동적 키를 이용한 블럭 암호 알고리즘의 설계 및 평가

정홍섭[†] · 이창두[†] · 박규석[††]

요 약

기존의 블럭 암호 알고리즘은 암호화 키 값이 변하지 않고 매 블럭의 라운드 함수에 적용되면서 최종적으로 암호화되도록 설계되어 있다. 따라서 라운드를 반복하는 구조의 블럭 암호를 해독하는 가장 강력한 방법인 차분 해독이나 선형 해독 등에 의해 평문이나 암호화 키가 노출되기 쉬운 단점을 가지고 있다. 이러한 단점을 보완하기 위해서는 좀더 효율적인 키를 사용하는 암호 알고리즘의 설계가 요구된다. 본 논문에서는 암호화 키 값을 매 블럭의 라운드마다 동적으로 생성 적용되도록 설계한 블럭 암호 알고리즘을 제안한다. 이 알고리즘은 자료의 암호화와 복호화 처리시간이 짧고, 암호화의 강도가 높아 전자상거래 보안 등 다양한 분야의 정보보호에 응용할 수 있다.

Design and Evaluation of A Block Encryption Algorithm using Dynamic-Key

Hong Seup Jeong[†], Chang Doo Lee[†] and Kyoo Seok Park[††]

ABSTRACT

The existing block encryption algorithms have been designed for the encryption key value to be unchanged and applied to the round functions of each block, and enciphered. Therefore, it has such a weak point that the plaintext or encryption key could be easily exposed by differential cryptanalysis or linear cryptanalysis, both are the most powerful methods for decoding block encryption of a round-repeating structure. In order to overcome with this weak point, an encryption algorithm using a more efficient key should be designed. In this paper, a block encryption algorithm which is designed for each encryption key value to be applied to each round block with different value is proposed. This algorithm needs a short processing time in an encryption and decryption, has a high intensity, can apply to electronic commerce and various applications of data protection.

Key words: block encryption, decryption, dynamic-key

1. 서 론

정보를 취급하는 과정에서 오는 취약성으로 인하여 정보 시스템의 파괴, 개인 신상 비밀의 누설 및 유출, 불건전 정보의 유통 등과 같은 피해가 증가하고 있다. 이러한 상황에서 정보 자산의 보호는 중요한 현안으로 떠오르고 있으며, 내 · 외부 침입에 의한 각종 정보 침해 사고의 해결을 위해 정보 암호화는 그 한 가지 대안으로 제시되고 있다.

암호화란 원래 내용을 숨겨 원본과 다른 전달문 또는 비밀문을 만드는 과정으로서, 컴퓨터 상에서 수학적인 기법들을 통해 메시지와 결합된 문자의 조합을 만드는 것을 의미한다[1]. 고전적 암호화 기술은 데이터의 기밀성을 위주로 연구 발전되어 왔으며, 혼돈(confusion)과 확산(diffusion)이론을 배경으로 하는 현대 암호화 기술은 인터넷과 같은 공중망을 이용한 정보 이용이 증가함에 따라 상대방의 신원 확인(authentication), 데이터 무결성(data integrity) 보

본 논문은 2002년도 경남대학교 학술연구비 지원에 의해 수행되었음.

[†] 정회원, 경남대학교 대학원 컴퓨터공학과 박사과정 수료
[††] 종신회원, 경남대학교 정보통신공학부 교수

장 및 부인봉쇄(non-repudiation) 등에 대한 연구가 활발히 이루어지고 있다[10,14].

현재의 암호 알고리즘은 공개키 암호 알고리즘과 대칭키 암호 알고리즘의 장점을 서로 혼합하여 구축하는 경우가 많다[2]. 즉, 평문은 암호화 속도가 빠른 대칭키 암호 알고리즘으로 암호화한 후 목적지로 전송하고, 암호화 키는 안전도가 높은 공개키 암호 알고리즘으로 암호화하여 전송하는 방법이 많이 운용되고 있다.

하지만 대칭키 암호 알고리즘에 속하는 블럭 암호 알고리즘 중 몇몇은 암호화 키 값이 변하지 않고 매 블럭의 라운드 함수에 적용되어 최종적으로 암호화되도록 설계되어 있다[11]. 따라서 차분 해독이나 선형 해독 등에 의해 평문 및 암호화 키가 노출되기 쉬운 단점을 가지고 있다. 이러한 단점을 보완하기 위해서는 좀더 효율적인 키 운용 방법을 사용하는 새로운 블럭 암호 알고리즘의 설계가 요구된다.

2. 관련연구

2.1 DES(Data Encryption Standard)

DES는 1977년 미국 국립 표준연구소 NIST (National Institute of Standard Technology)가 공개한 대칭키 암호화 시스템으로서 치환, 대입, XOR 등의 연산을 사용하기 때문에 속도가 매우 빠르다는 장점이 있으며, UNIX의 패스워드를 암호화하는 "crypt"에 이용되었고, 미국과 유럽의 은행에 의해 전자 자금 이체(EFT) 시스템 등에 널리 사용되었다[11].

DES는 평문 64비트를 암호문 64비트로 변환시키는 암호화 방식으로 64비트의 암호화 키를 사용하고 있다. 이 암호화 키는 8비트마다 패리트 비트 하나씩을 포함하고 있으며, 암호화 과정에서는 56비트만을 사용한다.

DES 암호 알고리즘의 기본 동작은 전치, 대치와 mod 2 연산으로 구성되어 있다. 전치는 평형 전치, 확대 전치 그리고 축약 전치 등의 세 종류가 있으며 대치는 S-Box라는 대치 장치에서 이루어진다.

2.2 Skipjack

Skipjack은 미국 NSA(National Security Agency)에서 1985년 개발한 Clipper Chip에 내장된 블럭 암호 알고리즘으로서 알고리즘 형태 및 구조는 비밀로 분류되어 있다가 1998년에 공개되었으며, 공표된 사양은 표 1과 같다.

표 1. Skipjack 알고리즘의 사양

특 성	반복 구조를 갖는 블럭 암호화
입출력 크기	64비트
키크기	80비트
라운드 수	32라운드

Skipjack은 G 함수(일반적으로 F 함수를 말함) 단위로 라운드 수를 계산하고, 4개의 word (1 word는 16비트)를 두 개의 변환 규칙(변환규칙 A, 변환규칙 B)으로 암호화한다. Skipjack의 G 함수는 4라운드 Feistel 구조(일반적으로 DES 암호 알고리즘 구조를 말함)로 되어 있다. G 함수 내부에는 8비트 입출력을 가지는 총 4개의 F 함수(일반적으로 S-Box를 말함)를 사용한다[8].

Skipjack 구조는 Feistel 구조와 달리 F 함수의 출력 결과가 다른 블록의 입력 블록으로 변화되어 암호화가 수행되므로 복잡도가 계속 증가되는 좋은 구조라 볼 수 있으며, 구조 자체의 안전성도 증명되었다[6].

