공학박사 학위논문

# 대칭단을 이용한 암호와 복호가 다른 블록 암호의 재설계


2010년 2월

부경대학교 대학원

컴 퓨 터 공 학 과

김 길 호

공학박사 학위논문

# 대칭단을 이용한 암호와 복호가 다른 블록 암호의 재설계

지 도 교 수  조 경 연

이 논문을 공학박사 학위논문으로 제출함

2010년 2월

부경대학교  대학원

컴 퓨 터 공 학 과

김 길 호

# 김길호의 공학박사 학위논문을 인준함

2009 년 12 월 4 일


주 심 공학박사 서 경 룡 (인)

위 원 공학박사 김 종 남 (인)

위 원 공학박사 송 홍 복 (인)

위 원 공학박사 권 장 우 (인)

위 원 공학박사 조 경 연 (인)

# 목 차

# 그 림 목 차

# 표 목 차

# Redesign of the block cipher which has different encryption and decryption algorithm, using symmetric layer

Gil Ho Kim

Department of Computer Engineering, The Graduate School
Pukyong National University

## ABSTRACT

The hardware implementation of block cipher algorithm is closely related to the structure of a cipher algorithm. Because Feistel structure has same encryption and decryption algorithm, only one of them is required to process both encryption and decryption, when implemented by hardware. However, since SPN structure has different encryption and decryption algorithm, both algorithms have to be implemented by hardware when both are needed. Therefore the block cipher algorithm, which has different encryption and decryption, has the significant disadvantage that hardware area increases, compared to the one, which has same encryption and decryption algorithm.

In this paper, in order to improve the block cipher which has different encryption and decryption algorithm, it has been composed of same encryption and decryption, using symmetric layer consisting of simple logical operations. The whole algorithm is organized with R round. A function (encryption algorithm) is used from first round to ((R/2) − 1)round and the inversed function (decryption algorithm) is used from (R/2) + 1 round to R round. Moreover, an irregular symmetric block, an independent round symmetric formed−layer, is inserted into the middle of the function and the inversed function to avoid the repetition of regular rounds.

The block ciphers which have different encryption and decryption

algorithm such as AES, RC6, SHACAL-1, and SHACAL-2, have been recomposed by software, adapting symmetric layer. The symmetric adapted-block ciphers are about 10% slower than original ones, and it is estimated that the adapted symmetric layers don't have a significant effect on encryption execution time. The symmetric layer can be easily applied to not only 128 bits block cipher but also 160 and 256 bits block cipher algorithms. It means that the symmetric layer can be applied to all block cipher algorithms which have different encryption and decryption.

In the aspect of security of symmetric layer adapted block algorithms, the symmetric layer prevents from the attacks such as Differential, Linear, Square, Impossible Differential, Boomerang, Related-key Slide, Related-key Differential-Nonlinear, Related-key Rectangle. It shows that symmetric layer is very efficient in the improvement of block cipher security. Like this, the reimplementation which applies characteristic about execution time and security of symmetric layer can be an efficient idea that is able to be easily adapted to a new block cipher algorithm design.

# 제 1장 서론

## 1.1 암호

현대는 정보화 사회라고 말한다. 이는 정보의 가치를 인식하고, 이러한 정보들을 통신 기술의 발달로 많은 사람들이 이용하면서 정보의 가치를 더욱더 향상시켜 나가는 것이다. 이와 같은 정보화 사회에서는 신뢰(Confidence)를 바탕으로 믿을 수 있는 정보통신 시스템이 필요하다. 신뢰할 수 있는 시스템의 구축에 가장 필요한 기반이 암호학(Cryptography)이다. 암호학이란 데이터의 인증(data origin authentication), 사용자 인증(entity authentication), 데이터 무결성(data integrity) 등의 정보보호(information security)와 관련 있는 수학적인 측면에서 수이론(number theory), 확률(probability), 통계(statistics) 뿐만 아니라 정보이론(information theory), 계산 복잡도(computational complexity), 코딩 이론(coding theory) 등을 포함 하는 굉장히 넓은 범위의 큰 학문이다[1]. 현재 사용자, 데이터 인증 및 데이터 무결성을 보장하는 다양한 암호 알고리즘이 개발 되어있고 또 많은 학자들이 연구 개발 중에 있다.

암호 알고리즘이란 전통적으로 군사, 외교, 정치 등 소수 특권 계층에서 어떤 목적을 위해 사용되어 온 비밀스러운 문자 코드 체계이다. 그리고 이러한 암호는 고대 시저 암호(Caesar)에서 근대의 전치 암호(Transposition), 치환 암호(Substitution)[2]와 2차 대전에 독일에서 사용한 회전자 기계(Rotor Machines) Enigma 등 많은 알고리즘이 잘 알려져 있다. 이와 같은 암호 알고리즘을 전통적인 암호 방식 즉 비밀 키(대칭 키, 단일 키) 암호(secret, symmetric, one key)라고 한다. 이러한 전통적인 방식의 암호는 비밀 키는 물론이고, 암호 알고리즘도 확실하게 비밀이 유지되어야 했다. 왜냐하면, 암호 알고리즘이 단순해 비밀 키를 몰라도 역으로 평문을 알 수 있는 역함수를 쉽게 만들 수 있기 때문이다. 그러나 현대 암호는 암호 알고리즘의 공개를 원칙으로 한다. 암호문의 안전성은 암호문 생성에 사용된 비밀 키의 비밀 유지가 핵심적인 사항이다. 그리고 현대 암호는 고전 암호와는 달리 컴퓨터의 빠른 연산 능력을 사용하지 않고서는 암호 및 복호를 수행할 수 없을 정도의 복잡하고 많은 연산을 요구한다. 이러한 현대 암호 알고리즘의 개발과 연구의 시초는 1977년 미국의 연방표준국(NIST)에서 DES(Data Encryption Standard)[3]를 표준 블록 암호 알고리즘으로 채택되고, 1976년 공개 키 암호(public key)개념[4]이

발표된 이후 지금까지 현대 암호학이 비약적으로 발전하게 되었다.

DES의 설계원리와 분석방법은 많은 암호학자들의 연구대상이었으며, 블록 암호 알고리즘의 대표적인 분석 방법인 차분 공격[5], 선형 공격[6] 등의 개발과 이와 같은 분석 방법에 내성이 있는 블록 암호 알고리즘의 개발 등에 초석이 되었다. 그리고 공개 키(비대칭 키, 두개 키)암호(asymmetric, two key)는 비밀 키 암호의 운영 시스템에서 비밀 키의 관리 및 분배에 취약점을 개선한 방식으로 비밀 키는 비밀 정보를 공유하는 쌍방이 같은 비밀 키를 가지고 운영하는 시스템이다. 그러나 현대 정보화 사회는 다수의 개인 및 단체와 정보 교환을 하고 있고, 이때 한 개의 비밀 키로 다수의 상대와 정보를 교환하는 것은 정보의 비밀을 유지할 수 없을 뿐만 아니라 많은 사회적인 문제도 야기 시킬 수 있다. 그렇다고 다수의 사용자에 대응 되는 각각의 비밀 키를 만들어 사용하는 것도 불가능에 가깝다. 이와 같은 문제점을 공개 키는 단 두 개의 키를 사용하므로 해서 해결할 수 있다. 공개 키 암호는 공개 키와 비밀 키라는 두 개의 키를 가지고 암호와 복호를 수행한다. 비밀 키는 자신만이 알고 있는 비밀 키이고, 공개 키는 세상에 공개하여 모든 사람이 사용할 수 있는 키이다. 어떤 사람이 나에게 비밀 정보를 보내려면, 먼저 나의 공개 키로 정보를 암호화 하여 나에게 보낸다. 나는 받은 암호문을 나의 비밀 키로 복호하여 정보를 공유 할 수 있다. 이와 같은 공개 키 시스템은 개인이나 단체에서 공개한 공개 키의 유지, 관리 및 인증을 할 수 있는 제 3의 인증기관(Certification Authority)이 필수 요소이다. 현재 구현되어 있는 대표적인 공개 키 암호는 RSA[7], ECC[8], ElGamal[9] 등이 있다.

현대 암호는 크게 비밀 키 암호와 공개 키 암호로 나눌 수 있다. 그 중 비밀 키의 대표적인 암호인 DES는 적은 키 공간과 획기적인 암호 분석법인 차분 공격과 선형 공격의 개발로 1990년대 후반 암호 알고리즘이 안전하지 않다는 것이 증명 되었다[10, 11]. 이후 미국을 비롯한 선진국들은 자국의 정보보호를 위한 새로운 블록 암호 알고리즘 선정을 위해 국제적인 공모 사업을 시작했다. 특히 미국의 AES(Advanced Encryption Standard)[12] 선정 프로젝트, 유럽의 NESSIE(New European Schemes for Signatures. Integrity. and Encryption)[13] 프로젝트, 일본의 CRYPTREC(Cryptography Research and Evaluation Committees)[14] 프로젝트 등이 대표적이며, 우리나라는 자체 개발한 SEED[15]와 ARIA[16]를 표준 128비트 블록 암호로 제정하였다. 초기 블록 암호는 소프트웨어로 구현되었지만 최근에는 빠른 정보처리를 위한 하드웨어 구현에도 많은 연구가 진행 중이다.

## 1.2 비밀키 암호의 분류

비밀 키 암호 시스템은 암호 알고리즘 E에 비밀 키 K가 적용되어 평문 P를 암호문 C = $E_K(P)$로 전환시키며, 복호화는 복호 알고리즘 $E^{-1}$에 비밀 키 K가 적용되어 암호문 C를 평문 P = $E^{-1}{}_K(C)$ = $E^{-1}{}_K(E_K(P))$로 복원시킨다. 그리고 일반적으로 비밀 키 암호는 블록 암호(block cipher)와 스트림 암호(stream cipher)로 분류할 수 있다. 먼저 블록 암호는 (그림 1-1)과 같이 시스템을 구성하고 있으며, 암호 알고리즘과 복호 알고리즘이 같을 수도 있지만 서로 다른 경우도 많다. 그리고 암호와 복호의 수행은 일정한 크기의 블록 단위로 평문과 암호문을 나누어 처리한다. 비밀 키는 암호와 복호에서 동일한 비밀 키를 사용한다. 그래서 비밀 키는 사전에 기밀성과 무결성이 보장된 채널을 통해 전달되어진다.

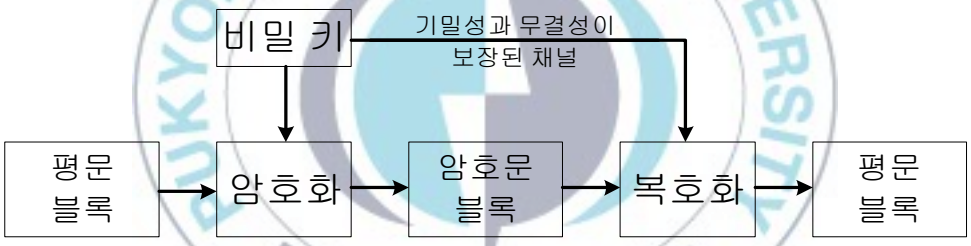



그림 1-1. 블록 암호의 원리

스트림 암호는 (그림 1-2)와 같이 구성되며, 1970년대 유럽을 중심으로 하드웨어 구현이 용이한 LFSR(Linear Feedback Shift Register) 기반의 이진 수열 발생기를 이용하여 평문과 이진 수열 발생기에서 생성된 이진 수열과 XOR 연산을 수행하여 이진 수열로 된 암호문을 만드는 암호 알고리즘이다. 즉 키 수열을 생성하는 키 생성(Key Generation) 알고리즘이라 할 수 있다. (그림 1-2)와는 다르게 키 생성 알고리즘에서 생성된 비밀 키를 전송하는 것이 아니라 키 생성 알고리즘을 서로 공유하고 있을 경우에 쌍방이 서로 사용되는 비밀 키는 사전에 공유되어야 하고 이 뿐만 아니라 키 생성 알고리즘의 초기값(Initialization Vector: IV)도 반드시 같게 맞추어야 한다. 그리고 스트림 암호의 특성상 송신자와 수신자간에 암호와 복호에 적용되는 키 수열에 대한 엄격

한 동기화(synchronization)가 요구된다. 이와 같은 동기화는 블록 암호에서는 필요 없다. 마지막으로 스트림 암호가 블록 암호 보다는 수행 속도가 5-10배 정도 빠른 것으로 알려져 있다. 대표적인 스트림 암호 알고리즘으로는 SSC2[17], A5[18], Salsa20[19] 등이 있다.

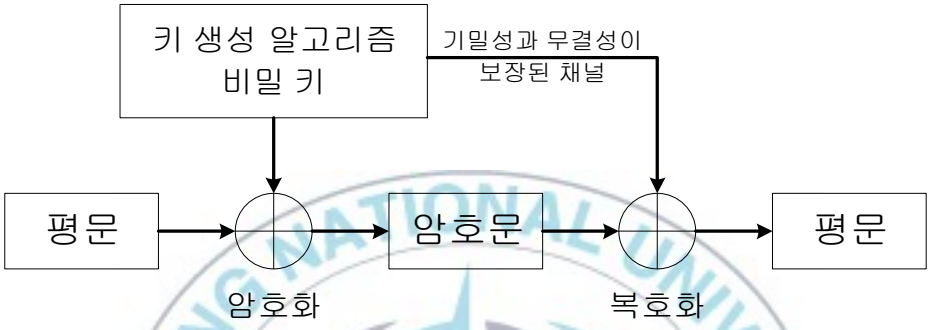



그림 1-2. 스트림 암호의 원리

## 1.3 블록 암호의 구조적 분류

블록 암호는 평문을 암호화 수행과정인 라운드(Round)가 반복적으로 수행된 결과 암호문을 출력하는 반복 구조(Iterated Structure)로 되어 있다. 그리고 각 라운드의 구조에 따라 블록 암호는 크게 Feistel 구조[20]와 SPN(Substitution Permutation Network) 구조로 나눌 수 있다. Feistel 구조는 라운드 함수에 입력되는 평문을 둘로 나누어 나누어진 한쪽 평문과 라운드 키와 결합된 비선형적 변환을 수행한 후 나머지 반쪽 평문과 XOR 연산을 수행 후 서로 자리를 바꾸는 구조로, 즉 평문 전체 블록에서 반은 변화를 주고 나머지 반은 변화 없이 서로 자리를 바꾸어 다음 라운드의 입력으로 들어가는 구조이다. (그림 1-3)은 Feistel 구조를 간략히 그림으로 표현한 것이다.

(그림 1-3)에서 라운드 함수 F는 라운드 키 $K^i$와 평문 블록의 반인 $R_{i-1}$을 입력으로 받아 비선형 연산(암호 알고리즘에 따라 서로 다른 연산을 수행)을 적용하는 함수로 라운드 수에 따라 F함수를 반복 적용한다. Feistel 구조는 암호와 복호 알고리즘이 같고, 다른 구조의 블록 암호 알고리즘에 비해 수행속도

가 빠르며, 하드웨어 및 소프트웨어 구현이 쉬우며, 아직까지 구조적으로 안전성에 대한 문제점이 발견되지 않은 많은 장점을 가진 좋은 구조이다.

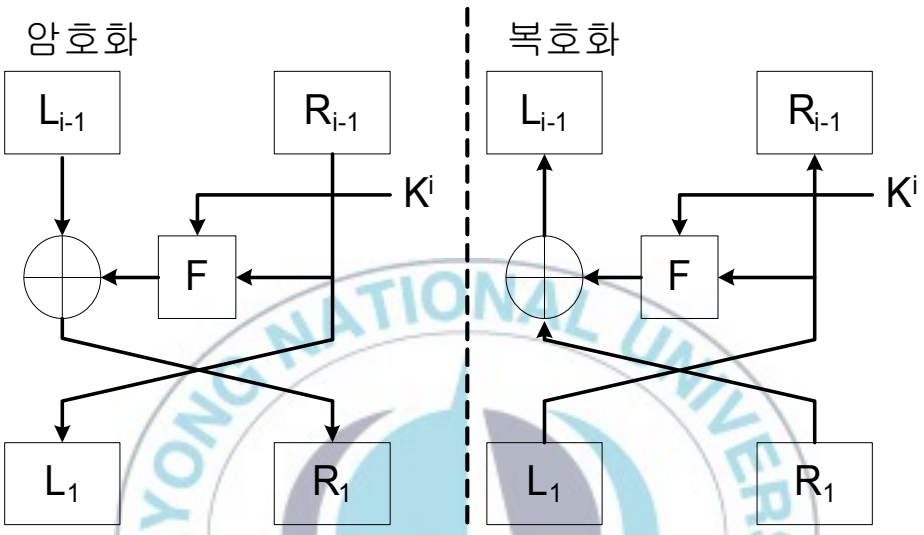



그림 1-3. Feistel 구조

SPN 구조는 암호 알고리즘이 치환 층(Substitution Layer), 확산 층(Permutation Layer), 키 작용 층(Key Addition Layer)과 같이 세 단계가 한 라운드로 구성되어 반복하는 구조를 말한다. (그림 1-4)는 SPN 구조를 그림으로 설명한 것이다.

SPN 구조는 C. E. Shannon의 혼돈(Confusion)과 확산(Diffusion)[21] 이론을 바탕으로 구성되었으며, 혼돈은 평문과 암호문과의 상관관계(correlation)를 숨기는 것을 말하고 치환(S)을 통해 이룰 수 있고, 확산(P)은 평문의 통계적 특성을 암호문 전반에 확산시키는 것을 말하고 이와 같은 혼돈과 확산을 반복 적용하여 안전성을 높인 블록 암호 알고리즘 구조이다. 그리고 SPN 구조는 암호와 복호 알고리즘이 서로 다른 특성을 갖고 있다. 다시 말해 암호화 과정에서 치환 연산인 S박스는 복호화 수행 시 역변환 S박스가 필요하며, S박스와 역변환 S박스는 일대일 대응되는 전단사(Bijection) 함수이다. 그리고 확산과정 역시 복호 수행을 위한 역변환 확산 알고리즘이 필요하다. 그래서 SPN 구조의 블록 암호 알고리즘의 개발은 암호 알고리즘과 그에 대응되는 복호 알고리즘을

같이 개발해야 한다.

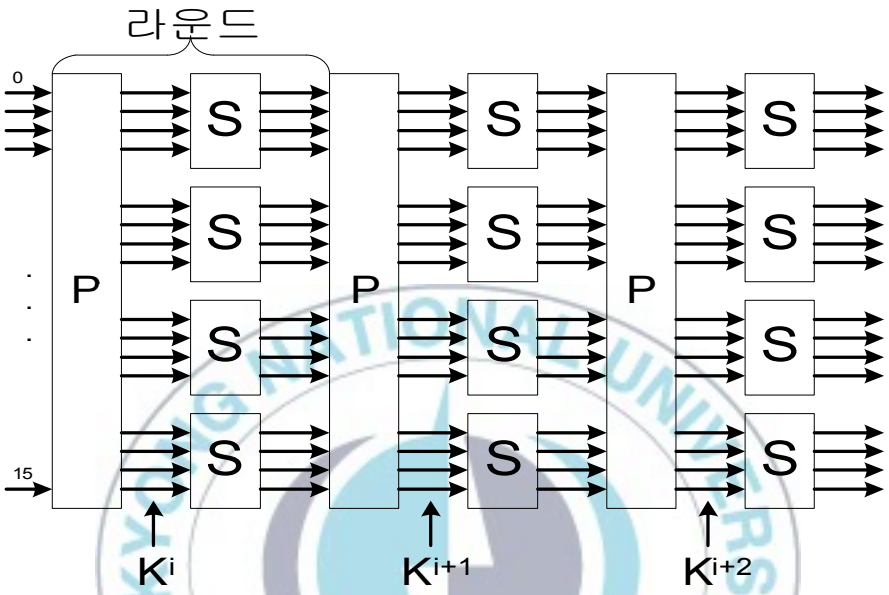



그림 1-4. SPN 구조

## 1.4 논문의 주제

현대 암호 알고리즘의 표준을 선정하는 방식은 대부분 공모[12, 13, 14]를 통해 선정하고 있다. 이는 많은 장점을 가진다. 암호 알고리즘 개발과정에서 얻은 기술이나 지식은 공모를 통해 다른 많은 개발자들에게 쉽게 전수되고, 이는 다시 더 좋은 암호 알고리즘 개발에 빠르게 적용되며, 또 개발과정에서 발견하지 못한 오류를 다른 개발자를 통해 쉽게 얻을 수 있다. 이와 같은 과정이 신뢰를 형성하여, 앞 절에서 설명한 정보화 사회를 이룰 수 있는 안전하고 신뢰성 있는 시스템을 만드는 기초가 된다.

정보처리 관점에서 오늘날과 같은 초고속 정보통신 이전의 데이터는 대부분 문자위주의 데이터였다. 그러나 현재는 고용량의 멀티미디어(Multimedia) 데이터(그림, 사운드, 동영상 등)가 더 높은 가치의 정보를 이루고 있으며, 이

와 같은 고용량의 멀티미디어 데이터의 암호화를 비롯한 정보보호를 위한 처리는 필연적으로 빠른 수행을 요구한다. 그래서 좀 더 빠른 암호화 수행을 위한 새로운 알고리즘 개발과 병행하여 기존의 알고리즘을 하드웨어로 구현하므로 해서 이와 같은 요구를 만족시킬 수 있다. 그리고 RFID(Raido Frequency Identification)를 이용한 보안이 적용된 센서 네트워크(sensor network)를 구축하는 과정에서 암호 및 복호 과정을 필요로 하는 능동형 태그(active tag)의 구현은 암호 알고리즘의 하드웨어 구현이 필수 요소가 된다. 이와 같은 이유들 때문에 암호 알고리즘을 공모한 선진국들은 공모 기준에 하드웨어 구현 및 성능 평가가 선정을 위한 필수 조건이었다.

블록 암호 알고리즘의 하드웨어 구현은 1.3절에서 설명한 대로 암호 알고리즘의 구조와 밀접한 관계가 있다. 왜냐하면 Feistel 구조인 경우 암호와 복호 알고리즘이 같기 때문에 하드웨어 구현 시 암호 알고리즘만 구현하면 암호와 복호 과정을 모두 수행할 수 있다. 그러나 SPN 구조의 경우는 암호와 복호 알고리즘이 서로 달라 암호와 복호 수행이 모두 필요한 경우에는 암호와 복호 알고리즘을 모두 하드웨어로 구현해야 한다. 그래서 암호와 복호가 서로 다른 블록 암호 알고리즘의 하드웨어 구현은 암호와 복호 알고리즘이 같은 블록 암호 알고리즘과 비교해서 하드웨어 면적이 많이 늘어나게(하드웨어 면적의 증가는 블록 암호 알고리즘에 따라 차이가 많이 남) 되는 단점이 된다.

본 논문에서는 암호와 복호가 다른 블록 암호 알고리즘의 단점을 개선하기 위해 간단한 논리 연산으로 구성된 대칭단(Symmetric Layer)[22, 23]을 적용해서 암호와 복호 알고리즘을 같게 구성했다. 대칭단에 대한 자세한 설명과 실제 블록 암호 알고리즘에 적용사례, 수행 평가, 안전성 분석은 3장, 4장, 5장, 6장에서 자세히 설명한다.

## 1.5 표기법

본 논문에서는 사용될 그림, 표, 알고리즘이나 연산 등에서 사용된 연산자의 설명은 다음과 같다.

- A + B : 정수덧셈 mod $2^{32}$.
- A − B : 정수뺄셈 mod $2^{32}$.
- A & B : 워드단위 비트 AND 연산.

● A | B : 워드단위 비트 OR 연산.
● ~A : 워드단위 비트 NOT 연산.
● A ⊕ B : 워드단위 비트 XOR 연산.
● A * B : 정수곱셈 mod $2^{32}$.
● A / B : 정수나눗셈 mod $2^{32}$.
● A ≪ B : B비트 만큼 A를 왼쪽 쉬프트연산.
● A ≫ B : B비트 만큼 A를 오른쪽 쉬프트연산.
● A ⋘ B : B비트 만큼 A를 왼쪽 회전연산.
● A ⋙ B : B비트 만큼 A를 오른쪽 회전연산.
● A ‖ B : 연결 연산.

각 변수들은 32비트 워드이며, 32비트가 아닌 경우는 따로 표시한다. 그리고 특별한 문자에 대한 표기법은 다음과 같다.

● P(Plaintext) : 평문.
● C(Ciphertext) : 암호문.
● K, Kr(Key, Round Key) : 비밀 키, 라운드 키.
● E(Encryption) : 암호화, 암호를 수행하다.
● $E^{-1}$(Decryption) : 복호화, 복호를 수행하다.
● R, r(Round) : 암호 또는 복호를 수행하는 회수.
● F, f(Function) : 암호를 수행하는 함수.
● $F^{-1}$, $f^{-1}$(Inverse Function): 복호를 수행하는 역함수.
● SL(Symmetric Layer) : 대칭단.

위 문자에 대한 표기법이 다른 의미로 중복되게 쓰일 경우는 따로 설명하고 사용한다.

## 1.6 논문의 구성

2장에서는 본 논문의 설명을 위해 선택된 4가지 블록 암호 AES, RC6, SHACAL-1, SHACAL-2를 먼저 설명하고, 블록 암호의 안전성 기준 및 블록 암호의 대표적인 공격 방법인 차분 공격, 선형 공격, Square 공격, 연관키 공격

에 대해서 설명한다.

3장에서는 본 논문의 핵심 주제인 대칭단 구조에 대해 설명하는 장으로 대칭단 구조의 목적과 암호 알고리즘의 블록 크기에 따른 128, 160, 256비트 대칭단의 적용 방법 및 대칭단의 안전성 분석과 결론으로 구성한다.

4장, 5장, 6장에서는 2장에서 설명한 4가지의 블록 암호에 대칭단을 적용하고 적용한 암호 알고리즘의 안전성과 수행 시간 테스트에 대한 분석을 하고 마지막으로 결론으로 구성한다.

7장은 본 논문의 최종적인 결론이다.

# 제 2장 선택된 블록 암호 알고리즘

본 논문에서 선택된 블록 암호 알고리즘의 선택 기준은 첫째 암호와 복호가 서로 다른 알고리즘이고, 둘째 많은 암호 알고리즘 개발자나 연구자들로부터 주목 받은 잘 알려진 알고리즘으로 하였고, 마지막으로 현재 널리 사용되고 있거나 앞으로 널리 사용된 가능성이 많은 암호 알고리즘으로 4가지 AES[24], RC6[25], SHACAL-1[26], SHACAL-2[27]를 선택하였다. 이번 장에서는 4가지 블록 암호 알고리즘의 암호와 복호에 대한 기본 설명을 바탕으로 설계 원리에 대해 설명하고, 블록 암호 분석 및 공격 방법에 대해서 설명한다.

## 2.1 AES

AES는 미국 국립 표준 기술 연구소(NIST)[28]에서 DES를 대체하기 위해 전 세계적으로 블록 암호 알고리즘의 공개 모집을 위한 프로젝트로, Joan Daemen과 Vincent Rijmen이 개발한 Rijndael을 2000년 10월에 AES 암호 알고리즘으로 선정하였다. 이후 AES는 전 세계적으로 가장 널리 사용되고 있는 블록 암호 알고리즘 중 가장 대표적인 암호 알고리즘이 되었다. AES 알고리즘의 입력 평문의 길이는 128비트로 고정이고, 암호화에 사용되는 라운드 키의 길이는 128비트, 196비트, 256비트 중에서 선택해서 사용할 수 있고, 라운드 수는 키 길이에 따라 가변적으로 10, 12, 14라운드가 적용된다.

AES는 암호 및 복호과정에서 생성되는 중간 결과 값 스테이트(State)를 바이트 단위로 4 * 4의 2차원 행렬로 간단히 표현할 수 있고, 4 * 4의 2차원 행렬은 기존의 행 우선이 아닌 열 우선으로 행렬의 순서를 표시한다. AES의 라운드 함수 내에 크게 4가지의 독립적인 함수가 있으며, 각각의 라운드 함수는 다음과 같다.

- SubBytes(SB): 8비트 S박스를 이용한 비선형 바이트 치환 함수.
- ShiftRows(SR): 행 단위 왼쪽 회전 함수로 첫 번째 행은 변환하지 않으며, 두 번째 행은 1바이트 회전 세 번째 행은 2바이트 회전 네 번째 행은 3바이트 회전을 적용하는 함수.

● MixColumns(MC): 열 단위로 혼합을 수행하는 32비트 선형 변환 함수.
● AddRoundKey(ARK): 라운드 키와 덧셈을 수행하는 함수.

AES의 암호화 과정에서 라운드 함수를 적용하기 전에 표백(whitening) 단계로서 평문과 라운드 키 덧셈을 먼저 적용하고 4가지 라운드 함수를 수행한다. 그리고 마지막 라운드는 MixColumn() 함수를 제외하고 라운드 함수를 수행한다. 복호화 과정은 암호화 과정에서 수행한 4개의 개별 함수들의 역 변환 함수가 존재하고 그 역 변환 함수와 암호화 과정에서 사용된 라운드 키를 역순으로 적용한다.

AES의 키 스케줄링 알고리즘은 128비트 키로 설명하고 192, 256비트 키는 128비트 키 스케줄링과 유사하며 자세한 내용은 참고문헌[24]에 있다. 먼저 128비트 키($K^0$, $K^1$, $K^2$, $K^3$)를 가지고 라운드 키 $K^4$, $K^5$, ... ,$K^{43}$을 생성한다. 알고리즘은 다음과 같다.

for i = 4, i ≤ 43, i = i + 1,
  if i ≡ 0 mod 4,
    then $K^i = K^{i-4} \oplus SB(RotByte(K^{i-1})) \oplus Rcon(i/4)$,
    else $K^i = K^{i-4} \oplus K^{i-1}$,
  end if
end for

SB()는 S박스 치환 함수이며, RotByte는 1바이트 왼쪽 회전연산 함수이고, Rcon은 고정된 상수이다.

## 2.2 RC6

RC6는 AES 선정 프로젝트에서 5개의 최종 후보 알고리즘에 포함된 효율성과 안전성이 검증된 블록 암호 알고리즘으로 최종적으로 선정되지는 않았다. 그러나 암호 알고리즘의 빠른 수행 필요로 하는 시스템에서는 RC6 암호 알고리즘이 AES보다는 수행속도가 빨라 많이 사용하고 있다. RC6의 설계 원리는 RC5[29]를 확장한 128비트 블록 암호 알고리즘으로 데이터 의존 회전연산(data-dependent rotation)과 32비트 곱셈연산으로 구성되어 있으며, 변형된

Feistel 구조로 암호와 복호 알고리즘이 서로 다르다.

RC6은 RC6-w/r/b로 표기한다. 여기서 w는 워드의 크기로 32비트이며, r은 라운드 수로 블록의 크기가 128비트인 경우 20라운드이다. b는 암호화 키의 바이트 수로 16바이트이다.

RC6의 암호와 복호는 32비트 워드단위로 A, B, C, D 4개의 저장 장소에 평문 또는 암호문을 가지고 20라운드를 반복 수행 후 암호문 또는 평문을 생성한다. RC6의 암호 알고리즘은 다음과 같다.

$B = B + K^0$,
$D = D + K^1$,
for $i = 1, i \le r, i = i + 1$,
  $t = (B * (2B + 1)) \lll 5$,
  $u = (D * (2D + 1)) \lll 5$,
  $A = ((A \oplus t) \lll u) + K^{2i}$,
  $C = ((C \oplus u) \lll t) + K^{2i+1}$,
  $(A, B, C, D) = (B, C, D, A)$,
end for
$A = A + K^{2r+2}$,
$C = C + K^{2r+3}$,

RC6의 복호 알고리즘은 다음과 같다.

$C = C - K^{2r+3}$,
$A = A - K^{2r+2}$,
for $i = r, i \ge 1, i = i - 1$,
  $(A, B, C, D) = (D, A, B, C)$,
  $u = (D * (2D + 1)) \lll 5$,
  $t = (B * (2B + 1)) \lll 5$,
  $C = ((C - K^{2i+1}) \ggg t) \oplus u$,
  $A = ((A - K^{2i}) \ggg u) \oplus t$,
end for
$D = D - K^1$,
$B = B - K^0$,

RC6의 라운드 키는 각각의 20라운드에서 32비트 2개의 키와 라운드 함수 시작 전과 후에 표백 단계로서 각각 2개씩 사용하여 총 32비트 44개의 라운드 키를 사용한다. RC6의 키 스케줄링 알고리즘은 다음과 같다.

**입력:** $L^0, L^1, ..., L^{c-1}$**개의 입력 키 워드**
**출력:** $K^0, K^1, ..., K^{43}$**개의 라운드 키 워드**

$K^0 = 0xB7E15163,$
for $i = 1, i \leq 2r + 3, i = i + 1,$
  $K^i = K^{i-1} + 0x9E3779B9,$
end for
$A = B = i = j = 0,$
for $l = 1, l \leq 132, l = l + 1,$
  $A = K^i = (K^i + A + B) \lll 3,$
  $B = L^j = (L^j + A + B) \lll (A + B),$
  $i = (i + 1) \bmod 44,$
  $j = (j + 1) \bmod 132,$
end for

## 2.3 SHACAL-1

2000년 Handschuh와 Naccache에 의해 소개된[26] 160비트 블록 암호 SHACAL은 표준 해시 함수인 SHA-1[30, 31]의 알고리즘을 핵심으로 하고 있다. 그리고 2001년 SHACAL의 개발자들은 SHACAL-1과 SHACAL-2로 2개 다른 버전으로 나누어 다시 소개했고, SHACAL-1은 처음 소개한 SHACAL을 그대로 적용하고 SHACAL-2는 SHA-256[32]의 압축 함수를 기반으로 한 256비트 블록 암호이다.

SHACAL-1과 SHACAL-2는 유럽의 암호 알고리즘 공개모집 프로젝트인 NESSIE에 제안된 블록 암호 알고리즘으로 SHACAL-1은 키 스케줄링 알고리즘의 취약점 때문에 최종적으로 선택되지 못했고, SHACAL-2는 2003년 NESSIE 프로젝트의 256비트 블록 암호 알고리즘으로 채택 되었다. 이 절에서는 먼저 SHACAL-1의 암호와 복호 과정을 설명한다.

SHACAL-1은 160비트 블록 암호 알고리즘으로 변형된 Feistel 구조로 암

호와 복호가 서로 다른 알고리즘으로 전체 80라운드로 구성되어 있다. 사용된 키 길이는 최소 128비트에서 최대 512비트로 가변적으로 적용 가능하며, 512비트 미만 일 경우 512까지 0으로 패딩(padding)하여 사용한다. 160비트의 평문은 32비트 워드 $A^0$, $B^0$, $C^0$, $D^0$, $E^0$에 나누어 저장한 후 80라운드를 수행한 후 $A^{80}$, $B^{80}$, $C^{80}$, $D^{80}$, $E^{80}$에 최종적인 암호문을 생성한다. 라운드 함수의 수행 과정은 다음과 같다.

for $r = 0, r \leq 79, r = r + 1,$
  $A^{r+1} = K^r + (A^r \lll 5) + f^r(B^r, C^r, D^r) + E^r + Y^r,$
  $B^{r+1} = A^r,$
  $C^{r+1} = B^r \lll 30,$
  $D^{r+1} = C^r,$
  $E^{r+1} = D^r,$
end for

라운드 함수에서 $K^r$은 32비트 라운드 키이며, $Y^r$은 32비트 라운드 상수이다. 그리고 $f^r(B^r, C^r, D^r)$는 SHACAL-1의 전체 80라운드에서 20라운드 마다 서로 다른 연산을 수행하는 라운드 함수이다. 라운드 함수의 구성은 다음과 같다.

for $r = 0, r \leq 19, r = r + 1,$
  $f^r(B^r, C^r, D^r) = (B^r \& C^r) \mid (\sim B^r \& D^r),$
end for

for $r = 20, r \leq 39, r = r + 1,$
  $f^r(B^r, C^r, D^r) = B^r \oplus C^r \oplus D^r,$
end for

for $r = 40, r \leq 59, r = r + 1,$
  $f^r(B^r, C^r, D^r) = (B^r \& C^r) \mid (B^r \& D^r) \mid (C^r \& D^r),$
end for

for $r = 60, r \leq 79, r = r + 1,$

$f^r(B^r, C^r, D^r) = B^r \oplus C^r \oplus D^r$,
end for

"X − Y = X + ($2^{32}$ − 1 − Y) + 1 = X + (~Y) + 1"과 같은 뺄셈 연산의 특징을 이용하여 SHACAL-1의 80라운드 복호 알고리즘은 다음과 같다.

for r = 79, r ≥ 0, r = r - 1,
$A^r = B^{r+1}$,
$B^r = C^{r+1} \ggg 30$,
$C^r = D^{r+1}$,
$D^r = E^{r+1}$,
$E^r = A^{r+1} + (\sim(B^{r+1} \lll 5)) + (\sim f^r((C^{r+1} \lll 2), D^{r+1}, E^{r+1})) +$
$(\sim Y^r) + (\sim K^r) + 4$,
end for

키 스케줄링 알고리즘은 초기 키 512비트를 이용하여 라운드 마다 32비트 라운드 키 1개씩 사용하여 80라운드에 사용될 2560비트의 키로 확장하는 과정으로 다음과 같다.

$K = K^0, K^1, \ldots, K^{15}$,
for i = 16, i ≤ 79, i = i + 1,
$K^i = (K^{i-3} \oplus K^{i-8} \oplus K^{i-14} \oplus K^{i-16}) \lll 1$,
end for

SHACAL-1은 20라운드 단위로 서로 다른 라운드 함수 f()와 라운드 상수 $Y^r$을 수행하는 특성이 있고, 라운드마다 32비트 라운드 키를 한 개씩 사용한다. 그리고 SHACAL-1은 비선형 변환인 S박스가 없으며, 20라운드 마다 다른 간단한 논리연산으로 구성된 비선형 변환 라운드 함수 f()를 사용하는 특징을 가지고 있다.

## 2.4 SHACAL-2

256비트 블록 암호 알고리즘인 SHACAL-2는 SHACAL-1과 유사한 변형

된 Feistel 구조로 암호와 복호가 서로 다른 알고리즘으로 전체 64라운드로 구성되어 있다. 키 스케줄링 역시 SHACAL-1과 유사하며, 사용된 키 길이는 최소 128비트에서 최대 512비트로 가변적으로 적용 가능하며, 512비트 미만 일 경우 512비트까지 0으로 패딩(padding)하여 사용한다. 256비트의 평문은 32비트 워드 $A^0$, $B^0$, $C^0$, $D^0$, $E^0$, $F^0$, $G^0$, $H^0$에 나누어 저장한 후 암호 알고리즘 64라운드를 수행한 후 $A^{64}$, $B^{64}$, $C^{64}$, $D^{64}$, $E^{64}$, $F^{64}$, $G^{64}$, $H^{64}$에 최종적으로 암호문을 생성한다. 라운드 함수의 수행 과정은 다음과 같다.

for $r = 0, r \leq 63, r = r + 1,$
$T_1^{r+1} = H^r + \sum_1(E^r) + Ch(E^r, F^r, G^r) + Y^r + K^r,$
$T_2^{r+1} = \sum_0(A^r) + Maj(A^r, B^r, C^r),$
$H^{r+1} = G^r,$
$G^{r+1} = F^r,$
$F^{r+1} = E^r,$
$E^{r+1} = D^r + T_1^{r+1},$
$D^{r+1} = C^r,$
$C^{r+1} = B^r,$
$B^{r+1} = A^r,$
$A^{r+1} = T_1^{r+1} + T_2^{r+1},$
end for

$T_1$과 $T_2$는 32비트 임시변수이고, $K^r$은 32비트 라운드 키이며, $Y^r$은 32비트 라운드 상수이다. Ch(X, Y, Z), Maj(X, Y, Z), $\sum_0(X)$, $\sum_1(X)$함수의 정의는 다음과 같다.

