해에 도움을 줄 것이다.

5라운드의 입력을 표현하는 4개의 16비트 데이터 채널 중 두 번째 데이터 채널만이 포화되도록 5라운드 입력 평문 집합을 $(\alpha^5, B^5, \gamma^5, \delta^5)$ 으로 구성한다. 이때 $B^5$는 $B^5 = I_{16}$이 성립하며, $\alpha^5, \gamma^5, \delta^5$는 임의의 16비트 상수가 되어도 무방하다. 5라운드 입력 평문 집합을 이렇게 구성할 경우 G함수가 전단사 함수라는 사실과 변환 규칙 A에 의하여 6, 7라운드를 거쳐 8라운드의 입력을 구성하게 되는 집합도 $(\alpha^8, B^8, \gamma^8, \delta^8)$과 같은 형태를 가지게 되며 또한 $B^8$은 포화집합이 되는 것이다. 우리는 물론 $\alpha^8, \gamma^8, \delta^8$가 어떤 값인지는 알 수 없으나 이것이 상수이라는 사실은 자명하다. 이제 이러한 집합은 8라운드를 거쳐 변환 규칙 B의 영향을 받게 되고 9라운드의 출력을 구성하게 되는 집합은 $(A^{10}, \beta^{10}, \gamma^{10}, \delta^{10})$의 형태를 가지게 된다. 여기서도 $A^{10}$은 포화집합이 된다. 이 집합은 8라운드의 입력을 구성하는 집합과는 포화 위치가 다르지만 하나의 워드만이 포화된다는 공통점을 가지고 있다.

[그림 2]를 참고로 하여 이런 식으로 분석해 보면 처음으로 포화되는 16비트 데이터 채널이 2개인 곳은 13라운드의 출력 부분임을 알 수 있다. 이때의 집합은 $(A^{14}, B^{14}, \gamma^{14}, \delta^{14})$의 형태를 가지며 $A^{14}$과 $B^{14}$는 포화 집합이 된다. 또한 처음으로 포화되는 워드가 3개가 되는 곳은 16라운드의 출력 부분임을 알 수 있다. 이때의 집합은 $(A^{17}, B^{17}, C^{17}, \delta^{17})$의 형태를 가지며 $A^{17}$, $B^{17}$과 $C^{17}$은 포화 집합이 된다. 그리고 처음으로 모든 워드가 포화되는 곳은 17라운드의 출력 부분임을 알 수 있다. 이때의 집합은 $(A^{18}, B^{18}, C^{18}, D^{18})$의 형태를 가지며 $A^{18}, B^{18}, C^{18}, D^{18}$은 모두 포화 집합이 된다.

18라운드 출력을 구성하는 집합은 $(A^{19}, B^{19}, C^{19}, D^{19})$의 형태를 가지며 $A^{19}, B^{19}, D^{19}$은 모두 포화 집합이 되지만 $C^{19}$는 일반적으로 $I_{16}$과 같다고 말할 수 없다. 왜냐하면 집합 $\{\gamma^{18} \oplus \delta^{19} \mid \gamma^{18} \in C^{18}, \delta^{19} \in D^{19}\}$가 항상 $I_{16}$과 같다고 볼 수 없기 때문이다. 그러나 참고로 이 집합은 포화집합이 되지는 못하지만 균일 집합은 된다. 19라운드 출력을 구성하는 집합인 $(A^{20}, B^{20}, C^{20}, D^{20})$의 성질을 보면 $A^{20}, D^{20}$은 모두 포화 집합이 되지만 위와 비슷한 이유로 $B^{20}, C^{20}$은 $I_{16}$과 같다고 말할 수 없다. 또한 이 두 집합은 일반적으로 균일 집합이 될 수도 없다. 이러한 사실을 [그림 2]에서는 $(A^{20}, ?, ?, D^{20})$와 같이 표시하고 있다.

본 소절에서 제시하는 16라운드 distinguisher의 마지막 라운드인 20라운드 출력을 구성하는 집합 $(A^{21}, B^{21}, C^{21}, D^{21})$의 성질을 보면 $D^{21}$은 $I_{16}$과 같으며 따라서 균일 집합이 된다. 위와 비슷한 이유로 $A^{21}, B^{21}, C^{21}$은 $I_{16}$과 같다고 말할 수 없다. 또한 이 세 집합은 균일 집합이 될 수도 없다. 이러한 사실을 [그림 2]에서는 $(?, ?, ?, D^{21})$와 같이 표시하고 있다.

요약하면 5라운드의 입력을 표현하는 4개의 16비트 데이터 채널 중 두 번째 데이터 채널만이 포화되도록 5라운드 입력 평문집합을 $(\alpha^5, B^5, \gamma^5, \delta^5)$의 형태로 구성하면 항상 확률 1로 19라운드 출력의 첫 번째 데이터 채널과 20라운드 출력의 네 번째 데이터 채널이 포화되며 균일한 성질을 가진다. 반면 랜덤한 전단사함수가 이러한 균일 성질을 만족할 확률은 $2^{-32}$으로 매우 작다. 포화공격이 18라운드(5~22)와 23라운드(5~27)에 적용되는 이유는 $A^{20}$과 $D^{21}$이 균일 집합이라는 사실에 기인한다. 21라운드의 출력을 구성하는 집합 $(A^{22}, B^{22}, C^{22}, D^{22})$의 성질을 보면 $A^{22}, B^{22}, C^{22}, D^{22}$ 모두 포화집합도 균일 집합도 아니므로 포화공격에 도움이 되지 못한다. 그러므로 21라운드 이상으로 확장한 distinguisher는 구성하지 못한다. 그러나 이러한 16라운드 distinguisher를 위쪽으로 4라운드 확장하여 20라운드 distinguisher를 구성하는 방법을 다음 소절에서 설명한다.