2.3 SEED

SEED 알고리즘은 한국정보통신기술협회에서 1999년 9월에 표준으로 제정한 블럭 암호 알고리즘이다. 이 알고리즘은 128비트 키를 이용하여 데이터를 128비트 블럭 단위로 암·복호화하는 전형적인 Feistel 구조로 설계되어 있다. 이 알고리즘의 특징적인 구조는 16개의 F 함수가 있고, F 함수 내에 3개의 G 함수로 구성되어 있으며, G 함수에는 두 개의 S-Box가 사용된다.

SEED는 128비트 입력 평문 블럭을 2개의 64비트 블럭으로 나누어서, 64비트 라운드 키 16개를 이용하여 16라운드의 암호화 과정을 수행한 후, 최종 128비트 암호문 블럭을 출력한다[13].

2.4 AES(Advanced Encryption Standard)

AES는 DES 암호 알고리즘을 대체하기 위해 1997년 NIST에서 AES 암호 알고리즘을 공모하여 최종 채택된 Rijndael(이하 AES) 암호 알고리즘으로서, 3

중-DES (triple-DES) 보다 더 효율적이고 안전하게 설계되었다. 암호화 키의 길이는 사용자 또는 환경과 특성에 따라 128, 192, 256비트 등을 유동적으로 선택할 수 있는 SPN(Substitution Permutation Network)구조의 매우 강력한 대칭키 블럭 암호 알고리즘이다.

AES는 코드가 간단하며, 디자인이 단순하여 여러 플랫폼에서 빠르게 수행되고, 알려진 여러 가지 해독에 강한 장점을 가지고 있으며, $GF(2^8)$로 덧셈, 곱셈, XOR 연산 등을 이용하여 암호화한다[16].

2.5 운영 모드

블럭 암호 알고리즘에서는 암호 알고리즘 뿐만 아니라 이 알고리즘을 운영하는 운영 모드도 상당히 중요한 역할을 담당한다. 즉, 어떤 운영 모드를 사용하여 알고리즘을 운영하느냐에 따라 암호화의 강도와 암호화 효율성이 달라질 수 있다. 그러므로 암호 알고리즘 운영 모드는 안전하고 효율적으로 설계되어야 한다.

현재 국제 표준인 ISO/IEC 10116에 등록되어 있는 운영 모드로는 ECB(Electronic Code Block), CBC(Cipher Block Chaining), CFB(Cipher Feedback), 그리고 OFB(Output Feedback) 등이 있다[3,9].

본 논문에서는 제안 블럭 암호 알고리즘의 운영을 위해서 [15]에서 제안한 RBF 운영모드를 이용한다. RBF 운영모드는 그림 1과 같이 매 블럭마다 랜덤 변수를 발생시켜 동적인 키($E_1, E_2, \cdots, E_n$) 값으로 암호화하여 차분 해독과 선형 해독을 어렵게 하는 장점이 있다.

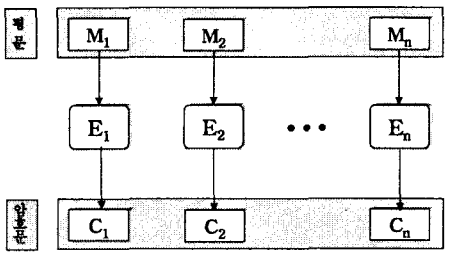


그림 1. RBF 운영 모드

3. 제안 알고리즘

기존의 블럭 암호 알고리즘은 일반적인 운영 모드를 이용하여 암호화를 수행함으로써 평문과 암호문의 쌍이 많아지면 해독 위험성이 높아질 수 있다. 따라서 차분 해독과 선형 해독 등에 강하게 대처할 수 있는 블럭 암호 알고리즘이 요구된다

제안 블럭 암호 알고리즘은 입력 비트와 출력 비트의 상관도를 낮추기 위해 그림 2와 같이 암호화 상위 단계, 중간단계, 하위 단계로 나누어 설계하여 침입자에 의한 암호 해독에 강하게 대항할 수 있도록 하였다.

3.1 암호 알고리즘 설계

본 논문에서 제안하는 블럭 암호 알고리즘의 설계 배경은 다음과 같다.

▪ 대칭적인 암호화 및 복호화

암호화 및 복호화 속도를 동일하게 하기 위해서 암호화와 복호화가 대칭적으로 수행하도록 설계한다. 이렇게 함으로써 선택 평문 해독과 선택 암호문 해독에도 동일한 암호화 강도를 적용할 수 있다.

▪ 키(key)의 길이

암호 알고리즘의 키 길이 권고사항인 128비트 키 암호 알고리즘으로 설계한다.

▪ 효율성과 안전성

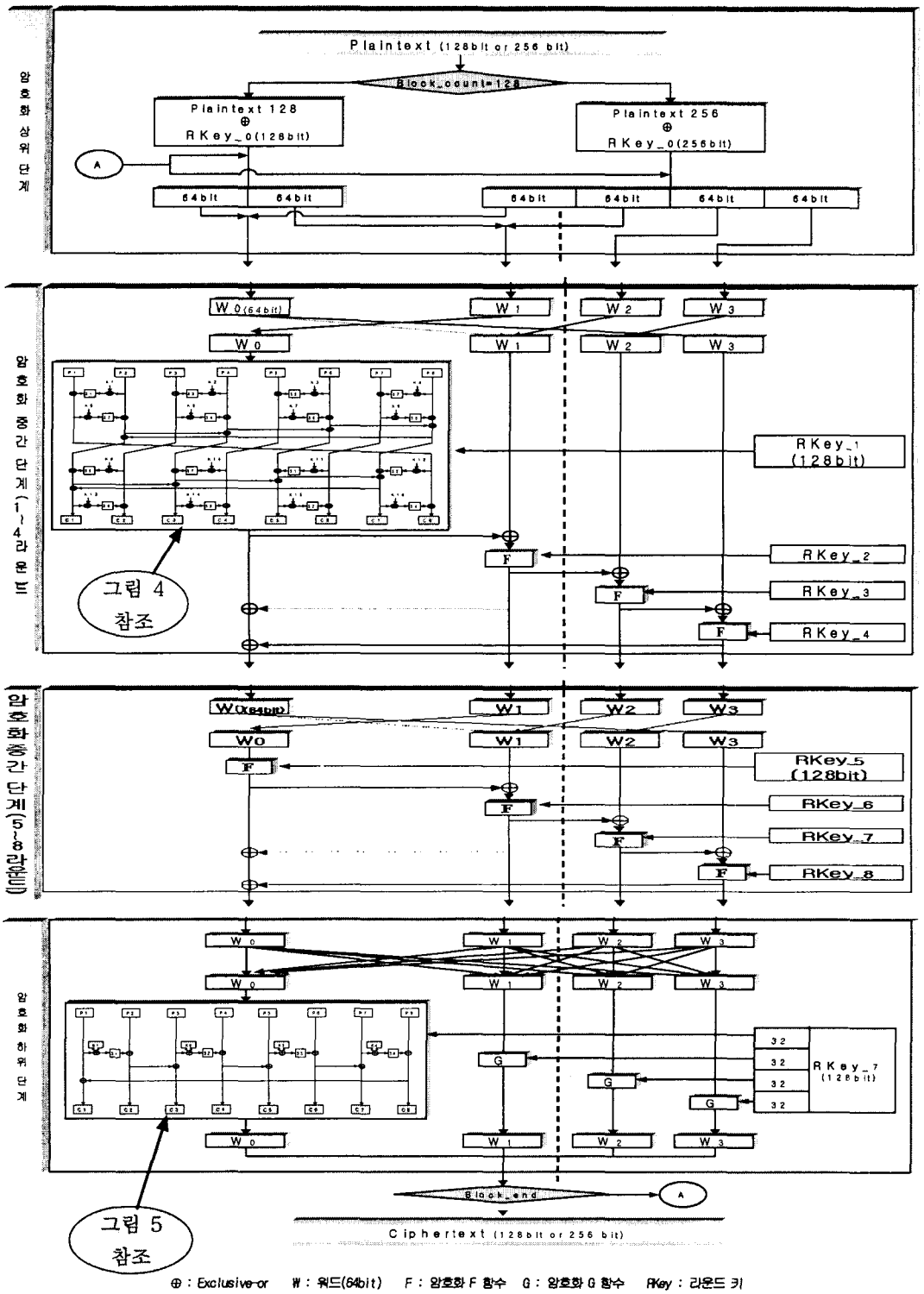
연산을 간단히 하여 빠른 암호화 및 복호화 속도를 가지도록 하고, 키 전수 탐색, 차분 해독, 선형 해독 등에 안전하도록 설계한다.