$Ch(X, Y, Z) = (X \& Y) \oplus (\sim X \& Z),$
$Maj(X, Y, Z) = (X \& Y) \oplus (X \& Z) \oplus (Y \& Z),$
$\sum_0(X) = (X \gg 2) \oplus (X \gg 13) \oplus (X \gg 22),$
$\sum_1(X) = (X \gg 6) \oplus (X \gg 11) \oplus (X \gg 25),$

SHACAL-2의 복호 알고리즘은 SHACAL-1과 같이 2의 보수(Complement)를 이용한 뺄셈으로 쉽게 구현 할 수 있다. 알고리즘은 다음과 같다.

for r = 63, r ≥ 0, r = r − 1,

$T_1^{r+1} = A^{r+1} - \sum_0(B^{r+1}) - Maj(B^{r+1}, C^{r+1}, D^{r+1})$,

$= A^{r+1} + (\sim\sum_0(B^{r+1})) + (\sim Maj(B^{r+1}, C^{r+1}, D^{r+1})) + 2$,

$H^r = T_1^{r+1} - \sum_1(F^{r+1}) - Ch(F^{r+1}, G^{r+1}, H^{r+1}) - K^r - Y^r$,

$= T_1^{r+1} + (\sim\sum_1(F^{r+1})) + (\sim Ch(F^{r+1}, G^{r+1}, H^{r+1}))$

$+ (\sim K^r) + (\sim Y^r) + 4$,

$G^r = H^{r+1}$,

$F^r = G^{r+1}$,

$E^r = F^{r+1}$,

$D^r = E^{r+1} - T_1^{r+1} = E^{r+1} + (\sim T_1^{r+1}) + 1$,

$C^r = D^{r+1}$,

$B^r = C^{r+1}$,

$A^r = B^{r+1}$,

end for

SHACAL-2의 키 스케줄링 알고리즘은 최소 128비트에서 최대 512비트까지 이며, 512비트 미만일 경우 0으로 512비트까지 채운다. 그리고 SHACAL-2 개발자들은 128비트 미만의 키 사용은 강력히 제한하고 있다[33]. 키 스케줄링 알고리즘은 초기 키 512비트로 라운드 마다 32비트 키 1개씩 사용하여 총 2048비트로 확장한다. 다음은 2048비트로 키 확장 과정이다.

for i = 16, i ≤ 63, i = i + 1,

$K^i = \sigma_1(K^{i-2}) + K^{i-7} + \sigma_0(K^{i-15}) + K^{i-16}$,

end for

$\sigma_0(X) = (X \gg 7) \oplus (X \gg 18) \oplus (X \gg 3)$,

$\sigma_1(X) = (X \gg 17) \oplus (X \gg 19) \oplus (X \gg 10)$,

SHACAL-2 역시 비선형 변환인 S박스가 없으며, 라운드 마다 1개의 32비트 라운드 키를 사용하며, 논리 연산만으로 구성된 비선형 변환 라운드 함수 Ch(X, Y, Z), Maj(X, Y, Z), $\sum_0(X)$, $\sum_1(X)$를 사용하는 특징을 가지고 있다.

## 2.5 블록 암호의 안전성 기준

암호 분석(Cryptanalysis, Codebreaking)은 평문이나 비밀 키 또는 두 가지 모두를 찾으려고 노력하는 모든 과정이라 할 수 있다. 암호 분석 기법은 암호 알고리즘의 특성과 암호 분석에 사용 가능한 유용한 정보에 따라 다양한 방법이 존재한다. 이 절에서는 암호 분석자(Cryptanalyst, Codebreaker)가 얻을 수 있는 유용한 정보에 따라 여러 가지 분석 유형에 대해서 설명한다[1].

- 암호문 단독 분석(Ciphertext only): 암호 알고리즘과 분석할 암호문만 암호 분석자는 사용할 수 있다. 이 경우는 가장 어려운 분석 방법으로 어떤 경우에는 암호 알고리즘도 알 수 없다. 이 경우는 유용한 분석 방법이 없으며, 단지 전수 조사(brute-force) 밖에는 다른 특별한 분석 방법이 없다. 현대 암호이전의 암호 분석자는 대부분 전수 조사를 통한 암호문 단독 분석 이었으며, 사용 가능한 키 공간(키 개수)이 적을 경우에는 매우 효과적인 분석 방법이다. 그러나 현대 암호에서는 사용 가능한 키 공간이 매우 크므로($2^{128}$이상) 이 분석 방법은 효과적이지 않다. 그리고 현실 세계에서 분석자들이 가장 많이 대면하게 되는 분석 시나리오이다.

- 기지 평문 분석(Known Plaintext): 암호 알고리즘, 분석할 암호문, 비밀 키에 의해 생성된 원하는 만큼의 평문-암호문 쌍을 가지고 암호 분석을 하는 방법이다. 실세계에서 분석자가 특정한 키에 의해서 생성된 평문-암호문 쌍을 원하는 만큼의 얻기는 불가능하다. 그러나 현대 암호학은 암호 알고리즘의 공개를 원칙으로 하고 있기 때문에 공개된 알고리즘으로 평문-암호문 쌍을 쉽게 얻을 수 있다. 그래서 기지 평문 분석은 가장 많이 그리고 가장 쉽게 분석 할 수 있는 시나리오이다.

- 선택 평문 분석(Chosen Plaintext): 암호 알고리즘, 분석할 암호문, 분석자가 원하는 평문과 그 평문을 비밀 키에 의해 생성된 암호문을 가지고 분석하는 방법이다. 기지 평문 분석보다는 강력한 분석 방법으로 분석자가 원하는 어떤 패턴이 있는 평문들과 이에 대응되는 암호문 쌍을 가지고 분석한다. 이는 단순한 평문-암호문 쌍으로 된 정보보다 더 유용한 정보를 얻을 수 있다. 이와 같은 분석 방법의 정보는 실세계에서는 얻을 수 없지만 암호 알고리즘의 공개에 의해 얼마든지 분석자가 원하는 정보를 만들어 사용할 수 있다.

● 선택 암호문 분석(Chosen Ciphertext): 암호 알고리즘, 분석할 암호문, 분석자가 원하는 특정 암호문과 그 암호문을 비밀 키에 의해 복호된 평문을 가지고 분석하는 방법이다. 선택 평문 분석과는 반대로 어떤 패턴의 암호문에 대응되는 평문 쌍을 가지고 분석하는 방법이다.

● 능동 평문/암호문 분석(Adaptive Plaintext/Ciphertext): 암호 알고리즘, 분석할 암호문, 분석자가 원하는 평문과 그 평문을 비밀 키에 의해 생성된 암호문, 분석자가 원하는 암호문과 그 암호문을 비밀 키에 의해 복호된 평문을 가지고 분석하는 방법이다. 선택 평문과 암호문을 조합하여 분석하는 방법으로 암호 개발자나 암호 학자에 의해 현대 암호 알고리즘의 분석 방법으로 가장 많이 사용 되는 분석 방법이다.

암호 분석자들은 이와 같은 실용적인 정보와 분석할 파일의 정보, *.HWP, *.DOC, *.EXE 등과 문장이 한글, 영어, 일본어, 중국어 등 그리고 문장의 형식에 대한 정보를 종합하여 모든 가능한 방법을 사용하여 암호를 분석한다. 이러한 분석 시나리오에 대응하는 안전한 암호 알고리즘의 설계는 쉽지 않으며, 안전한 암호 알고리즘의 설계를 위한 두 가지 용어를 정의할 수 있다. 첫 째는 절대 안전성(Unconditionally Secure)으로 어떤 암호 알고리즘으로 생성된 암호문을 무한대로 제공되어도 원하는 평문을 결정할 수 있는 정보를 포함하지 않는 경우로 필요한 정보가 암호문에 포함되어 있지 않기 때문에 무한대의 시간과 비용을 사용하여도 그 암호문은 해독되지 않는다. 대표적인 방법이 일회용 암호(One-time pad)[34]이고, 이는 한번 사용한 키는 다시 사용하지 않으며, 평문의 길이와 같은 길이의 비밀 키가 필요하므로 현대 암호 시스템에서는 실현 불가능하다.

두 번째는 계산상 안전성(Computationally Secure)으로 암호 분석에 사용된 비용이나 시간이 암호화된 정보의 가치나 유효기간을 초과하는 경우로 이 두 가지 기준을 만족할 경우 계산상 안전하다고 할 수 있다. 그리고 암호 알고리즘의 안전성은 분석자의 계산능력(Computing Power)에 좌우된다. 암호 알고리즘이 “절대적 또는 이론적으로 안전하다.”는 의미는 분석자의 계산능력이 무한대 일 때에도 암호 알고리즘의 분석이 불가능하다는 의미이며, 제한적인 계산능력을 갖는 분석자에 대해 안전할 경우는 “계산적 또는 실용적으로 안전하다.”라고 말할 수 있다.

마지막으로 사용 가능한 비밀 키 공간보다 더 적은 분석 방법으로 암호를

분석할 수 있다면, 그 암호 알고리즘은 분석되었다. 또는 안전하지 않다고 말할 수 있다. 다시 말해 비밀 키에 대해 전수조사 보다 효율적인 분석 방법이 있으면, 그 암호 알고리즘은 안전하지 않다고 말 할 수 있다.

## 2.6 블록 암호의 공격 방법

앞 절에서 분석자가 사용 가능한 정보에 의한 여러 가지 분석 방법과 암호 알고리즘의 안전성에 대해서 설명했다. 이 번 절에서는 블록 암호의 특성(구조, 사용된 연산 등)에 따른 암호 알고리즘의 공격 방법에 대해 설명한다. 특히 블록 암호의 대표적인 공격 방법인 차분 공격(Differential Attack), 선형 공격(Linear Attack), Square Attack[35], 연관 키 공격(Related-Key Attack)[36]에 대해 설명한다.

### 2.6.1 차분 공격

1990년 Biham과 Shamir에 의해 개발된 차분 공격은 가장 강력한 블록 암호 공격 방법 중 하나이다. 그래서 이 공격방법이 소개된 후 당시의 블록 암호의 상당수가 이 방법에 의해서 분석되었다.[37, 38, 39] 차분 공격의 강력함은 다른 많은 블록 암호 알고리즘에 적용되기 위해 다양한 변형된 공격 방법으로 진화되었다. 이와 같이 변형된 차분 공격은 부정 차분 공격(truncated differential attack)[40], 차분 선형 공격(differential-linear attack)[41], 불능 차분 공격(impossible differential attack)[42], boomerang attack[43], rectangle attack[44] 등이 있다.

차분 공격의 기본 개념은 (그림 2-1)과 같다. 두 개의 입력 P, $P^*$와 이에 대응되는 출력 C, $C^*$의 XOR 연산 결과인 β를 알고 있을 때 입력 P, $P^*$의 XOR 연산 결과인 α는 이때 사용된 키 K와는 상관없는 값이 되므로 α에서 β로 변환 가능한 입력 쌍 P ⊕ K와 $P^*$ ⊕ K를 미리 조사해서 알고 있다면 P와 P ⊕ K를 XOR 연산을 통해 키 K를 구할 수 있다는 사실을 이용하여 공격하는 방법이다.

차분 공격의 첫 번째 할 일은 높은 확률의 차분 특성(characteristic differential)을 구하는 것이다. 차분 특성을 구하는 방법은 암호 알고리즘 E에 대해 모두 같은 입력의 차분(α)을 갖는 입력 쌍에 대응되는 출력 차분의 분포(distribution)를 조사한다. 이 때 입력차분(α)과 출력차분(β)의 분포가 일정하

지 않고, 확률이 $p > 2^{-n}$이면, 유용한 차분 공격을 수행할 수 있다. 이와 같은 차분 특성을 수식으로 표현하면 (수식 2-1)과 같다.

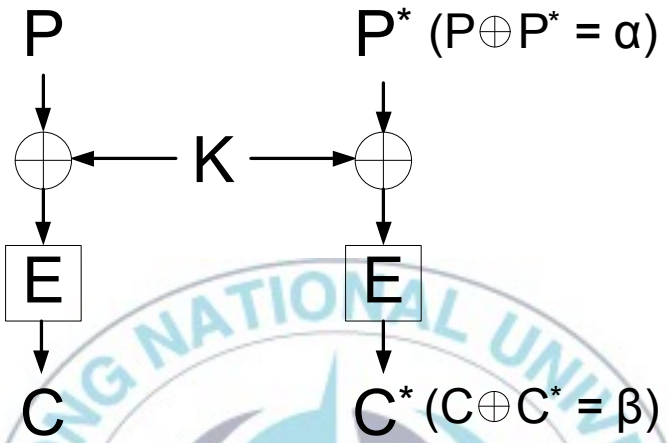



그림 2-1. 기본 차분 공격 원리

$$\Pr[E_K(P) \oplus E_K(P \oplus \alpha) = \beta] = p > 2^{-n} \qquad (2-1)$$

(수식 2-1)의 의미는 차분 특성이 α → β인 확률이 전수조사 보다 높은 확률 $p$일 경우 이 때 사용된 키 K를 전수조사보다 좋은 효율로 찾을 수 있다는 의미이다. 이와 같은 차분 특성을 선택 후 다음과 같은 방법으로 차분 공격을 할 수 있다.

① 차분 특성 확률 $p$에 대해서, $p$ / c개의 입력 차분이 α인 입력 쌍(P, $P^*$)을 수집한다. (상수 c > 1)
② 선택 평문 분석 시나리오에 의해 입력 쌍에 대응되는 암호문 쌍(C, $C^*$)을 구한다.
③ 암호 알고리즘에 사용된 각 후보 키에 대해서, (수식 2-2)를 만족하는 암호문 쌍의 개수를 카운트 한다.

$$E_K^{-1}(C) \oplus E_K^{-1}(C^*) = \beta \qquad (2-2)$$

단계 ②에서 구한 암호문 쌍이 (수식 2-2)를 만족할 경우, 옳은 쌍(right pair)이라 하고, 이 옳은 쌍의 기대되는 개수는 약 $(c * p) / 2^n$이 된다. 그리고 차분 특성 확률이 $p > 2^{-n}$ 이므로 옳은 쌍이 틀린 쌍(wrong pair)보다 크면, 이 경우 차분 공격은 성공한다.

### 2.6.2 선형 공격

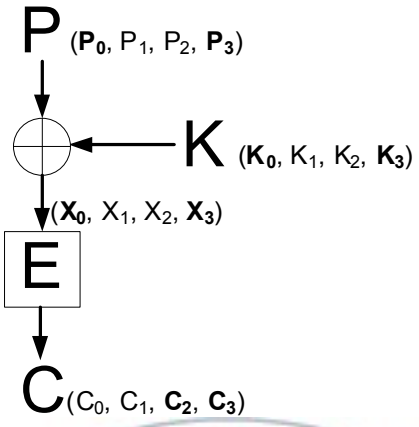
선형 공격은 1993년 Matsui에 의해 소개된 기지 평문(암호문) 공격으로 비밀 키 암호 알고리즘을 공격하는 대표적인 공격 방법이다. 그리고 어떤 블록 암호 분석에서는 선형 공격이 차분 공격과 유사한 특성을 보인다[45]. 선형 공격 역시 암호 알고리즘에 따라 다양한 변형된 공격 방법이 존재한다. 이와 같이 변형된 선형 공격은 다중 선형 공격(multiple linear attack)[46], 비선형 공격(nonlinear attack)[47], 선택 평문 선형 공격(chosen plaintext linear attack)[48], 이중 선형 공격(bilinear attack)[49] 등이 있다.

선형 공격의 첫 번째 할 일은 입력과 출력 사이에 높은 상관관계(correlation)가 있는 선형 근사값(linear approximation)을 구하는 것이다. 선형 근사 값을 구하는 방법은 암호 알고리즘 E에 대해 특정한 입력 비트(들)와 이에 대응되는 높은 편차(bias: κ)를 갖는 특정한 출력 비트(들)를 조사한다. 암호 알고리즘 E에 대해서 입력 ΓP → 출력 ΓC로 유도 되는 선형 근사값이 존재하면서, 편차 κ가 $\kappa^2 > (c / 2^n)$이면, 유용한 선형 공격을 수행할 수 있다. 이와 같은 선형 근사 값을 수식으로 표현하면 다음과 같다.

$$|\Pr[P \bullet \Gamma P \oplus E_K(P) \bullet \Gamma C = 0] - 1/2| = \kappa \qquad (2-3)$$

(수식 2-3)에서 • 는 비트 별 내적(inner products)이며, 이와 같은 선형 근사값을 선택 후 다음과 같은 방법으로 선형 공격을 할 수 있다.

① 선형 근사 값의 편차 κ에 대해, $c / \kappa^2$개의 평문 P를 수집한다.
(상수 $c > 1$)
② 기지 평문 분석 시나리오에 의한 평문 P에 대응되는 암호문 C를 구한다.
③ 암호 알고리즘에 사용된 각 후보 키에 대해서, (식 2-4)를 만족하는 평문, 암호문 쌍(P, C)의 개수를 카운트 한다.




$X_0 \oplus X_3 = C_2 \oplus C_3 \rightarrow P_0 \oplus P_3 \oplus C_2 \oplus C_3 = K_0 \oplus K_3$

그림 2-2. 기본 선형 공격 원리

$$P \cdot \Gamma P \oplus E_K^{-1}(C) \cdot \Gamma C = 0 \qquad (2-4)$$

선형 공격은 근사 확률(probability of approximation)이 (수식 2-4)에서 틀린 키(wrong key)가 1/2($\kappa = 0$)로 수렴하면, 옳은 키(right key)는 편차 $\kappa$로 수렴되는 사실은 기반으로 하고 있다. 선형 공격의 성공은 상수 c와 가능한 모든 후보 키에 의해 암호 알고리즘 E를 수행 후 생성된 암호문의 수와 관계가 많다.

### 2.6.3 Square 공격

Square 공격은 블록 암호 Square를 소개할 때 같이 소개된 블록 암호 공격 알고리즘으로 특히 Square 단위로(주로 8비트) 연산하는 SPN 구조의 블록 암호 알고리즘(AES, Square, CRYPTON[50]) 공격에 바로 적용할 수 있어서 이와 유사한 블록 암호 알고리즘 설계에는 반드시 고려해야 할 공격 방법이다. 이 공격 방법이 소개된 후 변형된 공격 방법으로 multiset attack[51], 적분 공격(integral attack)[52] 등이 있다.

Square 공격은 바이트 단위로 0~255까지 가질 수 있는 일부 특정한 바이트(active byte)와 이 특정한 바이트를 제외한 나머지 고정된 상수 값을 갖는 바이트(non-active byte)가 암호 라운드를 거치면서 모든 값(0~255)을 갖도록 균형 잡혀 진다는 성질을 이용한 공격이다. 균형성(balancedness)은 가능한

모든 값들의 합이 0이라는 의미고, 수식은 다음과 같다.

$$\sum_{i=0}^{255} \oplus A_i = 0 \qquad (2-5)$$

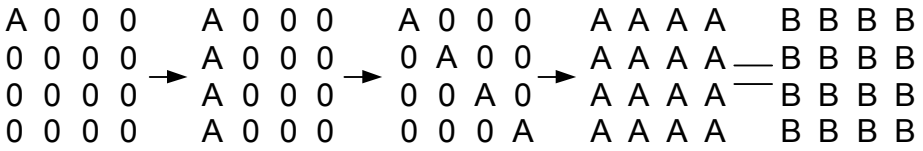



그림 2-3. 기본 Square 공격 원리

(그림 2-3)은 Square 단위(바이트 단위)로 연산을 수행하는 SPN 구조의 3단계 진행과정이다. 먼저 0~255의 값을 가지는 한 개의 활성화 바이트(A)와 나머지 15개의 고정된 상수(0)로 된 바이트가, (그림 2-3)과 같이 오른쪽으로 진행하면서(암호 알고리즘에 따라 연산과정이 다를 수도 있음) 3단계 후 출력이 모두 균형성(B)을 갖는 값을 갖게 된다. 균형성의 실질적인 의미는 활성화 바이트의 각 비트는 0과 1이 정확히 짝수 번 발생하고, 비활성화 바이트에서는 동일한 값이 짝수 번 발생한다. 이는 XOR 연산을 통해 균형성을 갖게 되고, 이 균형성과 랜덤(random)함수와 구별 가능해진다.

SPN 구조의 암호 알고리즘에서 초기 키 덧셈부분은 가능한 모든 키 값으로 XOR 연산을 통해 키 값을 상쇄(balancing)시킨다. 그래서 키 덧셈은 균형성에 영향을 주지 못한다. 이와 같이 매 라운드 마다 수행하는 키 덧셈은 균형성의 성질을 유지하므로 Square 공격에서는 키 덧셈을 고려할 필요가 없다. 그리고 활성화 바이트의 위치만 변화(Permutation)시키는 라운드 연산도 균형성을 그대로 유지한다. 그래서 (그림 2-3)에서 맨 마지막 라운드인 상쇄된 상태에서 적용된 라운드 키를 (수식 2-6)으로 찾을 수 있다.

$$a_{i,j} = SB(b_{i,j}) \oplus K_{i,j} \qquad (2-6)$$

(수식 2-6)에서 i는 행, j는 열을 나타내고 SB()는 8비트 치환 연산이다. 이는 상쇄된 라운드에서 사용된 라운드 키는 전수조사보다 매우 효율적인 방법이다.

### 2.6.4 연관 키 공격

연관 키 공격은 1992년과 1993년에 각각 독립적으로 Knudsen과 Biham에 의해 키 관계(related keys)를 이용한 비밀 키 암호 공격 방법으로 키들 간에 연관성은 알고 있지만 키 자체를 모를 때 각각의 키들로부터 생성된 평문, 암호문 쌍을 가지고 키를 찾는 공격 방법이다.

연관 키 공격의 기본은 암호, 복호 알고리즘과 그 암호 알고리즘에서 사용된 키 스케줄링 알고리즘이다. 특히 키 스케줄링 알고리즘이 약한 키(weakness key)를 생성할 경우 효과적이다. 이 공격은 먼저 연관 키에 대한 차분 특성을 구한다. 차분 특성의 수식은 다음과 같다.

$$\Pr[E_K(P) \oplus E_{K \oplus \Delta K}(P \oplus \alpha) = \beta] = p > 2^{-n} \qquad (2-7)$$

(수식 2-7) ΔK는 0이 아닌 키 차분이다. 이와 같이 키 차분을 이용하여 기본적인 차분 공격 알고리즘을 그대로 사용한다. 그러나 연관 관계(키 차분)가 있는 각각의 키로부터 암호화된 평문과 암호문 쌍을 구하는 것은 비현실적이므로 실제적인 공격 방법으로 보기는 어렵다. 그러나 키 사용이 규칙적으로 변하는 경우나 키 공유 프로토콜에 인위적으로 개입하여 조작 할 수 있는 경우에는 연관 키 공격이 적용될 수 있다. 연관 키 공격의 실질적인 분석은[53, 54] IBM 4758 cryptoprocessor에서 키 교환 프로토콜에 적용된 사례가 있다.

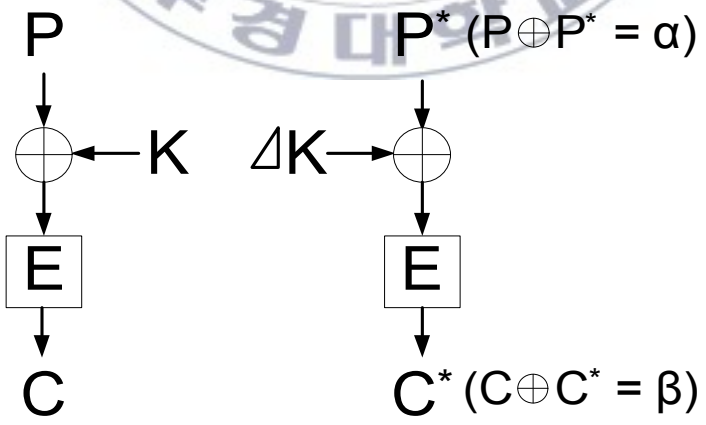



그림 2-4. 기본 연관 키 공격 원리

# 제 3장 대칭단과 대칭단을 적용한 구조

블록 암호 알고리즘은 3가지 부분, 암호 알고리즘 E, 복호 알고리즘 $E^{-1}$, 그리고 키 스케줄링 알고리즘으로 구성되며, 블록 암호의 설계는 이 3가지를 설계하는 것이다. 특히 암호 알고리즘 E의 블록 크기(현대 블록 암호의 블록 크기는 128비트)를 결정하여 평문을 블록 크기로 나누어 암호를 수행한다.

$$E: \{0,1\}^K \bullet \{0,1\}^n \rightarrow \{0,1\}^n \qquad (3-1)$$

(수식 3-1)과 같이 n비트 암호 알고리즘 E는 n비트 블록에 K비트 키를 적용하여 n비트 암호문을 만드는 과정을 수식으로 표현한 것이다. 그리고 특정 키 K에 대해서 n비트 평문과 n비트 암호문은 일대일 대응되는 전단사 함수이다. 암호화 과정을 적용할 때 만일 평문의 크기가 블록의 크기보다 클 경우는 블록 암호 운영모드(mode of operation)를 적용해야 한다. 블록 암호의 대표적인 운영모드에는 ECB 모드(Electronic Code Book Mode), CBC 모드(Cipher Block Chaining Mode), CFB 모드(Cipher Feed Back Mode), OFB 모드(Output Feed Back Mode), 그리고 CTR 모드(CounTeR Mode)[55]가 있다. 또 평문의 크기가 블록보다 작을 경우 0을 블록의 크기만큼 채워 암호 알고리즘을 수행한다.

현대 블록 암호 알고리즘 개발의 중요한 이론적 기반인 Shannon의 이론[21]은 치환(비선형변환: S박스)과 확산(선형변환)을 반복(iterative) 수행하면 암호 알고리즘의 안전성을 향상 시킬 수 있다는 것이다. 이 이론에 따라 암호 알고리즘의 효율성과 안전성을 고려한 반복 횟수(라운드 수)를 결정한다. 그리고 라운드 수는 짝수이다.

이 장에서의 주요 내용은 대칭단에 대한 전반적인 사실, 대칭단의 목적, 대칭단의 구조, 대칭단이 블록 크기에 따른 변화, 대칭단의 안전성과 최종적으로 결론 순으로 설명한다.

## 3.1 대칭단 구조의 목적

암호와 복호 과정이 다른 구조의 블록 암호 알고리즘을 암호와 복호 알고

리즘이 동일하게 구성하는 알고리즘을 제안한다. 즉, 암호 알고리즘의 전체를 짝수인 R라운드로 구성하고, 1라운드부터 (R / 2) - 1라운드까지는 정함수(암호 알고리즘)를 사용하고, (R / 2) + 1라운드부터 R라운드까지는 역함수(복호 알고리즘)를 사용한다. 또한 정함수 단과 역함수 단 사이에 규칙적인 라운드의 반복을 피하기 위해 불규칙적인 대칭 블록을, 하나의 독립적인 라운드 형태인 대칭단(Symmetric Layer)을 삽입한다.

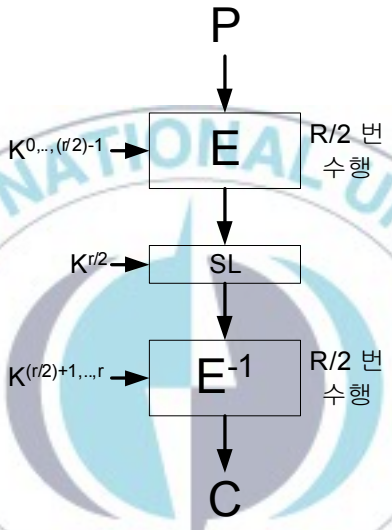



그림 3-1. 대칭단을 적용한 암호 알고리즘 구조

(그림 3-1)은 암호와 복호가 다른 암호 알고리즘에 대칭단을 중간에 적용하여 암호와 복호 알고리즘을 동일하게 구성한 모습을 그림으로 표현한 것이다. 일반적인 블록 암호 알고리즘의 키 적용 규칙은 암호 알고리즘에 적용된 키 순서는 복호 알고리즘에서는 암호의 역순으로 적용한다. 대칭단을 적용한 알고리즘도 복호 과정에서는 암호화 과정에서 적용된 키를 역순으로 적용한다.

암호와 복호가 다른 블록 암호 알고리즘을 대칭단 구조를 적용하여 암호와 복호가 동일하게 재설계를 위한 목적은 다음과 같다.

- 암호와 복호 알고리즘을 동일하게 만들 것.
- 대칭단의 삽입으로 불규칙성을 통해서 암호 알고리즘의 안전성을 향상

시킬 것.

- 소프트웨어 및 하드웨어 구현이 쉬울 것.
- 원본 암호 알고리즘과의 소프트웨어 및 하드웨어 수행속도에서 큰 차이가 없을 것.

## 3.2 암호와 복호 알고리즘 사용

암호와 복호 알고리즘을 사용하여 암호와 복호를 동일하게 구성한다는 아이디어는 비밀 키 암호의 특징에서 얻었다. 비밀 키 암호는 한 개의 비밀 키로 암호와 복호를 수행한다. 이와 같은 특징은 암호 알고리즘과 이에 대응되는 복호 알고리즘은 동일한 키를 사용한다는 것이고, 사용된 그 키에 의해서 평문은 암호문으로, 암호문은 평문으로 변환을 수행하는 전단사 함수이다. 그래서 암호 알고리즘과 복호 알고리즘을 번갈아 사용하고, 이때 암호와 복호에 사용된 비밀 키를 서로 독립적인 키를 사용하므로 해서 암호와 복호가 정상적으로 수행할 수 있도록 구성할 수 있다.

이러한 특징을 실질적으로 구현한 예가 Triple-DES[56]이다. (그림 3-2)는 Triple-DES의 구성을 그림으로 표현한 것이다. 1990년대 후반 DES의 안전성에 문제가 있어 Shannon의 이론에 따라 DES를 세 번 수행하여 안전성을 향상시킨 것이 Triple-DES이다. 즉 Triple-DES는 (그림 3-2)와 같이 암호, 복호, 암호 순으로 알고리즘을 구성하였고, 이때 적용된 비밀 키는 서로 독립적인 다른 키를 사용하였다. 그러나 (그림 3-2)의 DES와 $DES^{-1}$는 같은 알고리즘이며, 그 이유는 DES 알고리즘이 Feistel 구조이기 때문이다. 의미상으로 평문이 첫 번째 DES 수행 후 암호문이 생성되고, 이 암호문이 $DES^{-1}$를 수행하므로 복호 알고리즘이 되고, 마지막으로 DES를 수행하여 암호문을 만든다. 쉽게 말해서 DES 알고리즘을 서로 다른 키로 세 번 수행한다라고 생각하면 된다. 복호는 암호와 동일하게 DES를 세 번 수행하며, 단지 적용된 비밀 키만 암호의 역순으로 적용하면 된다.

암호와 복호가 다른 알고리즘 즉 SPN 구조의 암호 알고리즘도 위와 같은 구성이 가능하다. 이유는 SPN 구조의 암호 알고리즘도 비밀 키 암호의 특징이 그대로 적용되기 때문이다. 그래서 SPN 구조 역시 암호와 복호 알고리즘은 다르지만, 특정 비밀 키로 암호 알고리즘으로 평문을 암호문으로, 이에 대응되는 복호 알고리즘으로 암호문을 평문으로 변환시킨다. SPN 구조나 암호와 복호 알고리즘이 다른 암호 알고리즘의 개발은 반드시 암호 알고리즘에 대응되는 복호

알고리즘도 같이 개발한다. 그래서 암호 알고리즘과 복호 알고리즘을 반반씩 번갈아 구성하여 쉽게 암호와 복호 알고리즘이 동일하게 구성할 수 있다.

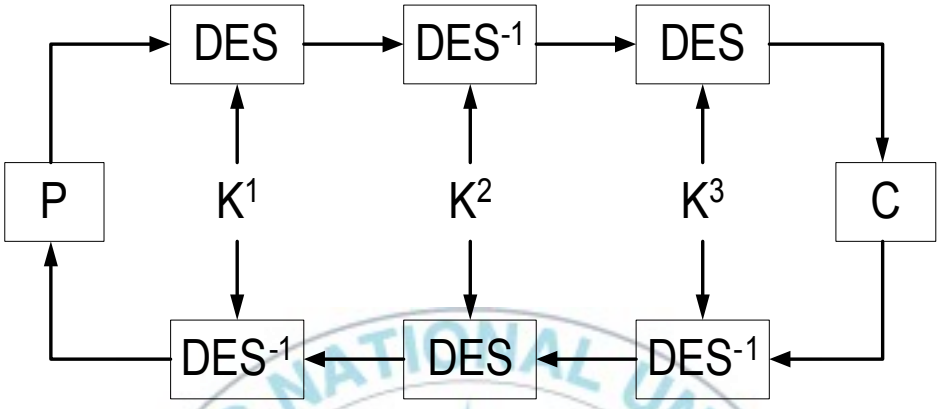



그림 3-2. Triple-DES 구조

암호와 복호가 다른 알고리즘을 암호 알고리즘과 복호 알고리즘을 연속해서 구성하는 것이 암호 안전성 측면에서 어떠한 영향을 주는지 생각해 볼 필요가 있다. Feistel 구조는 Triple-DES의 경우와 같이 단순히 라운드 수의 증가로 안전성을 향상시켰다. 그러나 SPN 구조인 경우는 암호 알고리즘과 복호 알고리즘이 달라 암호 알고리즘의 분석 방법을 복호 알고리즘에도 적용해 보아야 한다. 그래서 대표적인 SPN 구조의 암호 알고리즘인 AES의 복호 알고리즘을 분석했다. AES의 네 개의 복호용 라운드 함수의 분석 결과는 다음과 같다.

- $ARK^{-1}$(라운드 키 덧셈): 암호와 복호가 같은 라운드 키를 사용하므로 XOR 연산은 암호의 안전성에는 아무런 영향을 미치지 않는다.

- $SR^{-1}$(행 변환): 행 변환은 단순한 회전 연산이고, 복호는 암호의 역으로 수행하므로 안전성에 영향을 미치지 않는다.

- $MC^{-1}$(열 변환): 열 변환은 유한체($GF2^8$) 곱셈 연산으로 선형변환을 통해 암호의 특성을 확산 시키는 기능을 한다. 유한체 곱셈은 암호와 복호를 위한 각각 고정된 다항식(fixed polynomial)을 사용한다. 이 고정된 다항식은 유한체 내에서 서로 역함수(multiplicative inverse, 역수)관계이다.

$$c(X) * c^{-1}(X) = '01' \quad (3-2)$$

$$c(X) = '03'X^3 + '01'X^2 + '01'X + '02'$$

$$c^{-1}(X) = '0B'X^3 + '0D'X^2 + '09'X + '0E'$$

(수식 3-2)에서 c(X)는 암호용 고정된 다항식이고, $c^{-1}$(X)는 복호용 다항식이며, 사용된 계수는 16진수이다. 유한체 곱셈 연산이 암호와 복호가 역함수 관계라는 의미는 암호용 열 변환 연산이 예를 들어 MDS(Maximum Distance Separated)[57] 행렬을 생성하면 복호용 열 변환 역시 MDS 행렬을 이룬다는 것이다. 그래서 복호용 열 변환은 암호용 열 변환과 같은 암호학적 분석 안전성을 가지고 있다.

● $SB^{-1}$(바이트 치환): 바이트 치환은 AES 알고리즘에서 비선형 변환을 하는 8비트 S박스 치환을 수행한다. 이는 암호용 S박스와 이에 대응되는 복호용 역 S박스가 각각 있고, 정확히 일대일 대응된다. 그래서 복호용 역 S박스를 대표적인 블록 암호 분석 도구인 차분 공격과 선형 공격을 위한 최대 차분 특성과 선형 근사값을 컴퓨터 시뮬레이션을 통해서 분석했다. 결과는 두 가지 모두 $2^{-6}$으로 암호용 S박스와 같은 결과를 보였다. 바이트 치환 역시 암호와 복호는 안전성 분석측면에서 같다는 결론을 얻었다.

이와 같은 결과를 통해 SPN 구조와 같은 암호와 복호가 다른 알고리즘에서 복호의 안전성은 암호 알고리즘에 대한 안전성 분석 방식을 그대로 복호에 적용해서 안전성을 분석할 수 있고, 그 결과 또한 암호 안전성과 같다는 해석을 할 수 있다. 그러나 단순히 암호 알고리즘과 복호 알고리즘을 연속해서 연결하는 구조는 암호와 복호가 일대일 대응에 의한 서로 상쇄(balancing) 효과가 발생할 가능성이 있다. 이와 같은 현상은 지금은 정확히 분석할 순 없지만 가까운 미래에는 충분히 분석 가능한 취약점이 된다. 그래서 암호와 복호를 연속으로 연결하는 구조가 아니라 암호와 복호 사이에 독립적인 라운드 형태인 비선형 변환을 하는 대칭단을 삽입해서, 암호와 복호가 바로 일대일 대응되지 않고 서로 단절을 시키는 효과를 발생시키는 구조가 적당하다.

## 3.3 암호와 복호를 같게 만들기 위한 다른 노력

SPN 구조의 블록 암호 알고리즘을 암호와 복호가 같은 알고리즘으로 구성하기 위한 많은 연구를 지금도 진행하고 있다. 그 대표적인 예가 우리나라에서 개발된 ARIA[58]이다. ARIA는 128비트 블록에 128, 192, 256비트 가변 키를 지원하고 각각의 키 크기에 따라 10, 12, 14라운드를 적용한다. 치환은 바이트 단위로 적용되며, 2개의 S박스와 각각의 역 S박스를 사용하여 총 4가지 바이트 치환 연산이 있다. 그리고 확산연산은 16 * 16의 involution 이진 행렬을 사용하여 각각의 바이트들을 섞는 involution SPN 구조를 이루고 있다.

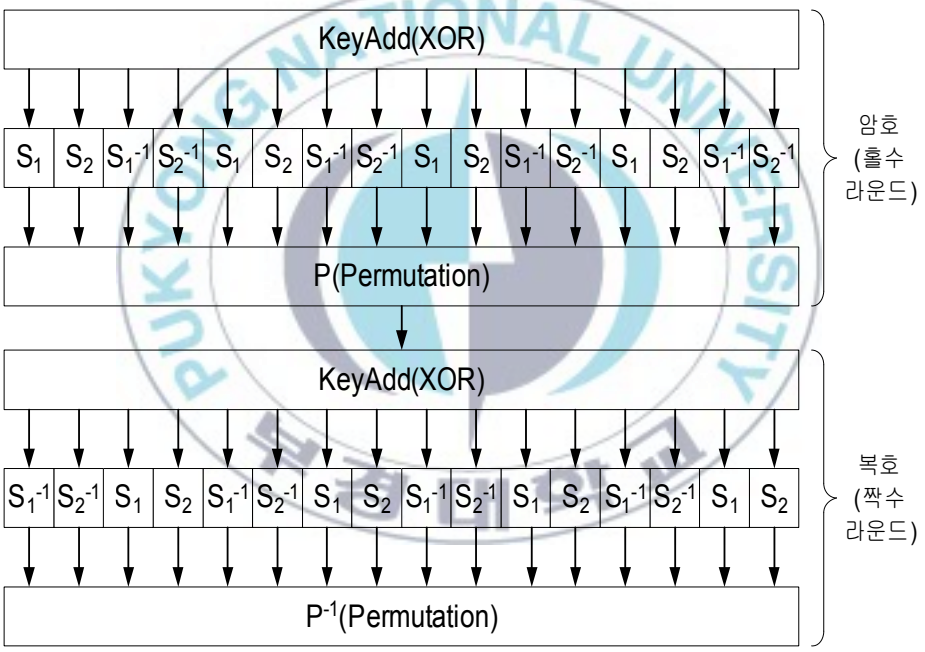



그림 3-3. ARIA의 라운드 함수

(그림 3-3)은 ARIA의 라운드 함수를 표현한 것으로 ARIA는 SPN 구조이지만 암호와 복호를 같게 하기 위해 2종류의 라운드 함수를 번갈아 적용하며 확산연산에서는 암호와 복호가 같은 involution 이진 행렬을 사용한다. 바이트 치환용 S박스는 짝수 라운드와 홀수 라운드가 서로 역변환 가능하도록 배열되어 있고, 확산연산 역시 짝수 라운드와 홀수 라운드가 역변환 가능하도록 배열

되어 있다. 여기서 확산연산에 사용된 involution 행렬이란 $P = P^{-1}$로 변환 가능한 행렬을 말한다. 이와 같은 ARIA의 라운드 구성은 홀수 라운드를 암호 알고리즘이라 했을 때, 짝수 라운드는 이에 대응되는 복호 알고리즘이라 할 수 있다. 그래서 ARIA는 암호와 복호가 같은 SPN 구조이다.