### 4.2. 20 라운드 distinguisher

본 소절에서는 16라운드(5~20) distinguisher

를 윗 부분으로 4라운드 확장하여 20라운드(1~20) distinguisher를 구성하는 방법을 소개한다. 이것은 다음절에서 소개할 22라운드(1~22)와 27라운드(1~27) SKIPJACK에 대한 포화공격을 가능하게 할 것이다. [그림 3]는 20라운드 distinguisher에 대한 이해에 도움을 줄 것이다. 여기서는 20라운드 distinguisher의 첫째 라운드는 원래 구조의 순서와 같이 1 라운드가 된다.

먼저 본 소절에서 이끌어낼 결론부터 언급하면 다음과 같다. 1라운드의 입력 평문 집합을 $2^{48}$개의 평문을 가지는 $(A^1, B^1, \gamma^1, D^1)$, $[A^1 = B^1 = D^1 = I_{16}]$으로 구성하면 이 집합에 대응하는 19, 20라운드 출력을 구성하는 집합 $(A^{20}, B^{20}, C^{20}, D^{20})$, $(A^{21}, B^{21}, C^{21}, D^{21})$에서 각각 $A^{20}$, $D^{21}$이 확률 1로 균일 집합이 된다. 반면 랜덤한 전단사함수가 이러한 균일 성질을 만족할 확률은 $2^{-32}$으로 매우 작다.

이 같은 결론은 앞 소절의 16라운드 distinguisher의 특성에 기인하여 도출된다. 이것을 증명하기 위하여 앞서 언급한 1라운드 입력 평문 집합을 $2^{32}$개의 부분집합들로 분할(partition)하여 설명하고자 한다. 구체적으로, 아래에 제시할 분할 방법에 의하여 분할된 부분집합들은 각각 그에 대응하는 4라운드 출력 집합이 앞 소절에서 제시한 16라운드 distinguisher의 5라운드 입력 평문 집합과 형태가 같아지게 된다. 따라서 이러한 16라운드 distinguisher의 특성에 기인하여, 분할된 각각의 부분집합들에 대응하는 19라운드 출력의 첫 번째 데이터 채널과 20라운드 출력의 네 번째 데이터 채널이 포화된다. 또한 $2^{32}$개의 분할된 각각의 집합들에 대하여 이러한 성질이 만족하므로 $2^{48}$개의 평문을 가진 1라운드의 입력 평문 집합에 대해서는 이에 대응하는 19라운드 출력의 첫 번째 데이터 채널과 20라운드 출력의 네 번째 데이터 채널이 각각 $2^{32}$-포화집합이 되게 된다. 그러므로 이 채널들은 균일 성질을 만족한다.

4라운드 출력 부분에 주목한 구체적인 분할 방법을 다음과 같이 제시한다([그림 3]는 이러한 분할 방법에 대한 이해에 도움을 줄 것이다). 4라운드 출력 부분 중 첫 번째와 네 번째 채널의 데이터를 $\alpha^5, \delta^5 \in I_{16}$로 고정시킨다. 그러면 이 때 $G(\gamma^1 \oplus \delta^5) = \gamma^5$가 성립하고 1라운드 입력 평문 집합 선택 시 $\gamma^1$이 결정되었으므로 $\gamma^5$는 $\delta^5$에 의존하여 결정된다. 이 때 $B^5(=I_{16})$의 원소 $\beta_i^5$가 결정되면 다음 식들에 의하여 $(\alpha_i^1, \beta_i^1, \delta_i^1)$쌍이 결정된다.

$$\alpha_i^1 = G^{-1}(\alpha^5 \oplus \beta_i^5)$$
$$\beta_i^1 = G^{-1}(\beta_i^5) \oplus \gamma^5$$
$$\delta_i^1 = (\alpha^5 \oplus \beta_i^5) \oplus G^{-1}(\delta^5)$$

이 때 G함수가 전단사함수이므로 $i \neq j$이면 $(\alpha_i^1, \beta_i^1, \delta_i^1) \neq (\alpha_j^1, \beta_j^1, \delta_j^1)$이 성립하며 $\beta_i^5$는 $I_{16}$의 원소이므로 총 $2^{16}$가지의 가능성이 있다. 그러므로 $\alpha^5, \delta^5$이 고정되면 이와 대응되는 서로 다른 $2^{16}$개의 원소를 가지는 집합 $\{(\alpha_i^1, \beta_i^1, \gamma^1, \delta_i^1) \mid 0 \le i \le 2^{16}-1\}$이 결정된다. 또한 역으로 이러한 집합에 대응하는 4라운드 출력을 구성하는 집합은 항상 $(\alpha^5, B^5, \gamma^5, \delta^5)$의 형태가 되며 이 때 $B^5$는 포화 집합 성질을 만족한다. 그러므로 이것은 앞 소절에서 제시한 16라운드 distinguisher의 입력 평문 집합과 같게 된다.