▪ 특성

· 성 격 : 128비트 또는 256비트를 처리하는 블럭 암호 알고리즘이며, 키 스케줄러에 의해 RBF 운영 모드로 암호화한다.
· 구 조 : Skipjack 구조를 기반으로 한다.
  - F 함수 내부 구조 : Feistel
· 입/출력물의 크기 : 128비트 또는 256비트
· 암호화 키의 크기 : 128비트 또는 256비트
· 회 전 수 : 상위 단계 - 1회전
  중간 단계 - 8회전
  하위 단계 - 4회전
· 내부함수 : 중간 단계 - F 함수 8개
  하위 단계 - G 함수 4개
· S-Box : 8비트 S-Box 8개

▪ 기호 정의

• $a \oplus b$: a xor b	• $\|$: concatenation
• $\lll$: Left Rotate	• $\ggg$: Right Rotate
• $\ll$: Left Shift	• $\gg$: Right Shift



⊕ : Exclusive-or W : 워드(64bit) F : 암호화 F 함수 G : 암호화 G 함수 RKey : 라운드 키

그림 2. 블럭 암호 알고리즘 구성도

3.1.1 암호화 상위 단계

그림 2의 암호화 상위 단계에서는 먼저 사용자의 암호화 요구에 따라 128비트 평문 암호화 또는 256비트 평문 암호화 작업을 선택한다. 128비트 암호화를 수행할 경우에는 128비트 평문과 그림 3에서 생성한 128비트 라운드 키(RKey_0)를 XOR한 후에 이것을 다시 두 개의 64비트 워드로 분할하여 중간 암호화 단계의 W_0, W_1 워드로 보낸다. 256비트 암호화를 수행할 경우에는 256비트 평문과 256비트 라운드 키(RKey_0)를 XOR한 다음 4개의 64비트 워드로 분할하여 암호화 중간 단계의 W_0, W_1, W_2, W_3 워드로 보내는 단계이다.

3.1.2 암호화 중간 단계

그림 2의 암호화 중간 단계는 본 암호 알고리즘의 핵심 단계로 128비트 암호화인 경우에는 암호화의 상위 단계에서 넘어온 W_0, W_1 워드만 가지고 암호화를 수행한다. 256비트 암호화인 경우에는 W_0, W_1, W_2, W_3 워드로 암호화를 수행한다.

중간 단계의 암호화 과정은 서브키 생성기와 라운드 키 생성기에서 생성한 동적 라운드 키 값으로 운용되는 F 함수를 가지고 총 8라운드의 암호화를 수행한다. 이 중간 단계의 주요 핵심은 입력과 출력에 대한 블럭과 블럭의 연관성, 라운드 키와 라운드 키의 연관성이 적도록 암호화 키를 생성하는 것이다.

먼저 암호화 중간 단계에서 사용하는 F 함수의 동적 라운드 키 값을 생성하는 서브 키 생성기와 라운드 키 생성기의 설계에 대해 기술한다. 다음으로 F 함수의 내부에서 사용하는 S-Box 설계에 대해 기술한다. 마지막으로 F함수의 구조 설계와 암호화 중간 단계의 설계에 대해 설명한다.

1) 서브 키 및 라운드 키 생성기의 설계

라운드 키를 동적으로 생성하기 위해 필요한 서브 키와 라운드 키의 생성 과정은 그림 3과 같다. 1차로 사용자 암호화 키를 패딩과 초기 치환을 한 다음 서브 키 생성기에서 제 1 서브 키를 만든 후, 이 결과 값으로 라운드 키 생성기에서 라운드 함수에 필요한 라운드 키 8개를 생성한다. 1차로 만들어진 1번 라운드 키와 8번 라운드 키 값을 가지고 서브 키 생성기에서 제 2 서브 키를 만들고, 다시 이 결과 값으로 라운드 키 생성기에서 2차 라운드 함수에 필요한 라운드 키 8개를 생성한다. 이 과정은 2차에서 N차까지 수행

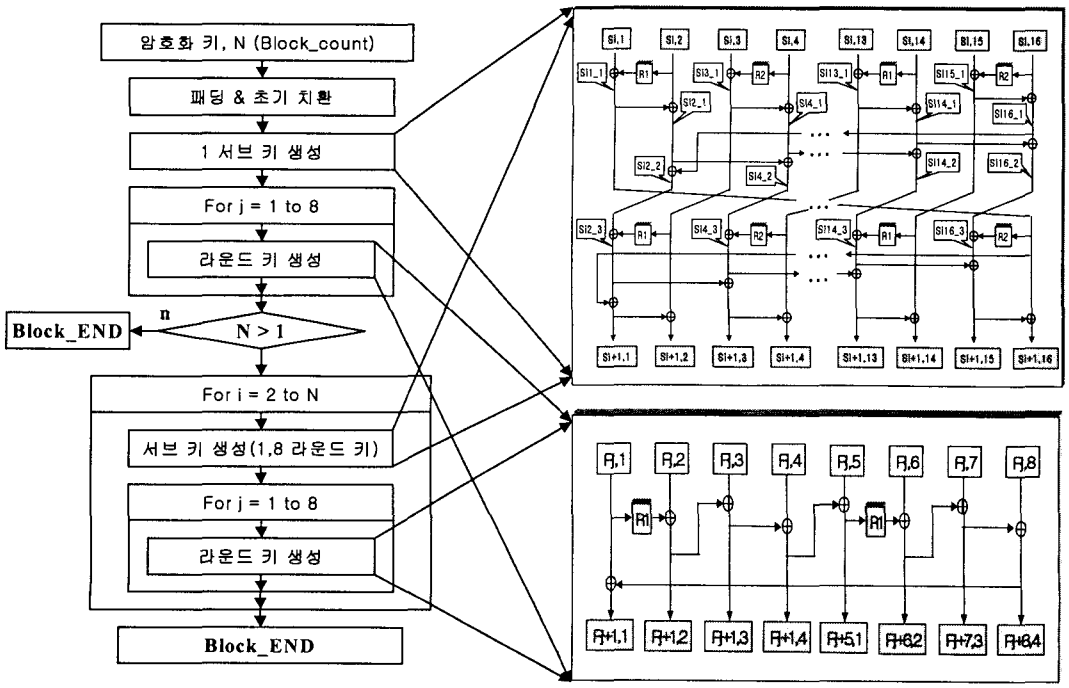


그림 3. 서브키를 이용한 라운드 키 생성 과정

되어 동적으로 라운드 키를 생성하게 된다.