그러나 앞 절에서 설명한대로 암호 알고리즘과 복호 알고리즘을 연속해서 연결하는 구조는 암호와 복호가 일대일 대응에 의한 서로 상쇄효과가 발생할 가능성이 있고 특히 확산연산 과정에서 involution 행렬의 사용으로 AES와는 다르게 MDS 코드를 생성하지 못하며 branch number는 8이다. 그리고 ARIA는 일반적인 대칭 키 암호와는 다르게 암호와 복호용 라운드 키가 서로 다른 특성이 있다. 이는 암호용 라운드 키를 복호에 사용하는 것이 아니라 암호용 라운드 키로 복호용 라운드 키를 새로 만들어 사용하므로 키 셋업 시간이 증가 되는 단점이 된다.

## 3.4 대칭단 구조

앞 절에서의 설명과 같이 대칭단을 암호와 복호 알고리즘 사이에 삽입하여 암호와 복호가 일대일 대응에 의한 상쇄효과를 없애고, 오히려 비선형 변환을 통해 불규칙성을 증가시킬 수 있는 기능을 하는 대칭단의 설계가 필요하다. 이와 같은 조건을 만족하는 대칭단의 구조와 대칭단에서의 연산(선형, 비선형 변환을 위한 연산)의 설계는 다음과 같다.

- 구조: Feistel 구조로 암호와 복호가 같은 알고리즘을 구성할 수 있는 구조다. SPN 구조의 대칭단은 암호와 이에 대응되는 복호용 대칭단을 각각 설계해야 하고, 이는 대칭단의 목표와 일치하지 않는다.

- 연산: 선형 변환의 대표적인 효과인 확산은 Feistel 구조적 특징에 의해 자연적으로 발생한다. 다음으로 비선형 변환을 위해 S박스를 사용할 수 있다. 그러나 모든 블록 암호 알고리즘이 S박스 사용으로 비선형 변환을 하지 않는다. 이는 S박스를 사용하지 않는 암호 알고리즘에 대칭단만을 위한 S박스를 만들어 사용하다는 것은 합리적이지 못하고, S박스 치환연산 자체가 암호와 복호용을 각각 설계해야 하므로 대칭단이 암호와 복호용으로 나누어진다. 이

와 같은 이유로 S박스가 아닌 다른 비선형 변환을 하는 연산이 필요하다. 대표적인 비선형 변환을 하는 연산이 AND와 OR 같은 논리 연산이다. 이는 SHACAL의 라운드 함수에서도 사용되고 있다. 그리고 이와 같은 논리 연산은 하드웨어 구현도 쉽고, 수행 속도도 빠르다는 장점을 가진다.

위의 설명과 같은 조건을 만족하는 구조나 연산을 수행하는 대칭단을 설계하는 것은 쉽지 않다.

대칭단의 구조와 연산은 Camellia[59]의 $FL/FL^{-1}$함수를 참조했다. Camellia는 일본에서 개발된 E2[60]와 MISTY[61]의 좋은 장점을 조합하여 만든 것으로 Feistel 구조의 128비트 블록으로 128, 192, 256비트의 가변적인 키를 지원하고 키의 크기에 따라 각각 18, 24, 24라운드 수행하는 알고리즘으로 라운드 함수 내부는 SPN 구조이며, 6라운드 마다 $FL/FL^{-1}$함수를 수행하는 특징이 있다. 그리고 유럽의 NESSIE 프로젝트에서 2003년 128비트 블록 암호로 선정되었다. Camellia에서 $FL/FL^{-1}$함수는 대칭단 개념은 아니지만 6라운드 마다 수행하면서 이전 라운드의 진행을 단절시키는 효과를 주고 있다. 이와 같은 이유로 $FL/FL^{-1}$함수를 참조했고, $FL/FL^{-1}$함수의 진행과정은 (그림 3-4)과 같다.

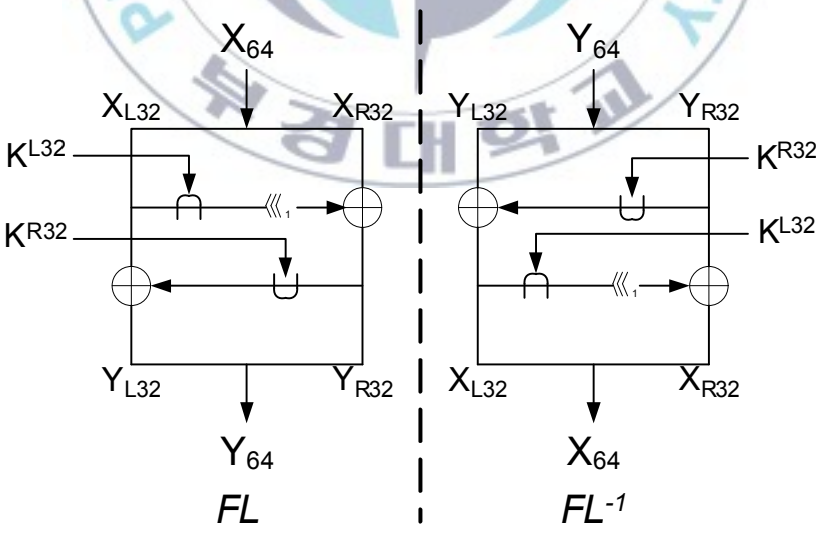



그림 3-4. Camellia의 $FL/FL^{-1}$함수

*FL*과 $FL^{-1}$은 암호와 복호 관계로서 서로 대응되는 역함수 관계이다. *FL*함수의 진행과정은 전체 128비트 블록 중에서 64비트 $X_{64}$를 가지고 다시 32비트 $X_{L32}$, $X_{R32}$로 나누다. 먼저 $X_{L32}$는 라운드 키 $K^{L32}$와 AND 연산을 수행 후 1비트 오른쪽 회전 연산을 수행한다. 그리고 $X_{R32}$와 최종적으로 XOR 연산을 수행 후 출력 32비트 $Y_{R32}$로 보낸다. 나머지 32비트 출력 $Y_{L32}$는 $Y_{R32}$와 두 번째 32비트 라운드 키 $K^{R32}$와 OR 연산을 수행 후 $X_{L32}$와 XOR 연산을 수행한 32비트 값이 된다. $FL^{-1}$은 (그림 3-4)에서와 같이 *FL*함수와 역함수 관계이므로 구조가 서로 대칭적이다.

*FL*함수가 대칭단의 구조와 연산에 합당한 측면도 있지만 *FL*함수의 안전성에 대한 문제점도 있다. (그림 3-5)는 *FL*함수의 낮은 확산 효과(small avalanche effect)를 보이고 있다.

(그림 3-5)의 1번은 왼쪽 32비트 입력이 0이고, 오른쪽 32비트 값은 알 수 없는 임의의 $X_R$일 때 출력으로 왼쪽 32비트는 $X_R \mid K^2$가 된다. 그리고 오른쪽 32비트는 입력 $X_R$이 그대로 출력으로 나간다. 2번의 경우는 1번의 경우와 입력이 반대의 경우이다. 3번의 경우는 64비트 입력이 모두 0일 경우로 두 번째 라운드 키인 $K^2$와 0이 출력으로 된다. 만일 대칭단의 입력을 제어할 수 있다면 2와 3번의 적용으로 64비트 라운드 키를 모두 알아 낼 수 있다. 물론 어떤 암호 알고리즘이던지 암호 진행 중간의 어떤 라운드 값을 추적하거나, 특정한 공격자가 원하는 값을 만들기 위한 입력을 알아내기는 어렵다. 그렇지만 이와 같은 낮은 확산 효과는 분명히 안전성에 영향을 주는 취약점이다.

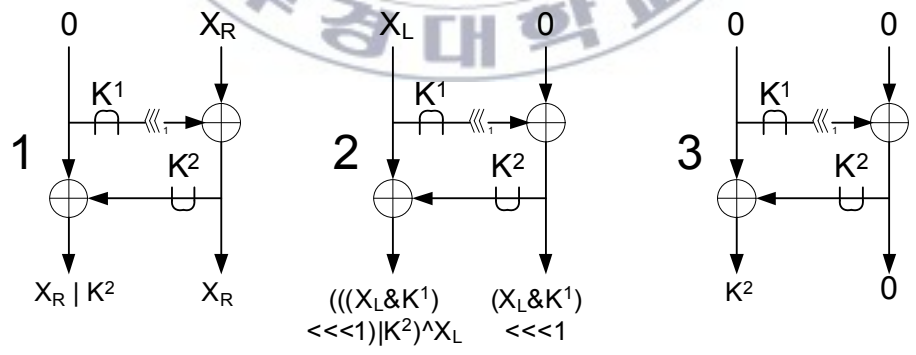



그림 3-5. *FL*함수의 낮은 확산 효과

*FL*함수의 낮은 확산 효과는 32비트 라운드 키 전체를 하나의 논리 연산만

수행해서 발생하는 것으로 32비트 라운드 키를 바이트 단위로 나누어 AND와 OR 연산을 교번 수행한 후 1비트 회전 연산보다 많은 12, 29비트 회전 연산으로 낮은 확산 효과의 취약점을 어느 정도 해결할 수 있다. 좀 더 명확한 개선 방법은 암호 알고리즘의 라운드 함수처럼 여러 번 반복 수행하면 되지만 대칭단은 암호와 복호가 일대일 연속해서 대응되어 서로 상쇄 효과를 없애주면 되고, 실행 효율적인 면을 고려하여 한번만 수행하면 충분하다. 그래서 이와 같은 개선된 대칭단 알고리즘은 다음과 같다.

$$Y_L = ( b(X_R, K^2) \lll 29) \oplus X_L,$$
$$Y_R = ( b(X_L, K^1) \lll 12) \oplus X_R,$$

$$b(X, Y) = (x_3 \,\&\, y_3) \parallel (x_2 \mid y_2) \parallel (x_1 \,\&\, y_1) \parallel (x_0 \mid y_0),$$

여기서 b() 함수의 입력 X, Y는 다음과 같다.

$$X = x_3 \parallel x_2 \parallel x_1 \parallel x_0,$$
$$Y = y_3 \parallel y_2 \parallel y_1 \parallel y_0,$$

$X_{64}$, $Y_{64}$는 각각 입력과 출력 64비트로 입력 32비트로 $X_L$, $X_R$로 나누어지고, 출력 역시 32비트로 $Y_L$, $Y_R$로 나누어진다. $K^1$과 $K^2$는 32비트 라운드 키이다. 그리고 b()함수의 32비트 입력 X, Y는 바이트 단위로 각각 $x_3$, $x_2$, $x_1$, $x_0$, $y_3$, $y_2$, $y_1$, $y_0$으로 나눈다. 그리고 바이트 단위 순서대로 AND, OR, AND, OR 연산을 수행한 후 모두 연결하여 32비트로 되돌린다.

## 3.5 128비트 블록의 대칭단

현대 블록 암호는 보통 128비트 이상의 블록 크기를 요구한다. 이는 128비트보다 작은 경우 전수 조사와 같은 가장 단순한 공격이 가까운 미래에 가능해질 수 있기 때문이다. 이 절에서는 64비트 *FL*함수를 개선한 128비트로 확장한 실질적인 대칭단을 소개한다.

64비트에서 128비트로 확장은 단순히 64비트를 두 번 적용했다. 대신에 출력 결과를 32비트 단위로 서로 자리 교환(Permutation)을 통해 암호와 복호를 같게 구성했다. Camellia의 *FL*과 대응되는 $FL^{-1}$처럼 64비트 단위로 나누어

구성할 수도 있지만, 3.2절에서 설명한 것과 같이 암호와 복호 알고리즘은 안전성이 같으므로 한 가지로만 구성해도 안전성에는 문제가 되지 않는다. 그리고 대칭단에서 사용되는 라운드 키의 크기는 대칭단과 같거나 커야 한다. 3.4절에서 설명한 작은 확산 효과의 원인 중 하나가 대칭단은 한 번 수행한다는 것이고, 이에 대한 대안으로 대칭단에서 사용된 키에 대한 의존성을 높여 최대의 확산 효과를 줄 수 있기 때문이다. 여기서 의존성에 대한 의미는 많은 라운드 키의 사용을 통해 낮은 확산의 취약점을 보완하는 것으로 논리 연산 뿐만 아니라 회전 연산에도 라운드 키를 사용해서 대칭단의 안전성을 향상시킬 수 있다. 그러나 고정된 값(12, 29)에 의한 회전 연산에서 라운드 키에 의한 가변 회전 연산은 하드웨어 구현 시 복잡해지고, 수행 시간 등의 이유로 128비트 대칭단에서는 고정된 값에 의한 회전 연산을 적용했다. 다음은 128비트 대칭단의 알고리즘이다.

$$C` = ( b(A, K^0) \lll 12) \oplus B,$$
$$D` = ( b(C`, K^3) \lll 29) \oplus A,$$
$$A` = ( b(C, K^2) \lll 12) \oplus D,$$
$$B` = ( b(A`, K^1) \lll 29) \oplus C,$$

대칭단의 128비트 입력은 32비트 단위로 A, B, C, D이고 이에 대응되는 출력은 각각 A`, B`, C`, D`이다. 그리고 대칭단에 적용된 라운드 키는 128비트로 32비트 단위로 나누어 각각 $K^0$, $K^1$, $K^2$, $K^3$이고, b()함수는 앞 절에서 소개한 것과 같다.

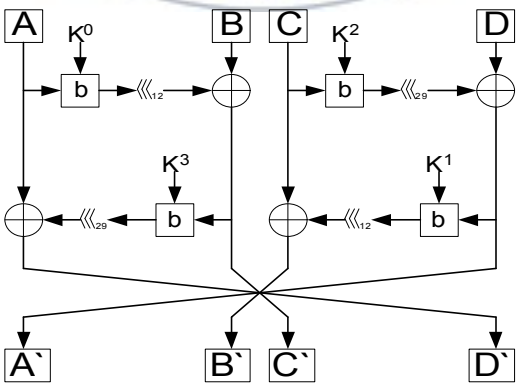



그림 3-6. 128비트 대칭단 구조

128비트 대칭단 진행과정을 그림으로 표현한 것이 (그림 3-6)이다. 128비트 블록을 먼저 64비트 블록으로 나누고, 다시 64비트 블록은 32비트 A, B로 나누어 진행한다. A와 32비트 첫 번째 라운드 키 $K^0$을 b()함수 입력으로 한다. b()함수는 바이트 단위로 논리 연산을 수행한 후 32비트 결과값을 만들고, 그 결과 값에 12비트 오른쪽 회전 연산을 수행 후 32비트 B와 XOR 연산을 수행 후 32비트 출력 C`로 보낸다. 그리고 C`와 네 번째 32비트 라운드 키 $K^3$을 b()함수 입력으로 보낸다. b()함수 수행 후 결과값은 29비트 오른쪽 회전 연산을 수행하고, 이번에는 32비트 A와 XOR 연산을 수행 후 32비트 출력 D`로 보낸다. 이와 유사한 방법으로 32비트 C, D 그리고 $K^2$, $K^1$을 사용하여 32비트 출력 B`, A`를 생성한다. (그림 3-6)에서 보는 것과 같이 구조적으로 진행되는 과정이 Feistel 구조와 비슷하며, 차이점은 64비트 블록 단위로 모두 업 데이터 한 후 서로 자리 교환이 일어난다는 것이다.

## 3.6 160비트 블록의 대칭단

160비트 블록 암호 알고리즘은 현재 많이 사용하고 있는 블록 크기는 아니며, 대칭단이 다양한 블록 크기에 맞게 변형이 가능함을 보이기 위해 적용한 것으로 전체 160비트 블록을 64, 32, 64비트 단위의 세부분으로 나누어 진행하고 사용된 라운드 키 역시 160비트이다. 진행과정은 64비트 단위 블록을 업 데이터 후 서로 교환하고, 32비트 블록은 라운드 키와 XOR 연산 수행 후 자리 교환 없이 그대로 출력으로 보낸다. 알고리즘은 다음과 같다.

$$D` = ( b(A, K^0) \lll 12) \oplus B,$$
$$E` = ( b(D`, K^3) \lll 29) \oplus A,$$
$$A` = ( b(D, K^1) \lll 12) \oplus E,$$
$$B` = ( b(A`, K^2) \lll 29) \oplus D,$$
$$C` = C \oplus K^4,$$

대칭단의 160비트 입력은 32비트 단위로 A, B, C, D, E이고 이에 대응되는 출력은 각각 A`, B`, C`, D`, E`이다. 그리고 대칭단에 적용된 라운드 키는 160비트로 32비트 단위로 나누어 각각 $K^0$, $K^1$, $K^2$, $K^3$, $K^4$이고, b()함수는 앞 절에서 소개한 것과 같다. 160비트 대칭단 진행과정을 그림으로 표현한 것이 (그림 3-7)이다.

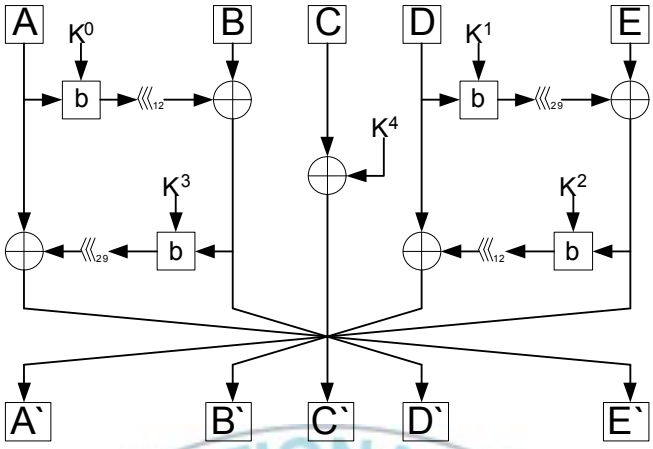



그림 3-7. 160비트 대칭단 구조

## 3.7 256비트 블록의 대칭단

256비트 대칭단은 128비트 대칭단을 연속해서 적용하여 확장했다. 진행과정은 256비트를 128, 128비트로 나누고, 나누어진 128비트는 64, 64비트로 나누어 64비트 블록 단위로 진행한다. 사용된 라운드 키는 256비트이다. 알고리즘은 다음과 같다.

$G` = ( b(A, K^0) \lll 12) \oplus B,$
$H` = ( b(G`, K^7) \lll 29) \oplus A,$
$E` = ( b(C, K^1) \lll 12) \oplus D,$
$F` = ( b(E`, K^6) \lll 29) \oplus C,$
$C` = ( b(E, K^2) \lll 12) \oplus F,$
$D` = ( b(C`, K^5) \lll 29) \oplus E,$
$A` = ( b(G, K^3) \lll 12) \oplus H,$
$B` = ( b(A`, K^4) \lll 29) \oplus G,$

대칭단의 256비트 입력은 32비트 단위로 A, B, C, D, E, F, G, H이고 이에 대응되는 출력은 각각 A`, B`, C`, D`, E`, F`, G`, H`이다. 그리고 32비트 단위 라운드 키는 각각 $K^0$, $K^1$, $K^2$, $K^3$, $K^4$, $K^5$, $K^6$, $K^7$이다. 256비트 대칭단 진행과정을 그림으로 표현한 것이 (그림 3-8)이다.

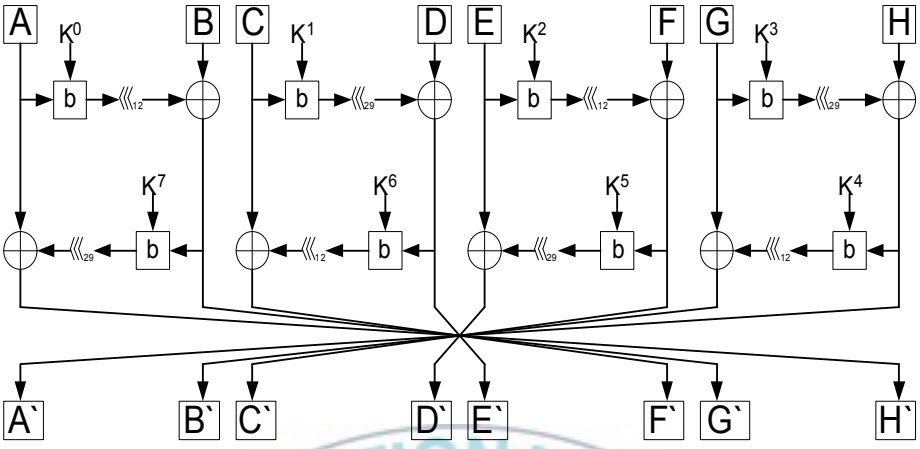



그림 3-8. 256비트 대칭단 구조

64비트 A, B 블록은 업 데이터 후 G`, H`로, C, D 블록은 E`, F`로, E, F블록은 C`, D`로 G, H 블록은 A`, B`로 서로 자리 교환을 한다. 이와 같은 자리 교환은 적용된 라운드 키의 순서에 따라 (그림 3-8)과 다른 다양한 변형이 가능하다.

## 3.8 대칭단의 안전성 분석

이 절에서는 대칭단의 안전성 분석으로 대칭단 연산에서 비선형 변환 연산인 라운드 키와 논리 연산(AND, OR)[62]의 안전성을 분석한다. 분석에 앞서 논리 연산의 특성에 대해 먼저 설명한다. 임의의 32비트 입력을 X, 이에 대응되는 출력은 Y, 라운드 키는 $K^r$로 표시하고, X[i]는 X의 i 번째 비트를 나타낸다. 출력 Y와 $K^r$ 역시 X와 같은 방법으로 표현한다. 그리고 입력에 대한 $\Delta X = X \oplus X`$의 결과이고, 대응되는 출력에 대한 $\Delta Y = Y \oplus Y`$이다.

X와 라운드 키 $K^r$의 AND와 OR 연산의 차분 분석에 어떤 영향을 미치는지 분석하면, $K^r[i]$의 값은 0 또는 1의 값을 가진다. 입력 차분 $\Delta X[i] = 1$일 경우 $K^r[i]$가 0으로 AND 연산을 수행하면 출력 차분 $\Delta Y[i] = 0$ 되며 차분 분석에 영향을 미친다. 또한 $\Delta X[i] = 1$일 경우 $K^r[i]$가 1 로 AND 연산을 수행하면 출력 차분 $\Delta Y[i] = 1$이 되며 입력과 출력이 같아지므로 차분 분석에는 영향을 미치지 않는다. 같은 방법으로 OR 연산을 분석하면, 입력 차분 $\Delta X[i] = 1$ 일 경우 $K^r[i]$가 0으로 OR연산을 수행하면 출력 차분 $\Delta Y[i] = 1$이 되며 차분 분

석에 영향을 미치지 않으며, 입력 차분 ΔX[i] = 1일 경우 $K^r[i]$가 1로 OR 연산을 수행하면 출력 차분 ΔY[i] = 0이 되며 차분 분석에 영향을 미친다.

표 3-1. AND와 OR의 차분 분석

| AND | $K^r[i]$ | ΔX[i], ΔY[i] | 영 향 |
|---|---|---|---|
| | 0 | ΔX[i] = 1 → ΔY[i] = 0 | yes |
| | 1 | ΔX[i] = 1 → ΔY[i] = 1 | no |
| OR | $K^r[i]$ | ΔX[i], ΔY[i] | 영 향 |
| | 0 | ΔX[i] = 1 → ΔY[i] = 1 | no |
| | 1 | ΔX[i] = 1 → ΔY[i] = 0 | yes |

라운드 키와 AND와 OR 연산이 선형 분석에 어떤 영향을 미치는지 분석하면, $K^r[i]$의 값이 0일 경우 AND연산을 수행하면 X[i]의 값과는 상관없이 Y[i] = 0 이 되어 선형 분석에 영향을 미치고, $K^r[i]$의 값이 1일 경우 Y[i] = X[i]가 되어 선형 분석에는 영향을 미치지 않는다. 같은 방법으로 OR 연산을 분석하면, $K^r[i]$의 값이 0일 경우 OR 연산을 수행하면 Y[i] = X[i]가 되어 선형 분석에는 영향을 미치지 않는다. $K^r[i]$의 값이 1일 경우 X[i]의 값과는 상관없이 Y[i] = 1이 되어 선형 분석에 영향을 미친다.

표 3-2. AND와 OR의 선형 분석

| AND | $K^r[i]$ | X[i], Y[i] 의 값 | 영 향 |
|---|---|---|---|
| | 0 | X[i]와 독립으로 Y[i]=0 | yes |
| | 1 | Y[i] = X[i] | no |
| OR | $K^r[i]$ | X[i], Y[i] 의 값 | 영 향 |
| | 0 | Y[i] = X[i] | no |
| | 1 | X[i]와 독립으로 Y[i]=1 | yes |

대칭단에 적용된 라운드 키 $K^r[i]$는 0 아니면 1 이므로 1/2의 확률을 가진다. 앞에서 설명한 바와 같이 입력 차분 ΔX는 대응되는 라운드 키 $K^r[i]$와 논리연산을 통해 출력 차분 ΔY가 예측할 수 없는 확산이 일어난다. 특히 ΔX의

Hamming weight가 h일 경우 ΔY = 0일 때 적용된 라운드 키의 확률은 $2^{-h}$ 이다. 예를 들어 ΔX가 n비트 길이를 가지고, ΔX ≠ 0 이며, ΔY = 0 일 때 적용된 라운드 키를 얻을 수 있는 확률은 $\frac{1}{2^n}\sum_{h=1}^{n} nC_h \times 2^{-h}$이다.

이와 같은 확률은 대칭단 내의 b()함수가 바이트 단위로 나누어 연산을 수행하고 이를 확률에 적용해 보면 약 0.1 = $2^{-3}$이고 각각의 바이트는 서로 독립이므로 약 $(2^{-3})^4 = 2^{-12}$이 된다. 최종적으로 128, 160비트 대칭단 논리 연산에 대한 분석 확률은 $2^{-48}$이고, 256비트는 $2^{-96}$이 된다.

마지막으로 대칭단에서 사용된 고정된(12, 29비트) 회전 연산은 암호의 안전성에 아무런 영향을 미치지 않고 단지 확산 효과만 있다. 이유는 회전 연산이 가변적이지 않고 고정되어 있기 때문에 100% 확률로 변화를 추적할 수 있기 때문이다. 대칭단이 Feistel 구조이기 때문에 확산 효과는 자동으로 발생하지만 고정된 회전 연산이 안전성에는 영향이 없지만 확산 효과를 더욱 강화시키는 역할을 한다. 그리고 대칭단의 구성을 간결하게 한 이유는 소프트웨어 및 하드웨어의 쉬운 구현과 수행속도의 향상 뿐만 아니라 안전성을 검증 가능한 수학식으로 표현하기 위함이다. AES 프로젝트에서 5개의 최종 후보 알고리즘 중 MARS[63]는 비대칭 Feistel 구조로 내부 라운드 함수의 구조가 너무 복잡하여 안전성 분석이 어려워 최종적으로 선택되지 않았다. 이는 암호 알고리즘의 안전성은 수학적으로 증명 가능한 안전성만 인증 받을 수 있다.

## 3.9 결론

암호와 복호가 다른 R라운드 진행의 블록 암호 알고리즘을 대칭단을 적용한 구조로 구성하여 암호와 복호를 같게 구성할 수 있다. 즉 1 ~ ((R / 2) - 1)라운드는 암호 알고리즘을, 그리고 ((R / 2) + 1) ~ R라운드는 복호 알고리즘을 적용하고 암호와 복호 중간에 대칭단을 적용하면 암호와 복호가 같게 구성된다. 대칭단은 간단한 논리 연산으로 구성되어 하드웨어 및 소프트웨어 구현이 쉽고, 수행 속도도 빠르며, 안전성 또한 대칭단을 통해 암호와 복호가 일대일 대응으로 상쇄 효과를 없애고, 암호 라운드와 복호 라운드의 단절 효과를 주고 있어 암호 알고리즘의 안전성 향상에 도움을 준다. 그리고 대칭단은 다양한 블록 크기에 맞게 쉽게 변형가능하며, 이는 암호와 복호가 다른 모든 블록 암호에 적용 가능하다.

# 제 4장 대칭단을 적용한 128비트 블록 암호 알고리즘

이 장에서는 암호와 복호가 서로 다른 128비트 블록 암호 알고리즘인 AES, RC6에 대칭단을 적용하여 암호와 복호를 같게 구성한 새로운 AES와 RC6을 소개하고, 새롭게 구성된 알고리즘들의 안전성을 중점적으로 설명하고, 실행 결과 및 결론 순으로 설명한다.

## 4.1 128비트 대칭단을 적용한 AES

NIST에 의해 AES 프로젝트에서 선전된 SPN 구조의 AES 암호 알고리즘은 많은 개발자들에 의해 여러 가지 다양한 방법으로 분석되었다. (표 4-1)은 기존의 AES와 같은 분석 방법을 대칭단을 적용한 AES 알고리즘에도 적용한 비교 결과표이다.

표 4-1. Square, 불능 차분, Boomerang 공격 비교

| 암호 알고리즘 | 공격 방법 | 라운드 | 복잡도 |
|---|---|---|---|
| AES | Square 공격[24] | 6 | $2^{72}$ |
| | 불능 차분 공격[64] | 6 | $2^{122}$ |
| | Boomerang 공격[65] | 6 | $2^{71}$ |
| 대칭단을 적용한 AES | Square 공격 | 6 | $2^{120}$ |
| | 불능 차분 공격 | 6 | $2^{186}$ |
| | Boomerang 공격 | 6 | $2^{83}$ |

AES 개발자들에 의해 소개된 Square 공격은 전체 16바이트 중에서 15바이트는 고정된 값을 가지고 나머지 1바이트는 모든 가능한 값(0 ~ 255)을 가지는 선택 평문 집합으로 라운드 함수에서 선형 변환을 통해 모든 바이트에 확산이 진행된 결과로 모든 선택 평문에 대해 XOR 연산을 수행하여 상쇄

(balanced)시킨 후 사용된 라운드 키를 찾는 공격 방법이다.

불능 차분 공격(Impossible Differentials Attack)[64]은 차분 확률이 0에 수렴하거나 차분이 존재하지 않는 경우, 평문쌍에 이와 같은 차분을 가지는 라운드 키를 제외한 후 남은 라운드 키를 가지고 틀린 키를 제외해 나가는 분석 방법으로 가능한 후보 라운드 키에 한정해서 적용하는 방법이다.

Boomerang 공격[65]은 SPN 구조의 암호 알고리즘에는 유용한 공격 방법으로 대칭단 개념과 비슷하며 암호와 복호 알고리즘을 서로 반대편에서 진행하여 특정 라운드에서 겹쳐질 때, 이 경우 확률적으로 라운드 키를 찾는 방법이다. AES 10라운드에 적용한 예는 없지만 AES 6라운드에 적용하여 전수 조사보다는 매우 효율적이며, Square 공격보다 두 배 더 빠른 결과를 보여주고 있다. (표 4-1)의 대칭단을 적용한 AES의 안정성에 관한 결과는 4.1.3절에서 자세히 설명한다.

### 4.1.1 구현

대칭단 구조를 AES 알고리즘에 적용할 때 SubBytes(), ShiftRows(), MixColumns(), AddRoundKey() 함수를 변경 없이 그대로 사용한다. 그러나 전체 라운드에서 1/2은 암호 알고리즘을 사용하고 나머지 1/2은 복호 알고리즘을 사용해 암호에서 복호로 바뀌는 중간에 대칭단 연산이 삽입된다.

AES의 암호 알고리즘 구성은 모든 라운드에 동일한 연산이 반복되면 수학식이 단순해져서 선형 공격에 취약하다. 그래서 1라운드 시작 전에 AddRoundKey() 함수를 수행하고 마지막 라운드에서는 MixColumns() 함수를 제외하고 있다. 대칭단을 적용한 AES에서는 기존의 AES와는 다르게 전체 라운드에서 중간에 불규칙적인 라운드인 대칭단을 삽입해서 수학식이 단순해지는 점을 극복하므로 AES와 같은 1라운드 시작 전에 AddRoundKey() 함수의 적용과 마지막 라운드에서 MixColumns() 함수를 제외하지 않고, 첫 번째 AddRoundKey() 함수를 1라운드 함수에 포함시켜 라운드를 진행한다. 그리고 마지막 라운드에서 MixColumns() 함수를 제외하는 것 역시 복호 알고리즘 마지막 라운드에서 InvMixColumns() 함수를 포함시켜 진행한다.

(그림 4-1)에서 보는 바와 같이 전체 라운드의 1/2은 AES 암호 알고리즘을 수행하는 부분으로 AES의 AddRoundKey(), SubBytes(), ShiftRows(), MixColumns() 함수 순서로 1 ~ 5라운드까지 적용한다. AES 알고리즘과의 차이는 없고, 단지 초기 키 덧셈 부분이 제안한 알고리즘에서는 라운드 함수에 포

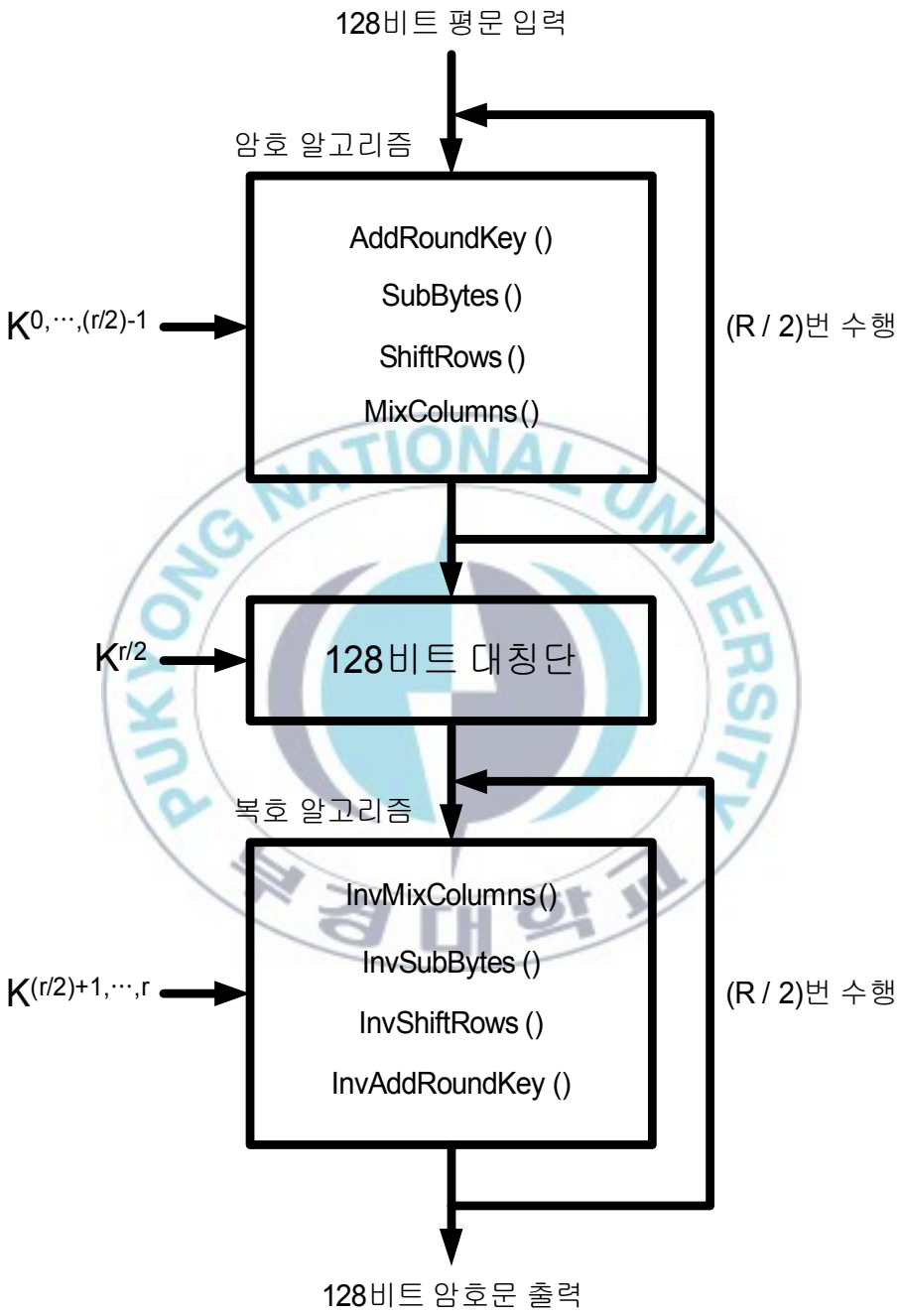



그림 4-1. 128비트 대칭단을 적용한 AES

함되어 진행하면서 5 라운드 이후 사용되는 라운드 키가 대칭단에서 사용되는 키로 된다는 차이가 있다. 이와 같이 1 ~ 5라운드까지 반복 수행 후 대칭단을 수행한다(그림 3-6 참조). 대칭단을 수행 후 AES 복호 알고리즘을 수행하고 암호 알고리즘에서 사용된 4개의 복호용 역 알고리즘을 사용한다. AES 복호 알고리즘의 적용 순서는 InvMixColumns(), InvSubBytes(), InvShiftRows(), InvAddRoundKey() 함수 순으로 적용하고 6 ~ 10라운드를 반복 수행하여 최종적으로 암호문을 만든다.

복호과정은 (그림 4-1)의 순서를 그대로 적용하고, 단지 라운드 키의 입력이 암호화 과정의 역순으로 입력하면 된다. 대칭단을 적용한 AES 암호 알고리즘의 키 스케줄링은 기존의 AES의 키 스케줄링을 그대로 사용한다.

### 4.1.2 AES의 부분 상쇄

대칭단을 적용하지 않고 암호와 복호를 연속해서 적용할 경우 복호 6라운드에서 부분적인 상쇄 현상이 발생한다. (그림 4-1)에서 중간의 대칭단이 없고 암호 5라운드 후 복호 6라운드가 바로 실행된다고 가정하면, 5라운드 키와 덧셈인 AddRoundKey() 수행 후 순차적으로 SubBytes(), ShiftRows(), MixColumns() 연산을 수행하여 5라운드 암호를 마무리한다. 그리고 그 5라운드 암호의 결과 값으로 6라운드 복호는 InvMixColumns(), InvSubBytes(), InvShiftRows(), InvAddRoundKey() 순으로 라운드 함수가 적용된다. 이때 5라운드의 MixColumns()과 6라운드의 InvMixColumns()이 연속으로 적용되면서 서로 완전한 상쇄로 5라운드 ShiftRows() 함수 적용 후의 값이 된다. 그리고 연속해서 InvSubBytes(), InvShiftRows()의 적용으로 SubBytes(), ShiftRows() 함수 역시 완전한 상쇄가 발생하여 5라운드 AddRoundKey() 함수의 적용값 만이 남아 있게 된다. 6라운드에서 InvSubBytes()와 InvShiftRows()의 적용 순서는 바이트 단위 연산이므로 결과값에는 아무런 영향이 없다. 마지막으로 6라운드의 InvAddRoundKey() 함수가 적용되며 이는 5라운드 키와 6라운드 키가 서로 독립적인 키이므로 5라운드와 6라운드 키를 단순히 XOR 연산만을 수행하게 된다.