또한 여기서 주목할 것은 제일 먼저 고정시켰던 $\alpha^5, \delta^5$가 될 수 있는 가능성은 총 $2^{32}$개라는 것이다. 그러므로 총 $2^{32}$개의 서로 다른 집합 $\{(\alpha_i^1, \beta_i^1, \gamma^1, \delta_i^1) \mid 0 \le i \le 2^{16}-1\}_{0 \le j \le 2^{32}-1}$이 결정된다. 또한 앞에서 이러한 각각의 집합은 모두 서로 다른 $2^{16}$개의 원소를 가지고 있음을 언급하였다. 그리고 이러한 총 $2^{32}$개의 집합은 각각 $2^{48}$개의 입력 평문 집합 $(A^1, B^1, \gamma^1, D^1)$, $[A^1 = B^1 = D^1 = I_{16}]$의 부분 집합들임을 알 수 있으며 이 집합을 $2^{32}$개로 분할하고 있음을 쉽게 알 수 있다. 따라서 이상의 과정은 본 소절의 도입부에서 제시한 결론이 올바른 것임을 나타낸다.

## V. 포화 공격

### 5.1 16라운드 distinguisher를 이용한 포화 공격

#### 5.1.1 18라운드(5~22) SKIPJACK에 대한 포화 공격

4.1절에서 구성한 16라운드 distinguisher는 18라운드(5~22) SKIPJACK에 대한 포화공격을 가능하게 한다. 여기서 우리는 선택 평문 공격인 포화공격을 이용하여 21라운드 부분키 $K_{21}$과 22라운드 부분키 $K_{22}$를 독립적인 과정을 이용하여 찾아낼 수 있다.

우선 5라운드의 입력을 표현하는 4개의 16비트 데이터 채널 중 두 번째 데이터 채널만이 포화되도록 $2^{16}$개의 원소를 가지는 5라운드 입력 평문 집합 $P = (\alpha^5, B^5, \gamma^5, \delta^5) (= \{(\alpha^5, \beta_i^5, \gamma^5, \delta^5) \mid 0 \le i \le 2^{16}-1\})$

을 구성한 후 이에 대응하는 22라운드출력 암호문 집합 $C=(A^{23},B^{23},C^{23},D^{23})(=\{(\alpha_i^{23},\beta_i^{23},\gamma_i^{23},\delta_i^{23}) \mid 0\le i\le 2^{16}-1\})$을 얻는다. 이 때 16라운드 distinguisher의 성질에 의하여 확률 1로 다음 식들이 성립함을 알 수 있다.

$$\bigoplus_{0\le i\le 2^{16}-1}\alpha_i^{20}=\bigoplus_{0\le i\le 2^{16}-1}G_{K_{21}}^{-1}(\alpha_i^{23})\oplus\beta_i^{23}=0, K_{21}\in I_{32} \quad (1)$$

$$\bigoplus_{0\le i\le 2^{16}-1}\delta_i^{21}=\bigoplus_{0\le i\le 2^{16}-1}\alpha_i^{23}\oplus G_{K_{22}}^{-1}(\delta_i^{23})=0, K_{22}\in I_{32} \quad (2)$$

그러므로 $K_{21}$이 right key라면 식 (1)이 항상 만족해야한다. 이 때 right key가 아니면서 식 (1)이 성립할 확률은 $2^{-16}$이다. 하지만 21라운드 부분키 공간의 크기는 $2^{32}$이므로 식 (1)을 통하여 여과시킬 수 있는 wrong key의 개수는 대략 $2^{32}-2^{16}$개가된다. 즉 $2^{16}$개 정도의 키가 식 (1)을 만족시키게 되므로 어떤 키가 right key인지를 정확히 알기 위해서는 또 다른 입력 평문 집합을 필요로 한다. 여과 과정에서 살아남은 $2^{16}$개의 키에 대하여 새로운 입력 평문 집합을 가지고 식 (1)을 통하여 다시 한 번 여과 과정을 실시하면 거의 하나의 키가 살아남게 되고 이 키는 right key가 되게 된다. 또한 이와 동일한 방법으로 $K_{22}$를 식 (2)을 이용하여 찾아낼 수 있다. 위에서 설명한 18라운드(5~22) SKIPJACK에 대한 포화 공격을 단계별로 정리하면 다음과 같다.

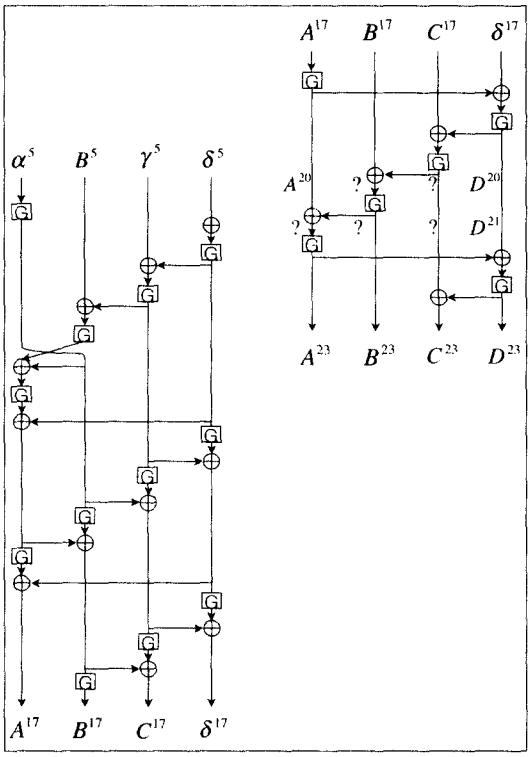
i) $2^{16}$개의 원소를 가지는 두 개의 5라운드 입력 평문집합 $P_1=(\alpha_1^5,B^5,\gamma_1^5,\delta_1^5)$, $P_2=(\alpha_2^5,B^5,\gamma_2^5,\delta_2^5)$를 선택하고 이에 대응하는 암호문 집합 $C_1, C_2$를 획득한다.