▪ 초기 치환

사용자가 키를 입력할 때 키의 값을 일정하게 반복한다든지(예를 들어, '11111···', 'aaaaa···'등) 또는 키 값이 순서적으로 입력된다면(예; '12345···', 'abcde···' 등), 키 노출의 위험성이 많아지게 된다. 따라서 사용자가 키 노출이 쉬운 암호를 입력하는 경우를 대비하여 키 스케줄러에서 먼저 사용자가 입력한 키 값이 128비트(16Byte) 또는 256비트(32Byte) 보다 적은 경우 그 나머지 비트 부분을 0 또는 1로 채우는 키 패딩을 한다. 즉, 사용자가 영문자 72비트만 입력하였다면, 나머지 56비트를 0 또는 1로 채우게 된다. 본 논문에서는 일반적으로 많이 사용하는 0으로 키 패딩을 한다. 다음으로 키 패딩한 값과 치환 상수를 XOR하여 사용자의 키를 일차적으로 치환한다. 여기서 사용되는 치환 상수는 의사 난수 생성기를 이용하여 만들어진다.

▪ 서브 키 생성기

서브 키 생성기는 라운드 키를 만들기 전에 동작하는 생성기로, 초기 치환된 키의 강도를 높이는 작업을 담당한다. 수행 절차는 다음과 같다. 1차적으로 초기 치환 값(128비트)을 8비트 16개로 분리하여 그림 3의 우측 상단에 있는 서브키 생성기의 $[S_i,1]$, $[S_i,2]$, · · ·, $[S_i,16]$에 입력한다.

최종적으로 아래의 서브키 생성수식으로 $[S_i+1,1]$,$[S_i+1,2]$, · · ·,$[S_i+1,16]$의 서브키를 생성한다.

- 서브 키 생성 수식

Si1_1 = R1($[S_i,2]$) ⊕ $[S_i,1]$

$[S_i+1,16]$= Si1_1 ⊕ $[S_i+1,15]$

Si2_1 = Si1_1 ⊕ $[S_i,2]$

Si2_2 = Si2_1 ⊕ Si16_1

Si2_3 = Si2_2 ⊕ R1(Si3_1)

$[S_i+1,1]$ = Si2_3 ⊕ Si1_1

Si3_1 = R2($[S_i,4]$) ⊕ $[S_i,3]$

$[S_i+1,2]$ = Si3_1 ⊕ $[S_i+1,1]$

Si4_1 = Si3_1 ⊕ $[S_i,4]$

Si4_2 = Si4_1 ⊕ Si2_1

Si4_3 = Si4_2 ⊕ R2(Si5_1)

$[S_i+1,3]$ = Si4_3 ⊕ Si2_3

·

·

·

Si16_1 = Si15_1 ⊕ $[S_i,16]$

Si16_2 = Si16_1 ⊕ Si14_1

Si16_3 = Si16_2 ⊕ R2(Si1_1)

$[S_i+1,15]$ = Si16_3 ⊕ Si14_3

위의 서브키 생성 수식에서 사용된 의사 난수 R1과 R2는 각각 식 1과 식 2로 생성한다.

R1 = ((입력값 * 175) + 11220) mod 256 (1)

(입력값 예: $[S_i,2]$, $[S_i,6]$, $[S_i,10]$, $[S_i,14]$ 등)

R2 = ((입력값 * 193) + 11220) mod 256 (2)

(입력값 예: $[S_i,4]$, $[S_i,8]$, $[S_i,12]$, $[S_i,16]$ 등)

▪ 라운드 키 생성기

그림 3의 우측 하단에 있는 라운드 키 생성기는 서브키 생성기에서 생성된 서브키를 받아 파이프라인 방식으로 운영하여 라운드 키 값을 생성한다. 라운드 키의 입력 워드 $[R_j,1]$(즉, $[R_j,1]$ = $[S_i+1,1]$ || $[S_i+1,2]$), $[R_j,2]$, · · ·, $[R_j,8]$은 16비트이다.

라운드 키 생성을 위해 사용되는 수식은 다음과 같다.

- 라운드 키 생성 수식

Rj+1,1 = (Rj,1) ⊕ (Rj+1,8)

Rj+1,2 = R1(Rj,1) ⊕ (Rj,2)

Rj+1,3 = (Rj+1,2) ⊕ (Rj,3)

Rj+1,4 = (Rj+1,3) ⊕ (Rj,4)

Rj+1,5 = (Rj+1,4) ⊕ (Rj,5)

Rj+1,6 = R2(Rj+1,5) ⊕ (Rj,6)

Rj+1,7 = (Rj+1,6) ⊕ (Rj,7)

Rj+1,8 = (Rj+1,7) ⊕ (Rj,8)

의사 난수 R1과 R2는 각각 식 3와 식 4로 생성한다.

R1 = ((입력값1 × 175) + 11220) mod 256

|| ((입력값2 × 193) + 11220) mod 256 (3)

단, 입력값1 : $[R_j,1]$의 좌측 8비트 값

입력값2 : $[R_j,1]$의 우측 8비트 값

R2 = ((입력값1 × 175) + 11220) mod 256

|| ((입력값2 × 193) + 11220) mod 256 (4)

단, 입력값1 : $[R_j+1,4] \oplus [R_j,5]$ 한 좌측 8비트 값

입력값2 : $[R_j+1,4] \oplus [R_j,5]$ 한 우측 8비트 값

2) S-Box 설계

F 함수의 내부에 사용하는 S-Box는 암호문의 강도에 영향을 주는 매우 중요한 요소 중의 하나로써 F 함수 내에 8개의 S-Box(S1,S2, · · · ,S8)를 설치한다. S-Box는 전단사함수 x^n ($0 \le n \le 255$)에서 차분 및 선형 특성(차분 및 선형 확률 2^{-6})이 가장 우수한 8개를 선택하고, 지수승을 구하기 위해 $GF(2^8)$로 모든 원시원소 a의 멱승으로 구한다. S-Box를 구하기 위해 사용된 원시원소는 $a=p(x)=x^8+x^6+x^5+1$ 이다.

3) F 함수 설계

제안 암호 알고리즘의 중간단계에서 사용되는 F 함수를 설계하기 위해서 식 5와 같이 최대 차분 확률이 만족되도록 설계한다.

$$DP^F_{\max} = \max_{\Delta X \neq 0, \Delta Y} DP^F(\Delta X \rightarrow \Delta Y) \qquad (5)$$

여기에서 최대 차분 확률 $DP^F_{\max}$는 F 함수의 안전성에 중요한 영향을 준다. F 함수를 설계하기 위한 수식은 다음과 같다.