이와 같이 대칭단 없이 연속적인 암호와 복호의 적용은 암호에서 복호 알고리즘으로 바뀔 때 암호 마지막 라운드와 복호 첫 라운드가 서로 완전한 상쇄로 각각의 라운드에서 적용된 라운드 키만이 남게 되고 이는 암호의 안전성에 심각한 취약점이 된다.

### 4.1.3 안전성 분석

대칭단을 적용한 AES 알고리즘은 기존의 AES 알고리즘과 똑같은 SPN 구조를 이루고 있으며, 각 라운드의 확산특성이 같다. (그림 4-2)는 AES 알고리즘의 확산 과정을 그림으로 설명한 것이다. 1라운드 초기 상태에서 하나의 바이트만을 공격하고 다른 바이트들은 공격하지 않는 경우를 상정했다. (그림 4-2)에서 변화가 발생하는 바이트를 'X'로 표시했다.

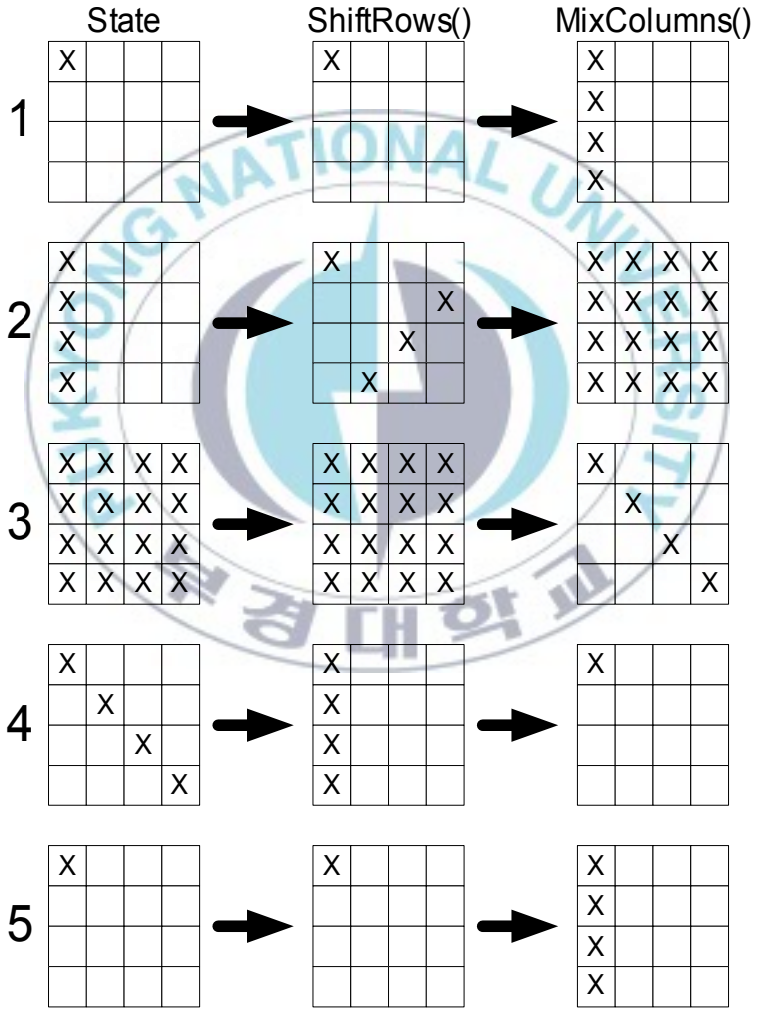



그림 4-2. AES의 5라운드 확산 진행과정

1라운드에서는 1개의 S박스가 활성화(Active)되고 MixColumns() 함수를 수행하면 하나의 열이 변화한다. 이는 AES의 선형 변환 함수 MixColumns()의 Branch number가 5인 MDS(Maximum Distance Separated)행렬을 이루기 때문이다. 2라운드에서는 4개의 S박스가 활성화되고 확산과정을 수행한 후 모든 바이트가 변화한다. 3라운드에서는 16개의 S박스가 활성화되고 각 열에서 1바이트만이 변화한다. 4라운드에서는 4개의 S박스가 활성화되고 초기 상태와 같이 하나의 바이트만이 변화한다.

이러한 과정이 공격에 가장 취약한 확산과정이다. 따라서 (그림 4-2)와 같은 확산과정은 기존의 AES 알고리즘에서 가장 적은 활성화된 S박스를 구할 수 있는 것으로, 이를 그대로 적용하여 대칭단을 적용한 AES 알고리즘의 안전성을 나타내는 기준이 된다. 그래서 대칭단을 적용한 AES 알고리즘은 (그림 4-2)에서 설명한 과정을 거친 후에 불규칙 라운드인 대칭단을 수행하고 이어서 AES의 복호과정을 수행 한다. 복호 알고리즘의 확산 진행 과정도 (그림 4-2)의 진행과 유사하다.

SPN 구조 암호 알고리즘의 안전성 평가는 Hong[66]등에 의해서 제안된 바 있다. 차분과 선형 공격은 차분과 선형 활동성을 가지는 S박스의 수를 구하므로 해서 SPN 구조 암호 알고리즘의 안전성을 구할 수 있다. AES S박스의 최대 차분과 선형 특성 및 근사 확률은 각각 $2^{-6}$이다. 그리고 AES의 경우 암호용 S박스와 복호용 역 S박스는 일대일 대응되는 전단사 함수로 같은 차분과 선형 확률 값을 가진다. 그리고 암호용 MixColumns() 함수 연산과 복호용 InvMixColumns() 함수 연산 역시 일대일 대응되는 전단사 함수다.

(그림 4-2)의 확산과정 설명에서 AES 알고리즘의 전체 10라운드의 확산과정을 진행해보면 활동성을 가지는 S박스의 수는 총 55개이므로 AES의 안전성은 $(2^{-6})^{55} = 2^{-330}$이라고 할 수 있다. AES 알고리즘의 암호와 복호가 일대일 대응으로 생기는 라운드간의 상쇄효과를 단절시키기 위해 불규칙 라운드인 대칭단을 삽입했고, 대칭단 내의 안전성 분석은 3.8절에서 설명 되어 있고, 크게 라운드 키와 AND와 OR 연산 수행과 대칭단 수행 후 워드 단위의 자리바꿈인 확산으로 나누어 볼 수 있다.

대칭단 수행 후 워드 단위 자리바꿈의 영향을 살펴보면 (그림 4-3)의 상위 3개의 4 * 4 행렬은 대칭단을 적용한 AES 알고리즘의 대칭단을 통과하면서 자리바꿈이 일어난 후 6라운드 확산 진행을 보여 주는 것이고, 하위 2개의 4 * 4 행렬은 기존의 AES의 정상적인 확산 진행 상황이다. 이는 (그림 4-3)에서 보는 봐와 같이 활성화된 S박스의 수는 한 개로 줄었지만 4 * 4 행렬의 바이트

순서는 추적할 수 없을 정도로 자리바꿈이 일어난 것을 볼 수 있다. 이후의 확산 진행 과정은 (그림 4-2)와 유사하며, 총 활성화된 S박스의 개수는 (표 4-2)와 같다.

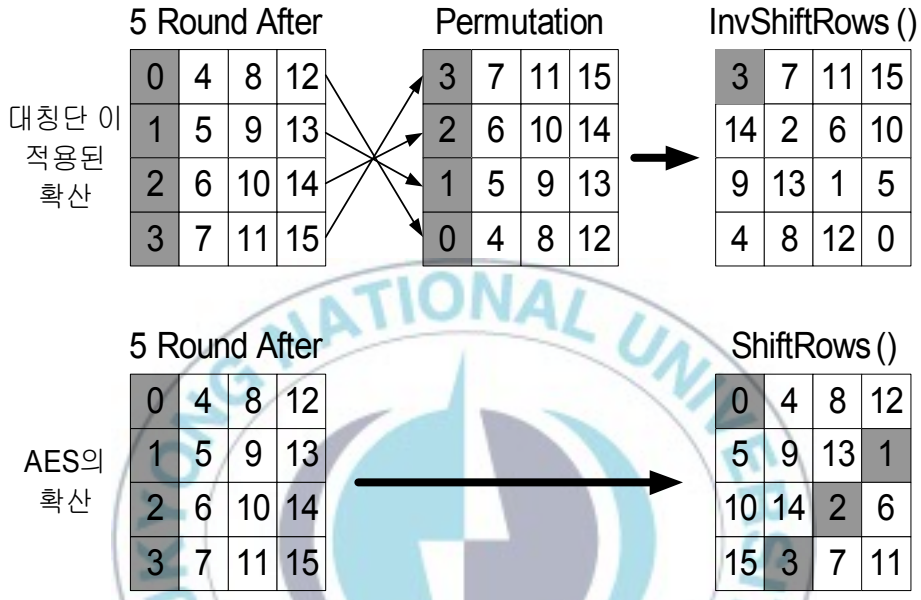



그림 4-3. 기존의 AES와 대칭단을 적용한 AES의 확산 진행과정 비교

표 4-2. 활성화 된 S박스의 개수 비교

| 라운드 / 알고리즘 | 1 | 2 | 3 | 4 | 5 | SL | 6 | 7 | 8 | 9 | 10 | 총계 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AES | 1 | 4 | 16 | 4 | 1 | - | 4 | 16 | 4 | 1 | 4 | 55 |
| 대칭단을 적용한 AES | 1 | 4 | 16 | 4 | 1 | - | 1 | 4 | 16 | 4 | 1 | 52 |

(표 4-2)에서는 대칭단의 확산과정을 정확하게 추론할 수 없으므로 최악의 경우를 상정하여 6라운드에서 하나의 S박스만이 활성화되는 것으로 가정했다. 이 경우에 52개의 S박스가 활성화됨으로 AES의 55개 S박스와 비교해서 유사한 안정성을 가지는 것으로 평가된다. 그러나 대칭단을 적용한 AES의 활성

화된 S박스의 가정은 가장 보수적인 가정으로 대칭단의 확산과정에 관한 연구는 추후 진행되어야 한다.

기존의 AES에 대한 Square 공격은 기본 4라운드 공격에서 시작 라운드 전과 마지막 4라운드 끝에 각각 추가적인 라운드를 적용하여 6라운드 Square 공격은 (표 4-1)과 같다. 대칭단을 적용한 AES의 6라운드 Square 공격에서는 5라운드와 6라운드 사이에 대칭단이 적용되며, 이는 대칭단 내의 연산중에서 고정된 회전 연산(12, 29비트)이 바이트 단위의 연산이 아니므로 바이트 단위의 확산 특성을 연결시키지 못하고, 특히 대칭단에서 사용되는 라운드 키와 논리연산(AND, OR)은 5라운드에서의 상쇄된(balanced)모든 바이트에 적용되므로 최종적인 6라운드 Square 공격 복잡도는 약 $2^{72} * 2^{48} = 2^{120}$정도 된다. 이와 같은 이유로 바이트 패턴이 각각의 라운드 사이에서 전파되는 특성을 이용한 Square 공격은 대칭단을 적용한 AES 알고리즘에서는 전체 라운드 중간에 있는 대칭단에서 32비트 단위의 12비트, 29비트 왼쪽 회전 연산과 라운드 키 의존적인 논리 연산을 통해 바이트 단절이 일어나 Square공격에도 대칭단 이후 라운드에서는 적용이 어렵다.

불능 차분 공격(Impossible Differentials Attack)은 AES의 전체 16바이트에서 1바이트를 제외한 모든 바이트가 동일한 선택 평문 쌍에 대해서 선형 변환인 Mixcolumns()의 출력 차분이 4바이트 중 1바이트가 0이 아닌 차분인 경우의 수는 $2^8-1$이고 출력 차분이 0이 아닌 바이트의 선택은 4가지가 있기 때문에 4바이트 중에서 1바이트가 0이 아닌 출력 차분을 가질 확률은 약 $(4 * 2^8) / (2^8)^4 = 2^{-22}$이 된다. 이는 AES의 선형 변환이 일대일 대응인 전단사 함수이므로 InvMixcolumns()에서도 동일한 확률이 적용된다. 이 확률은 고정된 바이트에 대해서 입력 차분의 값이 제한적일 때도 성립하고 이 확률값은 불능 차분 공격의 복잡도를 계산하는 과정에서 사용한다.

그리고 AES의 Mixcolumns()은 branch number가 5이기 때문에 4바이트 중 1바이트만 0이 아니 차분을 가지면 다음 라운드의 그 바이트가 속한 열의 모든 바이트가 0이 아닌 차분 값을 가지게 된다. 이후 ShiftRows()에 의해 모든 열에 1바이트씩 0이 아닌 차분 값을 가지게 된다. 그 후 다음 Mixcolumns()에 의해 모든 바이트가 0이 아닌 차분 값을 가지게 되며, 이와 같은 AES의 선형 변환 특성은 전체 16바이트 중 4바이트로 이루어진 금지된 조합 4개의 쌍((1,6,11,16), (2,7,12,13), (3,8,9,14), (4,5,10,15))은 1바이트

를 제외한 모든 바이트가 같은 평문쌍에 의한 4라운드 후의 대응되는 암호문은 금지된 조합 4개의 쌍안에서 절대로 동일한 바이트를 포함할 수 없다. 이러한 AES의 특성을 이용하여 6라운드 불능 차분 공격을 할 수 있으며, 결과는 (표 4-1)과 같다.

불능 차분 공격 역시 Square 공격과 마찬가지로 기본 4라운드 공격에서 라운드 시작 전과 마지막 4라운드 후에 각각 라운드를 추가하여 6라운드 공격을 할 수 있고 마지막 라운드에서는 Mixcolumns()이 제외 된다. 공격 방법은 다음과 같다.

① (1,8,11,14)의 위치에 있는 4바이트를 제외한 모든 바이트가 고정된 값(0)을 갖는 평문 집합을 정의한다. 그러면 4바이트에 대한 $2^{32}$개의 평문과 그에 대한 평문쌍의 개수는 $2^{63}$이 된다.

② 단계 ①에서 정의된 $2^{59.5}$개($2^{91.5}$개의 평문, $2^{122.5}$개의 평문 쌍)를 선택한다. 그리고 각 평문 쌍을 암호화하고, 암호문 쌍 중에서 2와 3행이 0인 차분 값을 갖는 평문 쌍을 고른다. 이때 평문 쌍의 기댓값은 약 $2^{122.5} * 2^{-64} = 2^{58.5}$이다.

③ 마지막 6라운드의 라운드 키는 0과 1행의 64비트 값만 유효한 값으로 가정한다.($K^6_{0,...,63}$)

④ 각각의 6라운드 암호문 쌍($C^6$, $C^{*6}$)에 대해서 5라운드 암호문 쌍을 다음과 같이 구한다.

$$C^5 = \text{InvSubBytes}() \cdot \text{InvShiftRows}(C^6 \oplus K^6),$$
$$C^{*5} = \text{InvSubBytes}() \cdot \text{InvShiftRows}(C^{*6} \oplus K^6),$$

구해진 5라운드 암호문 쌍을 InvMixColumns($C^5$, $C^{*5}$)에 적용하여 금지된 4개의 바이트 조합(1,6,11,16), (2,7,12,13), (3,8,9,14), (4,5,10,15)에서 0이 되는 쌍들을 고른다. 이때 각각의 평문 쌍에 대하여 금지된 4개의 바이트 조합에서 0이 될 확률은 $2^{-32} * 4 = 2^{-30}$이고, 평문 쌍의 개수에 대한 기댓값은 $2^{58.5} * 2^{-30} = 2^{28.5}$이다.

⑤ 단계 ④번에서 구해진 $2^{28.5}$개의 평문 쌍과 첫 번째 라운드 키 $K^1$의 4바이트(1,8,11,14)의 $2^{32}$비트 값으로 다음을 계산한다. ($K^1_{0,...,7 \| 57,...,63 \| 80,...,87 \| 104,...,111}$)

$$\text{MixColumns}() \cdot \text{ShiftRows}(\text{SubBytes}(P \oplus K^1) \oplus \text{SubBytes}(P^* \oplus K^1)),$$

계산 결과에서 1바이트를 제외한 모든 바이트에서 0의 차분 값을 가지는 쌍을 고른다. 이 경우의 확률은 $2^{-24} * 4 = 2^{-22}$이다.

⑥ 단계 ④, ⑤번에서 나타나는 바이트 조합은 절대로 동시에 나타날 수 없기 때문에 이러한 차분 값을 만드는 라운드 키는 모두 잘못된 라운드 키이다. 여기서 우리는 96비트의 라운드 키($K^1_{0,...,7 \| 57,...,63 \| 80,...,87 \| 104,...,111}$, $K^6_{0,...,63}$), $2^{96}$개의 라운드 키 쌍으로 불능 차분 공격을 한다. 암호문 쌍을 분석 후 4바이트 첫 번째 라운드 키 $K^1$이 잘못된 키 값은 약 $2^{32}(1-2^{-22})^{2^{28.5}} \approx 2^{-98.5}$정도 남는다. 이와 같은 $2^{-98.5}$의 의미는 잘못된 라운드 키는 ④번과 ⑤번의 과정을 통과할 가능성이 희박하다는 것이다.

⑦ 단계 ⑥번에서 64비트 라운드 키인 $K^6$이 옳은 키라고 가정 할 때 32비트 라운드 키 $K^1$이 전체 96비트 라운드 키에서 $K^6$의 값에 남아 있을 확률은 $2^{-98.5} / 2^{-64} = 2^{-34.5}$이 된다. 그리고 ③번에서 가정한 64비트 라운드 키를 바꾸어 2와 3행의 64비트 라운드 키로 실행하면, 6라운드의 128비트 라운드 키 전체를 알 수 있다.

⑧ 단계 ④번의 연산과정의 복잡도는 약 $2 * 2^{64} * 2^{58.5} = 2^{123.5}$이 되고, ⑤번의 연산과정의 복잡도는 다음과 같다.

$$2^{64} * 2 * 2^{32}\{1+(1-2^{-22})+(1-2^{-22})^2+...+(1-2^{-22})^{2^{28.5}}\} \approx 2^{119},$$

한번 잘못된 키로 판정된 라운드 키는 더 이상 연산이 필요 없고 이와 같은 연산 과정은 2번 반복하게 되므로 최종적인 AES의 6라운드 불능 차분 공격의 연산량은 약 $2^{122}$정도 된다.

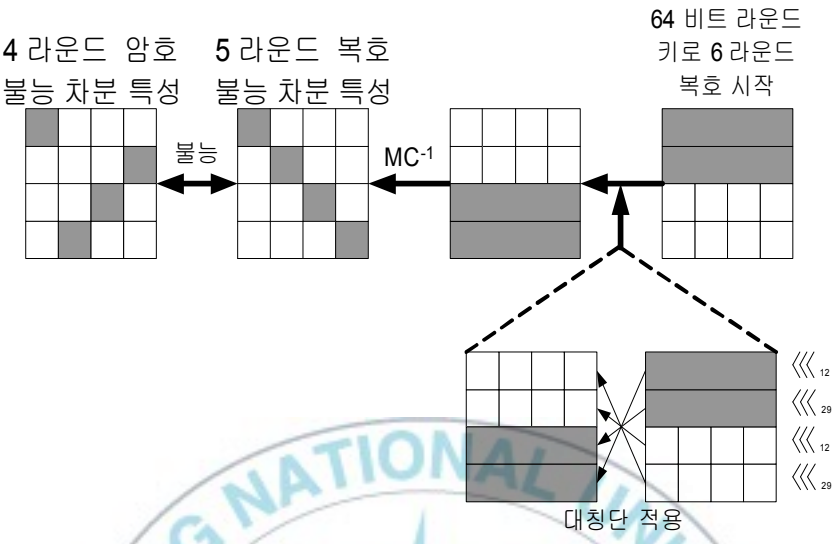



그림 4-4. 대칭단을 적용한 AES 6라운드 불능 차분 공격

(그림 4-4)는 대칭단을 적용한 AES 6라운드 불능 차분 공격을 그림으로 표현한 것으로 5라운드와 6라운드 사이에 대칭단의 삽입으로 6라운드 시작에 적용한 0과 1행의 64비트 라운드 키가 자리바꿈 때문에 2와 3행으로 확산된다. 행의 변화는 열 변환을 하는 MixColumns()에서는 열 단위에서 위치는 중요하지 않기 때문에 branch number에는 아무런 영향이 없다. 다만 대칭단에서 행 단위로 고정된 회전 연산이 열의 위치에 심각한 영향을 준다. 12와 29비트 회전 연산은 바이트 단위 분석으로 각각 1/2과 3/8의 확률을 갖는다. 그리고 대칭단에서 사용된 라운드 키와 논리 연산의 결과도 바이트 단위의 확률이 약 $2^{-3}$ 이므로 최종적인 대칭단이 적용된 6라운드 불능 차분 공격 복잡도는 약 $2^{122} * ((1/2)^4)^2 * ((3/8)^4)^2 * (2^3)^{16} \approx 2^{186}$이 된다. 이와 같은 대칭단의 적용은 전체 불능 차분 분석 복잡도의 급격한 상승뿐만 아니라 기존의 AES의 선형 변환인 MixColumns()의 불능 특성도 바뀌게 한다. 이것은 대칭단이 불능 차분 공격 특성도 대칭단에서 적용된 회전 연산과 라운드 키와 논리 연산에 의해 불능 차분 특성은 전수 조사보다 효율이 떨어지게 되어 대칭단 이후의 불능 차분 공격은 유효하지 않다.

Boomerang 공격은 선택 평문 또는 능동 선택 암호문 공격으로 바이트 단위 연산을 수행하는 SPN 구조의 블록 암호 알고리즘에 5 또는 6라운드 공격이

가능하다. Boomerang 공격은 암호화 과정을 두 부분으로 나누어($E = E_1 \cdot E_0$) 진행한다. 여기서 $E_0$은 첫 라운드부터 시작하는 암호화 과정을 수행하는 부분이고, $E_1$은 마지막 라운드부터 복호화 과정을 수행하는 부분이다. 먼저 기본 5라운드 Boomerang 공격 진행과정은 다음과 같다.

① 열 단위로 4바이트의 모든 가능한 값$(0,..,2^{32}-1)$을 가지며, 나머지 바이트는 임의의 상수값을 가지는 평문 집합 $P_i$를 선택한다. 선택된 평문과 대응되는 $2^{32}$개의  암호문 집합 $C_i$를 얻는다.

② 단계 ①번에서 얻어진 암호문 집합 $C_i$에 0이 아닌 고정된 차분 값(Δ)을 가지는 새로운 $2^{32}$개의 집합 $D_i$를 얻는다.

$$D_i = C_i \oplus \Delta,$$

③ 단계 ②번에서 얻은 $D_i$를 복호화 하여 새로운 $2^{32}$개의 평문 $Q_i$를 얻는다.

④ 단계 ③번에서 얻은 $Q_i$에서 8바이트만 비활성화된 S박스 치환을 하는 것을 골라 정렬한다. 새롭게 정렬된 $Q_i$를 서로 비교하여 8바이트만 차분 값이 0인 쌍$(Q_i, Q_j)$을 선택한다. 만일 조건에 맞는 쌍이 없으면 ①번부터 다시 시작한다.

⑤ 단계 ④번에서 얻은 새로운 평문 쌍과 이에 대응되는 초기 평문 쌍인 $Q_i$, $Q_j$, $P_i$, $P_j$의 4개의 쌍 중에서 4바이트만 0이 아닌 차분 값을 가지는 쌍은 4개의 S박스만 활성화 되고 그 S박스 입력에 대응되는 32비트 라운드 키를 추측 할 수 있다. 추측된 라운드 키로 다시 1라운드 암호화를 수행하면 1개의 S박스만 활성화 된다. $(P_i, P_j)$와 $(Q_i, Q_j)$ 각각의 쌍은 4개의 활성화 S박스에서 1라운드 암호화 후 1개의 활성화 S박스로 될 확률은 각각 $(4 * 2^8) / 2^{32} = 2^{-22}$이다. 이 확률은 22비트의 필터링(filtering) 조건을 의미하며, 두 쌍이므로 44비트 조건에 의해 한 라운드에서 한 개의 S박스가 활성화 될 때 그 라운드 이전과 이후는 4개의 활성화된 S박스가 출현하는 것을 말하며, 이를 부메랑(Boomerang)이라 한다.

이와 같이 한 라운드의 전과 후에서 총 12개의 활성화된 S박스가 있을 경우, $(P_i, P_j)$쌍으로부터 4개, $(Q_i, Q_j)$쌍으로부터 8개의 활성화된 S박스가 겹쳐지며, 최소 2배의 후보 라운드 키를 조사해야 한다. 그리고 4개의 활성화된 S박스가 2개의 활성화된 S박스로 확산 될 확률은 $(6 * 2^8 * 2^8) / 2^{32} = 2^{-13.5}$이 된다. 마지막으로 5라운드 공격 수행 복잡도는 약 $2^{22} * 2^{13.5} = 2^{36}$정도이고 이는 32비트 라운드 키 공격 복잡도이다. 전체 128비트 라운드 키 공격은 32비트 단위로 3번 더 수행하므로 약 $2^{39}$정도 된다.

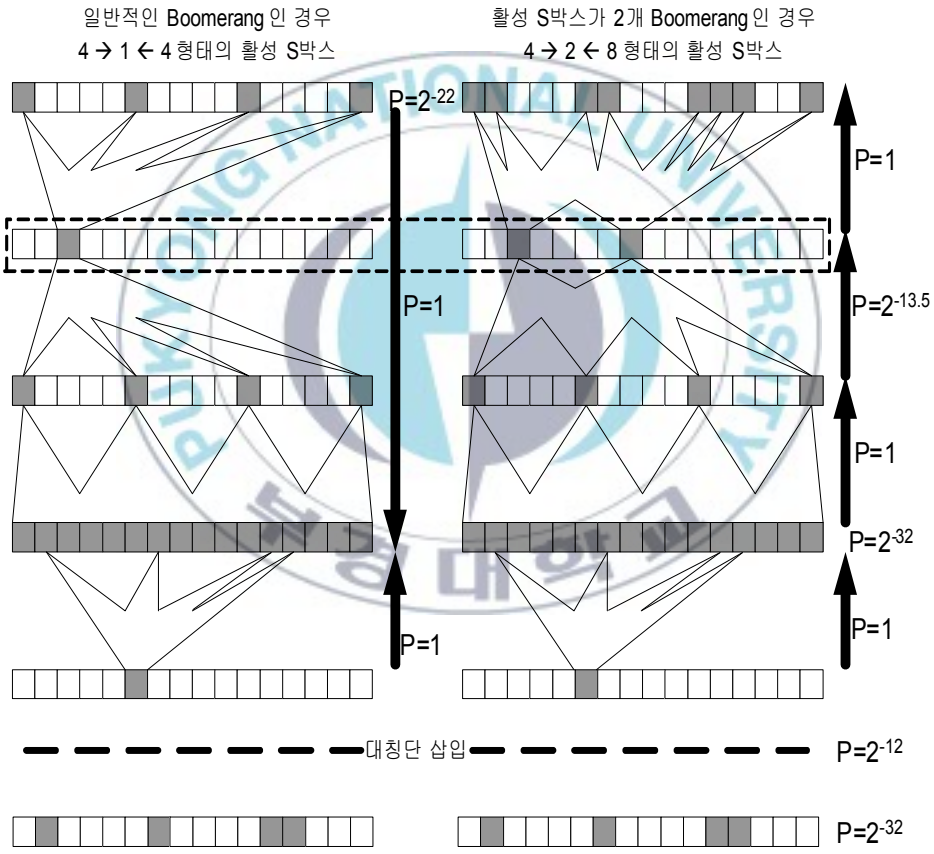



그림 4-5. 대칭단을 적용한 AES 6라운드 Boomerang 공격

(그림 4-5)는 대칭단을 적용하여 기본 5라운드 Boomerang 공격에 마지막 라운드에 1라운드 더 추가하여 6라운드 Boomerang 공격이고 점선으로 된 직사

각형이 Boomerang이다. 5라운드에서 1개의 활성화된 S박스를 만들기 위해 4개의 활성화된 S박스가 필요하고 이때 확률은 $2^{-32}$이 된다. 그리고 대칭단의 적용은 4개의 활성화된 S박스와 대칭단의 라운드 키와 논리 연산의 확률은 $2^{-12}$이 되어 최종적인 대칭단이 적용된 6라운드 Boomerang 공격의 수행 복잡도는 $2^{39} * 2^{32} * 2^{12} = 2^{83}$이 된다.

부정 차분 공격(Truncated Differentials Attack)은 바이트 단위로 연산이 적용되는 암호에서 바이트 패턴이 서로 다른 경우 1, 같은 경우 0으로 정의된 차분 값을 가지고 분석하는 것으로 입력 차분과 출력 차분의 전부를 고려해야 하는 차분 공격보다 쉽고 정확하게 확률을 계산할 수 있다. 대칭단을 적용한 AES 알고리즘에서 대칭단은 기존의 AES의 암호와 복호 알고리즘이 반반씩 적용되고, 그 중간에 대칭단 삽입이 되며, 기존 AES의 암호와 복호 라운드와는 다른 독립적인 불규칙라운드이다. 대칭단이 전체 라운드 중간에 삽입 될 때 암호 전체의 안전성을 나타내주는 확률뿐만 아니라 차분 특성도 변하게 된다. 이는 대칭단에서 적용되는 라운드 키와 논리 연산이 차분 특성 분석을 더욱 어렵게 하고 있으며, 효과적인 차분 특성을 찾기 위한 대칭단에서 사용된 라운드 키를 추론하는 것은 그 라운드 키에 의존한다는 것을 의미한다. 결론적으로 대칭단에서 사용하고 있는 2가지 논리 연산이 위와 같은 효과로 인해 대칭단 이후의 유용한 부정 차분 공격의 구성을 어렵게 하고 있다.

대칭단에서의 선형 공격(Linear Attack)은 적용된 논리 연산이 라운드 키 $K^r[i]$에 의존적이며 XOR 연산이 아닌 AND와 OR 연산이므로 출력 Y[i]는 입력 X[i]에 독립적이다. 이는 입력과 출력의 비트쌍(X[i], Y[i])이 어떠한 선형 근사식(Linear approximation)을 가질 수 없다는 것을 말한다. 다시 말해서 대칭단에 적용된 라운드 키를 구하기 위해 입력과 출력의 비트쌍(X[i], Y[i])의 평균 반이 선형 근사식을 만들 수 없다. 이와 같은 이유로 대칭단의 삽입은 높은 키 의존성에 의해 만들어진 선형 근사식 및 선형 근사식을 구성하는 방법의 한계를 보여준다. 유사한 방법으로 Linear hull 효과 또한 대칭단의 AND와 OR 연산으로 감소하고 Linear hull 특성도 교란시킨다. 결론적으로 대칭단의 적용은 선형 분석에 저항성이 있으며, 대칭단 이후의 라운드 분석에 어려움이 많다.

## 4.2 128비트 대칭단을 적용한 RC6

간단하고, 빠르며, 안전한 블록 암호 알고리즘인 RC6은 미국의 AES 선정 프로젝트, 유럽의 NESSIE 프로젝트, 일본의 CRYPTREC 프로젝트의 128비트 및 가변길이 블록 암호 선정의 최종 후보 암호 알고리즘이었다. RC6은 변형된 Feistel 구조로 암호와 복호 알고리즘이 다른 특징을 가지고 있고 이러한 특징은 RC6을 하드웨어로 구현할 때 암호와 복호 알고리즘이 같은 것보다 면적이 상당히 증가하는 단점이 된다.

표 4-3. 차분, 선형 공격 비교

| 암호 알고리즘 | 공격 방법 | 라운드 | 복잡도 |
|---|---|---|---|
| RC6 | 차분 공격[67] | 20 | $2^{238}$ |
| | 선형 공격[67] | 20 | $2^{101}$ |
| 대칭단을 적용한 RC6 | 차분 공격 | 20 | $2^{274}$ |
| | 선형 공격 | 20 | $2^{125}$ |

기존의 RC6 개발자에 의한 20라운드 차분, 선형 공격 결과와 대칭단을 적용한 20라운드 RC6의 차분, 선형 공격 비교는 (표 4-3)에 나타냈다. 대칭단을 적용한 RC6의 차분, 선형 공격의 자세한 결과는 4.2.3절의 안전성 분석에서 설명한다.

### 4.2.1 구현

대칭단을 RC6 알고리즘에 적용할 때 기존의 라운드 함수 내의 연산은 변경 없이 그대로 사용한다. 그러나 전체 진행 라운드의 반은 암호 알고리즘을 나머지 반은 복호 알고리즘을 적용하고, 중간에 대칭단을 삽입한다.

(그림 4-6)는 대칭단을 적용한 RC6 알고리즘의 전체 진행과정을 그림으로 표현한 것으로 먼저 암호화 과정은 라운드 함수 진행 전에 표백(whitening) 단계로 라운드 키와 덧셈 연산을 수행한 후 10라운드 암호화 라운드를 수행한

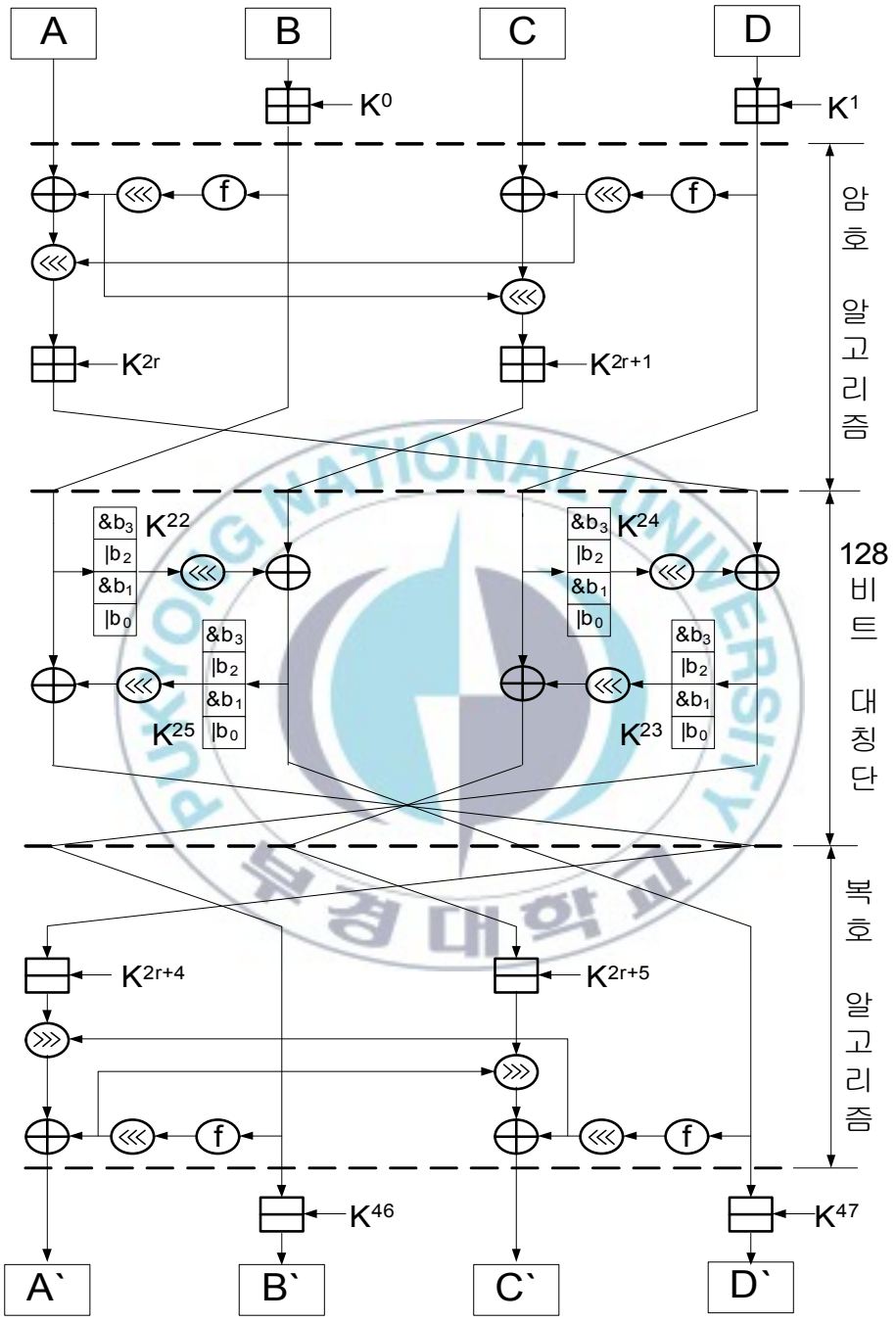



그림 4-6. 128비트 대칭단을 적용한 RC6

다. 각 라운드 연산에서 32비트 라운드 키를 2개씩 덧셈 연산에 사용한다. 다음으로 대칭단의 적용은 3.5절 128비트 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 표백 과정을 거친 후 128비트 암호문을 생성한다.

대칭단을 적용한 RC6 알고리즘의 복호는 (그림 4-6)의 과정을 그대로 수행하며, 라운드 키의 적용을 암호화 과정의 역순으로 적용한다. 그리고 표백 단계에서 수행한 덧셈 연산을 뺄셈 연산으로 전환하고, 대칭단의 수행과정에서 적용된 라운드 키 역시 암호의 역순으로 적용한다.

대칭단을 적용한 RC6의 키 스케줄링은 기존의 RC6의 키 스케줄링을 그대로 사용하며, 단지 대칭단에서 사용된 32비트 4개의 라운드 키를 더 생성해서 총 32비트 48개의 라운드 키를 사용한다.