ii) $P_1$과 $C_1$를 이용하여 가능한 모든 $K_{21}$에 대하여 식 (1)을 만족시키는지 조사한다.

iii) 과정 ii)을 통과한 키들에 대하여 $P_2$과 $C_2$를 이용하여 다시 한번 (1)식을 만족시키는지 조사한다.

iv) 과정 iii)까지 통과한 키를 right key로 결정한다.

v) (2)식을 가지고 과정 ii), iii), iv)를 동일하게 반복하여 $K_{22}$을찾아낸다.




〔그림 2〕 16라운드 distinguisher를 적용한 18라운드 (5~22) SKIPJACK

위의 공격 과정에서 요구되는 선택 평문수는 $2\cdot 2^{16}=2^{17}$개이며, 공격 복잡도는 $2(2^{16}\cdot 2^{32}\cdot 2^{-5}+2^{16}\cdot 2^{16}\cdot 2^{-5})\simeq 2^{44}$이 된다. 여기서 $2^{-5}$은 한 번의 G함수 연산 시간을 나타내며 괄호 안의 식 중 앞 부분과 뒷 부분은 각각 과정 ii)와 과정 iii)을 실행하는 데에 걸리는 시간을 나타낸다. 위의 과정을 통하여 알아낸 부분키 $K_{21}$, $K_{22}$와 SKIPJACK의 단순한 키 스케줄에 의하여 80비트 마스터 키 중 64비트를 바로 알아낼 수 있다. 또한 전수조사를 통하여 나머지 16비트를 알아낼 수 있다.

### 5.1.2 23라운드(5~27) SKIPJACK에 대한 포화 공격

우선 SKIPJACK에는 5라운드마다 같은 부분키가 사용된다는 것을 다시 한번 언급한다. 이러한 부분키들의 특성과 앞에서 구성한 16라운드 distinguisher는 23라운드(5~27) SKIPJACK에 대한 포화 공격을 가능하게 한다. 여기서 우리는 포화 공격을 이용하여 22, 26, 27라운드 부분키 $K_{22}, K_{26}, K_{27}$을 찾아낼 수 있다. 이 때 물론 부분키의 성질에 의하여 $K_{22}=K_{27}$가 성립한다.

우선 5라운드의 입력을 표현하는 4개의 16비트 데이터 채널 중 두 번째 데이터 채널만이 포화되도록 $2^{16}$개의 원소를 가지는 5라운드 입력 평문 집합 $P=(a^5, B^5, \gamma^5, \delta^5)(=\{(a^5, \beta_i^5, \gamma^5, \delta^5) \mid 0\le i\le 2^{16}-1\})$을 구성한 후 이에 대응하는 27라운드 출력 암호문 집합 $C=(A^{28}, B^{28}, C^{28}, D^{28})$ $(=\{(\alpha_i^{28}, \beta_i^{28}, \gamma_i^{28}, \delta_i^{28}) \mid 0\le i\le 2^{16}-1\})$을 얻는다. 이 때 16라운드 distinguisher의 성질에 의하여 확률 1로 다음 식이 성립함을 알 수 있다.

$$\bigoplus_{0\le i\le 2^{16}-1}\delta_i^{21} = \bigoplus_{0\le i\le 2^{16}-1}\beta_i^{28}\oplus G_{K_{22}}^{-1}(G_{K_{26}}^{-1}(G_{K_{27}}^{-1}(\gamma_i^{28})\oplus\delta_i^{28}))=0 \quad K_{22}, K_{26}, K_{27}\in I_{32} \quad (K_{22}=K_{27}) \tag{3}$$

그러므로 $(K_{22}, K_{26})$이 right key쌍이라면 위 식이 항상 만족해야 한다(물론 $K_{27}$은 $K_{22}$에 의하여 결정된다). 이 때 right key쌍이 아니면서 위 식이 성립할 확률은 $2^{-16}$이고 부분키 쌍이 될 수 있는 총 가지 수는 $2^{64}$이므로 $2^{48}$개 정도의 키가 위 식을 만족하게 된다. 따라서 어떤 키 쌍이 right key쌍인지 가려내기 위해서는 다른 세 개의 입력 평문 집합을 필요로 한다. 이렇게 다른 세 개의 입력 평문 집합에 대하여 위 식을 만족시키는 부분키 쌍은 거의 하나만이 남게 된다. 그러므로 이 키 쌍을 right key쌍이라고 결정할 수 있다.