• F 함수 설계 수식

P1_1 = S1(P2 ⊕ K1) ⊕ P1
P2_1 = S2(P1_1 ⊕ K5) ⊕ P2
P3_1 = S3(P4 ⊕ K2) ⊕ P3
P4_1 = S4(P3_1 ⊕ K6) ⊕ P4
P5_1 = S5(P6 ⊕ K3) ⊕ P5
P6_1 = S6(P5_1 ⊕ K7) ⊕ P6
P7_1 = S7(P8 ⊕ K4) ⊕ P7
P8_1 = S8(P7_1 ⊕ K8) ⊕ P8

P2_2 = (P6_1 ⊕ P8_1) ⊕ P2_1

P4_2 = P2_1 ⊕ P4_1
P6_2 = P4_1 ⊕ P6_1
P8 2 = P6 1 ⊕ P8 1

P2_3 = S5(P3_1 ⊕ K9) ⊕ P2_2
P4_3 = S7(P5_1 ⊕ K10) ⊕ P4_2
P6_3 = S1(P7_1 ⊕ K11) ⊕ P6_2
P8_3 = S3(P1_1 ⊕ K12) ⊕ P8_2

C1 = (P6_3 ⊕ P8_3) ⊕ P2_3
C2 = S6(C1 ⊕ K13) ⊕ P3_1
C3 = P2_3 ⊕ P4_3
C4 = S8(C3 ⊕ K14) ⊕ P5_1
C5 = P4_3 ⊕ P4_3
C6 = S2(C5 ⊕ K15) ⊕ P7_1
C7 = P6_3 ⊕ P8_3
C8 = S4(C7 ⊕ K16) ⊕ P1_1

여기서, K1~K16은 서브 키 생성기와 라운드 키 생성기에 의해 생성된 라운드 키 값이고 S1~S8은 S-BOX의 번호이다.

위의 수식으로 그림 4와 같이 F 함수를 설계한다.

4) 중간 단계의 암호화 과정

그림 3의 서브키를 이용한 라운드 키 생성 과정에서 만들어진 동적 라운드 키 RKey_1, RKey_2, · · · , RKey_8은 그림 3.2의 암호화 중간단계에서 사용되는 8개의 F함수에 각각 입력하여 8라운드의 암호화를 수행한다.

중간 단계의 암호화 과정은 다음과 같다.

• 중간 단계의 암호화 과정

W0 = W1 : W1 = W2 : W2 = W3 : W3 = W4
W0,1 = F(W0, RKey_1)
W1,1 = F(RKey_2,(W0,1 ⊕ W2))
W2,1 = F(RKey_3,(W1,1 ⊕ W3))
W3,1 = F(RKey_4,(W2,1 ⊕ W4))

W0,1 = W0,1 ⊕ W3,1

W0 = W1 : W1 = W2 : W2 = W3 : W3 = W4
W0,2 = F(W0, RKey_5)
W1,2 = F(RKey_6,(W0,1 ⊕ W1))
W2,2 = F(RKey_7,(W1,1 ⊕ W2))
W3,2 = F(RKey_8,(W2,1 ⊕ W3))
W0,2 = W0,1 ⊕ W3,1

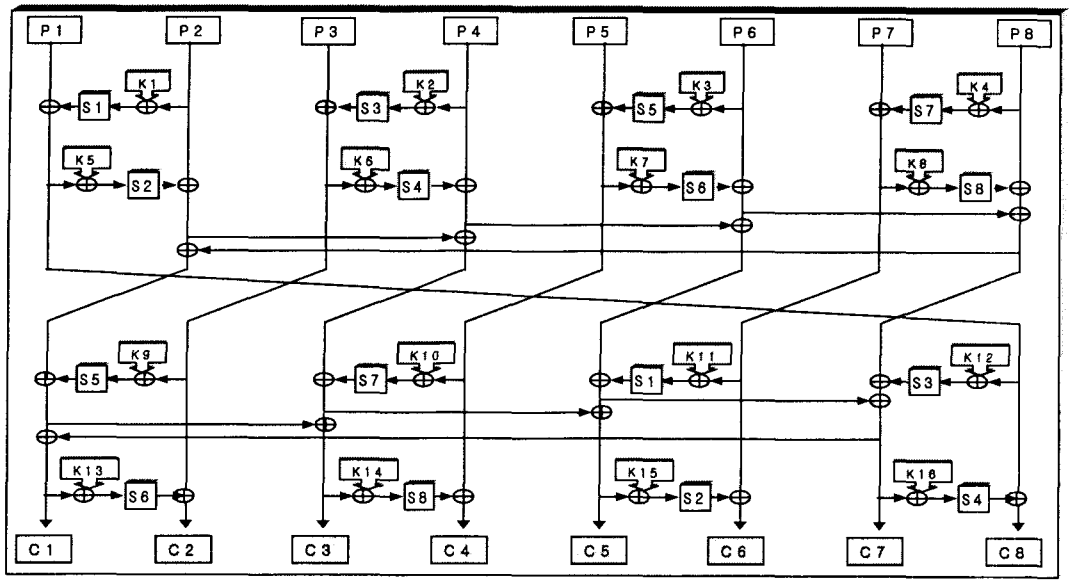


그림 4. F 함수 구조도

3.1.3 암호화 하위 단계

암호화 하위 단계는 암호화 중간 단계에서 만들어진 암호문을 G 함수를 사용하여 다시 치환하는 단계이다. 즉, G 함수는 암호문의 한 비트 변경으로 전체 비트에 영향을 주어 중간 단계에서 생성한 암호문에 대한 해독을 더욱 어렵게 만드는 단계이다.

암호화의 하위 단계에서의 수행과정은 다음과 같다.

① 암호화 중간 단계에서 생성된 워드(W_0, W_1, W_2, W_3)를 아래와 같이 각각 섞는다.

- W_0 = W_0의 C_1 || W_0의 C_5 || W_1의 C_1 || W_1의 C_5 || W_2의 C_1 || W_2의 C_5 || W_3의 C_1 || W_3의 C_5
- W_1 = W_0의 C_2 || W_0의 C_6 || W_1의 C_2 || W_1의 C_6 || W_2의 C_2 || W_2의 C_6 || W_3의 C_2 || W_3의 C_6
- W_2 = W_0의 C_3 || W_0의 C_7 || W_1의 C_3 || W_1의 C_7 || W_2의 C_3 || W_2의 C_7 || W_3의 C_3 || W_3의 C_7
- W_3 = W_0의 C_4 || W_0의 C_8 || W_1의 C_4 || W_1의 C_8 || W_2의 C_4 || W_2의 C_8 || W_3의 C_4 || W_3의 C_8

② 섞여진 각 W_0, W_1, W_2, W_3워드를 128비트 라운드 키(RKey_7)와 G 함수의 연산으로 암호화 과정을 재차 수행한다.

③ 암호화 하위 단계 4라운드를 수행하고 난 후 각 W_0, W_1, W_2, W_3워드를 연접하여 1 블럭을 저장하고, 마지막 블럭의 암호화가 수행되었는지를 판단한 다음, 끝이 아니면 암호화 상위단계부터 하위단계까지 반복하고, 끝이면 암호화의 모든 과정을 마친다.