### 4.2.2 RC6의 부분 상쇄

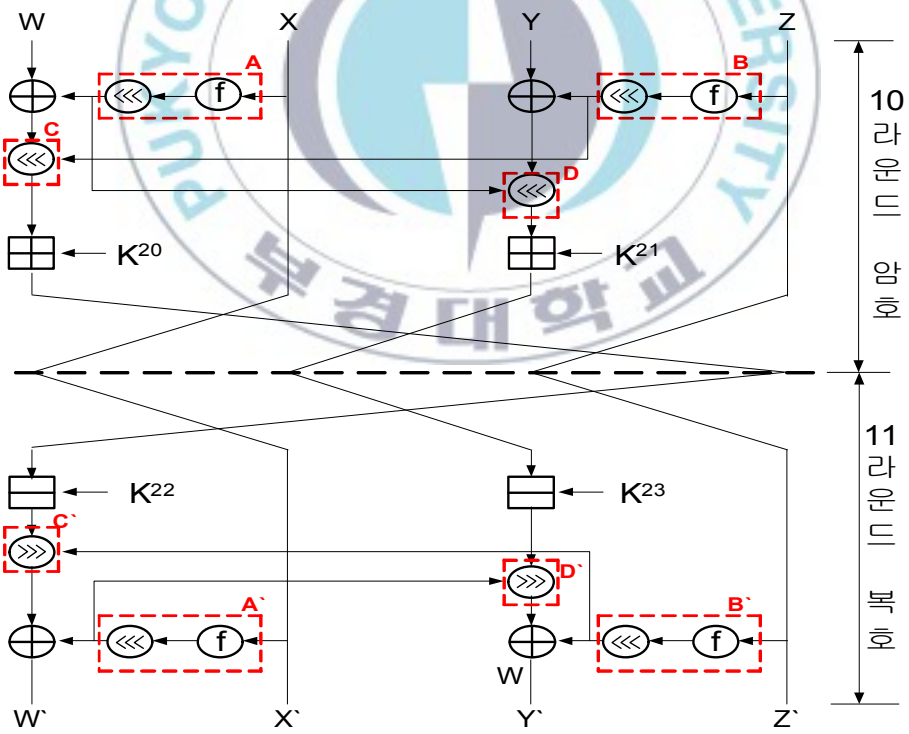



그림 4-7. RC6의 10, 11라운드 부분 상쇄

(그림 4-7)은 대칭단을 적용하지 않고 10라운드 암호 알고리즘 수행 후 연속해서 11라운드 복호과정 적용에 의해 상쇄가 발생한 경우를 그림으로 표현한 것이다. (그림 4-7)에서 10라운드 입력으로 4개의 워드 W, X, Y, Z에서 X, Z는 아무런 변화 없이 11라운드의 같은 워드 위치로 입력된다. 이는 (그림 4-7)에서 각각 X는 A, A`와 Z는 B, B`로 입력되어 RC6의 f() 함수와 5비트 회전 연산의 결과가 같게 된다. 이와 같은 결과는 C, C`와 D, D`의 데이터의존 회전연산 또한 같은 회전연산을 수행하게 되므로 11라운드의 복호과정에서 RC6의 f() 함수와 데이터의존 회전연산은 완전한 상쇄가 발생하며, 단지 10라운드의 32비트 라운드 키 2개의 덧셈과 11라운드 키 2개의 뺄셈연산만 남는다.

이와 같이 대칭단 없이 연속적인 암호와 복호의 적용은 암호에서 복호 알고리즘으로 바뀔 때 암호 마지막 라운드와 복호 첫 라운드가 서로 완전한 상쇄로 각각의 라운드에서 적용된 라운드 키만이 남게 되고 이는 암호의 안전성에 심각한 취약점이 된다.

### 4.2.3 안전성 분석

RC6의 안전성 평가는 [67]에 잘 나타나 있으며 대칭단을 적용한 RC6에서는 차분 공격, 선형 공격 방법으로 안전성을 평가해서 제안한 알고리즘의 안전성을 평가한다.

그리고 다음과 같은 표기 방법을 안전성 분석에서 사용한다. X[i]는 특정 입력으로 32비트 X의 i번 째 비트를 나타내며, 출력 Y[i], 라운드 키 K[i]도 X[i]와 같이 표기한다. X, Y, K만 표시할 때는 32비트 워드이다. 그리고 $X_0$, $X_1$, $X_2$, $X_3$은 RC6의 128비트 입력을 32비트 4개로 나눈 것이고, 아래 첨자는 워드 순서 번호이다. ΔX는 입력 차분이고, ΔY는 출력 차분이다. 마지막으로 $e_i$는 32비트 워드 e에서 i번째 비트만 1이고 나머지는 모두 0인 32비트 값이며, 이 절에서 입력이나, 출력의 차분으로 $e_i$를 사용한다. 이후 이와 같은 표기법은 계속해서 사용한다.

첫 째로 RC6에서 사용된 모든 연산과 차분 공격과의 관계를 알아보겠다. 먼저 표백 단계의 라운드 키와 덧셈 연산의 차분의 차이를 생각해보면, 입력 X와 X`가 아주 조금의 차이가 있으면 X + K와 X`+ K의 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 + K와 X`+ K 또한 MSB만 다르다. 같은 방법으로 X[i]와 X[i]`가 다를 경우(i < 31) X[i] + K와 X[i]` + K

가 다를 확률은 1/2이다.

$$Z[i] = X[i] + K[i] + w[i-1], \qquad (0 \le i \le 31)$$
$$w[i] = X[i] * K[i] + X[i] * w[i-1] + K[i] * w[i-1], \qquad (4-1)$$

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

다음은 데이터의존 회전 연산과 차분 공격과의 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 차분은 예측하기 매우 어렵다. 이는 두 워드가 반드시 같은 값의 회전 연산을 수행하는 조건에서만 유용한 차분 공격이 가능하다는 것을 의미한다. 그리고 회전 연산은 LSB 5비트만 유효한 회전 연산 값이 되므로 확률 $\rho = 2^{-5}$이 된다.

다음은 2차 함수 f(X) = X(2X + 1)의 차분 공격을 생각해 보자. 확률이 1인 2차 함수 f(X) = X(2X + 1)의 미세한 차이는 쉽게 계산이 된다. 다시 말해 32비트 입력 X와 X`의 MSB만 다를 경우 f(X)와 f(X`) 역시 MSB만 다르다. 이와 같은 경우가 확률 1이 된다. 다음은 X와 X`가 두 번째 MSB만 다를 경우 f(X)와 f(X`)는 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 특성을 유지하면서 RC6 내의 모든 라운드 함수에 적용할 수 있다. 그러나 문제는 높은 차분 확률의 특성을 찾는 것이 가능하냐가 문제이다.

그리고 RC6의 개발자들의 차분 공격은 차분의 차이를 XOR 연산이 아니라 뺄셈 연산으로 분석하였다. 다시 말해 입력 $X_0$과 $X_1$의 차이의 차분은 $X_0 - X_1$

mod $2^{32}$이 된다. 먼저 두 워드 $X_0$과 $X_1$의 차이의 입력 차분은 α이며, $f(X_0)$와 $f(X_1)$의 차이의 출력 차분은 Υ인 경우, 출력 차분 Υ에 대해 2차 함수 f(X)에 적용해 보면 (수식 4-2)와 같다.

$$\begin{aligned} f(X_0) - f(X_1) &= X_0(2X_0 + 1) - X_1(2X_1 + 1) \\ &= (X_1 + \alpha)(2(X_1 + \alpha) + 1) - X_1(2X_1 + 1) \\ &= 4\alpha X_1 + 2\alpha^2 + \alpha \end{aligned} \qquad (4-2)$$

(수식 4-2)에서 $X_0 = X_1 + \alpha$이며 다시 $X_0 = X_1 \oplus 2^i$으로 가정했을 때 $X_1[i] = 0$이 된다. 확장하여 어떤 홀수(보통 1이 적용됨) v와 $1 \le i \le 30$일 때 입력 차분 $\alpha = v2^i$이며, $\Upsilon = \alpha$이 될 확률은 $2^{i-30}$이고, 앞에서 설명한 것과 같이 입력 차분 $\alpha = 2^{31}$일 경우 $\Upsilon = \alpha$이 될 확률은 1이 되고, i = 0일 경우는 확률이 0이다. 이는 (수식 4-3)를 통해 쉽게 증명 가능하다.

$$\begin{aligned} \Upsilon &= 4X_1\alpha + 2\alpha^2 + \alpha = 0 \bmod 2^{32} \\ \Upsilon - \alpha &= 4X_1\alpha + 2\alpha^2 = 0 \bmod 2^{32} \\ 4X_1v2^i + 2(v2^i)^2 &= 0 \bmod 2^{32} \\ 4X_1 + 2(v2^i) &= 0 \bmod 2^{32-i} \\ X_1 + v2^{i-1} &= 0 \bmod 2^{30-i} \end{aligned} \qquad (4-3)$$

(수식 4-3)에서 v = 1이고, $1 \le i \le 15$인 경우이고, $16 \le i \le 30$인 경우는 $X_1 = 0 \bmod 2^{30-i}$가 된다. 두 경우 모두 $X_1$의 MSB (i + 2)비트의 어떠한 값이던지 $X_1$의 LSB (30 − i)비트로 변환되며, 모든 가능한 $X_1$에 대한 $\Upsilon = \alpha$이 될 확률 평균은 $2^{i+2} / 2^{32} = 2^{i-30}$이다.

(수식 4-2)와 (수식 4-3)의 결과로 확률이 높은 어떤 특성이 있는 입력 차분 α와 $X_1$값에 의존적이다. 이와 같은 방법을 1차 차분 공격하며, 특히 Hamming Weight가 1인( $\Upsilon = \alpha = 2^i$) 차이를 갖는 차분 특성은 유용한 RC6의 차분 공격을 할 수 있다.

1차 차분 공격과 유사한 방법으로 4개의 워드 입력을 적용하는 2차 차분 공격을 적용해 보면 다음과 같다. $X_0$과 $X_1$, $X_2$와 $X_3$의 차이는 α이고, $X_1$과 $X_3$의 차이가 β인 경우 이 4개의 입력 워드를 f(X) 함수에 적용해 보면

$$f(X_0) - f(X_1) - (f(X_2) - f(X_3))$$

$$= 4aX_1 + 2a^2 + a - (4aX_3 + 2a^2 + a)$$
$$= 4a(X_1 - X_3)$$
$$= 4a\beta \qquad (4-4)$$

(수식 4-4)의 결과는 입력 워드와 관계없이 입력 워드 간의 차분만으로 유용한 차분 공격을 할 수 있음을 보이는 것이다. 2개의 차분을 이용하여 RC6을 공격하는 구체적인 방법은 다음과 같다. $\delta = w_0 - w_1$이라 두고 $w_0 = X_1 \oplus f(X_1)$, $w_1 = X_3 \oplus f(X_3)$일 때 $(\alpha, \beta) = (2^i, 0)$ 또는 $\delta = 2^i$이 되는 $(\alpha, \beta) \rightarrow \delta$인 차분 특성 확률은 pi이고, 이와 유사하게 $(\alpha, \beta) = (2^i, 2^i)$ 또는 $\delta = 0$이 되는 $(\alpha, \beta) \rightarrow \delta$인 차분 특성 확률은 qi이다. 그리고 언제나 pi = qi가 같을 확률은 다음과 같다.

$$(pi = qi) = \begin{cases} 2^{i-31}, & 15 \le i \le 31, \\ (\text{표 } 4\text{-}4), & 0 \le i \le 14, \end{cases} \qquad (4-5)$$

(표 4-4)의 결과 값은 참고문헌 [67]에 RC6 개발자가 직접 실행한 결과로 이 절에서는 그 결과를 그대로 사용한다.

표 4-4. $0 \le i \le 14$일 경우 pi = qi인 확률

| i | $2^{31-i}$ | pi = qi | i | $2^{31-i}$ | pi = qi |
|---|---|---|---|---|---|
| 0 | $2^{31}$ | $2^{-30.10}$ | 8 | $2^{23}$ | $2^{-25.92}$ |
| 1 | $2^{30}$ | $2^{-29.15}$ | 9 | $2^{22}$ | $2^{-25.44}$ |
| 2 | $2^{29}$ | $2^{-28.57}$ | 10 | $2^{21}$ | $2^{-24.98}$ |
| 3 | $2^{28}$ | $2^{-28.17}$ | 11 | $2^{20}$ | $2^{-23.09}$ |
| 4 | $2^{27}$ | $2^{-27.71}$ | 12 | $2^{19}$ | $2^{-19.58}$ |
| 5 | $2^{26}$ | $2^{-27.21}$ | 13 | $2^{18}$ | $2^{-18.10}$ |
| 6 | $2^{25}$ | $2^{-26.89}$ | 14 | $2^{17}$ | $2^{-17.01}$ |
| 7 | $2^{24}$ | $2^{-26.42}$ | | | |

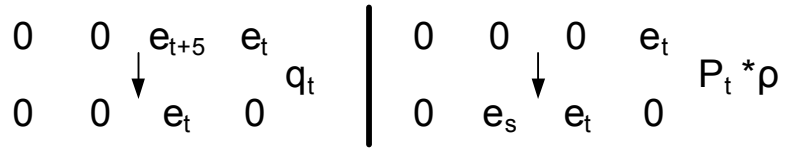



그림 4-8. 1차 차분과 2차 차분 진행과정

(그림 4-8)의 왼쪽은 2차 차분공격의 1라운드 진행이고 오른쪽은 1차 차분공격의 진행과정이다. 대칭단을 적용한 20라운드 RC6의 차분 공격 과정은 (그림 4-9)에서 $B_6$은 시작 6라운드로서 초기 값으로 첫 번째 32비트 워드의 차분은 MSB만 1로 하고 두 번째 워드의 차분은 26번째 비트만 1로 셋팅 후 6라운드를 진행한다. 32비트 값이 0인 것은 차분이 없는 것을 뜻한다. 그 다음 $I_6$은 $B_6$의 6라운드 진행 후 그대로 이어 받아 진행 하게 된다. 그리고 $I_6$의 4라운드 다음에 대칭단이 적용되고, 마지막으로 $E_6$의 6라운드를 적용한 대칭단을 적용한 20라운드 RC6의 전체 차분 공격을 한다. 공격과정은 대칭단이 있는 $I_6$을 자세한 설명하고 $B_6$, $E_6$은 $I_6$과 같은 방법으로 공격하며 참고문헌 [67]에 잘 설명되어 있다.

$$
\begin{aligned}
I_6 &= \rho^6 * q_t * P_s * P_v * P_u * q_{u-5} * q_{u-10} * 2^{-12} * 2^{-12} * 2^{-12} \\
&= \rho^6 * q_{11} * P_{26} * P_{26} * P_{26} * q_{21} * q_{16} * 2^{-12} * 2^{-12} * 2^{-12} \\
&= 2^{-30} * 2^{-23} * 2^{-4} * 2^{-4} * 2^{-5} * 2^{-10} * 2^{-15} * 2^{-12} * 2^{-12} * 2^{-12} \\
&= 2^{-127}
\end{aligned}
$$

$I_6$의 공격 과정에서 t = 11, s = v = u = 26인 차분이다. $P_s$, $P_v$는 1차 차분 공격으로 (수식 4-3)의 확률을 적용이며, $P_u$, $q_t$, $q_u$는 2차 차분 공격으로 (수식 4-5)와 (표 4-4)의 확률을 적용한다. $2^{-12}$, $2^{-12}$, $2^{-12}$는 각각 대칭단의 첫 번째, 두 번째, 네 번째의 논리 연산이 적용된 확률이다. 최종적으로 $I_6 = 2^{-127}$이 된다.

그래서 대칭단을 적용한 RC6의 20라운드 전체 차분 확률은 $2^{-76} * 2^{-127} * 2^{-71} = 2^{-274}$이다.

두 번째는 선형 공격이다. 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 공격과 유사하며, 간단한 2라운드 반복(iterative) 선형 근사값(Approximation)으로 (r − 2)라운드 선형 분석을 할 수 있다. 물론 차분 공격

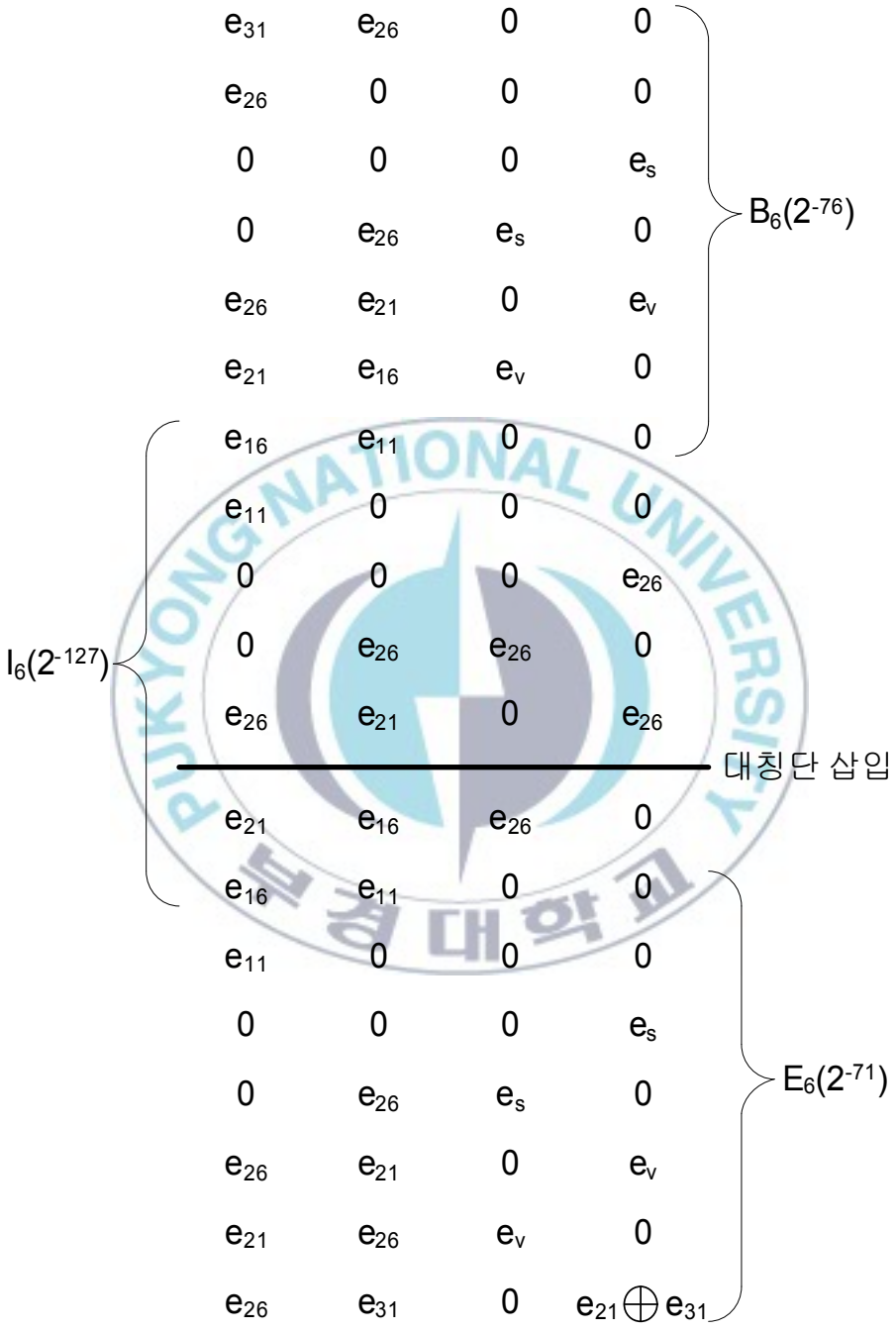



그림 4-9. 대칭단을 적용한 RC6의 20라운드 차분 공격

과 마찬가지로 높은 확률 특성을 찾는 것이 쉽지 않다. 먼저 회전 연산 A = B ≪ C 대한 선형 근사 확률 ρ = (t / 32) + ((32 − t) / 32) * (1 / 2) (ρ = $2^{-6}$)이며, 이 때 t = 1로 LSB의 5비트까지만 유효한 단일 선형 근사 편차(bias)이다. 이는 차분 공격의 회전 연산에 대한 차분 특성 확률과 같은 유사한 값이다. 그리고 회전 연산에 대한 단일 선형 근사 편차를 구하는데 32비트 A, B, C의 각각의 1비트만으로 구하는 방법과 A, B의 1비트만으로 구하는 방법이 있다. 이 중 A, B의 1비트만으로 구하는 방식이 좀 더 유용하다.

마지막으로 Y = X + K와 같은 어떤 고정된 워드 K와 입력 워드 X의 덧셈 연산이 0 ≤ i ≤ 31, Y • $e_i$ = X • $e_i$가 되는 선형 근사값의 편차를 $\kappa_i$라 하면 모든 가능한 K에 대한 평균 $\kappa_i$는 다음과 같다.

$$\kappa_i \quad \begin{matrix} 1/2, & i = 0 \\ 1/4, & 1 \le i \le 31 \end{matrix} \qquad (4-6)$$

(그림 4-10)은 단일 선형 근사값을 이용한 2라운드 반복적 선형 공격으로 2라운드 반복적 분석을 9번 적용해서 20라운드 선형 공격을 했다. 선형 공격 과정은 첫 번째 세 번째 워드에서 LSB만 취하여 2라운드를 진행하여 $2^{r-1}\prod_{i=0}^{r-1} \cdot \kappa_i$인 piling-up lemma[11]를 적용한 다음과 같은 식을 얻을 수 있다.

$$a_u * \rho^2 * a_v * 2^3 \qquad (4-7)$$

(수식 4-7)에서 $a_u$, $a_v$는 u와 v위치(position)에서 덧셈 연산의 단일 선형 근사값의 편차이고, ρ는 회전 연산의 선형 근사값의 편차이다. (수식 4-7)을 (수식 4-6)에 적용하여 계산한 결과는 $2^{-11}$이다. 이 결과를 20라운드에 적용하는 식은 다음과 같다.

$$(2^{-11})^{r/2} * 2^{(r/2)-1} \qquad (4-8)$$

(수식 4-8)에서 $r$은 라운드 수이다. 20라운드의 결과는 $2^{-101}$이다. 최종적으로 대칭단을 적용한 20라운드 RC6의 선형 공격 확률은 $2^{-101} * 2^{-12} * 2^{-12} = 2^{-125}$가 된다. 여기서 $2^{-12}$과 $2^{-12}$는 각각 대칭단의 첫 번째와 세 번째 워드

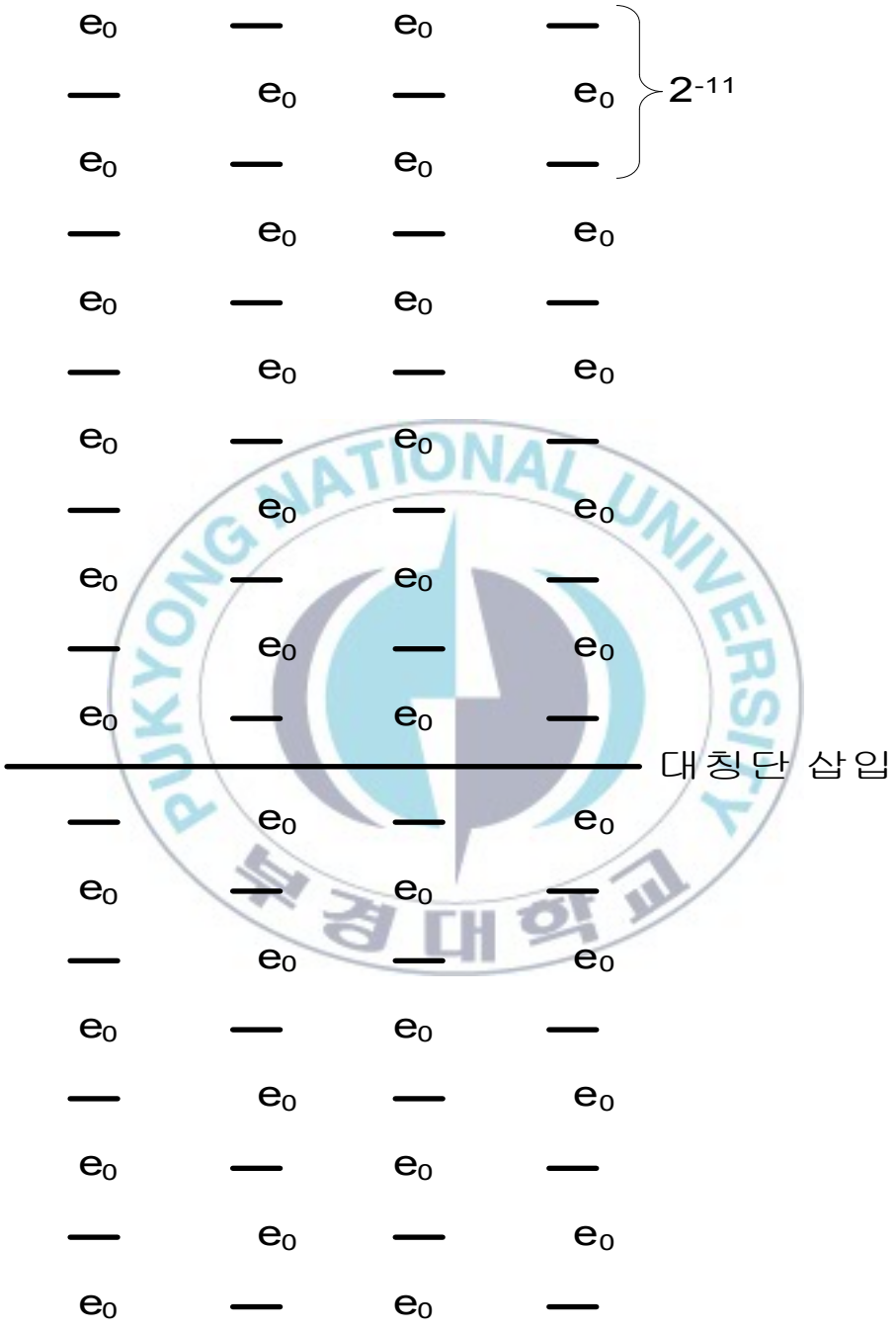



그림 4-10. 대칭단을 적용한 RC6의 20라운드 선형 공격

의 논리 연산의 확률이다.

## 4.3 실행 결과 분석

대칭단을 적용한 AES, RC6 알고리즘은 CBC(Cipher Block Chaining) 운영 모드를 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 Vista, Intal core2 Duo 2.26Ghz, 2.27Ghz, 2GB RAM의 환경에서 기존의 AES, RC6 알고리즘과 대칭단을 적용한 AES, RC6 알고리즘의 수행 시간을 테스트했다. 결과는 (표 4-5)와 같으며, 대칭단을 적용한 알고리즘들의 암호와 복호 수행 시간이 조금 증가하는 것으로 나타났다. 그러나 이 정도의 속도 증가는 대칭단에서 적용한 간단한 논리 연산들이 전체적인 암호와 복호 알고리즘 수행에 거의 영향을 미치는 않는 것으로 판단된다.

표 4-5. AES, RC6와 대칭단을 적용한 AES, RC6의 수행 시간 비교

| 암호 알고리즘 | 암호 | 복호 |
|---|---|---|
| AES | 8.040<br>(100%) | 8.303<br>(100%) |
| 대칭단을<br>적용한 AES | 8.176<br>(101.69%) | 8.439<br>(101.64%) |
| RC6 | 7.098<br>(100%) | 6.887<br>(100%) |
| 대칭단을<br>적용한 RC6 | 7.148<br>(100.7%) | 7.051<br>(102.38%) |

(표 4-6)은 AES와 RC6의 라운드 함수에서 사용된 주요 연산자로 각각 10, 20라운드를 적용한 연산 회수이고, 대칭단은 128비트 대칭단에 적용된 주요 연산자 회수이다.

표 4-6. AES, RC6와 대칭단의 주요 연산자의 비교

| | AES | RC6 | 대칭단 |
|---|---|---|---|
| 테이블 참조 | 160 | - | - |
| XOR(32) | 40 | 40 | 4 |
| rotate | 30 | 80 | 4 |
| XOR(8) | 1008 | - | - |
| if | 144 | - | - |
| AND/OR(8) | - | - | 16 |
| 곱셈(32) | - | 40 | - |
| 덧셈/뺄셈(32) | - | 40 | - |

## 4.4 결론

AES 공모 프로젝트에 의해 선택된 SPN 구조의 대표적인 블록 암호 알고리즘인 AES의 암호와 복호 알고리즘이 서로 다르다는 단점을 간단한 논리 연산과 자리바꿈으로 구성된 대칭단을 적용하여 암호와 복호 알고리즘이 동일한 SPN 구조 블록 암호 알고리즘을 구현했다. 즉 AES의 전체 라운드 중 1라운드에서 ((R / 2) - 1)라운드까지는 AES의 암호 알고리즘을 적용하고, ((R / 2) + 1)라운드부터 R라운드까지는 AES의 복호 알고리즘을 적용하면서 암호와 복호 사이에 하나의 독립적인 라운드인 대칭단을 적용했다.

간단한 논리 연산으로 구성한 대칭단을 적용한 AES 알고리즘은 기존의 AES 알고리즘 보다 향상된 안전성을 보여주고 있으며 암호와 복호가 동일하기 때문에 하드웨어 구성을 간단히 할 수 있는 장점을 가진다.

또한 RC6 역시 암호와 복호 알고리즘이 다른 것을 AES와 동일한 대칭단을 삽입하여 RC6의 암호와 복호를 같게 개선했다. 즉 RC6의 10라운드는 암호 알고리즘을, 나머지 10라운드는 복호 알고리즘을 사용하고 중간에 대칭단을 삽입해서 암호와 복호를 같게 구현했다. 기존의 RC6과 안전성 평가에서도 대칭단의 적용이 암호 알고리즘을 분석하는데 유효한 차분 및 선형 특성을 단절 또

는 변화를 통해 안전성이 향상되는 것을 보여 주고 있다.

그리고 암호와 복호 수행 시간 테스트에서 대칭단을 적용한 AES, RC6이 약간의 속도 증가를 보여주고 있는데 이는 전체적인 수행 속도를 볼 때 미세한 증가로 대칭단이 수행 속도에서 거의 영향을 미치지 않는다고 볼 수 있다.

암호와 복호가 다른 블록 암호 알고리즘을 간단한 대칭단을 적용하여 스마트카드 및 RFID 태그와 같은 제한된 하드웨어 및 소프트웨어 환경에서도 쉽게 구현 가능하고 AES와 같은 SPN 구조의 블록 암호 알고리즘과 RC6과 같은 Feistel 구조의 블록 암호 알고리즘에 대칭단을 삽입해서 암호와 복호 알고리즘을 동일하게 구현할 수 있으므로 새로운 블록 암호 설계 및 개발에도 유용한 아이디어로 사용될 것이다.

# 제 5장 대칭단을 적용한 160비트 블록 암호 알고리즘

이 장에서는 암호와 복호가 서로 다른 160비트 블록 암호 알고리즘인 SHACAL-1에 대칭단을 적용하여 암호와 복호를 같게 구성한 새로운 SHACAL-1을 소개하고, 새롭게 구성된 알고리즘의 안전성을 중점적으로 설명하고, 실행 결과 및 결론 순으로 설명한다.

## 5.1 160비트 대칭단을 적용한 SHACAL-1

유럽의 NESSIE 프로젝트에 160비트 블록 암호 알고리즘으로 제안된 SHACAL-1은 키 스케줄링 알고리즘의 취약점 때문에 여러 가지 키 스케줄링과 관련된 공격으로 SHACAL-1이 안전하지 않다는 것이 입증되어 최종적인 NESSIE 프로젝트에서 160비트 블록 암호 알고리즘으로 선정되지 않았다.

표 5-1. 차분, 선형, 연관 키 Slide 공격 비교

| 암호 알고리즘 | 공격 방법 | 라운드 | 복잡도 |
|---|---|---|---|
| SHACAL-1 | 차분 공격[68, 69] | 80 | $2^{116}$ |
| | 선형 공격[68] | 80 | $2^{80}$ |
| | 연관 키 Slide 공격[70] | 80 | $2^{98}$ |
| 대칭단을 적용한 SHACAL-1 | 차분 공격 | 80 | $2^{128}$ |
| | 선형 공격 | 80 | $2^{87}$ |
| | 연관 키 Slide 공격 | 80 | $2^{244}$ |

참고문헌[68, 69]은 SHACAL-1 개발자들에 의한 기본적인 차분, 선형 공

격으로, 연관 키 공격 기술을 이용하지 않은 순수한 차분 특성과 선형 근사 확률을 이용한 SHACAL-1의 80라운드 공격이다. [70]은 연관 키와 Slide 공격 [71]을 결합한 선택 평문 공격 시나리오로 전체 80라운드 SHACAL-1 공격으로 32비트 2개의 라운드 키를 공격한다.

(표 5-1)의 대칭단을 적용한 SHACAL-1의 안성성에 대한 결과는 5.4절 안전성 분석에서 다시 자세히 설명한다.

## 5.2 구현

대칭단을 SHACAL-1 알고리즘에 적용할 때 기존의 4개의 서로 다른 라운드 함수는 변경 없이 그대로 사용한다. 그러나 각 20라운드마다 다른 라운드 함수를 10라운드로 줄여서 40라운드는 SHACAL-1의 암호 알고리즘을 적용하고, 대칭단을 적용한 다음 나머지 40라운드는 SHACAL-1의 복호 알고리즘을 적용한다.

(그림 5-1)은 대칭단을 적용한 SHACAL-1 알고리즘의 전체 진행과정을 그림으로 표현한 것으로 먼저 암호화 과정은 라운드 함수 진행 전에 표백 단계로 라운드 키와 XOR 연산을 수행한 후 10라운드씩 40라운드 SHACAL-1암호화 라운드를 수행한다. 각 라운드 연산에서 32비트 라운드 키를 1개씩 덧셈 연산에 사용한다.

다음으로 대칭단의 적용은 32비트 라운드 키를 5개 사용한다. 나머지 10라운드씩 40라운드는 SHACAL-1의 복호 알고리즘을 적용하고 각 라운드마다 32비트 라운드 키 1개씩 뺄셈 연산을 수행하고 최종적으로 마지막 표백 과정을 거친 후 160비트 암호문을 생성한다.

대칭단을 적용한 SHACAL-1의 키 스케줄링 알고리즘은 기존의 SHACAL-1의 알고리즘을 그대로 사용하고 단지 대칭단에서 사용되는 32비트 5개의 라운드 키와 암호 알고리즘 수행 전과 마지막 라운드 후 XOR 연산을 위한 10개의 라운드 키를 더 생성하여 전체 라운드 키의 개수는 총 95개를 생성하여 사용한다.

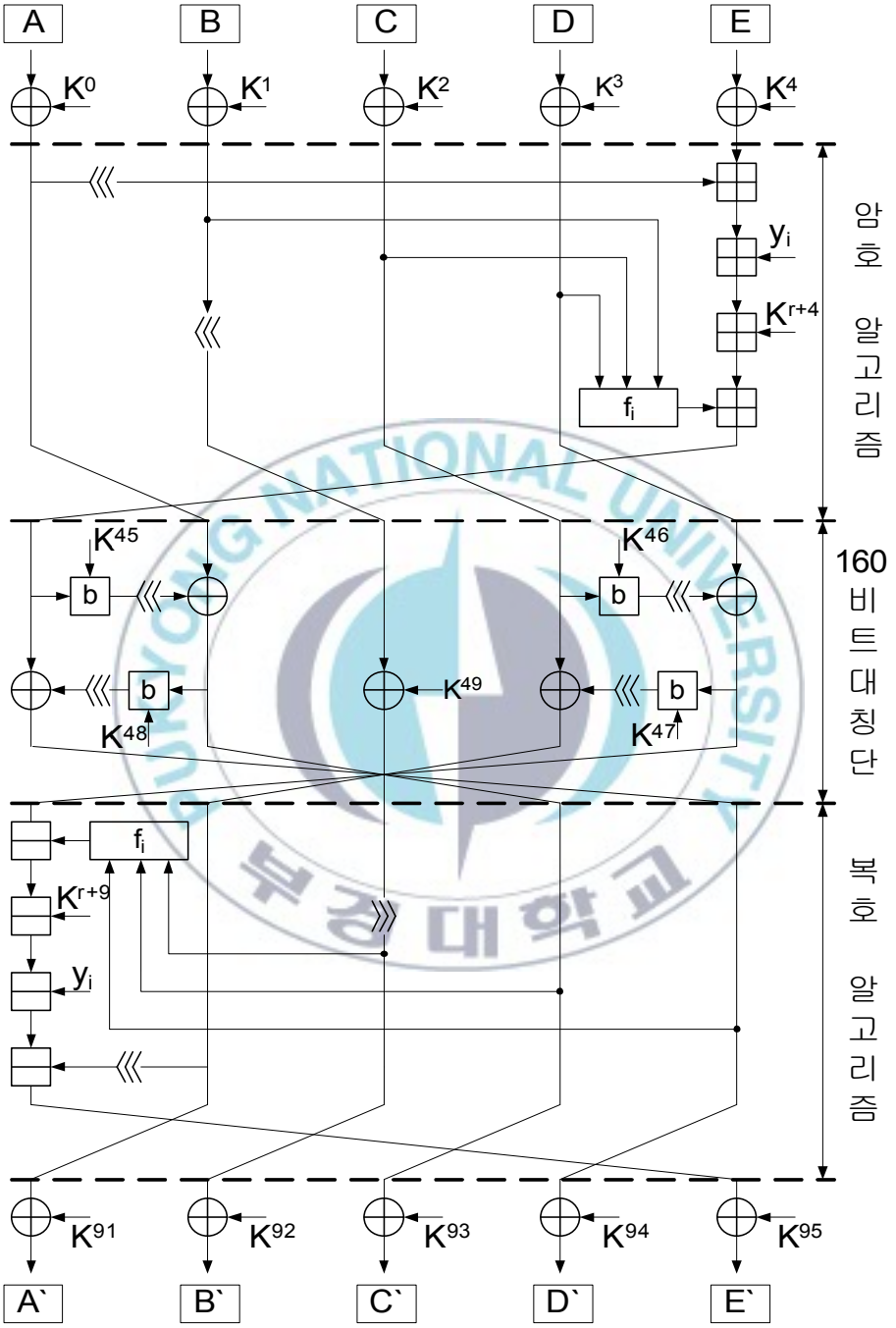



그림 5-1. 160비트 대칭단을 적용한 SHACAL-1

## 5.3 SHACAL-1의 부분 상쇄

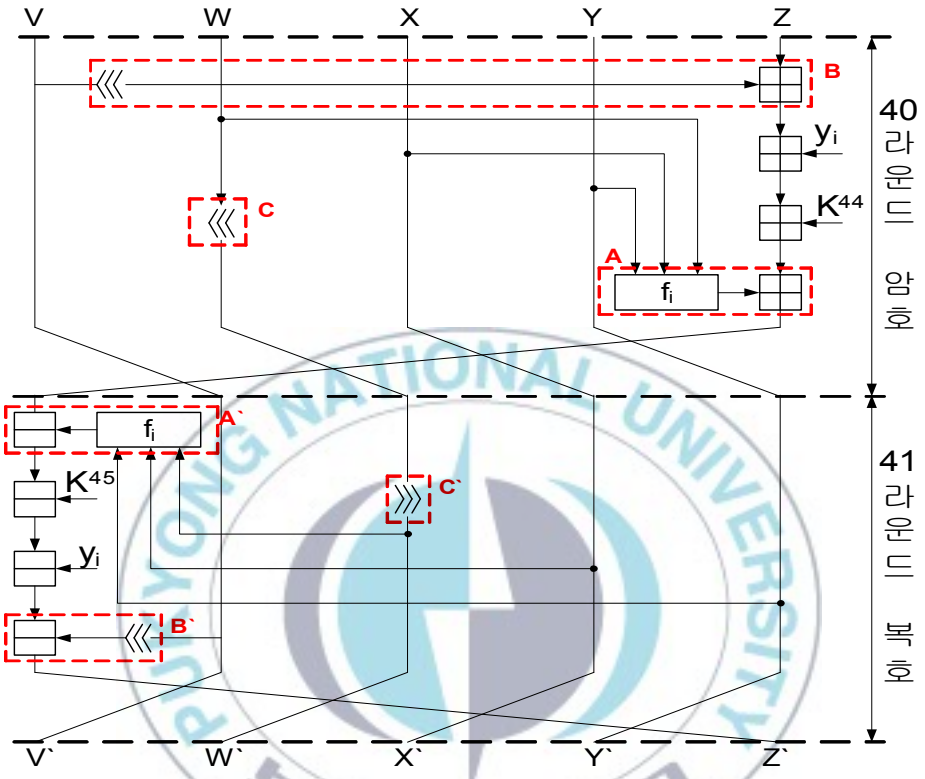



그림 5-2. SHACAL-1의 40, 41라운드 부분 상쇄

(그림 5-2)는 대칭단을 적용하지 않고 SHACAL-1의 40라운드 암호 알고리즘 수행 후 연속해서 41라운드 복호과정 적용에 의해 상쇄가 발생한 경우를 그림으로 표현한 것이다. (그림 5-2)에서 40라운드 입력으로 5개의 워드 V, W, X, Y, Z에서 V는 40라운드에서 B를 적용하고 아무런 변화 없이 41라운드에서 B`를 적용한다. 이는 같은 V에 5비트 회전연산을 수행한 후 덧셈과 뺄셈을 수행하므로 완전한 상쇄가 발생한다. W는 40라운드에서 30비트 왼쪽 회전연산(C)을 수행하고 41라운드에서는 30비트 오른쪽 회전 연산(C`)을 수행하므로 역시 완전한 상쇄가 발생한다. 마지막으로 3개의 워드 W, X, Y를 입력받아 SHACAL-1의 40라운드 함수 $f_{xor}()$를(A) 수행하고 41라운드에서 같은 워드 W, X, Y를 입력받아 41라운드 함수 $f_{if}()$를(A`) 수행한다. 이는 완전한 상쇄는

아니지만 같은 값을 입력 받으므로 암호의 안전성에 영향을 미친다. 그리고 라운드 상수($y_i$)의 덧셈과 뺄셈 역시 상쇄처럼 소거할 수 있다. 그래서 최종적으로 40라운드의 라운드 키와 41라운드 키의 덧셈과 뺄셈 그리고 40라운드 함수 $f_{xor}()$와 41라운드 함수 $f_{if}()$만 남게 된다.