위의 공격 과정에서 요구되는 선택 평문수는 $4\cdot 2^{16}=2^{18}$개이며, 공격 복잡도는 $2^{16}\cdot 2^{64}\cdot\frac{3}{2^5}+2^{16}\cdot 2^{48}\cdot\frac{3}{2^5}+2^{16}\cdot 2^{32}\cdot\frac{3}{2^5}+2^{16}\cdot 2^{32}\cdot\frac{3}{2^5}\simeq 3\cdot 2^{75}$이된다. 위의 과정을 통하여 알아낸 부분키 $K_{22}$, $K_{26}$과 SKIPJACK의 키 스케줄에 의하여 80비트 마스터 키 중 64비트를 바로 알아낼 수 있다. 또한 전수조사를 통하여 나머지 16비트를 알아낼 수 있다.

### 5.2 20라운드 distinguisher를 이용한 포화 공격

#### 5.2.1 22라운드(1~22) SKIPJACK에 대한 포화 공격

4.2절에서 구성한 20라운드 distinguisher는 22라운드(1~22) SKIPJACK에 대한 포화공격을 가능하게 한다. 여기서 우리는 포화공격을 이용하여 21라운드 부분키 $K_{21}$과 22라운드 부분키 $K_{22}$를 독립적인 과정을 이용하여 찾아낼 수 있다. 우선 1라운드의 입력을 표현하는 4개의 16비트 데이터 채널 중 첫 번째, 두 번째, 네 번째 데이터 채널이 포화되도록 $2^{48}$개의 원소를 가지는 1라운드 입력 평문집합 $P=(A^1, B^1, \gamma^1, D^1)(=\{(\alpha_i^1, \beta_i^1, \gamma^1, \delta_i^1) \mid 0\le i\le 2^{48}-1\})$을 구성한 후 이에 대응하는 22라운드 출력 암호문 집합 $C=(A^{23}, B^{23}, C^{23}, D^{23})(=\{(\alpha_i^{23}, \beta_i^{23}, \gamma_i^{23}, \delta_i^{23}) \mid 0\le i\le 2^{48}-1\})$을 얻는다. 이 때 20라운드 distinguisher의 성질에 의하여 확률 1로 다음 식들이 성립함을 알 수 있다.

$$\bigoplus_{0\le i\le 2^{48}-1}\alpha_i^{20} = \bigoplus_{0\le i\le 2^{48}-1}G_{K_{21}}^{-1}(\alpha_i^{23})\oplus\beta_i^{23}=0, K_{21}\in I_{32} \tag{4}$$

$$\bigoplus_{0\le i\le 2^{48}-1}\delta_i^{21} = \bigoplus_{0\le i\le 2^{48}-1}\alpha_i^{23}\oplus G_{K_{22}}^{-1}(\delta_i^{23})=0, K_{22}\in I_{32} \tag{5}$$

그러므로 $K_{21}$이 right key라면 식 (4)이 항상 만족해야 한다. 이 때 right key가 아니면서 식 (4)이 성립할 확률은 $2^{-16}$이다. 하지만 21라운드 부분키 공간의 크기는 $2^{32}$이므로 이 중에서 $2^{16}$개 정도의 키가 식 (4)을 만족시키게 된다. 그러므로 어떤 키가 right key인지를 정확히 알기 위해서는 또 다른 입력 평문 집합을 필요로 한다. 여과 과정에서 살아남은 $2^{16}$개의 키에 대하여 새로운 입력 평문 집합을 가지고 식 (4)을 통하여 다시 한번 여과 과정을 실시하면 거의 하나의 키가 살아남게 되고 이 키는 right key가 되게 된다. 또한 이와 동일한 방법으로 $K_{22}$를 식 (5)을 이용하여 찾아낼 수 있다.

하나의 워드는 16비트이고 집합 $C$는 $2^{48}$개의 암호문들로 구성되므로 각각의 16비트 데이터 채널에는 중복되는 값이 반드시 존재한다. 만약 어떤 워드가 짝수번 나타난다면 이러한 값들을 모두 XOR한 값은 0이 된다는 사실을 쉽게 알 수 있다. 만일 중복되는 개수가 홀수이면 이러한 값들을 모두 XOR한 값은 자기 자신이 된다는 사실을 알 수 있다.

이러한 사실을 이용하면 공격 복잡도를 상당히 줄일 수 있게 된다. 구체적으로 설명하면 다음과 같다. 어떤 부분키가 식 (4)을 만족하는지 알아보기 위해서는 $2^{48}$번의 $G^{-1}$함수 연산이 필요하다. 그러나 먼저 집합 $C$를 이용하여 $\alpha_i^{23}$이 될 수 있는 $2^{16}$개의 각각의 값에 대하여 중복되는 개수가 짝수인지 홀수인지 조사해 놓을 수 있다. 이렇게 조사된 결과와 위에서 언급한 사실을 이용하면 $2^{48}$번이 아닌 최대 $2^{16}$번

의 $G^{-1}$함수 연산을 통하여 $\bigoplus_{0\le i\le 2^{48}-1} G^{-1}_{K_{21}}(\alpha_i^{23})$을 계산할 수 있게 된다. 위에서 설명한 22라운드(1~22) SKIPJACK 대한 포화 공격을 단계별로 정리하면 다음과 같다.