1) G 함수 설계

암호화 하위 단계에서 수행되는 G함수는 아래의 수식으로 설계하며, 설계한 G 함수 구조는 그림 5와 같다.

- G 함수 설계 수식

C1 = P1 ⊕ C8
C2 = S1(P1 ⊕ K1) ⊕ P2
C3 = C2 ⊕ P3
C4 = S2(P3 ⊕ K2) ⊕ P3
C5 = C4 ⊕ P5

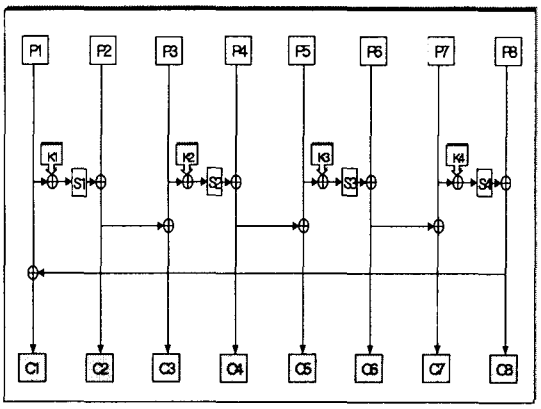


그림 5. G 함수 구조

C6 = S3(P5 ⊕ K3) ⊕ P6
C7 = C6 ⊕ P7
C8 = S4(P7 ⊕ K4) ⊕ P8

위의 수식에서 사용한 P1, P2, · · ·, P8은 매 라운드마다 각각의 W_0, W_1, W_2, W_3 워드를 8비트로 분리하여 입력되는 값이다. S1, S2, S3, S4는 S-Box 번호이며 K1, K2, K3, K4는 라운드 키 값이다.

3.2 복호화 알고리즘 설계

제안 블록 암호 알고리즘의 복호화는 암호화 단계의 역순으로 하위 단계, 중간 단계, 상위 단계 순으로 진행한다. 먼저 복호화 알고리즘을 설계하기 위하여 암호화 중간단계에서 사용된 F함수와 암호화 하위 단계에서 사용된 G 함수의 역구조를 설계하여 전체 복호화 알고리즘을 설계하였다. 먼저 암호화에 사용했던 키를 입력받아 다음과 같이 복호화 알고리즘을 설계한다.

• 복호화 상위 단계(암호화 상위 단계의 역순)

먼저 암호화가 128비트로 되었는지 아니면 256비트로 되었는지 판단한다. 128비트인 경우에는 64비트 워드 2개로 분할하여 복호화를 수행하고, 256비트인 경우에는 64비트 워드 4개로 분할한 다음 복호화를 수행한다. 복호화를 수행하는 과정은 다음과 같다. RKey_7을 이용하여 암호화 하위 단계의 역순으로 4 라운드의 G 함수 복호화부터 1라운드의 G함수 복호화까지 수행한다. 여기서 생성된 W_0, W_1, W_2, W_3 워드를 암호화의 하위 단계에서의 수행과정 ①의 역순으로 섞어 최종적으로 W_0, W_1, W_2, W_3 워드를 생성하여 복호화 중간 단계로 넘긴다.

• 복호화 중간 단계(암호화 중간 단계의 역순)

암호화 중간 단계의 역순으로 마지막 8라운드의 F함수는 RKey_8를 이용하여 복호화하고 계속적으로 1라운드의 F함수를 RKey_1로 복호화하여 복호화 중간단계를 수행한다.

• 복호화 하위 단계(암호화 상위 단계의 역순)

복호화 하위 단계에서는 복호화 계속 유무를 판정한다. 복호화가 마지막이면 복호화된 평문과 RKey_0를 XOR 연산하여 평문을 생성하고 복호화 과정을 모두 마친다. 마지막이 아니면 다시 복호화 상위단계로 분기하여 복호화를 마칠 때까지 계속한다.

4. 성능 평가

일반적으로 많이 사용되는 암호 해독 방법으로는 한 쌍의 입력 차분에 대한 출력 차분의 비균일성을 이용해 해독하는 차분 해독(공격 또는 분석) 방법과 특정 출력 비트들의 선형 결합과 특정 키(또는 입력 비트)들의 선형 결합과의 연관성을 이용해 해독하는 선형 해독 방법 등이 있다[4,7].

본 논문에서는 기존의 암호 알고리즘 및 제안 알고리즘에 대한 차분 특성 확률 비교에 의한 안전성 비교 및 암호화·복호화 처리 속도 비교, 그리고 암호 알고리즘 강도 비교에 의한 성능 평가 작업을 수행한다.

4.1 안전성 평가

암호 알고리즘의 차분 특성 확률 비교에는 크게 키 전수 탐색(exhaustive key search) 조사 수(2^{128}: 2^{128}인 경우는 확률 값)를 비교하는 방법과 기존 암호 알고리즘의 차분 특성 확률 값과 비교하는 방법이 있다.

차분 해독은 암호 알고리즘에 따라 분석 방법이 다를 수 있다. 본 알고리즘의 차분 분석을 위하여 하나의 F함수로 전체 차분 특성을 구한다.

그림 6에서 F 함수의 차분 특성을 알기 위하여 우선 해독자의 입장에서 공격하기 쉬운 경로를 찾는다. 평문은($P1$, $P2$, …, $P8$)이고 암호문을 ($C1$, $C2$, …, $C8$)이라 할 때,. 해독자의 공격은 P1, P3, P5, P7과 같은 홀수 워드의 공격과 P2, P4, P6, P8의 짝수 워드의 공격으로 나누어 볼 수 있다.

홀수 워드인 P3 공격의 경우, 우선 $P3$에만 0이 아닌 차분을 주고, 그외 모든 평문 워드 ($P1$, $P2$, …, $P8$)에 고정된 차분 0으로 대입하고 경로를 따라가면 라운드 키인 K6을 만난다. 여기서 P3와 K6을 XOR한 후 이 값을 S4(S-Box)에 입력하면, S4는 $\alpha \neq 0$ 값을 생성한다. 이 값을 가지고 계속 경로를 따라가면서 계산할 경우, $P4$도 이 값의 영향을 받아 $\alpha \neq 0$값으로 차분되고, $P6$으로 가는 경로 역시 $\alpha \neq 0$값으로 된다. 따라서 모든 평문들이 $\alpha \neq 0$ 인 차분을 갖게 되어 전 워드의 해독이 어려워진다.

다음으로, 입출력 차분을 갖는 평문쌍 P, P^*과 그것에 대응되는 암호문 쌍 C, C^*를 얻는다고 가정하

고, 그림 6의 F함수에 대한 차분해독을 한다. S-Box의 입력 차분 α에 대하여 출력 차분β를 갖게 될 확률이 p_1이라고 하고, 입력차분 β에 대하여 출력차분α를 갖게 될 확률이 p_2라 하면, S-Box는 전단사 함수 $x^n(0\le n\le 255)$ 이므로$\beta\neq 0$된다. 따라서, (0, β,0,0,0,0,0,0)이 될 확률은 p_1p_2가 된다. 임의의γ, ε, η에 대해서는(γ, ε, γ, η,0,0,0,0)가 되며, 이 때 ε은 임의의 1바이트(8비트) 값이 되고, γ, η는 임의의 0이 아닌 1바이트 값이 된다.