이와 같이 대칭단 없이 연속적인 암호와 복호의 적용은 SHACAL-1의 암호에서 복호 알고리즘으로 바뀔 때 암호 마지막 라운드와 복호 첫 라운드가 서로 부분적인 상쇄로 각각의 라운드에서 적용된 라운드 키와 라운드 함수만 남게 되고 이는 암호의 안전성에 심각한 취약점이 된다.

## 5.4 안전성 분석

SHACAL-1에 대한 안전성 분석[68]은 알고리즘 개발자에 의한 차분, 선형 공격을 했으며, 개발자와 같은 방법으로 10라운드씩 4가지 라운드 함수를 적용하여 40라운드의 차분, 선형 공격으로 대칭단을 적용한 SHACAL-1의 80라운드의 안전성을 분석한다.

표 5-2. 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 |

대칭단을 적용한 SHACAL-1의 차분 공격은 첫 번째 XOR 연산과 정수 덧셈을 적용하는 것에 대한 차분 확률과 두 번째로 SHACAL-1의 라운드 함수 중 비선형 변환 함수 $f_{if}()$, $f_{xor}()$, $f_{maj}()$의 적용에 대한 차분 특성을 생각해 볼 수 있다.

Z = X + Y, Z` = X` + Y`에서 X와 Y가 i번째 비트만 다를 때 X ⊕ Y = $e_i$로 표시하며, 이와 같은 XOR 연산의 차분과 덧셈 연산 사이의 관계는 다음과 같다. X ⊕ X` = $e_{31}$이고 Y = Y`일 때 Z ⊕ Z` = $e_{31}$은 확률 1이고, X ⊕ X` = $e_{31}$이고 Y ⊕ Y` = $e_{31}$일 때 Z = Z`는 확률 1이고, X ⊕ X` = $e_j$이고 Y = Y`일 때 Z ⊕ Z` = $e_j$는 확률 1/2이고, X ⊕ X` = $e_j$이고 Y ⊕ Y` = $e_j$일 때 Z = Z`는 확률 1/2이다[72]. (0 ≤ j ≤30)

두 번째로는 비선형 변환 함수 $f_{if}()$, $f_{xor}()$, $f_{maj}()$에 대한 차분 분석이다. 위의 비선형 함수들은 3비트를 입력 받아 1비트를 출력하는 함수로 (표 5-2)는 이러한 함수들의 XOR 차분 분포를 나타낸 것이다. (표 5-2)에서 X, Y, Z는 비선형 함수들의 1비트 입력으로 3비트 조합은 8가지의 경우가 있으며 $f_{if}()$, $f_{xor}()$, $f_{maj}()$의 각각의 XOR 차분 분포는 (표 5-2)의 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 각각 1/2이 되는 것을 의미한다[68].

표 5-3. $f_{if}()$와 $f_{maj}()$ 함수의 10라운드 차분 특성 확률

| 라운드 | A | B | C | D | E | 확률 |
|---|---|---|---|---|---|---|
| | 0 | $e_1$ | $e_{26}$ | 0 | 0 | |
| 1 | 0 | 0 | $e_{31}$ | $e_{26}$ | 0 | 1/4 |
| 2 | 0 | 0 | 0 | $e_{31}$ | $e_{26}$ | 1/4 |
| 3 | $e_{26}$ | 0 | 0 | 0 | $e_{31}$ | 1/4 |
| 4 | 0 | $e_{26}$ | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | $e_{24}$ | 0 | 0 | 1/2 |
| 6 | 0 | 0 | 0 | $e_{24}$ | 0 | 1/2 |
| 7 | 0 | 0 | 0 | 0 | $e_{24}$ | 1/2 |
| 8 | $e_{24}$ | 0 | 0 | 0 | 0 | 1/2 |
| 9 | $e_{29}$ | $e_{24}$ | 0 | 0 | 0 | 1/2 |
| 10 | $e_2$ | $e_{29}$ | $e_{22}$ | 0 | 0 | 1/4 |

앞서 설명한 차분 분석 방법과 SHACAL-1 개발자들이 소개한 가장 좋은 차분 공격 특성을 이용하여 10라운드씩 적용한 4가지 비선형 라운드 암호 알고리즘 중 $f_{if}()$와 $f_{maj}()$의 차분 공격 확률은 다음과 같다.

$f_{if}()$와 $f_{maj}()$ 함수는 각각 1 ~ 10라운드와 21 ~ 30라운드에 적용되며, (표 5-3)과 같은 차분 특성을 이용할 수 있다. (표 5-3)에서 초기 상태는 두 번째, 세 번째 32비트 워드 B와 C가 각각 $e_1$, $e_{26}$으로 다음 라운드 진행 후 1라운드와 같은 상태로 될 확률이 1/4이 된다. 그 이유는 $f_{if}()$와 $f_{maj}()$ 함수가 적용되는 워드가 B, C, D이며 (표 5-2)를 적용한 확률이다. 또 1라운드를 적용한 $f_{if}()$와 $f_{maj}()$ 함수 역시 1/4의 차분 확률을 가진다. 그리고 1라운드를 더 적용한 경우 이번에는 $f_{if}()$와 $f_{maj}()$의 차분 확률은 1/2이고, 덧셈 확률 1/2과 곱한 확률 1/4이 전체 차분 확률이 된다. 이와 같은 방법으로 $f_{if}()$와 $f_{maj}()$ 함수의 10라운드 차분 특성 확률은 각각 $2^{-13}$이다.

표 5-4. $f_{xor}()$ 함수의 10라운드 차분 특성 확률[69]

| 라운드 | A | B | C | D | E | 확률 |
|---|---|---|---|---|---|---|
| | $e_{1,3}$ | $e_{1,8}$ | 0 | $e_{3,6,31}$ | $e_{1,3,31}$ | |
| 1 | 0 | $e_{1,3}$ | $e_{6,31}$ | 0 | $e_{3,6,31}$ | 1/16 |
| 2 | $e_1$ | 0 | $e_{1,31}$ | $e_{6,31}$ | 0 | 1/8 |
| 3 | $e_1$ | $e_1$ | 0 | $e_{1,31}$ | $e_{6,31}$ | 1/4 |
| 4 | 0 | $e_1$ | $e_{31}$ | 0 | $e_{1,31}$ | 1/2 |
| 5 | 0 | 0 | $e_{31}$ | $e_{31}$ | 0 | 1/2 |
| 6 | 0 | 0 | 0 | $e_{31}$ | $e_{31}$ | 1 |
| 7 | 0 | 0 | 0 | 0 | $e_{31}$ | 1 |
| 8 | $e_{31}$ | 0 | 0 | 0 | 0 | 1 |
| 9 | $e_4$ | $e_{31}$ | 0 | 0 | 0 | 1/2 |
| 10 | $e_{9,31}$ | $e_4$ | $e_{29}$ | 0 | 0 | 1/2 |

그리고 $f_{xor}()$ 함수는 각각 11 ~ 20, 31 ~ 40라운드에 적용되며, 앞서 설명한 방법으로 차분 특성을 구성한 경우가[69] (표 5-4)이다. 그래서 $f_{xor}()$ 함

수의 각각의 차분 특성 확률은 $2^{-13}$이다. 이와 같이 10라운드씩 적용된 $f_{if}()$, $f_{maj}()$, $f_{xor}()$ 함수의 전체 40라운드의 차분 특성 확률은 $2^{-52}$이 되며, SHACAL-1 복호 알고리즘의 비선형 변환 함수 $f_{if}()$, $f_{maj}()$, $f_{xor}()$ 함수는 암호와 같다. 그리고 덧셈 연산이 복호 알고리즘에서는 뺄셈 연산으로 바뀌게 되며 이는 안전성 분석 방법이 덧셈과 같고 안전성 역시 덧셈과 같은 안전성을 보인다. 그래서 SHACAL-1의 복호 알고리즘 또한 암호와 마찬가지로 $2^{-52}$의 차분 특성 확률을 갖는다. 최종적으로 대칭단을 적용한 SHACAL-1의 80라운드 차분 공격 확률은 $2^{-52} * 2^{-24} * 2^{-52} = 2^{-128}$이 된다. $2^{-24}$은 대칭단에서 적용된 라운드 키와 논리 연산의 결과로 $f_{xor}()$ 함수의 마지막 10라운드 결과인 첫 번째 워드, 두 번째 워드가 대칭단에서 적용되며 전체 5개의 워드 중에서 2개의 워드가 논리 연산의 적용을 받는다. 그래서 대칭단의 논리연산 공격 확률이 $2^{-24}$이 된다.

표 5-5. $f_{xor}()$ 함수의 10라운드 선형 편차

| 라운드 | A | B | C | D | E | 편차 |
|---|---|---|---|---|---|---|
| | $e_{24} \oplus e_2$ | $e_{29} \oplus e_4$ | $e_{29} \oplus e_2$ | $e_{29} \oplus e_2$ | $e_{29}$ | |
| 1 | $e_{29}$ | $e_2$ | $e_2$ | $e_2$ | $e_2$ | |
| 2 | $e_2$ | - | - | - | - | 1/2 |
| 3 | - | $e_2$ | - | - | - | 1/2 |
| 4 | - | - | $e_0$ | - | - | 1/2 |
| 5 | - | - | - | $e_0$ | - | 1/2 |
| 6 | - | - | - | - | $e_0$ | 1/2 |
| 7 | $e_0$ | $e_{27}$ | $e_{30}$ | $e_0$ | $e_0$ | 1/2 |
| 8 | $e_0$ | $e_{27} \oplus e_0$ | $e_{30} \oplus e_{25}$ | $e_{30} \oplus e_0$ | - | 1/2 |
| 9 | - | $e_0$ | $e_{25} \oplus e_{30}$ | $e_{25} \oplus e_{30}$ | $e_{30} \oplus e_0$ | 1/2 |
| 10 | $e_{30} \oplus e_0$ | $e_{27} \oplus e_{25}$ | $e_{28}$ | $e_{25} \oplus e_0$ | $e_{25} \oplus e_0$ | |

대칭단을 적용한 SHACAL-1의 선형 공격은 piling-up lemma와 간단한 정수 덧셈 Y[i] = K[i] + X[i]에서 선형 근사값을 생각할 수 있다. 여기서 X[i]는 32비트 평문의 i번째 비트이며, K[i]는 고정된 라운드 키의 i번째 비트이다. i = 0이면 편차는 언제나 1/2로 수렴하고, i = 1이고 K[0] = 0이면 1/2로 수렴하지만 K[0] = 1이면 확률이 1/2로 편차(bias)는 0이 된다. 그러므로 $i \geq 1$이고 $K \& (2^i - 1) = 0$인 경우 선형 근사값의 최대 편차를 가질 수 있으며, i보다 낮은 위치에서 캐리가 없는 것을 고려한다. 그래서 i > 1인 어떤 경우에도 최대 편차는 $2^{-2}$과 같거나 작다. 여기서 최대 편차를 주는 약한 키의 개수는 i의 위치에 의존적이며, 예를 들어 i = 2일 때는 4개의 키 중 한 개가 최대 편차를 가지고, i = 30일 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 $K \& (2^i - 1) = 2^{i-1}$이다.

표 5-6. $f_{if}()$ 함수의 10라운드 선형 편차

| 라운드 | A | B | C | D | E | 편차 |
|---|---|---|---|---|---|---|
| | $e_2$ | - | - | - | - | |
| 1 | - | $e_2$ | - | - | - | 1/2 |
| 2 | - | - | $e_0$ | - | - | 1/2 |
| 3 | - | - | - | $e_0$ | - | 1/2 |
| 4 | - | - | - | - | $e_0$ | 1/2 |
| 5 | $e_0$ | $e_{27}$ | - | $e_0$ | - | 1/4 |
| 6 | - | $e_0$ | $e_{25}$ | - | $e_0$ | 1/2 |
| 7 | $e_0$ | $e_{27}$ | - | $e_{25} \oplus e_0$ | - | 1/4 |
| 8 | - | $e_0$ | $e_{25}$ | - | $e_{25} \oplus e_0$ | 1/2 |
| 9 | $e_{25} \oplus e_0$ | $e_{27} \oplus e_{20}$ | - | $e_0$ | - | |
| 10 | - | $e_{25} \oplus e_0$ | $e_{25} \oplus e_{18}$ | - | $e_0$ | |

이와 같은 방법으로 대칭단을 적용한 SHACAL-1의 10라운드씩 적용한 $f_{if}()$, $f_{maj}()$, $f_{xor}()$ 함수의 40라운드 암호 알고리즘의 선형 공격은 우선 XOR 연산으로 이루어진 $f_{xor}()$ 함수는 각각 11 ~ 20, 31 ~ 40라운드에 적용된다.

(표 5-5)는 10라운드 $f_{xor}()$ 라운드 함수의 진행과정으로 2 ~ 8라운드까지는 확률 1로 편차는 1/2이 되고 앞서 설명한 선형 공격 방법으로 1라운드와 10라운드를 적용한 전체 10라운드 $f_{xor}()$ 경우 최대 편차는 $2^{-6}$이다. 그리고 $f_{xor}()$는 11 ~ 20, 31 ~ 40라운드에 적용되므로 20라운드 선형 근사값의 편차는 $(2^{-6})^2 * 2 = 2^{-11}$이 된다.

(표 5-6)은 1 ~ 10라운드에 적용되는 $f_{if}()$의 진행과정을 설명한 것으로 1 ~ 4라운드까지는 최대 편차가 $2^{-1}$이고, 5 ~ 8라운드까지는 최대 편차가 $2^{-3}$이다. 그리고 마지막 9 ~ 10라운드를 적용한 전체 10라운드 $f_{if}()$의 최대 편차는 약 $2^{-6}$이 된다.

표 5-7. $f_{maj}()$ 함수의 10라운드 선형 편차

| 라운드 | A | B | C | D | E | 편차 |
|---|---|---|---|---|---|---|
| | $e_2$ | - | - | - | - | |
| 1 | - | $e_2$ | - | - | - | 1/2 |
| 2 | - | - | $e_0$ | - | - | 1/2 |
| 3 | - | - | - | $e_0$ | - | 1/2 |
| 4 | - | - | - | - | $e_0$ | 1/2 |
| 5 | $e_0$ | $e_{27}$ | - | $e_0$ | - | 1/4 |
| 6 | - | $e_0$ | $e_{25}$ | - | $e_0$ | 1/2 |
| 7 | $e_0$ | $e_{27}$ | - | $e_{25}$ | - | 1/4 |
| 8 | - | $e_0$ | $e_{25}$ | - | $e_{25}$ | 1/2 |
| 9 | $e_{25}$ | $e_{20}$ | $e_{30}$ | - | - | |
| 10 | - | $e_{25}$ | $e_{18}$ | $e_{30}$ | - | |

(표 5-7)은 21 ~ 30라운드에 적용되는 $f_{maj}()$의 진행과정을 설명한 것으로 1 ~ 4라운드까지는 최대 편차가 $2^{-1}$이고, 5 ~ 8라운드까지는 최대 편차가 $2^{-3}$이다. 그리고 마지막 9 ~ 10라운드를 적용한 전체 10라운드 $f_{maj}()$의 최대 편차는 약 $2^{-5}$이 된다.

대칭단을 적용한 SHACAL-1의 80라운드 선형 공격은 지금까지 설명한 40라운드 SHACAL-1 암호 알고리즘의 선형 공격 확률은 $2^{-11} * 2^{-6} * 2^{-5} * 2^{2} = 2^{-20}$이다. 그리고 선형 공격 역시 SHACAL-1의 암호와 복호를 같은 분석 방법을 그대로 적용할 수 있고, 안전성 또한 같다. 그래서 최종적인 대칭단을 적용한 SHACAL-1의 선형 공격 확률은 $(2^{-20})^{2} * 2 * 2^{-48} = 2^{-87}$이다. 여기서 $2^{-48}$은 대칭단에서 적용된 라운드 키와 SHACAL-1의 40라운드 암호 알고리즘 수행 후 논리 연산 적용 확률이다.

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

$$Kr_0^{i} = Kr_1^{i+1} \qquad (5-1)$$

(수식 5-1)에서 $Kr_0^{i}$, $Kr_1^{i+1}$는 초기 키 $K_0$, $K_1$에 의해 각각 생성된 라운드 키이고, i는 라운드 수이다.

어떤 연관 키의 쌍($K_0$, $K_1$)으로 $P_0 = f_{Kr1}^{1}(P_1)$와 같은 관계를 만족하는 평문쌍($P_0$, $P_1$)이 있고 이때 $f_{Kr1}^{1}$는 초기 키 $K_1$로 만들어진 1라운드 키를 사용하여 암호화하는 과정이다. 이 평문에 대응되는 $C_0 = f_{Kr0}^{i}(C_1)$를 만족하는 암호문쌍($C_0$, $C_1$)의 $f_{Kr0}^{i}$은 초기 키 $K_0$으로 만들어진 i라운드 키를 사용하여 암호화하는 과정이다. 이와 같은 연관 관계가 있는 키와 위에서 설명한 조건의 평문의 쌍이 주어지면 쉽게 라운드 키에 대한 정보를 얻을 수 있다. 위에서 설명한 분석방법은 간단한 연관 관계가 있으면서 라운드 키에 독립적인 암호화 라운드 함수로 구성된 블록 암호 알고리즘의 공격에 사용하는 기법으로 Slide 공격이라 하고, Slide 공격은 n비트 블록 크기에 대한 $2^{n/2}$의 공격 복잡도를 갖는다. SHA-1에 대한 Saarinen[73]의 연구를 결합하여 80라운드 SHACAL-1 연관 키 Slide 공격을 수행할 수 있다. Saarinen의 공격 시나리오는 두 개의 서로 다른 연관 키 $K_0 = (Kr_0^{0}, kr_0^{1}, \ldots, kr_0^{15})$와 $K_1 = (Kr_1^{0}, kr_1^{1}, \ldots, kr_1^{15})$의 관

계는 다음과 같이 정의한다.

$$\begin{aligned} Kr_1^i &= Kr_0^{i+1}, && (0 \le i \le 14), \\ &= Kr_0^{16} = (Kr_0^{13} \oplus Kr_0^8 \oplus Kr_0^2 \oplus Kr_0^0) \lll 1, && (i = 15), \\ Kr_1^i &= Kr_0^{i+1}, && (0 \le i \le 78), \end{aligned}$$

그리고 $K_0$과 $K_1$에 대한 평문을 각각 $P_0 = (A_0^0, B_0^0, C_0^0, D_0^0, E_0^0)$, $P_1 = (A_1^0, B_1^0, C_1^0, D_1^0, E_1^0)$을 정의 하면 (수식 5-2)를 만족 한다.

$$A_1^0 = A_0^1, \quad B_1^0 = B_0^1, \quad C_1^0 = C_0^1, \quad D_1^0 = D_0^1, \quad E_1^0 = E_0^1, \qquad (5-2)$$

(수식 5-2)는 SHACAL-1의 20라운드까지 만족한다. 왜냐하면 SHACAL-1은 20라운드 마다 다른 비선형 변환 라운드 함수를 적용하기 때문이며 다음과 같은 등식으로 표현할 수 있다.

$$\begin{aligned} A_0^{21} &= Kr_0^{20} + (A_0^{20} \lll 5) + f_{20}(B_0^{20}, C_0^{20}, D_0^{20}) + E_0^{20} + Y^{20} \\ &= Kr_1^{19} + (A_1^{19} \lll 5) + f_{19}(B_1^{19}, C_1^{19}, D_1^{19}) + E_1^{19} + Y^{19} \\ &= A_1^{20}, \end{aligned}$$

그리고 SHACAL-1의 5개의 워드가 랜덤 순열(random permutation)이라면 (수식 5-3)을 만족하는 확률이 $2^{-32}$이 된다[72].

$$f_{xor}(B^{20}, C^{20}, D^{20}) + Y^{20} = f_{if}(B^{20}, C^{20}, D^{20}) + Y^{19}, \qquad (5-3)$$

그리고 (수식 5-3)과 유사한 식이 40라운드, 60라운드에도 적용할 수 있으며, 이는 각각 20라운드 마다 다른 비선형 라운드 함수를 적용하는 SHACAL-1의 특성 때문이다. 그래서 Saarinen의 최종적인 결론은 (수식 5-2)를 만족하는 Slide 쌍($P_0$, $P_1$)의 확률이 $2^{-96}$이 된다. 그리고 Saarinen의 공격 시나리오는 SHACAL-1의 복호 과정에서도 그대로 적용 할 수 있다. 왜냐하면 SHACAL-1의 복호 알고리즘은 암호 알고리즘에서 사용된 비선형 변환 함수를 그대로 사용한다. 단지 덧셈 연산을 뺄셈 연산으로 바꾸면 되기 때문이다.

이와 같은 이론을 바탕으로 대칭단을 적용한 80라운드 SHACAL-1의 연관키 Slide 공격과정은 다음과 같다. (수식 5-1)과 같은 연관 관계가 있는 두 개

의 초기 키($K_0$, $K_1$)에 의한 암호화 과정을 수행한 후 생성된 암호문쌍과 이때 사용된 라운드 키 정보를 얻기 위한 Slide 공격에 필요한 후보 Slide 쌍(Slide pairs)을 찾는다.

(그림 5-3)는 대칭단을 적용한 연관 키 Slide 공격 과정으로 160비트 SHACAL-1의 평문 집합 $P_0$ = (A, B, C, D, x)은 초기 키 $K_0$에 의해 생성된 라운드 키로 SHACAL-1의 암호화 과정을 수행한다. 이때 임의의 고정된 32비트 A, B, C, D와 32비트의 모든 값을 다 가지는 x의 집합이다. 그리고 평문 집합 $P_1$ = (y, A, B ≪ 30, C, D)은 초기 키 $K_1$에 의해 생성된 라운드 키로 SHACAL-1의 암호화 과정을 수행한다. 이때 32비트 A, B, C, D는 평문 집합 $P_0$에서 사용된 값과 같은 값이며, y는 32비트의 모든 값을 포함하는 집합이다. 각각의 평문 집합 $P_0$, $P_1$은 $2^{32}$개의 원소가 있는 집합이다. 따라서 $2^{64}$개의 높은 확률의 연관성을 가지는 Slide 쌍이 생성된다. (수식 5-4)와 같은 Slide 쌍을 찾기 위해 평문 집합 $P_0$, $P_1$과 이에 대응되는 각각의 암호문에서 4개의 32비트 값으로 최종적으로 128비트만 체크하면 된다.

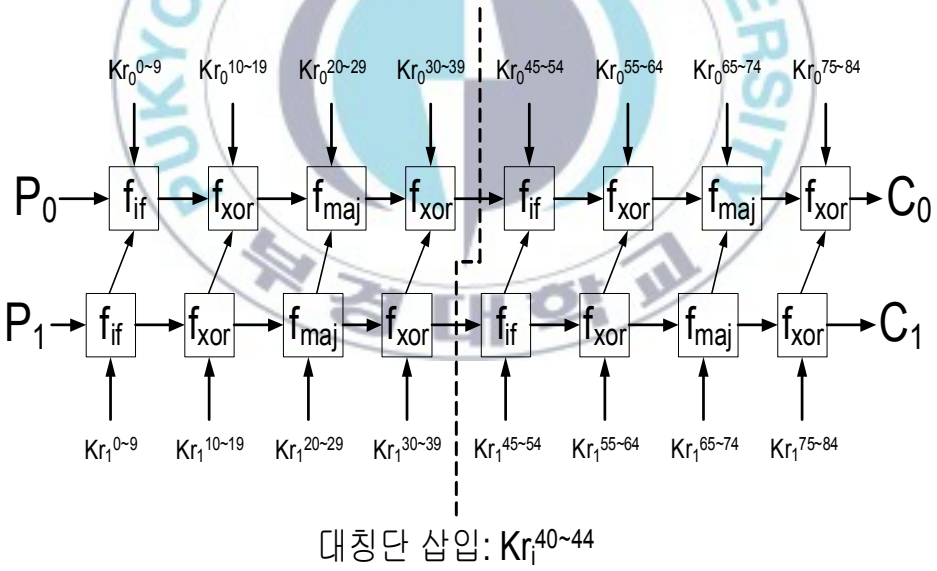



그림 5-3. 대칭단을 적용한 SHACAL-1의 80라운드 연관 키 Slide 공격

$$A_0^{80} = A_1^{79},\ B_0^{80} = B_1^{79},\ C_0^{80} = C_1^{79},\ D_0^{80} = D_1^{79},\ E_0^{80} = E_1^{79}, \quad (5-4)$$

그리고 Saarinen의 공격 시나리오에 따라 최종적으로 분석 확률인 $2^{96}$번을

4번 체크해보면 마지막 라운드의 라운드 키 쌍을 얻을 수 있으므로 SHACAL-1의 연관 키 Slide 공격 확률은 $2^{-98}$이 된다. 이와 같은 방법을 1 ~ 40라운드까지 암호 알고리즘을 (그림 5-3)에 그대로 적용하고 대칭단을 적용한 다음 마지막으로 41 ~ 80라운드까지 복호 알고리즘을 적용하면 된다. 앞서 설명한 대로 SHACAL-1의 복호 알고리즘의 Saarinen의 공격 시나리오는 암호 알고리즘과 같은 안전성을 가지므로 최종적인 대칭단을 적용한 80라운드 SHACAL-1의 공격 복잡도는 $(2^{98})^2 * 2^{48} = 2^{244}$이 되고 이는 대칭단의 적용이 SHACAL-1의 연관 키 Slide 공격에 충분한 저항성(resistance)이 있는 것으로 판단된다.

## 5.5 실행 결과 분석

대칭단을 적용한 SHACAL-1 알고리즘은 CBC(Cipher Block Chaining) 운영 모드를 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 Vista, Intal core2 Duo 2.26Ghz, 2.27Ghz, 2GB RAM의 환경에서 기존의 SHACAL-1 알고리즘과 대칭단을 적용한 SHACAL-1 알고리즘의 수행 시간을 테스트했다. 결과는 (표 5-8)과 같으며, 대칭단을 적용한 알고리즘의 암호와 복호 수행 시간이 약 4.1% 증가하는 것으로 나타났다. 이와 같은 증가는 대칭단에서 적용한 간단한 논리 연산들이 전체적인 암호와 복호 알고리즘 수행에 거의 영향을 미치는 않는 것으로 판단된다.

표 5-8. SHACAL-1와 대칭단을 적용한 SHACAL-1의 수행 시간 비교

| 암호 알고리즘 | 암호 | 복호 |
|---|---|---|
| SHACAL-1 | 7.383<br>(100%) | 7.511<br>(100%) |
| 대칭단을 적용한 SHACAL-1 | 7.738<br>(104.8%) | 7.765<br>(103.4%) |

(표 5-9)는 SHACAL-1의 라운드 함수에서 사용된 주요 연산자로 80라운드를 적용한 연산 회수이고, 대칭단은 160비트 대칭단에 적용된 주요 연산자 회수이다.

표 5-9. SHACAL-1과 대칭단의 주요 연산자의 비교

| | SHACAL-1 | 대칭단 |
|---|---|---|
| XOR(32) | 80 | 5 |
| rotate | 320 | 4 |
| AND/OR(32) | 320 | - |
| AND/OR(8) | - | 16 |
| NOT(32) | 20 | - |
| 덧셈/뺄셈(32) | 640 | - |

## 5.6 결론

유럽의 NESSIE 프로젝트에 160비트 블록 암호 알고리즘으로 제안된 SHACAL-1은 암호와 복호가 서로 다른 80라운드 블록 암호 알고리즘으로 20라운드 마다 다른 비선형 라운드 연산을 수행하는 특성을 가지고 있다. 그래서 SHACAL-1의 20라운드 마다 다른 비선형 변환 연산을 연결한 80라운드를 10라운드마다 다른 비선형 변환으로 라운드 함수를 적용한 40라운드 암호 알고리즘을 적용하고 나머지 40라운드 역시 10라운드 마다 다른 비선형 변환 함수를 적용한 복호 알고리즘을 적용한 다음 암호와 복호 알고리즘 사이에 대칭단을 적용하여 암호와 복호 알고리즘이 같게 SHACAL-1을 구성하였다.

간단한 논리 연산으로 구성된 대칭단을 적용한 SHACAL-1 알고리즘은 기존의 SHACAL-1 알고리즘 보다 차분, 선형, 연관 키 Slide 공격에 대해서 향상된 안전성을 보여주고 있으며, 특히 SHACAL-1의 안전성에 대한 최대 취약점인 연관 키 Slide 공격에 대해서도 대칭단의 적용이 충분한 안전성을 보여주고 있다. 그리고 대칭단을 적용한 SHACAL-1의 새로운 구성은 암호와 복호가 동일하기 때문에 하드웨어 구성을 간단히 할 수 있는 장점을 가진다.

대칭단을 적용한 SHACAL-1과 기존의 SHACAL-1을 소프트웨어로 구현

한 후 암호와 복호 수행 시간 테스트한 결과 대칭단을 적용한 SHACAL-1이 약간의 속도 증가를 보여주고 있는데 이는 전체적인 수행 속도를 볼 때 미세한 증가로 대칭단이 전체적인 암호와 복호 수행 속도에서 거의 영향을 미치지 않는다고 볼 수 있다.

암호와 복호가 다른 블록 암호 알고리즘을 간단한 대칭단을 적용하여 스마트카드 및 RFID 태그와 같은 제한된 하드웨어 및 소프트웨어 환경에서도 쉽게 구현 가능하고 SHACAL-1과 같은 변형된 Feistel 구조의 블록 암호 알고리즘에 대칭단을 삽입해서 암호와 복호 알고리즘을 동일하게 구현할 수 있으므로 새로운 블록 암호 설계 및 개발에도 유용한 아이디어로 사용될 것이다.

# 제 6장 대칭단을 적용한 256비트 블록 암호 알고리즘

이 장에서는 암호와 복호가 서로 다른 256비트 블록 암호 알고리즘인 SHACAL-2에 대칭단을 적용하여 암호와 복호를 같게 구성한 새로운 SHACAL-2를 소개하고, 새롭게 구성된 알고리즘의 안전성을 중점적으로 설명하고, 실행 결과 및 결론 순으로 설명한다.

## 6.1 256비트 대칭단을 적용한 SHACAL-2

유럽의 NESSIE 프로젝트에 256비트 블록 암호 알고리즘으로 제안된 SHACAL-2는 SHA-2의 해시 함수를 기반으로 설계된 블록 암호 알고리즘으로 2003년 256비트 블록 암호 알고리즘으로 최종적으로 선정되었다. 현재까지 SHACAL-2의 안전성에 대한 분석은 많이 없지만 안전성에 대한 문제점도 아직까지는 없다.

표 6-1. 연관 키 차분-비선형, 연관 키 Rectangle 공격 비교

| 암호 알고리즘 | 공격 방법 | 라운드 | 복잡도 |
|---|---|---|---|
| SHACAL-2 | 연관 키 차분-비선형 공격[74] | 35 | $2^{451.1}$ |
| | 연관 키 Rectangle 공격[74] | 40 | $2^{449.4}$ |
| 대칭단을 적용한 SHACAL-2 | 연관 키 차분-비선형 공격 | 35 | $2^{547.1}$ |
| | 연관 키 Rectangle 공격 | 40 | $2^{473.4}$ |

참고문헌[74]는 SHACAL-2의 분석 방법 중 가장 효율적이고 성능 좋은

분석 방법으로 연관 키 차분-비선형 공격은 Langford와 Hellman의 차분-선형 공격[41]과 Hawkes의 연관 키 차분-선형 공격[75]을 바탕으로 35라운드 분석으로 전수조사보다 좋은 성능으로 공격한다. 그리고  연관 키 Rectangle 공격은 40라운드 분석으로 연관키 차분-비선형 공격과 마찬가지로 전수조사보다 좋은 성능으로 공격한다. 이와 같은 SHACAL-2의 공격방법은 연관 키, 차분-비선형, Rectangle 공격[76]과 같은 단순한 한 가지의 공격이 아니라 2가지의 기본 공격을 결합한 복합공격 방법이다.

(표 6-1)의 대칭단을 적용한 SHACAL-2의 안성성에 대한 결과는 6.4절 안전성 분석에서 다시 자세히 설명한다.

## 6.2 구현

대칭단을 SHACAL-2 알고리즘에 적용할 때 기존의 비선형 변환 라운드 함수는 변경 없이 그대로 사용한다. 그러나 전체 64라운드의 1/2인 32라운드를 SHACAL-2의 암호 알고리즘을 적용하고, 대칭단을 적용한 다음 나머지 32라운드는 SHACAL-2의 복호 알고리즘을 적용한다.

(그림 6-1)은 대칭단을 적용한 SHACAL-2 알고리즘의 전체 진행과정을 그림으로 표현한 것으로 먼저 암호화 과정은 라운드 함수 진행 전에 표백 단계로 라운드 키와 XOR 연산을 수행한 후 32라운드 SHACAL-2 암호화 라운드를 수행한다. 각 라운드 연산에서 32비트 라운드 키를 1개씩 덧셈 연산에 사용한다.

다음으로 대칭단의 적용은 32비트 라운드 키를 8개 사용하고, 나머지 32라운드는 SHACAL-2의 복호 알고리즘을 적용하고 각 라운드마다 32비트 라운드 키 1개씩 뺄셈 연산을 수행하고 최종적으로 마지막 표백 과정을 거친 후 256비트 암호문을 생성한다.

대칭단을 적용한 SHACAL-2의 키 스케줄링 알고리즘은 기존의 SHACAL-2의 알고리즘을 그대로 사용하고 단지 대칭단에서 사용되는 32비트 8개의 라운드 키와 암호 알고리즘 수행 전과 마지막 라운드 후 XOR 연산을 위한 16개의 라운드 키를 더 생성하여 전체 라운드 키의 개수는 총 88개를 생성하여 사용한다.

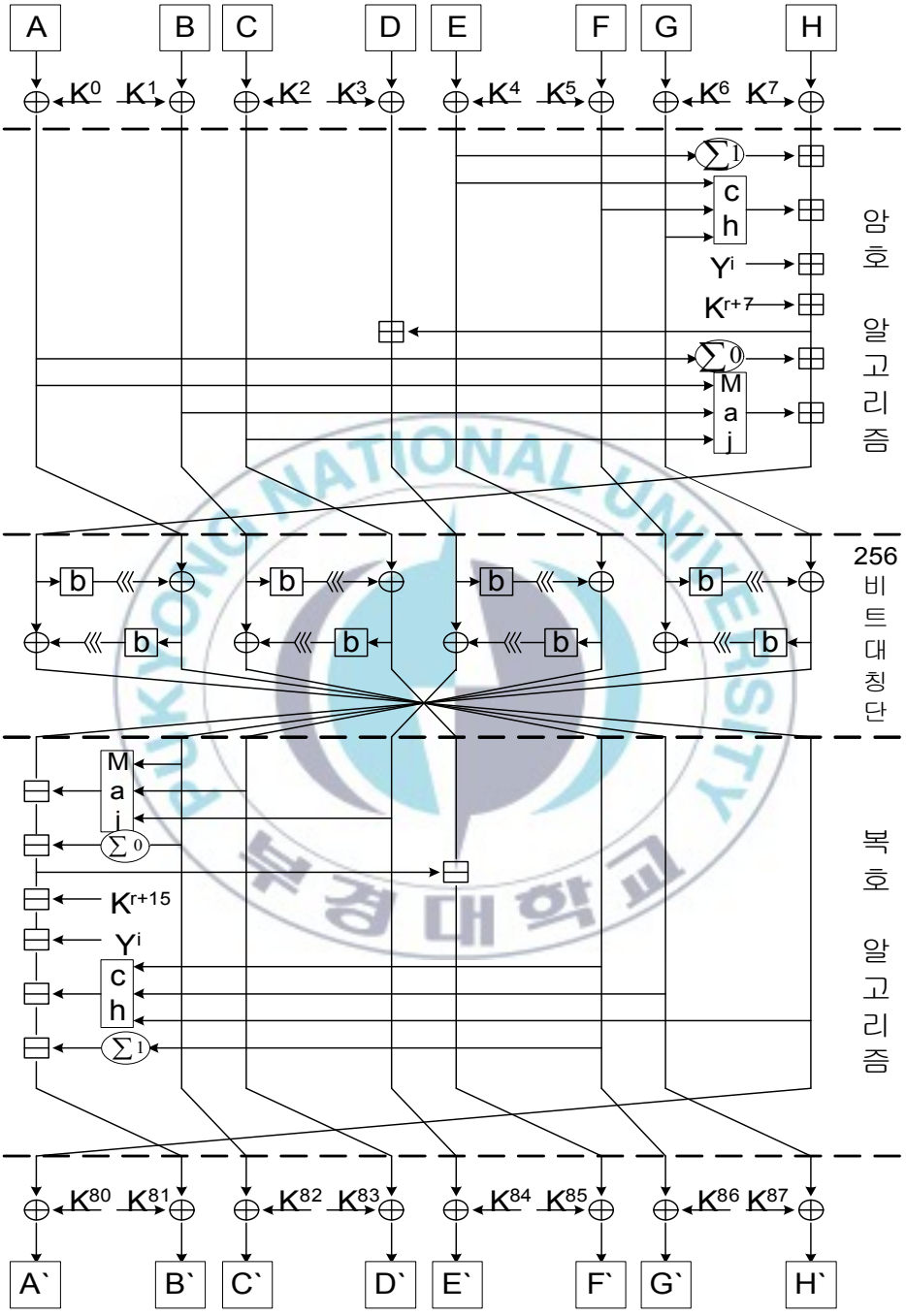



그림 6-1. 256비트 대칭단을 적용한 SHACAL-2

## 6.3 SHACAL-2의 부분 상쇄

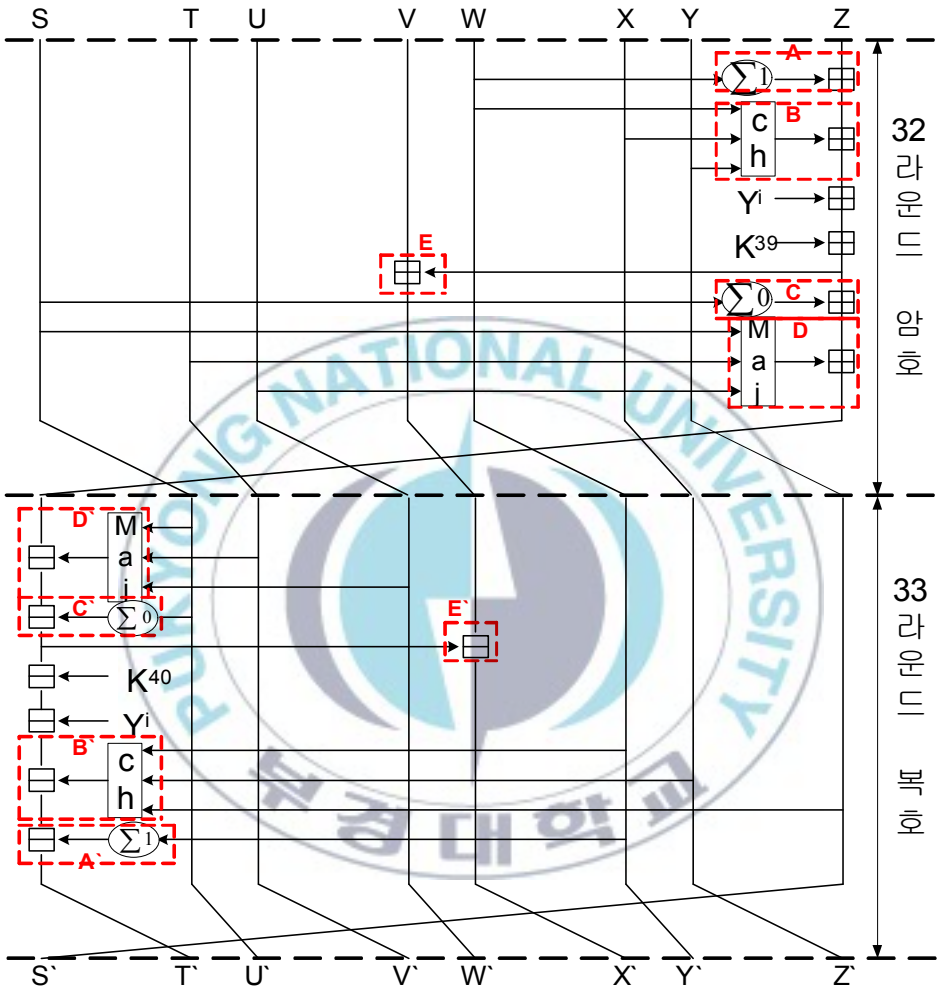



그림 6-2. SHACAL-2의 32, 33라운드 부분 상쇄

(그림 6-2)는 대칭단을 적용하지 않고 SHACAL-2의 32라운드 암호 알고리즘 수행 후 연속해서 33라운드 복호과정 적용에 의해 상쇄가 발생한 경우를 그림으로 표현한 것이다. (그림 6-2)에서 32라운드 입력으로 8개의 워드 S, T, U, V, W, X, Y, Z에서 W는 32라운드에서 A를 적용하고 아무런 변화 없이 33라운드에서 A`를 적용한다. 이는 같은 W에 $\Sigma_1$ 연산을 수행한 후 덧셈과 뺄

셈을 수행하므로 완전한 상쇄가 발생한다. W, X, Y는 32라운드 함수 $f_{ch}()$의 입력(B)이고, W, X, Y의 값은 변화 없이 33라운드의 B`연산인 $f_{ch}()$를 수행하므로 같은 값의 덧셈과 뺄셈 수행으로 상쇄가 발생한다. 32비트 입력 S는 C, C`연산을 적용하며, 이는 32라운드에서 $\Sigma_0$ 연산 결과의 덧셈과 33라운드에서는 뺄셈을 수행하므로 서로 상쇄된다. 그리고 32라운드의 라운드 함수 $f_{maj}()$의 입력 S, T, U는 33라운드에서도 $f_{maj}()$의 입력으로 적용되어 같은 결과를 만들어 낸다. 이는 덧셈과 뺄셈의 결과로 역시 상쇄가 발생한다. 마지막으로 E와 E`의 경우 덧셈과 뺄셈의 연산자 우선순위가 같고 라운드 상수의 적용은 쉽게 소거 되므로 최종적인 32라운드 키 덧셈과 33라운드 키 뺄셈의 차이가 발생하게 된다. 이는 상쇄는 아니지만 SHACAL-2의 안전성에 심각한 영향을 미치는 취약점이다. 그래서 최종적으로 32라운드의 라운드 키와 33라운드 키의 덧셈과 뺄셈의 차이만 남게 된다.