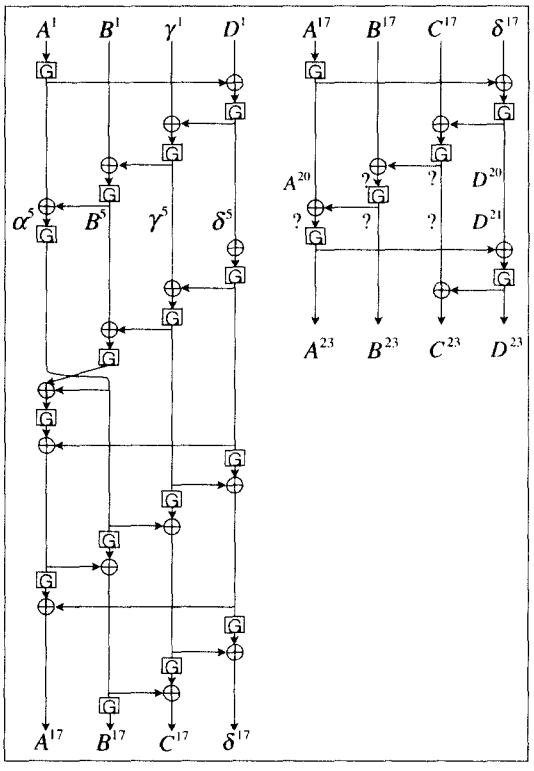
i) $2^{16}$개의 원소를 가지는 두 개의 5라운드 입력 평문집합 $P_1=(A^1, B^1, \gamma_1^1, D^1)$, $P_2=(A^1, B^1, \gamma_2^1, D^1)$를 선택하고 이에 대응하는 암호문 집합 $C_1, C_2$를 획득한다.

ii) $P_1$과 $C_1$를 이용하여 가능한 모든 $K_{21}$에 대하여 (4)식을 만족시키는지 조사한다. (이 때 공격 복잡도를 줄이기 위하여 위에서 소개한 방법을 이용하여 조사한다.)

iii) 과정 ii)을 통과한 키들에 대하여 $P_2$과 $C_2$를 이용하여 다시 한번 (4)식을 만족시키는지 조사한다.

iv) 과정 iii)까지 통과한 키를 right key로 결정한다.

v) 식 (5)을 가지고 과정 ii), iii), iv)를 동일하게 반복하여 $K_{22}$을 찾아낸다.




〔그림 3〕 20라운드 distinguisher를 적용한 22라운드 (1~22) SKIPJACK

앞의 공격 과정에서 요구되는 선택 평문수는 $2\cdot 2^{48}=2^{49}$개이며, 공격 복잡도는 $2(2^{16}\cdot 2^{32}\cdot 2^{-5}\cdot 2^{-5}+2^{16}\cdot 2^{16}\cdot 2^{-5})\approx 2^{44}$이 된다. 위의 과정을 통하여 알아낸 부분키 $K_{21}, K_{22}$와 키 스케줄에 의하여 80비트 마스터 키 중 64비트를 바로 알아낼 수 있다. 또한 전수조사를 통하여 나머지 16비트를 알아낼 수 있다.

### 5.2.2 27라운드(1~27) SKIPJACK에 대한 포화 공격

20라운드 distinguisher는 또한 27라운드(1~27) SKIPJACK에 대한 포화 공격을 가능하게 한다. 여기서 우리는 포화 공격을 이용하여 22, 26, 27라운드 부분키 $K_{22}, K_{26}, K_{27}$ 을 찾아낼 수 있다. 이 때 물론 부분키의 성질에 의하여 $K_{22}=K_{27}$가 성립한다.

우선 1라운드의 입력을 표현하는 4개의 16비트 데이터 채널 중 첫 번째, 두 번째, 네 번째 데이터 채널이 포화되도록 $2^{48}$개의 원소를 가지는 1라운드 입력 평문 집합 $P=(A^1, B^1, \gamma^1, D^1)(=\{(a_i^1, \beta_i^1, \gamma^1, \delta_i^1) \mid 0\le i\le 2^{48}-1\})$을 구성한 후 이에 대응하는 27라운드 출력 암호문 집합 $C=(A^{28}, B^{28}, C^{28}, D^{28})(=\{(a_i^{28}, \beta_i^{28}, \gamma_i^{28}, \delta_i^{28}) \mid 0\le i\le 2^{48}-1\})$을 얻는다. 이 때 20라운드 distinguisher의 성질에 의하여 확률 1로 다음 식이 성립함을 알 수 있다.

$$\bigoplus_{0\le i\le 2^{16}-1}\delta_i^{21} = \bigoplus_{0\le i\le 2^{16}-1}\beta_i^{28}\oplus G^{-1}_{K_{22}}(G^{-1}_{K_{23}}(G^{-1}_{K_{24}}(\gamma_i^{28})\oplus\delta_i^{28}))=0$$
$$K_{22}, K_{26}, K_{27}\in I_{32} \quad (K_{22}=K_{27}) \tag{6}$$

그러므로 $(K_{22}, K_{26})$가 right key쌍이라면 위 식이 항상 만족해야 한다(물론 $K_{27}$은 $K_{22}$에 의하여 결정된다). 이 때 right key쌍이 아니면서 위 식이 성립할 확률은 $2^{-16}$이고 부분키 쌍이 될 수 있는 총 가지 수는 $2^{64}$이므로 $2^{48}$개 정도의 키가 위 식을 만족하게 된다. 따라서 어떤 키 쌍이 right key쌍인지 가려내기 위해서는 다른 세 개의 입력 평문 집합을 필요로 한다. 이렇게 다른 세 개의 입력 평문 집합에 대하여 위 식을 만족시키는 부분키 쌍은 거의 하나만이 남게 된다. 그러므로 이 키 쌍을 right key쌍이라고 결정할 수 있다.