따라서 S-Box의 입출력 차분을 알고 있으므로, S-Box의 입력차분 γ를 만족하는 2^8개의 입력 순서쌍을 모두 조사하여 S-Box에서 출력차분ε을 만족하는 모든 순서쌍을 구한다.

예를 들어 이것을 만족하는 순서쌍이 두 개 있다고 하면, 이것은 식 6과 같이 표현할 수 있다.

$$(x_1, x_1\oplus\gamma)\rightarrow(y_1, y_1^*), y_1\oplus y_1^*=\varepsilon\oplus\beta$$
$$(x_2, x_2\oplus\gamma)\rightarrow(y_2, y_2^*), y_2\oplus y_2^*=\varepsilon\oplus\beta \quad (6)$$

위 식 6을 만족하는 순서쌍이 두 개인 경우는 $x_2=x_1\oplus a$의 관계가 성립할 때이다. 이 사건이 일어날 확률이 p_1p_2이므로 $(p_1p_2)^{-1}$개의 평문쌍이 있다면 그 중에 1개의 순서 쌍이 존재한다고 볼 수 있다. 따라서 $(p_1p_2)^{-1}$의 두 배 정도의 평문쌍을 이용한다면 위와 같은 해독이 가능하다. 여기서의 자료 복잡도(data complexity)는 $2(p_1p_2)^{-1}$이 된다.

S-Box 차분 분포표에서 만족되는 평문쌍을 right pair라하고, 그렇지 않은 것을 wrong pair라고 부른다.

여기서 '확실히' wrong pair인 것들은 모두 버린다. 이것을 filtering이라고 하며, 임의의 순서쌍이 filtering을 통과할 확률은 식 7과 같다.

$$(2^8-1)^{-1}\cdot 2^{-8}\approx 2^{-16} \quad (7)$$

위의 식 7에서 S-Box 차분 특성상 2개의 쌍으로 나타남으로 실제 이 값은 2^{-8}이 된다.

위의 결과와 덧붙여 실제로 해독에 이용하게 되는 순서쌍의 개수는 식 8과 같다.

$$(p_1p_2)^{-1}\cdot filltering_{확률}=2^{-12}\cdot 2^{-8}=2^{-20} \quad (8)$$

따라서, 본 논문의 F 함수의 차분 특성 값 2^{-20}을 얻을 수 있다.

또한, 2^{-20}이 차분 해독으로 가능한지 검증해야 한다. 이 해독에 대한 구현 개념은 키 후보의 개수 만큼 카운터를 설치하고, 하나의 평문쌍-암호문쌍이 키 후보를 제시할 때마다 각각에 해당되는 카운터가 1

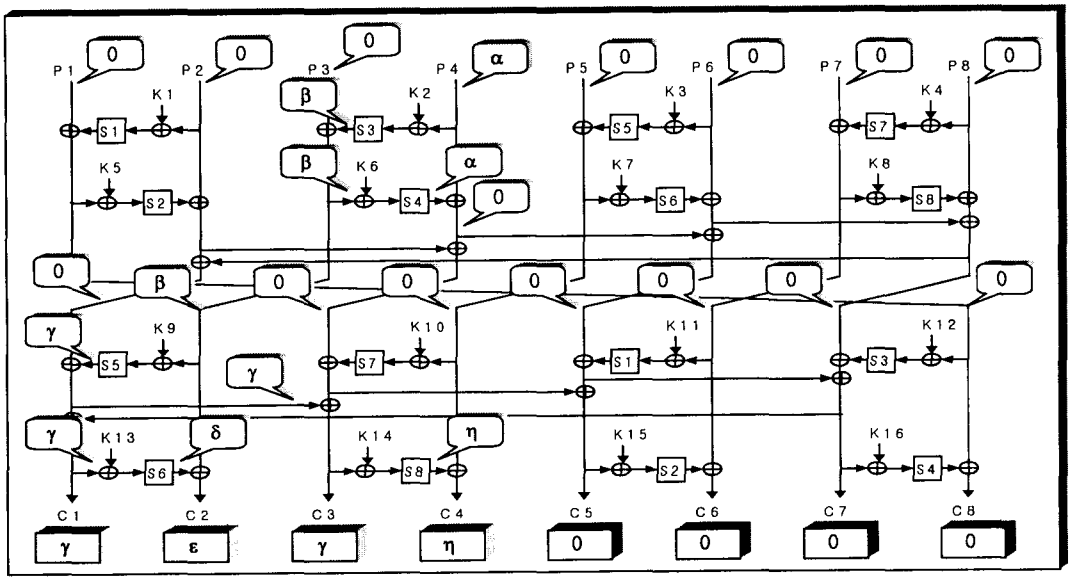


그림 6. F함수 차분 해독 참고도

씩 증가한다. 최종적으로, 가장 높은 값을 갖는 키 후보를 실제 키로 선택한다. 즉, 실제 키는 키 후보로 가장 많이 제시되는 값이다. 이것이 성공하려면 실제 키는 다른 키보다 월등히 많이 카운트되어야 한다. 이 비율을 계산한 것이 S/N(Signal to Noise)이며, 식 9와 같이 계산된다.

$$S/N=\frac{mp}{\frac{m\alpha\beta}{2^k}}=\frac{2^k p}{\alpha\beta}=\frac{2^6\cdot 2^{-2}}{1\cdot 2^{-8}}=2^2=4 \quad (9)$$

단, m : 해독에 이용한 평문 쌍의 개수

2^k : 키 후보의 개수

α : 임의의 평문쌍-암호문 쌍이 제시하는 키 후보의 개수 평균

β : filtering후 살아남는 평문 쌍들의 비율

식 9에서 첫 번째 등식의 분자는 실제 키가 카운트되는 회수이고, 분모는 임의의 키 후보가 평균적으로 카운트되는 회수이다.

여기에서 S/N이 4임으로 차분 해독이 성공적이라고 말할 수 있다. 또한, 본 논문의 F 함수 한 개의 차분 특성은 2^{-20}이 된다. 따라서 본 논문의 암호화 중간 단계에서 8라운드로 F 함수를 운영하므로 차분 특성 확률은 2^{-160}이 된다.

SEED의 자체 개발 및 분석 보고서에 의하면 15라운드의 확률값이 $2^{-137.6}$이고, 실측은 $2^{-137.9}$이다[13]. AES는 'The Rijndael Block Cipher AES Proposal'에서 4라운드에 도달전에 2^{-150}이 넘고 8라운드 도달전에 2^{-300}이 넘는 것으로 나타나 있다[12]. DES는 Biham과 Shamir에 의해 16라운드의 차분 값이 2^{-47}임을 증명하였다[5].

본 논문에서 제안하는 알고리즘과 기존 알고리즘과의 차분 특성 확률 비교 결과는 표 2와 같다.

제안 알고리즘의 경우, AES보다 차분 특성확률이 높게 나타났지만, 키의 전수 탐색 조사 수인 2^{-128}보다 차분 특성 확률이 낮고, 또한 SEED와 DES의 차분 특성 확률보다 낮음을 알 수 있었다. 따라서 SEED와 DES보다 안전성이 다소 높음을 알 수 있다.