이와 같이 대칭단 없이 연속적인 암호와 복호의 적용은 SHACAL-2의 암호에서 복호 알고리즘으로 바뀔 때 암호 마지막 라운드와 복호 첫 라운드가 서로 부분적인 상쇄로 각각의 라운드에서 적용된 라운드 키와 라운드 키의 차이만 남게 되고 이는 암호의 안전성에 심각한 취약점이 된다.

## 6.4 안전성 분석

SHACAL-2에 대한 안전성 분석은 차분-선형 특성을 이용한 32라운드 공격[77]과 연관 키를 이용한 42라운드 공격[78]이 있다. 대칭단을 적용한 SHACAL-2의 안전성 분석은 이 두 공격 방법을 기초로 하여 각각 35라운드, 40라운드 SHACAL-2의 안전성을 분석한다.

먼저 35라운드 연관 키 차분-비선형 공격은 14라운드 부정 차분 특성과 3라운드 비선형 관계식[79]을 이용한 17라운드 차분-비선형 특성을 기반으로 하고 있으며 대칭단을 적용한 SHACAL-2의 안전성 분석에서도 [79]의 3라운드 비선형 관계식을 이용한다. SHACAL-2의 3라운드 비선형 관계식을 간단히 요약하면, SHACAL-2의 r라운드의 마지막 32비트 워드 LSB($h_0^r$)는 비선형 함수 NF(Non-Linear Function: $A^{r+3}$, $B^{r+3}$, ..., $H^{r+3}$, $Y^r$, $Y^{r+1}$, $Y^{r+2}$, $K^r$, $K^{r+1}$, $K^{r+2}$)의 출력으로 표현 가능하고 이 비선형 함수를 간단히 $NF^{r+3}$으로 표시한다. 단 $0 \leq r \leq 61$. 그리고 $X_i^r$의 표기는 r라운드 32비트 X의 i번째 비트를 나타낸다.

$$h_0^r = c_0^{r+3} \oplus d_2^{r+3} \oplus d_{13}^{r+3} \oplus d_{22}^{r+3} \oplus (d_0^{r+3} \& (e_0^{r+3} \oplus t_{1,0}^{r+3})) \oplus (d_0^{r+3} \& (f_0^{r+3} \oplus t_{1,0}^{r+2})) \oplus ((e_0^{r+3} \oplus t_{1,0}^{r+3}) \& (f_0^{r+3} \oplus t_{1,0}^{r+2})) \oplus h_6^{r+3} \oplus h_{11}^{r+3} \oplus h_{25}^{r+3} \oplus (h_0^{r+3} \& h_0^{r+2}) \oplus ((\sim h_0^{r+3}) \& h_0^{r+1}) \oplus Y_0^r \oplus K_0^r, \quad (6-1)$$

(수식 6-1)에서 $h_0^{r+1}$, $t_{1,0}^{r+2}$, $h_0^{r+2}$, $t_{1,0}^{r+3}$은 다음과 같이 표현할 수 있으며 유도과정은 [79]에 자세히 설명하고 있고, 2.4절의 SHACAL-2의 암호와 복호 알고리즘으로 충분히 설명할 수 있다.

$$h_0^{r+1} = t_{1,0}^{r+2} \oplus g_6^{r+3} \oplus g_{11}^{r+3} \oplus g_{25}^{r+3} \oplus (g_0^{r+3} \& h_0^{r+3}) \oplus ((\sim g_0^{r+3}) \& h_0^{r+2}) \oplus Y_0^{r+1} \oplus K_0^{r+1},$$

$$t_{1,0}^{r+2} = b_0^{r+3} \oplus c_2^{r+3} \oplus c_{13}^{r+3} \oplus c_{22}^{r+3} \oplus (c_0^{r+3} \& d_0^{r+3}) \oplus (c_0^{r+3} \& (e_0^{r+3} \oplus t_{1,0}^{r+3})) \oplus (d_0^{r+3} \& (e_0^{r+3} \oplus t_{1,0}^{r+3})),$$

$$h_0^{r+2} = t_{1,0}^{r+3} \oplus f_6^{r+3} \oplus f_{11}^{r+3} \oplus f_{25}^{r+3} \oplus (f_0^{r+3} \& g_0^{r+3}) \oplus ((\sim f_0^{r+3}) \& h_0^{r+3}) \oplus Y_0^{r+2} \oplus K_0^{r+2},$$

$$t_{1,0}^{r+3} = a_0^{r+3} \oplus b_2^{r+3} \oplus b_{13}^{r+3} \oplus b_{22}^{r+3} \oplus (b_0^{r+3} \& c_0^{r+3}) \oplus (b_0^{r+3} \& d_0^{r+3}) \oplus (c_0^{r+3} \& d_0^{r+3}),$$

SHACAL-2의 35라운드 연관 키 차분-비선형 공격을 효과적으로 표현하기 위해 위의 수식들이 포함된 비선형 함수 $MNF^{r+3}$을 사용한다. 단 $MNF^{r+3} = NF^{r+3} \oplus Y_0^r \oplus K_0^r$이다. 연관 키 차분-선형 공격은 키 K에 대한 평문 P와 키 $K^*$에 대한 평문 $P^*$의 암호문 쌍을 필요로 한다. 여기서 K와 $K^*$는 서로 다른 키이지만 연관된 키이다. 차분 특성을 표현하기 위한 입력 차분은 $\Delta_P$, 출력 차분은 $\Delta_T$로 하고, 선형 근사식을 표현하기 위해 입력, 출력, 부분 키 비트 마스크는 각각 $\kappa_P$, $\kappa_T$, $\kappa_K$로 표기 한다.

블록 암호가 두 개의 부분 암호 $E^0_K$, $E^1_K$의 합성 형태 $E_K = E^0_K \bullet E^1_K$ 표현 될 때 $E^0$가 확률이 $p^* \leq 1$인 연관 키 차분 특성 $\Delta_P \rightarrow \Delta_T(Pr_P[E^0_K(P) \oplus E^0_{K*}(P^*) = \Delta_T | P \oplus P^* = \Delta_P] = p^*)$이 존재하고, $E^1$이 확률 1/2 + q 또는 편차가 q인 선형 근사식 $\kappa_P \rightarrow \kappa_T(\kappa_P \bullet P \oplus \kappa_T \bullet E^1(P) \oplus \kappa_K \bullet K = 0 = 1/2 + q$, K는 부분 키)가 존재 한다고 가정 한다. $P \oplus P^* = \Delta_P$를 만족하는 평문

쌍이라 할 때 P와 $P^*$가 확률 $p^*$의 차분 특성 $\Delta_P \rightarrow \Delta_T$를 만족하는 경우에 확률 100%의 1비트 방정식 $\kappa_P \cdot (E^0_K(P) \oplus E^0_{K*}(P^*)) = \alpha$를 얻을 수 있다.

만약 평문쌍 P와 $P^*$가 확률 $1 - p^*$의 차분 특성 $\Delta_P \rightarrow \Delta_T$를 만족하지 않는 경우에 1비트식 $\kappa_P \cdot (E^0_K(P) \oplus E^0_{K*}(P^*))$는 랜덤한 값을 갖는다. 이 두 경우를 고려한 다음과 같은 확률 $1/2 + p^*/2 (= p^* * 1 + (1 - p^*) * 1/2)$의 1비트 방정식을 얻을 수 있다.

$$\kappa_P \cdot (E^0_K(P) \oplus E^0_{K*}(P^*)) = \alpha, \quad (6-2)$$

그리고 $E^1$의 선형 근사식 입력 조건에 따라 다음과 같은 확률 $1/2 + q$ 또는 편차 q를 갖는 선형 근사식 두 개를 얻을 수 있다.

$$\kappa_P \cdot E^0_K(P) \oplus \kappa_T \cdot E^1_K(E^0_K(P)) \oplus \kappa_K \cdot K = 0, \quad (6-3)$$
$$\kappa_P \cdot E^0_{K*}(P^*) \oplus \kappa_T \cdot E^1_{K*}(E^0_{K*}(P^*)) \oplus \kappa_K \cdot K^* = 0, \quad (6-4)$$

(수식 6-3, 6-4)는 piling-up lemma를 사용하여 확률 $1/2 + 2p^*q^2 (= 1/2 + 2^{3-1} * p^*/2 * q^2)$를 갖는 선형 근사식을 다음과 같이 얻을 수 있다.

$$\kappa_T \cdot E^1_K(E^0_K(P)) \oplus \kappa_T \cdot E^1_{K*}(E^0_{K*}(P^*)) \oplus \kappa_K \cdot K \oplus \kappa_K \cdot K^* = \alpha, \quad (6-5)$$

즉 편차 $2p^*q^2$의 다음 선형 근사식을 얻을 수 있다.

$$\kappa_T \cdot E_K(P) \oplus \kappa_T \cdot E_{K*}(P^*) = 0, \quad (6-6)$$

따라서 (수식 6-6)을 이용한 선형 공격을 수행하기 위해 약 $(p^*)^2 * q^4$개의 연관 키 선택 평문쌍이 요구된다. 이와 같은 이론을 바탕으로 SHACAL-2의 28라운드 연관 키 차분-비선형 특성을 구성하는 방법을 소개한 후 이 특성을 이용한 35라운드 연관 키 차분-비선형 공격을 설명하고 대칭단을 적용한 SHACAL-2의 35라운드 연관 키 차분-비선형 공격의 안전성을 설명한다.

SHACAL-2의 키 스케줄링 알고리즘은 선형 궤한 쉬프트 레지스터(LFSR: Linear Feedback Shift Register)를 기반으로 수행한다. 그래서 처음 몇 라운드에 대한 차분 확산 효과가 적다. 즉 6라운드 키 $K^6$을 제외하고 모두가 같은 연관 키를 고려할 때 확장된 라운드 키 $K^{16}$, $K^{17}$, ..., $K^{20}$은 모두 같으며, $K^{21}$은

$e_{13,\sim}$의 차분 값을 가지며, $K^{22}$는 $e_{31}$의 차분 값을 갖는다. 여기서 $e_{13,\sim}$는 32비트 워드에서 13번째 비트는 1이고, 14 ~ 31번째 비트는 0또는 1의 임의의 값이며, 0 ~ 12번째 비트는 0의 값을 갖는다. 이러한 연관 키 차분 특성은 높은 확률을 갖는 25라운드 연관 키 부정 차분 특성을 구성할 수 있도록 한다. 즉 확률 $2^{-16}$으로 라운드 0 ~ 24($E^0$)에 대한 25라운드 부정 차분 특성 $\Delta_P \rightarrow \Delta_T$를 구성할 수 있다.

$$\Delta_P = (0, e_{31}, 0, 0, e_{6,20,25}, 0, 0, e_{9,13,19}),$$
$$\Delta_T = (?, ?, ?, e_{13,\sim}, ?, ?, ?, e_{13,\sim}),$$

$$a_{31} = c_{31},\ f_6 = g_6,\ f_{20} = g_{20},\ f_{25} = g_{25}, \qquad (6-7)$$

출력 차분 $\Delta_T$의 32비트 워드 중에서 ?는 알 수 없는 32비트 값이다. 평문쌍 P와 $P^*$는 (수식 6-7)과 같은 고정된 비트 값을 갖는다. 연관 키 부정 차분 특성 흐름에 대한 자세한 내용은 (표 6-2)에 나타냈다. (표 6-2)의 각각의 라운드 별 부정 차분 확률은 SHACAL-1의 차분 특성 분석과 같으며, 25라운드 연관 키 부정 차분 특성 확률은 $1/2 + 2^{-17}(= 2^{-16} + 1/2 * (1 - 2^{-16}))$을 만족하는 선형 근사 특성으로 바꾸어 적용할 수 있다. 다시 말해 평문쌍 P와 $P^*$가 차분 $(0, e_{31}, 0, 0, e_{6,20,25}, 0, 0, e_{9,13,19})$을 만족하고, (수식 6-7)의 고정된 값을 갖는다면 확률 $1/2 + 2^{-17}$으로 $h_0^{25} = h_0^{*25}$를 만족한다.

(표 6-2)의 25라운드 부정 차분 특성식을 앞서 설명한 3라운드 비선형 관계식을 연결함으로써 더욱 강력한 특성식을 얻을 수 있다. 주어진 평문쌍 P와 $P^*$는 $1/2 + 2^{-17}$의 근사 확률을 가지고 $h_0^{25} = h_0^{*25}$를 만족하며, 이에 3라운드 비선형 관계식을 연결하는 구조로 확률 $1/2 + 2^{-17}$으로 $NF^{28} = NF^{*28}$을 만족하는 28라운드 특성식을 구성할 수 있다. 이와 같은 의미로 선형 근사 편차 $2^{-17}$을 만족하는 식은 다음과 같다.

$$MNF^{28} = MNF^{*28}, \qquad (6-8)$$

표 6-2. SHACAL-2의 25라운드($E^0$) 연관 키 부정 차분 특성

| r | ΔA | ΔB | ΔC | ΔD | ΔE | ΔF | ΔG | ΔH | ΔK | 확률 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | $e_{31}$ | 0 | 0 | $e_{M1}$ | 0 | 0 | $e_{M2}$ | 0 | $2^{-3}$ |
| 1 | 0 | 0 | $e_{31}$ | 0 | 0 | $e_{M1}$ | 0 | 0 | 0 | $2^{-4}$ |
| 2 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | $e_{M1}$ | 0 | 0 | $2^{-3}$ |
| 3 | 0 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | $e_{M1}$ | 0 | $2^{-4}$ |
| 4 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | 0 | $2^{-1}$ |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | $2^{-1}$ |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | $e_{31}$ | 1 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $e_{13,\sim}$ | 1 |
| 22 | $e_{13,\sim}$ | 0 | 0 | 0 | $e_{13,\sim}$ | 0 | 0 | 0 | $e_{31}$ | 1 |
| 23 | ? | $e_{13,\sim}$ | 0 | 0 | ? | $e_{13,\sim}$ | 0 | 0 | ? | 1 |
| 24 | ? | ? | $e_{13,\sim}$ | 0 | ? | ? | $e_{13,\sim}$ | 0 | ? | 1 |
| 25 | ? | ? | ? | $e_{13,\sim}$ | ? | ? | ? | $e_{13,\sim}$ | | |

※ M1 = {6, 20, 25}, M2 = {9, 13, 19}

1/2 + $2^{-17}$을 만족하는 (수식 6-8)과 같은 28라운드 연관 키 차분-비선형 특성을 통해 35라운드 SHACAL-2의 연관 키를 찾는 방법은 다음과 같다.

① 차분 $\Delta_P$를 만족하고 8비트 고정된 값을 갖는 $2^{39}$개의 평문쌍($P_{i,j}$, $P^*_{i,j}$), i = 0, 1, ..., 4, j = 0, 1, ..., $2^{39}-1$로 구성된 5개의 집합을 선택한다. 여기서 $P_{i,j}$는 키 K를 사용하여 암호화 하며, $P^*_{i,j}$는 키 K와 차분 (0,0,0,0,0,0,$e_{31}$,0,0,0,0,0,0,0,0,0)을 갖는 키 $K^*$를 사용하여 암호화 한다. 5개의 평문쌍 집합에 각각 대응되는 암호문쌍 집합은 ($C_{i,j}$, $C^*_{i,j}$)가 된다.

② 207비트 부분 키쌍(Kr, $Kr^*$)을 추측한다. 부분 키 Kr은 $K^{34}$, $K^{33}$, $K^{32}$, $K^{31}$, $K_0^{30}$, $K_1^{30}$, ..., $K_{25}^{30}$, $K_0^{29}$, $K_1^{29}$, ..., $K_{25}^{29}$, $K_0^{28}$, $K_1^{28}$, ..., $K_{24}^{28}$, $K_0^{27}$, $K_0^{26}$이고, 다른 부분 키 $Kr^*$는 $K^{*34}$, $K^{*33}$, $K^{*32}$, $K^{*31}$, $K_0^{*30}$, $K_1^{*30}$, ..., $K_{25}^{*30}$, $K_0^{*29}$, $K_1^{*29}$, ..., $K_{25}^{*29}$, $K_0^{*28}$, $K_1^{*28}$, ..., $K_{24}^{*28}$, $K_0^{*27}$, $K_0^{*26}$이다.

③ For i = 0, 1, ..., 4까지 다음을 수행한다.
추측한 Kr의 모든 $2^{39}$개의 암호문 $C_{i,j}$를 부분적으로(28라운드까지) 복호화 하여 (수식 6-8)과 비교한다. 그리고 $Kr^*$를 사용한 $2^{39}$개의 $C^*_{i,j}$ 역시 부분 복호화를 하여 (수식 6-8)과 비교한다. (수식 6-8)과 비교한 암호문쌍의 개수가 $2^{38} - 2^{21.6}$보다 크고 $2^{38} + 2^{21.6}$보다 작으며 단계 ②로 돌아간다.

④ 단계 ③을 통과한 부분 키 Kr에 대한 나머지 305비트 키에 대한 전수조사를 수행한다. 이 과정을 통과한다면 512비트 키 K`를 35라운드 SHACAL-2의 마스터 키로 K` ⊕ (0,0,0,0,0,0,$e_{31}$,0,0,0,0,0,0,0,0,0)를 연관된 512비트 키로 출력한다. 그렇지 않다면 단계 ②로 돌아간다. 207비트 부분 키쌍(Kr, $Kr^*$)을 모두 테스트 했지만 K`가 출력되지 않는 다면 전체적인 이 과정의 출력값 없이 종료한다.

이와 같은 연관 키 차분-비선형 공격의 계산 복잡도(Complexity)는 단계 ①에서 $2^{42.32}$($\approx 5 * 2 * 2^{39}$)번의 35라운드 SHACAL-2의 암호화 연산이 필요하고 단계 ③에서 수행하는 i에 대한 부분 키쌍(Kr, $Kr^*$)이 옳은 키 쌍인지 계산하고 그 비율을 조사해야 한다. SHACAL-2의 비선형 함수들이 랜덤 치환 함수인 경우 옳은 키 쌍과 그렇지 않은 키 쌍이 같은 비율 1/2로 (수식 6-8)을 만족할 것이다. 그러므로 (수식 6-8)을 만족하는 평문 쌍의 개수는 이항 분포(Binomial Distribution) $B(2^{39}, 1/2)$를 따른다. 이항 분포는 바로 정규 분포(Normal Distribution)로 전환되며, $N(2^{38}, 2^{37})$이 된다. 따라서 $Z = (X - 2^{38}) / 2^{37}$의 정규 분포 $N(0, 1^2)$이고, $Pr[X \geq 2^{38} + 2^{21.6}$ or $X \leq 2^{38} - 2^{21.6}] = Pr[Z \geq 8.5742$ or $Z \leq -8.7542] \approx 2^{-53.3}$이 된다. 각각의 i에 대한 단계 ③을 통과한 부분 키 쌍의 비율이 약 $2^{-53.3}$이며, 단계 ③에서 통과된 i번째 부분 키 쌍의 개수는 약 $(2^{207})^2 * 2^{-53.3*(i+1)}$이 된다. 그러므로 단계 ③의 복잡도는

$2^{450.6}(\approx \sum_{i=0}^{4} 2^{39} \times 2 \times (2^{207})^2 \times 2^{-53.3 \times i} \times (7/35) \times (1/2))$번 35라운드 SHACAL-2의 암호화 수행이 요구되고, 단계 ③을 통과한 부분 키 쌍의 개수는 약 $2^{146.7}(\approx (2^{207})^2 * 2^{-53.3*5} * (1/2))$이기 때문에 단계 ④의 복잡도는 약 $2^{449.2}(\approx 2^{146.7} * 2^{305} * (7/35))$의 35라운드 SHACAL-2 암호 수행이 요구된다. 따라서 최종적인 35라운드 SHACAL-2의 연관키 차분-비선형 공격의 복잡도는 약 $2^{451.1}(\approx 2^{449.2} + 2^{450.6})$이 된다.

이와 같은 공격 과정에서 32라운드 후 대칭단이 적용되어 35라운드 연관 키 차분-비선형 공격은 단계 ①과 같은 평문 쌍과 대응되는 암호문 쌍을 얻는다. 그리고 단계 ②와 같이 부분 키쌍(Kr, $Kr^*$)를 추측하는 과정은 먼저 부분 키 Kr은 대칭단에서 적용되는 라운드 키 8개 256비트($K^{33}$, $K^{34}$, $K^{35}$, $K^{36}$, $K^{37}$, $K^{38}$, $K^{39}$, $K^{40}$)를 포함한 $K^{42}$, $K^{41}$, $K^{32}$, $K^{31}$, $K_0^{30}$, $K_1^{30}$, ..., $K_{25}^{30}$, $K_0^{29}$, $K_1^{29}$, ..., $K_{25}^{29}$, $K_0^{28}$, $K_1^{28}$, ..., $K_{24}^{28}$, $K_0^{27}$, $K_0^{26}$의 463비트를 추측하고, 이에 대응되는 $Kr^*$ 역시 대칭단에서 적용되는 라운드 키 8개 256비트($K^{*33}$, $K^{*34}$, $K^{*35}$, $K^{*36}$, $K^{*37}$, $K^{*38}$, $K^{*39}$, $K^{*40}$)를 포함한 $K^{*42}$, $K^{*41}$, $K^{*32}$, $K^{*31}$, $K_0^{*30}$, $K_1^{*30}$, ..., $K_{25}^{*30}$, $K_0^{*29}$, $K_1^{*29}$, ..., $K_{25}^{*29}$, $K_0^{*28}$, $K_1^{*28}$, ..., $K_{24}^{*28}$, $K_0^{*27}$, $K_0^{*26}$의 463비트를 추측한다. 그리고 단계 ③에서 28라운드까지 부분 복호화는 35 ~ 33라운드 까지는 암호화 과정을 수행하고 대칭단을 적용한 다음 32 ~ 28라운드까지는 복호화 과정을 적용 한다 이는 (그림 6-1) 알고리즘의 복호화 과정이다. 단계 ④는 앞서 설명한 연관 키 차분-비선형 공격과정과 같으며, 단지 대칭단에서 적용된 256비트의 라운드 키를 포함한 768비트를 검사하고 출력한다. 그래서 최종적인 35라운드 대칭단이 적용된 연관 키 차분-비선형 공격 복잡도는 약 $2^{547.1}(\approx 2^{451.1} * 2^{96})$이 된다. 여기서 $2^{96}$은 대칭단에서 적용된 라운드 키와 논리 연산의 분석 복잡도이다.

다음은 대칭단을 적용한 40라운드 연관 키 Rectangle 공격으로 이 공격은 연관 키와 Rectangle 공격을 결합한 것으로 연속된 2개의 차분 특성을 이용하는 공격이다. 여기서 2개의 차분 특성은 연관 키 차분과 다른 하나는 차분 특성을 이용한다. 연관 키 차분-비선형 공격은 다음과 같은 3단계로 공격을 진행한다.

① 2개의 랜덤한 n비트 평문 P와 P` 그리고, 이 평문에 각각 대응되는 차분 값이 α인 $P^* = P \oplus \alpha$와 $P`^* = P` \oplus \alpha$인 4개로 묶인 평문(P, $P^*$,

P`, P`*)를 선택한다.

② 선택 평문 공격 시나리오에 의해 4개의 평문에 각각 대응되는 4개의 암호문 $C = E_K(P)$, $C^* = E_{K*}(P^*)$, $C` = E_{K`}(P`)$, $C`^* = E_{K`*}(P`^*)$을 얻는다. 여기서 $K^* = K \oplus \Delta K$, $K` = K \oplus \Delta K`$, $K`^* = K \oplus \Delta K \oplus \Delta K`$ (즉 $K \oplus K^* = K` \oplus K`^* = \Delta K$이며, $K \oplus K` = K^* \oplus K`^* = \Delta K`$)이다. 그리고 $\Delta K$와 $\Delta K`$는 암호 공격자가 선택한 고정된 연관 키 차분이다.

③ 단계 ②에서 얻어진 4개의 암호문을 $C \oplus C` = C^* \oplus C`^* = \delta$ 또는 $C \oplus C`^* = C^* \oplus C` = \delta$인지 판별한다.

이와 같은 연관 키 차분-비선형 공격으로부터 연관 키 Rectangle 특성(distinguisher)은 다음과 같은 4가지 차분 조건을 만족하는 4개로 묶인 평문 쌍(P, P*, P`, P`*)을 판별하는 것이다.

차분 조건 1: $P \oplus P^* = P` \oplus P`^* = \alpha$,
차분 조건 2: $I \oplus I^* = I` \oplus I`^* = \beta$, (임의의 β에 대해서)
차분 조건 3: $I \oplus I` = \Upsilon$(또는 $I \oplus I`^* = \Upsilon$), (임의의 Υ에 대해서)
차분 조건 4: $C \oplus C` = C^* \oplus C`^* = \delta$(또는 $C \oplus C`^* = C^* \oplus C` = \delta$),

4개의 차분 조건에서 $I = E^0_K(P)$, $I^* = E^0_{K*}(P^*)$, $I` = E^0_{K`}(P`)$, $I`^* = E^0_{K`*}(P`^*)$이고, 4개의 차분 값 중에서 α와 $\delta$는 공격자가 의도한 차분 값이며, β와 Υ는 임의의 차분 값이다. 그리고 차분 조건 2와 3은 $I^* \oplus I`^* = \Upsilon$ (또는 $I^* \oplus I` = \Upsilon$)가 확률 100%라는 것을 의미한다. 이와 같은 4개의 차분 조건을 만족하는 각각의 4개로 묶인 평문 쌍(P, P*, P`, P`*)을 right quartet라 부른다.

차분 α를 만족하는 m개의 평문 쌍(키 K를 사용하는 평문과 키 K*를 사용하는 평문과 각각의 키 차분 값은 ΔK)이 있다고 가정하자. 부분 암호 $E^0$가 확률 $p^*$인 연관 키 차분 특성 $\alpha \rightarrow \beta$가 만족한다. 즉 $Pr_{P,K}[E^0_K(P) \oplus E^0_{K*}(P^*) = \beta \mid P \oplus P^* = \alpha] = p^*$이면, 평문 쌍은 약 $mp^*$개 이며, 차분 조건 1과 2에 의해 $mp^*$개의 평문 쌍은 약 $(mp^*)^2 / 2$개의 quartet를 생성한다. 블록 암호의 암

호화 과정에서 중간 과정의 값이 랜덤하게 분포 한다고 가정하면, 확률 $2^{-n}$으로 차분 I ⊕ I` = ϒ와 I ⊕ I`* = ϒ를 얻는다. 이 경우에 right quartet가 되기 위해 I와 I`는 부분 암호 $E^1$의 확률 q를 갖는 연관 키 차분 특성 ϒ → δ를 만족해야 한다. 또 I와 I`* 역시 부분 암호 $E^1$의 확률 q*를 갖는 연관 키 차분 특성 ϒ → δ를 만족해야 한다.($Pr_{P,K}[E^1{}_K(P) \oplus E^1{}_{K*}(P \oplus \Upsilon) = \delta] = q^*$) 따라서 차분 α를 만족하는 m개의 평문 쌍에 대한 right quartet의 평균 기댓값은 $\sum_{any\ \beta,\gamma} ((m\times p^*)^2/2)\times 2^{-n}\times(q^2+q^{*2}) = m^2\times 2^{-n-1}\times \widehat{p^{*2}}\times(\widehat{q^2}+\widehat{q^{*2}})$이다. 여기서, $\widehat{p^*} = \sqrt{\sum_\beta p^{*2}}$, $\hat{q} = \sqrt{\sum_\gamma q^2}$, $\widehat{q^{*2}} = \sqrt{\sum_\gamma q^{*2}}$ 이다.

한편 랜덤한 분포를 보이는 암호문에서 right quartet의 기댓값은 약 $m^2 * 2^{-2n}$($\approx {}_mC_2 * 2 * 2^{-2n}$)이며, 확률 $2^{-n}$으로 차분 δ를 만족하는 암호문 2쌍(C, C`), (C*, C`*) (또는 (C, C`*), (C*, C`))이 가능한 quartet는 ${}_mC_2 * 2$ 이다. 그러므로 만약 $\widehat{p^*}\times((1/2)\times(\widehat{q^2}+\widehat{q^{*2}}))^{1/2} > 2^{-n/2}$을 만족하고, m이 충분히 크다면 연관 키 Rectangle 특성을 갖는 E와 랜덤한 분포를 갖는 암호문과 구별이 가능하다.

이와 같은 이론을 바탕으로 40라운드 SHACAL-2에 대한 연관 키 Rectangle 공격을 먼저 설명하고, 이어서 대칭단을 적용한 40라운드 SHACAL-2의 연관 키 Rectangle 공격을 설명한다. 그리고 SHACAL-2 알고리즘과 키 스케줄링 알고리즘의 특별한 성질(property)에 대해서 먼저 간단히 설명한다.

성질 1: 어떤 연속적인 4라운드(i ~ i+3라운드)의 차분 특성 흐름을 쉽게 얻을 수 있다. 만일 i라운드($\Delta A^i$, $\Delta B^i$, ..., $\Delta H^i$)의 차분 값을 알고 있으면, $\Delta B^{i+1} = \Delta A^i$, $\Delta C^{i+1} = \Delta B^i$, $\Delta D^{i+1} = \Delta C^i$, $\Delta F^{i+1} = \Delta E^i$, $\Delta G^{i+1} = \Delta F^i$, $\Delta H^{i+1} = \Delta G^i$,이고, $\Delta C^{i+2} = \Delta A^i$, $\Delta D^{i+2} = \Delta B^i$, $\Delta G^{i+2} = \Delta E^i$, $\Delta H^{i+2} = \Delta F^i$, 이며, $\Delta D^{i+3} = \Delta A^i$, $\Delta H^{i+3} = \Delta E^i$가 되는 것을 쉽게 알 수 있다.

성질 2: 서로 다르지만 0번째와 9번째 라운드 키의 차분 값은 $e_{31}$이고 나머지 16번째 라운드 키까지는 차분 값이 0인 연관된 2개의 키 K와 K*는 16 ~ 23번째 라운드 키 모두가 차분 값이 0을 갖는다. 이와 같은 특성은 다음과 같은 라운드 키에 대한 방정식을 확률

100%로 만족한다.

$$
\begin{aligned}
K^{*16} &= \sigma_1(K^{*14}) + K^{*9} + \sigma_0(K^{*1}) + K^{*0} \\
&= \sigma_1(K^{14}) + (K^{9} \oplus e_{31}) + \sigma_0(K^{1}) + (K^{0} \oplus e_{31}) \\
&= \sigma_1(K^{14}) + K^{9} + \sigma_0(K^{1}) + K^{0} \\
&= K^{16},
\end{aligned}
$$

더욱이 연관된 키 K와 $K^*$의 24라운드 키 $K^{24} = L_0 + L_1$, $K^{*24} = L_0 + (L_1 \oplus e_{13,24,28})$을 쉽게 얻을 수 있음을 알 수 있다. 여기서 $L_0 = \sigma_1(K^{22}) + K^{17} + K^{8}$, $L_1 = \sigma_1(K^{9})$이다. 이와 같은 성질과 참고문헌[70], (표 5-2)를 기반으로 24라운드 연관 키 차분 특성을 구성할 수 있다. (표 6-3)은 1 ~ 24라운드 SHACAL-2의 연관 키 차분 특성을 설명한 것으로, 확률 $2^{-38}$으로 연관 키 차분 특성 $\alpha \rightarrow \beta$: $(0, 0, e_{6,9,18,20,25,29}, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}) \rightarrow (e_{13,24,28}, 0, 0, 0, e_{13,24,28}, 0, 0, 0)$을 구성할 수 있다. 이와 같은 연관 키 차분 특성을 구성하기 위해 1라운드 2개의 입력 ($A^1$, $B^1$, $C^1$, $D^1$, $E^1$, $F^1$, $G^1$, $H^1$)과 ($A^{*1}$, $B^{*1}$, $C^{*1}$, $D^{*1}$, $E^{*1}$, $F^{*1}$, $G^{*1}$, $H^{*1}$)의 차분 값 α를 유지하기 위해 12비트의 고정된 조건식이 필요하다.

$$
\begin{aligned}
&a^1_6 = b^1_6, \quad a^1_9 = b^1_9, \quad a^1_{18} = b^1_{18}, \quad a^1_{20} = b^1_{20}, \quad a^1_{25} = b^1_{25}, \\
&a^1_{29} = b^1_{29}, \quad a^1_{31} = b^1_{31}, \quad e^1_9 = 0, \quad e^1_{13} = 0, \quad e^1_{18} = 1, \\
&e^1_{19} = 0, \quad e^1_{29} = 1, \qquad\qquad (6\text{-}9)
\end{aligned}
$$

(수식 6-9)에서 $a^1_i$, $b^1_i$, $e^1_i$는 각각 $A^1$, $B^1$, $E^1$의 i번째 비트이다. 만일 1라운드에 (수식 6-9)와 차분 값 α를 만족하고, 2라운드에서 (수식 6-9)의 $a^1_{31} = b^1_{31}$의 조건식이 적용되면 1라운드와 2라운드 사이에 적용되는 함수 $f_{ch()}$, $f_{maj()}$에 의해 1라운드 차분 특성 확률이 1이 된다.

다음은 연관 키 Rectangle 특성을 구성하기 위한 10라운드 차분 특성을 $2^{-65}$의 확률로 25 ~ 34라운드에 적용한다. (표 6-4)는 SHACAL-2의 25 ~ 34라운드 차분 특성을 설명하는 표이다. 차분 특성 $\Upsilon \rightarrow \delta$: $(e_{31}, e_{31}, e_{6,9,18,20,25,29}, 0, 0, e_{9,13,19}, e_{18,29,31}, 0) \rightarrow (e_{6,9,18,20,25,29}, e_{31}, 0, 0, e_{6,20,25}, e_{31}, 0, 0)$으로 구성된다.

표 6-3. SHACAL-2의 1 ~ 24라운드($E^0$) 연관 키 차분 특성

| r | ΔA | ΔB | ΔC | ΔD | ΔE | ΔF | ΔG | ΔH | ΔK | 확률 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | $e_{M1}$ | $e_{31}$ | 0 | $e_{M3}$ | $e_{M4}$ | $e_{31}$ | . | $e_{31}$ | . |
| 1 | 0 | 0 | $e_{M1}$ | $e_{31}$ | 0 | $e_{M3}$ | $e_{M4}$ | $e_{31}$ | 0 | 1 |
| 2 | $e_{31}$ | 0 | 0 | $e_{M1}$ | 0 | 0 | $e_{M3}$ | $e_{M4}$ | 0 | $2^{-12}$ |
| 3 | 0 | $e_{31}$ | 0 | 0 | $e_{M2}$ | 0 | 0 | $e_{M3}$ | 0 | $2^{-7}$ |
| 4 | 0 | 0 | $e_{31}$ | 0 | 0 | $e_{M2}$ | 0 | 0 | 0 | $2^{-4}$ |
| 5 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | $e_{M2}$ | 0 | 0 | $2^{-3}$ |
| 6 | 0 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | $e_{M2}$ | 0 | $2^{-4}$ |
| 7 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | 0 | $2^{-1}$ |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | $2^{-1}$ |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | $e_{31}$ | 1 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | . | $2^{-6}$ |
| 25 | $e_{M5}$ | 0 | 0 | 0 | $e_{M5}$ | 0 | 0 | 0 | . | . |

※ M1 = {6, 9, 18, 20, 25, 29}, M2 = {6, 20, 25},
M3 = {9, 13, 19}, M4 = {18, 29}, M5 = {13, 24, 28}

right quartet의 평균 기댓값을 계산하기 위한 $\widehat{p^*}$(대응 되는 $\hat{q}$)는 부분 암호 $E^0$의 차분 값 α를 갖는 모든 연관 키 차분 특성 확률의 제곱의 합(대응 되는 부분 암호 $E^1$의 차분 값 $\delta$를 갖는 차분 특성 확률의 제곱의 합)의 계산 복잡도가 필요하다. 이는 모든 특성 경로를 고려하는 것은 실현 불가능하기 때문에 다양한 입력 차분을 고려한 (표 6-3)의 연관 키 차분 특성의 마지막 라운드와 (표 6-4)의 차분 특성의 첫 라운드를 연결 했다. 결론적으로 $\widehat{p^*} = 2^{-37}$ 이며 $\hat{q} = 2^{-63.4}$이다[74]. 그러므로 34라운드 SHACAL-2의 연관 키

Rectangle 특성 확률의 하한(lower bound)값은 $2^{-456.8}$(= $(2^{-37} * 2^{-63.4})^2 * 2^{-256}$)이 된다.