위의 공격 과정에서 요구되는 선택 평문 수는 $4\cdot 2^{48}=2^{50}$개이며, 공격 복잡도는 $2^{64}\cdot 2^{16}\cdot \frac{3}{2^5}$

$+2^{16}\cdot 2^{48}\cdot\frac{3}{2^5}+2^{16}\cdot 2^{32}\cdot\frac{3}{2^5}+2^{16}\cdot 2^{16}\cdot\frac{3}{2^5}$ $\simeq 3\cdot 2^{75}$이 된다. 위의 과정을 통하여 알아낸 부분키 $K_{22}, K_{26}$과 키 스케줄에 의하여 80비트 마스터 키 중 64비트를 바로 알아낼 수 있다. 또한 전수조사를 통하여 나머지 16비트를 알아낼 수 있다.

## Ⅵ. 결 론

본 논문에서는 포화 공격을 SKIPJACK에 최초로 적용한 결과를 제시하였다. 우리는 SKIPJACK에 대한 16라운드 distinguisher의 구성 방법을 설명한 후 이것에 기반한 20라운드 distinguisher의 구성 방법을 제시하였다. 이렇게 구성된 20라운드 distinguisher를 이용하면 최대 27라운드 SKIPJACK까지 포화 공격이 가능함을 보였다. 27라운드 SKIPJACK에 대한 공격 시 필요한 선택 평문은 $2^{50}$개이며 이때의 공격 복잡도는 $3\cdot 2^{75}$임을 살펴 보았다.

〔표 2〕는 축소된 SKIPJACK에 대한 포화 공격의 결과들을 포함하여 지금까지 알려진 여러 가지 축소된 SKIPJACK에 대한 다양한 공격 결과들을 정리한 것이다. 〔표 2〕에 제시된 부메랑 공격은 다른 공격들과는 공격자 모델이 다른 선택 평문 선택 암호문 공격이므로 선택 평문 수와 선택 암호문 수를 따로 분리하여 나타내었다. 본 논문에서 제시한 포화 공격의 결과는 Biham 등이 발표한 31라운드 SKIPJACK에 대한 불능 차분공격 결과를 향상시키지는 못하였다.

〔표 3〕 기존 공격들과 포화 공격의 공격 복잡도 비교

| 공격 방법 | 라운드 수 | 선택 평문 수 | 복잡도 |
|---|---|---|---|
| 불능 차분 | 25(5-29) | $2^{38}$ | $2^{27}$ |
| 불능 차분 | 26(4-29) | $2^{38}$ | $2^{49}$ |
| 불능 차분 | 28(1-28) | $2^{34}$ | $2^{77}$ |
| 불능 차분 | 29(1-29) | $2^{34}$ | $2^{77}$ |
| 불능 차분 | 30(1-30) | $2^{34}$ | $2^{77}$ |
| 불능 차분 | 31(1-31) | $2^{41}$ | $2^{78}$ |
| 불능 차분 | 31(2-32) | $2^{34}$ | $2^{78}$ |
| 부정 차분 | 16(1-16) | $2^{17}$ | $2^{34}$ |
| 부정 차분 | 28(5-32) | $2^{41}$ | $2^{77}$ |
| 포화 | 18(5-22) | $2^{17}$ | $2^{44}$ |
| 포화 | 22(1-22) | $2^{49}$ | $2^{44}$ |
| 포화 | 22(5-26) | $2^{18}$ | $2^{76}$ |
| 포화 | 23(5-27) | $2^{18}$ | $3\cdot 2^{75}$ |
| 포화 | 26(1-26) | $2^{50}$ | $2^{76}$ |
| 포화 | 27(1-27) | $2^{50}$ | $3\cdot 2^{75}$ |
| 부메랑 | 25(4-28) | $2^{18.5}, 2^{34.5}$ | $2^{61.5}$ |

그러나 포화 공격 관점에서의 SKIPJACK에 대한 분석 결과를 최초로 제시했다 점에서 본 논문의 의미를 찾을 수 있다. 포화 공격을 이용하여 본 논문의 결과를 더욱 향상시킬 수 있는가에 대한 문제는 계속하여 연구를 진행할 여지가 남아 있다고 생각된다. 이와 더불어 SKIPJACK과 유사한 다른 구조에 대한 포화 공격의 적용 가능성을 연구하는 것 또한 남겨진 과제가 되고 있다.

## 참 고 문 헌


〔1〕 E. Biham, A. Biryukov, and A. Shamir. "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials", *In J. Stern, editor, Advances in Cryptology Eurocrypt '99*, volume 1592 of Lecture Notes in Computer Science, pp. 12~23, 1999.