표 6-4. SHACAL-2의 25 ~ 34라운드($E^1$) 차분 특성

| r | ΔA | ΔB | ΔC | ΔD | ΔE | ΔF | ΔG | ΔH | 확률 |
|---|---|---|---|---|---|---|---|---|---|
| 25 | $e_{31}$ | $e_{31}$ | $e_{M2}$ | 0 | 0 | $e_{M4}$ | $e_{M5}$ | 0 | $2^{-15}$ |
| 26 | $e_{31}$ | $e_{31}$ | $e_{31}$ | $e_{M2}$ | 0 | 0 | $e_{M4}$ | $e_{M5}$ | $2^{-12}$ |
| 27 | 0 | $e_{31}$ | $e_{31}$ | $e_{31}$ | $e_{M3}$ | 0 | 0 | $e_{M4}$ | $2^{-7}$ |
| 28 | 0 | 0 | $e_{31}$ | $e_{31}$ | $e_{31}$ | $e_{M3}$ | 0 | 0 | $2^{-8}$ |
| 29 | 0 | 0 | 0 | $e_{31}$ | $e_{31}$ | $e_{31}$ | $e_{M3}$ | 0 | $2^{-7}$ |
| 30 | 0 | 0 | 0 | 0 | $e_{31}$ | $e_{31}$ | $e_{31}$ | $e_{M3}$ | $2^{-4}$ |
| 31 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | $e_{31}$ | $e_{31}$ | 1 |
| 32 | 0 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | $e_{31}$ | $2^{-1}$ |
| 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $e_{31}$ | 1 |
| 34 | $e_{31}$ | 0 | 0 | 0 | $e_{31}$ | 0 | 0 | 0 | $2^{-11}$ |
| 35 | $e_{M1}$ | $e_{31}$ | 0 | 0 | $e_{M3}$ | $e_{31}$ | 0 | 0 | . |

※ M1 = {6, 9, 18, 20, 25, 29}, M2 = {6, 9, 18, 20, 25,29, 31}, M3 = {6, 20, 25}, M4 = {9, 13, 19}, M5 = {18, 29, 31}

(그림 6-3)는 대칭단을 적용한 SHACAL-2의 40라운드 연관 키 Rectangle 공격을 그림으로 표현한 것으로 차분 값이 ($e_{31}$, 0, 0, 0, 0, 0, 0, 0, 0, $e_{31}$, 0, 0, 0, 0, 0, 0)인 연관된 키 K와 K*를 사용하여 40라운드 SHACAL-2 암호화를 수행한 것으로 가정한다. 그리고 34라운드 연관 키 Rectangle 특성을 이용해 0, 35, 36, 37, 38, 39라운드의 작은 부분 후보 키를 얻을 수 있으며, 나머지 부분 후보 키는 전수조사를 통해 전체 512비트 연관된 K와 K*를 찾는다. 이와 같은 연관 키 Rectangle 특성을 적용하기 위해 1라운드에 (수식 6-9)와 차분 값 α를 만족하는 충분히 많은 평문 쌍이 필요하다. 상세한 40라운드 연관 키 Rectangle 공격은 다음과 같은 절차에 의해서 진행된다.

① 32비트 D와 H는 모든 가능한 값 0 ~ $2^{64}-1$을 가지는 $2^{64}$개의 묶음이고, 나머지 A, B, C, E, F, G는 192비트 고정된 값을 가지는 $2^{178.4}$개의

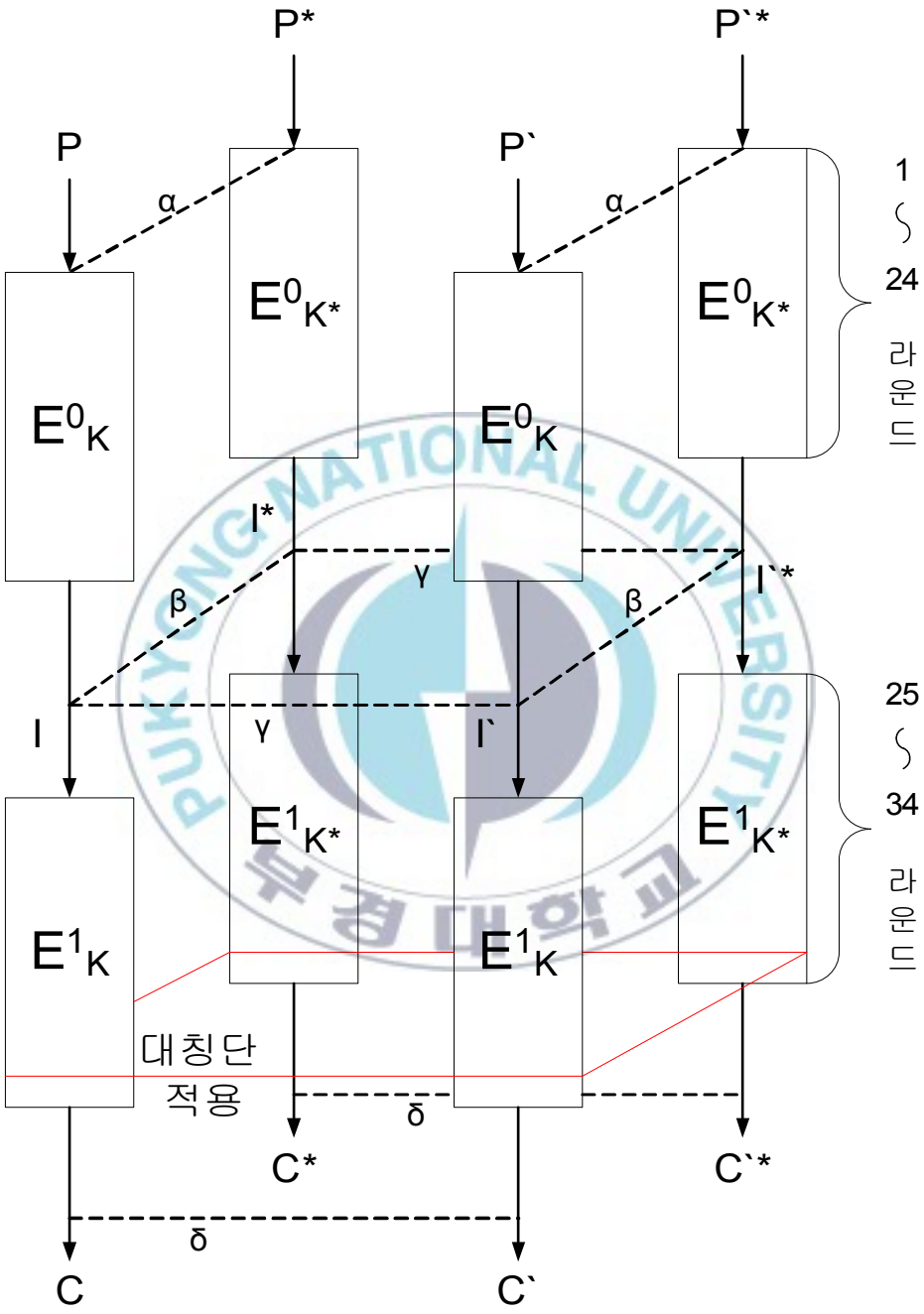



그림 6-3. 대칭단을 적용한 SHACAL-2의 40라운드 연관 키 Rectangle 공격

집합 구조로 이루어진 $P_{i,j}$(i = 1, 2, ..., $2^{178.4}$, j = 1, 2, ..., $2^{64}$)를 선택한다. 선택 평문 공격 시나리오에 의해 $P_{i,j}$ 각각의 평문에 대응 되는 키 K에 의해 암호화된 암호문 $C_{i,j}$를 얻는다.

② 단계 ①에서 선택된 $2^{178.4}$개의 평문 $P_{i,j}$에 각각 대응 되는 차분 값(0, $e_{6,9,18,20,25,29}$, $e_{31}$, 0, $e_{9,13,19}$, $e_{18,29}$, $e_{31}$, 0)인 $P^*_{i,j}$를 계산한다. 선택 평문 공격 시나리오에 의해 각각의 평문 $P^*_{i,j}$에 대응 되는 키 $K^*$에 의해 암호화된 암호문 $C^*_{i,j}$를 얻는다.

③ 0라운드 32비트 라운드 키 $K^0$을 추측하고 이에 대응 되는 연관 키 $K^{*0} = K^0 \oplus e_{31}$을 계산한다. 단계 ①에서 선택된 평문 $P_{i,j}$를 $K^0$으로 1라운드 암호화 한 후 그 암호문을 $x_{i,j}$라 한다. $x_{i,j}$와 (수식 6-9)를 만족하고, $x^*_{i,j} = x_{i,j} \oplus \alpha$를 계산한다. 여기서 $x^*_{i,j}$는 평문 $P^*_{i,j}$의 라운드 키 $K^{*0}$로 1라운드 암호화 한 암호문이다. 그리고 전체 평문 집합에서 1라운드 복호화 한 후 대응 되는 $x^*_{i,j}$의 평문 $P^*_{i,j}$만 수집한다.

④ 96비트 37, 38, 39라운드 라운드 키쌍(($K^{37}$, $K^{38}$, $K^{39}$), ($K^{*37}$, $K^{*38}$, $K^{*39}$))을 추측한다. 추측된 라운드 키($K^{37}$, $K^{38}$, $K^{39}$)로 37, 38, 39라운드 부분 복호화를 수행한 후 그 결과 값 $C^{37}_{i,j}$를 메모리 테이블에 저장한다. 같은 방법으로 추측된 라운드 키($K^{*37}$, $K^{*38}$, $K^{*39}$)로 37, 38, 39라운드 부분 복호화를 수행한 후 그 결과 값 $C^{*37}_{i,j}$를 메모리 테이블에 저장 한다. 그리고 $C^{37}_{i0,j0} \oplus C^{37}_{i1,j1}$와 $C^{*37}_{i0,j0} \oplus C^{*37}_{i1,j1}$이 $\delta^2$를 만족하는지 검사한다.($1 \le i_0 < i_1 \le 2^{178.4}$, $1 \le j_0, j_1 \le 2^{64}$ 그리고 $1 \le i_0 = i_1 \le 2^{178.4}$, $1 \le j_0 < j_1 \le 2^{64}$) 여기서 $\delta^2$는 2라운드 후 차분 $\delta$를 만족할 수 있는 모든 가능한 차분 집합이다. 이와 같은 조건을 만족하는 라운드 키($K^0$, $K^{37}$, $K^{38}$, $K^{39}$)를 저장한다.

⑤ 32비트 36라운드 라운드 키쌍($K^{36}$, $K^{*36}$)을 추측한다. 나머지 quartet($C^{37}_{i0,j0}$, $C^{37}_{i1,j1}$, $C^{*37}_{i0,j0}$, $C^{*37}_{i1,j1}$)에 대해 추측된 36라운드 키 $K^{36}$으로 $C^{37}_{i0,j0}$와 $C^{37}_{i1,j1}$을 1라운드 복호화를 수행한다. 같은 방법으로 추측된 36라운드 키 $K^{*36}$으로 $C^{*37}_{i0,j0}$와 $C^{*37}_{i1,j1}$을 1라운드 복호화를 수행한다. 그 결과 quartet는 ($C^{36}_{i0,j0}$, $C^{36}_{i1,j1}$, $C^{*36}_{i0,j0}$, $C^{*36}_{i1,j1}$)이다. 그리고 $C^{36}_{i0,j0} \oplus C^{36}_{i1,j1}$와 $C^{*36}_{i0,j0} \oplus C^{*36}_{i1,j1}$이 $\delta^1$를 만족하는지 검사한다. 여

기서 $\delta^1$은 1라운드 후 차분 $\delta$를 만족할 수 있는 모든 가능한 차분 집합이다. 이와 같은 조건을 만족하는 라운드 키($K^0$, $K^{36}$, $K^{37}$, $K^{38}$, $K^{39}$)를 저장한다.

⑥ 32비트 35라운드 라운드 키쌍($K^{35}$, $K^{*35}$)을 추측한다. 나머지 quartet($C^{36}_{i0,j0}$, $C^{36}_{i1,j1}$, $C^{*36}_{i0,j0}$, $C^{*36}_{i1,j1}$)에 대해 추측된 35라운드 키 $K^{35}$로 $C^{36}_{i0,j0}$와 $C^{36}_{i1,j1}$을 1라운드 복호화를 수행한다. 같은 방법으로 추측된 35라운드 키 $K^{*35}$로 $C^{*36}_{i0,j0}$와 $C^{*36}_{i1,j1}$을 1라운드 복호화를 수행한다. 그 결과 quartet는 ($C^{35}_{i0,j0}$, $C^{35}_{i1,j1}$, $C^{*35}_{i0,j0}$, $C^{*35}_{i1,j1}$)이다. 그리고 $C^{35}_{i0,j0} \oplus C^{35}_{i1,j1}$와 $C^{*35}_{i0,j0} \oplus C^{*35}_{i1,j1}$이 $\delta$($e_{6,9,18,20,25,29}$, $e_{31}$, 0, 0, $e_{6,20,25}$, $e_{31}$, 0, 0)를 만족하는 5개 이상의 quartet가 존재하면 라운드 키($K^0$, $K^{35}$, $K^{36}$, $K^{37}$, $K^{38}$, $K^{39}$)를 저장한다. 그렇지 않으면 단계 ⑥의 다른 키 쌍을 추측한다. (만일 단계 ⑥의 모든 키쌍을 검사 후 만족하는 키가 없으면 단계 ⑤의 36라운드 다른 키를 추측하고 단계 ⑤의 모든 키 쌍을 검사 후 만족하는 키가 없으면 단계 ④의 다른 37, 38, 39라운드 키를 추측하고, 단계 ④의 모든 37, 38, 39라운드 키를 검사 후 만족하는 키가 없으면 단계 ③으로가 다른 0라운드 키를 추측한 후 처음부터 다시 시작한다.)

⑦ 이와 같은 조건을 만족하는 라운드 키($K^0$, $K^{35}$, $K^{36}$, $K^{37}$, $K^{38}$, $K^{39}$)에 대한 나머지 320비트에 대한 전수조사를 실시한다. 이 과정을 통과하면 512비트 키 $K^*$를 40라운드 SHACAL-2의 마스터 키로 $K^* \oplus$ ($e_{31}$, 0, 0, 0, 0, 0, 0, 0, $e_{31}$, 0, 0, 0, 0, 0, 0)를 적용하여 연관된 512비트 마스터 키를 출력한다.

이와 같은 알고리즘을 적용하는 연관 키 Rectangle 공격의 계산 복잡도는 우선 ①, ②단계에서 40라운드 SHACAL-2의 암호화에 $2^{(178.4 + 64)} * 2 = 2^{243.4}$의 계산 복잡도를 요구하고, 단계 ③은 (수식 6-9)를 만족하는 12비트 검사를 포함한 계산 복잡도는 $(2^{242.4} + 2^{230.4}) * 2^{32} * (1/40) * (1/2) = 2^{268.1}$이 된다. 단계 ④에서 추측된 quartet 검사의 계산 복잡도는 약 $2^{230.4 * 2} / 2 = 2^{459.8}$이며, 3라운드 부분 복호화는 약 $2^{231.4} * 2^{192} * (3/40) * (1/2) = 2^{418.6}$이 된다. 그리고 차분 값 $\delta$로부터 32비트 워드 C, D, G, H의 차분을 결정하고 2라운드 후의 모든 가능한 차분 값 $\delta^2$을 검사한다. $\delta^2$로부터 32비트 워드 B는

약 $2^{28}$이고 F는 $2^{17}$이다. 그래서 $\delta^2$의 가능한 차분 값은 $2^{64+28+17} = 2^{109}$이 된다. 단계 ④를 통과할 예상된 quartet 수는 약 $2^{459.8} * 2^{(-256+109) * 2} = 2^{165.8}$이다. 단계 ⑤는 $2^{287}$번의 추측이 필요하며, 최종적인 복잡도는 $2^{165.8} * 4 * 2^{287} * (1/40) = 2^{449.4}$이며, 추가적으로 64비트와 45비트 $\delta^1$, $\delta$의 검사는 각각 $2^{165.8} * 2^{-64*2} = 2^{37.8}$, $2^{37.8} * 2^{-45*2} = 2^{-52.2}$가 된다. 단계 ⑥은 약 $2^{37.8} * 4 * 2^{352} * (1/40) * (1/2) = 2^{385.4}$이 된다. 그래서 최종적으로 SHACAL-2에 대한 연관 키 Rectangle 공격은 약 $(2^{243.4} + 2^{243.4} + 2^{268.1} + 2^{418.6} + 2^{449.4} + 2^{385.4}) = 2^{449.4}$의 40라운드 SHACAL-2의 암호화 시간이 요구된다.

대칭단을 적용한 40라운드 SHACAL-2 연관 키 Rectangle 공격은 32라운드 후 대칭단을 적용하고 33, 34, 35라운드는 복호 알고리즘을 적용한다. 이때 SHACAL-2의 복호 알고리즘은 암호 분석 특성이 암호 알고리즘과 동일하므로 (표 6-4)의 차분 특성을 만족할 수 있다. 그러나 32라운드 후 적용된 대칭단은 라운드 키와 논리 연산으로 33라운드 차분 특성 흐름을 단절 시킬 수 있으며, 이와 같은 이유로 가장 보수적인 분석 평가를 해서 32라운드 G, H의 차분 특성을 유지하면서 33라운드로 진행할 수 있는 64비트에 대한 논리 연산 분석이 필요하다 그래서 최종적인 대칭단을 적용한 40라운드 SHACAL-2의 연관 키 Rectangle 공격 복잡도는 $2^{449.4} * 2^{24} = 2^{473.4}$이 된다.

## 6.5 실행 결과 분석

대칭단을 적용한 SHACAL-2 알고리즘은 CBC(Cipher Block Chaining) 운영 모드를 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 Vista, Intal core2 Duo 2.26Ghz, 2.27Ghz, 2GB RAM의 환경에서 기존의 SHACAL-2 알고리즘과 대칭단을 적용한 SHACAL-2 알고리즘의 수행 시간을 테스트했다. 결과는 (표 6-5)와 같으며, 대칭단을 적용한 알고리즘들의 암호와 복호 수행 시간이 약 9.6% 증가하는 것으로 나타났다. 그러나 이정도의 속도 증가는 대칭단에서 적용한 간단한 논리 연산들이 전체적인 암호와 복호 알고리즘 수행에 크게 영향을 미치는 않는 것으로 판단된다.

표 6-5. SHACAL-2와 대칭단을 적용한 SHACAL-2의 수행 시간 비교

| 암호 알고리즘 | 암호 | 복호 |
|---|---|---|
| SHACAL-2 | 11.516<br>(100%) | 11.295<br>(100%) |
| 대칭단을<br>적용한<br>SHACAL-2 | 12.657<br>(109.9%) | 12.345<br>(109.3%) |

(표 6-6)는 SHACAL-2의 라운드 함수에서 사용된 주요 연산자로 64라운드를 적용한 연산 회수이고, 대칭단은 256비트 대칭단에 적용된 주요 연산자 회수이다.

표 6-6. SHACAL-2와 대칭단의 주요 연산자의 비교

| | SHACAL-2 | 대칭단 |
|---|---|---|
| XOR(32) | 448 | 8 |
| rotate | 384 | 8 |
| AND(32) | 320 | - |
| AND/OR(8) | - | 32 |
| NOT(32) | 64 | - |
| 덧셈/뺄셈(32) | 448 | - |

## 6.6 결론

유럽의 NESSIE 프로젝트에 256비트 블록 암호 알고리즘으로 제안된 SHACAL-2는 암호와 복호가 서로 다른 64라운드 블록 암호 알고리즘으로 비선형 변환으로 S박스를 사용하지 않고 간단한 논리 연산으로 구성된 비선형 라운드 함수를 수행하는 특성을 가지고 있다. 이와 같은 SHACAL-2의 32라운드까지 암호 알고리즘을 적용하고 나머지 32라운드는 복호 알고리즘을 적용한 후 암호와 복호 알고리즘 사이에 대칭단을 적용하여 암호와 복호 알고리즘이 같게 SHACAL-2를 구성하였다.

간단한 논리 연산으로 구성된 대칭단을 적용한 SHACAL-2 알고리즘은 기존의 SHACAL-2 알고리즘 보다 연관 키 차분, 연관 키 Rectangle 공격에 대해서 향상된 안전성을 보여주고 있으며, 대칭단을 적용한 SHACAL-2의 새로운 구성은 암호와 복호가 동일하기 때문에 하드웨어 구현을 간단히 할 수 있는 장점을 가진다.

대칭단을 적용한 SHACAL-2와 기존의 SHACAL-2를 소프트웨어로 구현한 후 암호와 복호 수행 시간 테스트한 결과 대칭단을 적용한 SHACAL-2가 약간의 속도 증가를 보여 주고 있는데 이는 전체적인 수행 속도를 볼 때 미세한 증가로 대칭단이 전체적인 암호와 복호 수행 속도에서 많은 영향을 미치지 않는다고 볼 수 있다.

암호와 복호가 다른 블록 암호 알고리즘을 간단한 대칭단을 적용하여 스마트카드 및 RFID 태그와 같은 제한된 하드웨어 및 소프트웨어 환경에서도 쉽게 구현 가능하고 SHACAL-2과 같은 변형된 Feistel 구조의 블록 암호 알고리즘에 대칭단을 삽입해서 암호와 복호 알고리즘을 동일하게 구현할 수 있으므로 새로운 블록 암호 설계 및 개발에도 유용한 아이디어로 사용될 것이다.

# 제 7장 결론

정보처리 관점에서 오늘날과 같은 초고속 정보통신이전의 데이터는 대부분 문자위주의 데이터였다. 그러나 현재는 고용량의 멀티미디어(Multimedia) 데이터(그림, 사운드, 동영상 등)가 더 높은 가치의 정보를 이루고 있으며, 이와 같은 고용량의 멀티미디어 데이터의 암호화를 비롯한 정보보호를 위한 처리는 필연적으로 빠른 수행을 요구한다. 그래서 좀 더 빠른 암호화 수행을 위해 하드웨어로 구현하므로 해서 이와 같은 요구를 만족시킬 수 있다. 그리고 RFID(Raido Frequency Identification)를 이용한 보안이 적용된 센서 네트워크(sensor network)를 구축하는 과정에서 암호 및 복호 과정을 필요로 하는 능동형 태그(active tag)의 구현은 암호 알고리즘의 하드웨어 구현이 필수 요소가 된다. 이와 같은 이유들 때문에 암호 알고리즘을 공모한 선진국들은 공모 기준에 하드웨어 구현 및 성능 평가가 선정을 위한 필수 조건이었다.

블록 암호 알고리즘의 하드웨어 구현은 암호 알고리즘의 구조와 밀접한 관계가 있다. 왜냐하면 Feistel 구조인 경우 암호와 복호 알고리즘이 같기 때문에 하드웨어 구현 시 암호 알고리즘만 구현하면 암호와 복호 과정을 모두 수행할 수 있다. 그러나 SPN 구조의 경우는 암호와 복호 알고리즘이 서로 달라 암호와 복호 수행이 모두 필요한 경우에는 암호와 복호 알고리즘을 모두 하드웨어로 구현해야 한다. 그래서 암호와 복호가 서로 다른 블록 암호 알고리즘의 하드웨어 구현은 암호와 복호 알고리즘이 같은 블록 암호 알고리즘과 비교해서 하드웨어 면적이 많이 늘어나게 되는 단점이 된다.

본 논문에서는 암호와 복호가 다른 블록 암호 알고리즘의 단점을 개선하기 위해 간단한 논리 연산으로 구성된 대칭단(Symmetric Layer)을 적용해서 암호와 복호 알고리즘을 같게 구성했다. 즉 암호 알고리즘의 전체를 R라운드로 구성하고, 1라운드부터 (R / 2) - 1라운드까지는 정함수(암호 알고리즘)를 사용하고, (R / 2) + 1라운드부터 R라운드까지는 역함수(복호 알고리즘)를 사용한다. 또한 정함수 단과 역함수 단 사이에 규칙적인 라운드의 반복을 피하기 위해 불규칙적인 대칭 블록을, 하나의 독립적인 라운드 형태인 대칭단을 삽입한다. 이와 같은 블록 암호의 구성은 암호와 복호 알고리즘을 같게 하는 특징이 된다.

암호와 복호가 다른 블록 암호 AES, RC6, SHACAL-1, SHACAL-2를 대칭단을 적용해서 암호와 복호가 같도록 소프트웨어로 새롭게 구성 했다. 기존의 AES, RC6, SHACAL-1, SHACAL-2와 대칭단을 적용한 각각의 새로운 블록

암호 알고리즘을 수행 시간 테스트 결과 대칭단을 적용한 암호 알고리즘이 10% 미만의 시간 증가를 보였고, 이는 대칭단의 적용이 전체적인 암호 수행 시간에 별다른 영향을 미치지 않는 것으로 판단된다. 그리고 128비트 블록뿐만 아니라 160, 256비트 블록 암호 알고리즘에도 대칭단은 쉽게 적용할 수 있으며, 이는 모든 암호와 복호가 다른 블록 암호 알고리즘에 대칭단을 적용할 수 있음을 의미한다.

대칭단을 적용한 블록 암호 알고리즘의 안전성 또한 대칭단이 암호 공격 흐름에 대한 단절로 효과적인 차분, 선형, Square, 불능 차분, Boomerang, 연관 키 Slide, 연관 키 차분-비선형, 연관 키 Rectangle 공격을 방해하고, 또한 공격 복잡도를 급격하게 향상 시키는 효과를 나타내고 있다. 이는 대칭단이 블록 암호 알고리즘의 안전성 향상에도 매우 효과적임을 보여 주고 있다. 이와 같은 대칭단의 실행 시간, 안전성에 대한 특징을 적용한 블록 암호 알고리즘의 재설계는 새로운 블록 암호 알고리즘의 디자인에도 쉽게 적용 가능한 유용한 아이디어가 된다.

# 참고문헌


[1] A.J. Menezes, P.C. ven Oorschot and S.A. Vanstone, "Handbook of Applied Cryptography", CRC Press, ISBN 0-8493-8523-7, http://www.cacr.math.uwaterloo.ca/hac/, 1996.

[2] A. Sinkov, "Elementary Cryptanalysis: A Mathematical Approach", Washington DC: The Mathematical Association of America, 1966.

[3] National Bureau of Standards, Data Encryption Standard, FIPS-Pub 46, 1977.

[4] W. Diffie and M. Hellman, "Multiuser Cryptographic Techniques", IEEE Transaction on Information Theory, 1976.

[5] E. Biham and A. Shamir, "Differential Cryptanalysis of the Full 16-Round DES", LNCS 537, page 2-21, 1990.

[6] M. Matsui, "Linear Cryptanalysis Method for DES", LNCS 765, page 386-397, 1994.

[7] R.L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, Vol. 21, No. 2, page 120-126, 1978.

[8] N. Koblitz,, "Elliptic Curve Cryptosystems", Mathematics of Computation, Vol. 48, No. 177, page 203-209, 1987.

[9] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transaction on Information Theory, Vol. 31, page 469-472, 1985.

[10] E. Biham and A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard", Springer-verlag, 1993.

[11] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", LNCS 765, page 386-397, 1994.

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

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

[14] "Cryptography Research and Evaluation Committees(CRYPTREC)",

http://www.cryptrec.go.jp/.
[15] SEED, http://www.kisa.or.kr/seed/.
[16] ARIA, http://www.nsri.re.kr/ARIA/.
[17] C. Carroll, A. Chan and M. Zhang, "The software-oriented stream cipher SSC-Ⅱ", FSE 2000, LNCS 1978, page 39-56, 2000.
[18] GSM interception, http://www.dia.unisa.it/ads.dir/corso-security/www/CORSO-9900/a5/Netsec/netsec.html.
[19] D.J. Bernstein, "Synchronous Stream Cipher Salsa20", http://www.ecrypt.eu.org/stream/salsa20.html.
[20] H. Feistel, "Cryptography and Computer Privacy", Scientific American, Vol. 228, No. 5, page 15-23, 1973.
[21] C.E. Shannon, "Communication Theory of Secrecy System" Bell System Technical Journal, Vol. 28, No. 4, page 656-715, 1949.
[22] 김 길호, 박 창수, 조 경연, "대칭구조 SPN 블록 암호 알고리즘", 한국멀티미디어학회, 2008.
[23] 김 길호, 박 창수, 김 종남, 조 경연, "대칭구조 RC6 블록 암호 알고리즘", 한국해양정보통신학회, 2009.
[24] J. Daemen and V. Rijmen, "The Design of Rijndael", Springer-Verlag, 2002.
[25] R.L. Rivest, M.J.B. Robshaw, R. Sidney and Y.L. Yin, "The RC6 Block Cipher v1.1", http://www.rsa.com/rsalabs/aes, 1998.
[26] H. Handschuh and D. Naccache, "SHACAL", preproceedings of NESSIE first workshop, Leuven, 2000.
[27] H. Handschuh and D. Naccache, "SHACAL: A Family of block Ciphers", Submission to the NESSIE project, 2002.
[28] NIST(National Institute of Standards and Technology), http://www.nist/gov/.
[29] R.L. Rivest, "The RC5 encryption algorithm", LNCS 1008, page 86-96, 1995.
[30] H. Handschuh, L.R. Knudsen and M.J. Robshaw, "Analysis of SHA-1 in Encryption Mode", LNCS 2020, page 70-83, 2001.
[31] FIPS 180-1: "Secure Hash Standard", Federal Information Processing

Standards Publication, NIST, 1995.
[32] FIPS 180-2: "Secure Hash Standard", Federal Information Processing Standards Publication, NIST, 2002.
[33] H. Handschuh and D. Naccache, "SHACAL: A Family of Block Ciphers", Submission to the NESSIE project, 2002.
[34] D. Kahn, "The Codebreakers", New York Times Book Review, 1968.
[35] J. Daemen, L.R. Knudsen and V. Rijmen, "The Block Cipher Square", LNCS 1267, page 149-165, 1997.
[36] E. Biham, "New Type of Cryptanalysis Attacks Using Related Keys", Journal of Cryptology, Vol. 7, No. 4, page 229-246, 1994.
[37] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like cryptosystem", LNCS 537, page 2-21, 1991.
[38] E. Biham and A. Shamir, "Differential Cryptanalysis of FEAL and N-hash", Advances in Cryptology-Eurocrypt `91, page 1-16, 1991.
[39] E. Biham and A. Shamir, "Differential Cryptanalysis of Snefru, Khafru, REDOC-II, LOKI and Lucifer", LNCS 576, page 156-171, 1992.
[40] L.R. Knudsen, "Trucated and Higher Order Differentials", LNCS 1008, page 196-211, 1995.
[41] S.K. Langford and M.E. Hellman, "Differential-Linear Cryptanalysis", LNCS 839, page 17-25, 1994.
[42] E. Biham, A. Biryukov and A. Shamir, "Cryptanalysis of Skipjack Reduced to 32 Rounds Using Impossible Differentials", Journal of Crytology, Vol. 18, No. 4, page 291-311, 2005.
[43] D. Wagner, "The Boomerang Attack", LNCS 1636, page 156-170, 1999.
[44] E. Biham, O. Dunkelman and N. Keller, "The Rectangle Attack - Rectangling the Serpent", LNCS 2045, page 340-357, 2001.
[45] K. Aoki and K. Ohta, "Strict Evaluation of the Maximum Average of Differential Probability and the Maximum Average of Linear Probability", IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences, Vol. E-80A, No. 1, page 2-8, 1997.
[46] B.S. Kaliski Jr. and M.J.B. Robshaw, "Linear Cryptanalysis Using

Multiple Approximations", LNCS 839, page 26-39, 1994.
[47] L.R. Knudsen and M.J.B. Robshaw, "Non-Linear Approximations in Linear Cryptanalysis", LNCS 1070, page 224-236, 1996.
[48] L.R. Knudsen and J.E. Mathiassen, "A Chosen-Plaintext Linear Attack on DES", LNCS 1978, page 262-272, 2001.
[49] N.T. Courtois, "Feistel Schemes and Bi-Linear Cryptanalysis", LNCS 3152, page 23-40, 2004.
[50] CRYPTON, http://www.securitytechnet.com/main/research/crypton.html.
[51] A. Biryukov and A. Shamir, "Structural Cryptanalysis of SASAS", LNCS 2045, page 394-405, 2001.
[52] L.R. Knudsen and D. Wagner, " Integral Cryptanalysis", LNCS 2365, page 112-127, 2002.
[53] J. Kelsey, B. Schneier and D. Wagner, "Key Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER and Triple-DES", LNCS 1109, page 237-251, 1996.
[54] R.C.W. Phan and H. Handschuh, "On Related-Key and Collision Attack: The Case for the IBM 4758 Cryptoprocessor", LNCS 3225, page 111-122, 2004.
[55] Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", NIST, http://csrc.nist.gov/piblications/fips/.
[56] FIPS 46-3: "Data Encryption Standard", NIST, Specifies the use of Triple-DES, 1999.
[57] A.M. Youssef, S. Mister and S.E. Tavares, "On the Design of linear Transformation for Substitution and Permutation Encryption Networks", SAC, page 40-48, 1997.
[58] A. Biryukov, C. D. Canniere, J. Lano, S. B. Ors, B. Preneel, "Security and Performance Analysis of ARIA", Katholieke Universiteit Leuven, 2003.
[59] K. Aoki, T. Ichikawa, M. Kanda, M. Matusi, S. Moriai, J. Nakajima and T. Tokita, "Camellia - a 128bit block cipher suitable for miltiple platforms", ASC 2000.
[60] Nippon Telegraph and Telephone Corporation, "Specification of E2 -

a 128-bit Block Cipher", http://info.isl.ntt.co.jp/e2/, 1999.
[61] M. Matusi, "New Block Encryption Algorithm MISTY", In Proceedings of the 4th Fast Software Encryption Workshop, 1997.
[62] Y.L. Yin, "A Note on the Block Cipher Camellia", a contribution for ISO/IEC JTC1/SC27, 2000.
[63] IBM Corporation, "MARS - a candidate cipher for AES", 1999.
[64] J. Cheon, M. Kim, K. Kim, J. Lee and S. Kang, "Improved impossible differential cryptanalysis of Rijndael and Crypton", LNCS 2288, page 39-49, 2001.
[65] A. Birynkov, "The Boomerang attack on 5 and 6-round reduced AES", LNCS 3373, page 42-57, 2005.
[66] S. Hong, S. Lee, J. Lim, J. Sung, D. Cheon and I. Cho, "Provable security against differential and linear cryptanalysis for the SPN structure", LNCS 1978, page 273-283, 2001.
[67] S. Contini, R.L. Rivest, M.J.B. Robshaw, and Y.L. Yin, "The Security of the RC6 Block Cipher", 1998.
[68] H. Handschuh, L.R. Knudsen and M.J. Robshaw, "Analysis SHA-1 in Encryption Mode", LNCS 2020, page 70-83, 2001.
[69] 김 종성, 문 덕재, 이 원일, 홍 석희, 이 상진, "SHACAL의 축소 라운드에 대한 확장된 부메랑 공격", 정보보호학회 논문지 12권 5호, 2002.
[70] E. Biham, O. Dunkelman and N. Keller, "A Simple Related-Key Attack on the Full SHACAL-1", Topice in Cryptology-Proceeding of CT-RSA 2007.
[71] A. Biryukov and D. Wagner, "Slide Attacks", LNCS 1636, page 245-259, 1999.
[72] H. Lipmaa, "On Differential Properties of Pseudo-Hadamard Transform and Related Mappings", LNCS 2551, page 48-61, 2002.
[73] Markku-Juhani O. Saarinen, "Cryptanalysis of Block Cipher Based on SHA-1 and MD5", LNCS 2887, page 36-44, 2003.
[74] J. Kim, "Combined Differential, Linear and Related-Key Attacks on Block Cipher and MAC Algorithms", KATHOLIEKE UNIVERSITEIT LEUVEN, 2006.
[75] P. Hawkes, "Differential-Linear WeakKey Classes of IDEA", LNCS

1403, page 112-126, 1998.
[76] E. Biham, O. Dunkelman and N. Keller, "Rectangle Attacks on 49-Round SHACAL-1", LNCS 2887, page 22-35, 2003.
[77] Y. Shin, J. Kim, G. Kim, S. Hong and S. Lee, "Differential-Linear Type Attacks on Reduced Rounds of SHACAL-2", LNCS 3108, page 110-122, 2004.
[78] J. Lu, J.S. Kim, N. Keller and O. Dunkelman, "Related-Key Rectangle Attack on 42-Round SHACAL-2", LNCS 4176, page 85-100, 2006.
[79] S. Hong, J. Kim, G. Kim, J. Sung, C. Lee and S. Lee, "Impossible Differential Attack on 30-Round SHACAL-2", LNCS 2904 page 97-106, 2003.

# 감사의 글

대학원 박사과정 5년 동안 많은 어려움이 있었지만 오늘이 있게 해주신 모든 분들께 감사의 인사 올립니다. 특히 많이 부족한 저를 많은 관심과 지도편달로 이끌어주신 조경연 교수님께 진심으로 감사드립니다. 그리고 바쁘신 가운데에도 본 논문의 심사와 많은 학문적 조언, 더 나은 논문이 될 수 있도록 보완점을 제시해 주신 서경룡 교수님, 김종남 교수님, 송홍복 교수님, 권장우 교수님께도 진심으로 감사드립니다. 그리고 대학원 생활동안 저에게 많은 가르침을 주신 전자컴퓨터정보통신공학부 여러 교수님께도 진심으로 감사드립니다.

긴 시간동안 많은 조언으로 나에게 큰 버팀목이 되어준 박창수 박사님과 김성기 선배님께도 깊은 감사드립니다. 그리고 연구실에서 동고동락하며 같이 지내면서 큰 힘이 되어준 고려대학교 대학원에 진학한 서상우, 울산과학기술대학교 대학원에 진학한 조정욱군과 조호현, 한주승 그리고 새로 우리 연구실에 들어온 변두호, 한성환군에게도 고마움을 표시 합니다.

아직도 많이 부족함을 느끼지만 지금까지의 배움을 밑바탕으로 이 사회 어디서든지 꼭 필요한 사람으로 남고 싶으며, 학문에 대한 배움의 열정을 잃지 않고 끝임 없이 노력 하겠습니다. 마지막으로 오늘과 같은 영광이 있게 해주신 모든 분들께 다시 한 번 더 감사드리고, 지금까지 인내로 기다려 주신 사랑하는 어머니께 감사드립니다.

2009년 12월

김 길 호 올림