〔2〕 E. Biham, A. Biryukov, O. Dunkelmann, E. Richardson and A. Shamir, "Initial Observations on the Skipjack Encryption Algorithm", June 25, 1998. Available at http://www.cs.technion.ac.il/biham/reports / Skipjack/.

〔3〕 E. Biham, A. Biryukov, O. Dunkelmann, E. Richardson and A. Shamir, "Cryptanalysis of Skipjack-3XOR in $2^{20}$ time and using $2^9$ chosen plaintexts.", July 2, 1998. Available at http://www.cs.technion.ac. il/biham/reports/Skipjack.

〔4〕 E. Biham, A. Biryukov, O. Dunkelmann, E. Richardson and A. Shamir, "Cryptanalysis of Skipjack-4XOR", June 30, 1998. Available at http://www.cs.technion.ac. il/biham/reports/Skipjack.

〔5〕 J. Daemin, L. Knudsen and V. Rijmen, "The block cipher SQUARE", *In Fast Software Encryption*, Springer LNCS 1267, pp. 137~151, 1997.

〔6〕 Lars R. Knudsen, M. J. B. Robshaw "Truncated Differentials and Skipjack", *Advances in Cryptology, Proceedings Crypto '99*, Springer, LNCS 1666, pp. 165~180.

〔7〕 S. Lucks, "The saturation attack-a bait

for Twofish", *In Fast Software Encryption Third International Workshop*, 2001.

[8] C. H. Lim, "Crypton: a new 128 bit block cipher", AES Submission, AES Development effort, NIST,http://www.nist.gov/aes.

[9] C. H. Lim, "A revised version of Crypton Crypton v.1.0", *In Fast Software Encryption* Springer, LNCS 1636, pp. 31~45, 1999.

[10] J. Daemem and V. Rijmen, "AES proposal : Rijndael(2nd version)", *AES Submission, AES Developement Effort*, NIST. http://www.nist.aes.

[11] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D Wang and D. Whiting, "Improved cryptanalys is of RIJNDAEL", *In Fast Software Encryption*, Springer LNCS 1978, pp. 213~230, 2000.

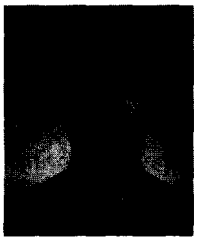
[12] Louis Granboulan, "Flaws in differential Cryptanalysis of Skipjack", *Fast Software Encryption Workshop* 2001, LNCS 1039, Springer Verlag April 2001 pp. 341~346.

[13] V. Rijmen, B. Preneel, C. D'Hallui and G. Bijnens, "Attakc on six round of Cryption", *Fast Software Encryption*, Springer LNCS 1636, pp. 46~59, 1999.

[14] National Institute of Standards and Technology, "SKIPJACK and KEA algorithm specifications, version 2.0", http://crsc.nist.encryption/SKIPJACK-kea.htm,1998.

[15] National Institute of Standards and Technology, "NSA Releases Fortezza Algorit hms", available at http://crsc.nist.gov/encryption/encryption/nsa-press.pdf.

## 〈著者紹介〉

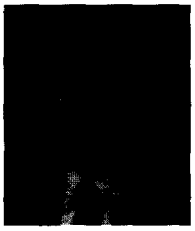


**황 경 덕 (Kyung-deok Hwang)**
2000년 2월 : 고려대학교 수학과 학사
2000년 3월~현재 : 고려대학교 정보보호기술학과 석사 과정
〈관심분야〉 블록 암호 및 스트림 암호 분석 및 설계

**이 원 일 (Won-il Lee)**
1998년 2월 : 고려대학교 수학과 학사
2000년 2월 : 고려대학교 수학과 석사
2000년 3월~현재 : 고려대학교 수학과 박사 과정
〈관심분야〉 블록 암호 및 스트림 암호 분석 및 설계

**이 성 재 (Sung-jae Lee)**
1997년 8월 : 고려대학교 수학과 학사
1999년 8월 : 고려대학교 수학과 석사
1999년 9월~현재 : 한국정보보호진흥원 연구원
〈관심분야〉 블록 암호 및 스트림 암호 분석 및 설계

**이 상 진 (Samg-jin Lee) 정회원**
1987년 2월 : 고려대학교 수학과 학사
1989년 2월 : 고려대학교 수학과 석사
1994년 2월 : 고려대학교 수학과 박사
1989년 2월~1999년 2월 : 한국전자통신연구원 선임 연구원,
1999년 2월~현재 : 고려대학교 자연과학대학 부교수,
고려대학교 정보보호대학원 겸임교수, 고려대학교 정보보호기술연구센터 연구실장
〈관심분야〉 블록 암호 및 스트림 암호의 분석 및 설계, 암호 프로토콜, 공개키 암호 알고리즘의 분석.



**임 종 인 (Jong-in Lim) 정회원**
1980년 2월 : 고려대학교 수학과 학사
1982년 2월 : 고려대학교 수학과 석사
1986년 2월 : 고려대학교 수학과 박사
1999년 2월~현재 : 고려대학교 자연과학대학 정교수, 한국통신정보보호학회 편집위원장
고려대학교 정보보호대학원 원장, 고려대학교 정보보호기술연구센터 센터장
〈관심분야〉 블록 암호 및 스트림 암호의 분석 및 설계, 암호 프로토콜, 공개키 암호 알고리즘의 분석