4.2 처리 속도

암호화·복호화 처리 속도 비교는 기존 암호 알고리즘의 암호문·복호문 산출 결과에 따른 실제 데이터 처리량에 초점을 맞추었다. 컴퓨터가 처리하는데 걸리는 clocks 단위로 10KB 자료를 100회 수행하여 평균을 정리하였으며, 분석을 위한 시스템 구현 환경은 다음과 같다.

- 시스템 구현 환경
  - 시스템 : Pentium Ⅲ 800MHz
  - RAM : 256MB
  - 운영체제 : Windows 2000 Server
  - 소프트웨어 : Borland C++ Builder 6

표 2. 암호 알고리즘 차분 특성 확률 비교

순	알고리즘 명	키 길이	설계자 권장 라운드	차분특성 비교 라운드	차분 특성 확률
1	AES	128,192,256	10,12,14	8	2^{-300}
2	SEED	128	16	15	2^{-137}
3	DES	64	16	16	2^{-47}
4	제안알고리즘	128	8	8	2^{-160}

제안 알고리즘의 암호화 속도는 0.014초로 나타났다. AES에 비해서는 0.002초 느리지만, SEED와 DES에 비해서는 0.002초 빠르고, 3중-DES에 비해서는 0.031초 빠른 것으로 나타났으며, 복호화 속도는 암호화 속도와 비슷하게 나타났다.

표 3. 암호화 처리 속도 비교

순	알고리즘 명	처리clocks수	암호화속도(sec)
1	AES	11.205	0.012
2	SEED	15.631	0.016
3	DES	15.241	0.016
4	3중-DES	42.917	0.045
5	제안 알고리즘	13.524	0.014

4.3 암호화의 강도

암호화의 강도는 키를 해독하는데 걸리는 소요 시간(CPU 속도 : 나노초)이 어느 정도인가에 따라 판별하였다. 암호 알고리즘의 암호화 강도 산정은 식

10과 같으며, 각 알고리즘 별 계산 결과는 표 4와 같이 산출되었다.

$$\frac{\text{차분 특성 확률의 50\% } \times \text{ 컴퓨터 속도}}{\text{1시간에 대한 초 } \times \text{ 24시간 } \times \text{ 365일}}$$

$$= \frac{\frac{2^{160}}{2} \times \frac{1}{10^{12}}}{3600 \times 24 \times 365} \fallingdotseq 2.3^{28}\text{년} \qquad (10)$$

표 4. 암호화 강도 비교

순	알고리즘 명	키 길이	소요 시간(년)
1	AES	128, 192, 256	3.2^{70}
2	SEED	128	2.7^{21}
3	DES	64 (128)	2.2^{6} (4.4^{12})
4	제안 알고리즘	128	2.3^{28}
5	키 전수 탐색	128	5.3^{18}

※ ( )는 키 길이를 128로 환산한 값

제안 알고리즘의 암호화 강도가 2.3^{28}으로 측정되어 AES 보다는 강도가 약하지만 SEED나 DES보다 강도가 다소 나음을 알 수 있다.

4.4 시뮬레이션

대칭키 암호 알고리즘에서는 송신자와 수신자가 제 3자에게 공개되지 않은 비밀키를 공유하고 있어야 한다. 이 비밀키는 보통 자료의 전송 전에 결정되거나 공개키 암호화 시스템에서 공유한다고 가정하고 시뮬레이션을 수행하였다.

- 암호화에 사용된 자료
  - 성명 3자, 영문자수 6자
  - 카드번호 숫자 16자
  - 유효기간 월년 숫자 4자
  - 추가보안번호 숫자 3자
  - 내용별 공백 3자를 비롯하여 총 32자

그림 7은 서버에서 ID abcd00~abcd99의 100명에게 자료를 암호화하여 송신한 내용을 나타낸다.

본 논문의 암호 알고리즘으로 성명이 "홍길동"이고, 카드 번호가 "1234223432344234", 카드의 유효기간이 "0312", 추가 보안번호가 "019" 인 데이터를 100명의 사용자에게 송신한 암호는 그림 7의 우측 부분

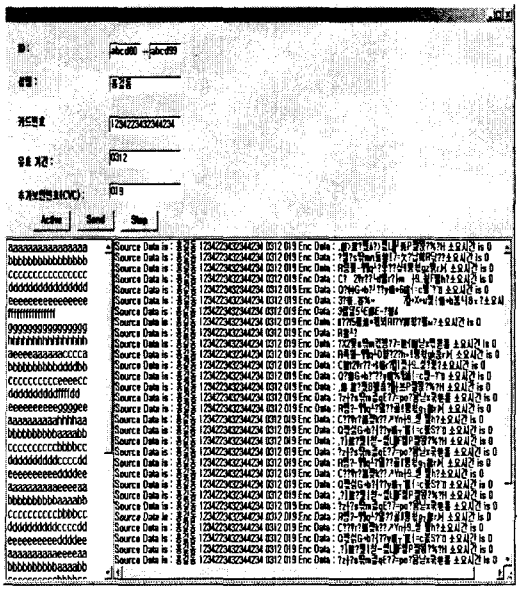


그림 7. 송신 암호문

처럼 모두 다르다. 따라서, 해독자의 공격에 강하게 대응할 수 있음을 알수 있다.

그림 8은 서버가 자료를 암호화하여 송신한 내용에 대해 ID abcd00에서 수신한 결과를 나타내고 있다.

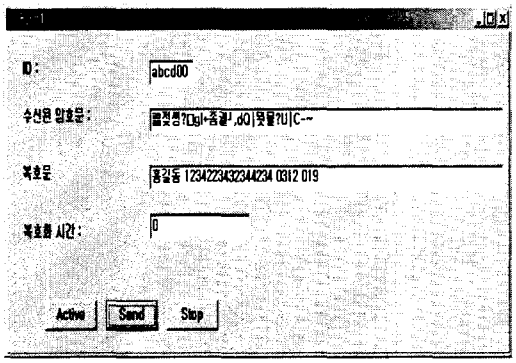


그림 8. 수신된 암호문과 복호문

여기에서 복호화된 평문은 그림 7의 송신 내용과 일치함을 알 수 있으며, 복호화 시간도 1초 이내여서 제안 알고리즘의 효율성을 입증할 수 있었다.

따라서, 본 논문의 제안 알고리즘은 자료의 암호화와 복호화 처리시간이 짧고 암호문의 해독이 어려워 전자상거래 보안 등의 다양한 곳의 정보보호에 응용될 수 있을 것으로 본다.

5. 결 론

기존의 블럭 암호 알고리즘들은 암호화 키 값에 변화를 주지 않고 매 블럭의 라운드 함수에 적용하여 암호화되도록 설계되어 있어서 차분 해독이나 선형 해독 등에 의해 평문이나 암호화 키가 노출되기 쉬운 단점을 가지고 있다. 이러한 단점을 보완하기 위하여 본 논문에서는 가변 암호화 키가 적용되어 운영되는 F 함수를 제안하고, 이 제안 F함수에 의해 암호화 과정을 수행하여 강도가 향상된 블럭 암호 알고리즘임을 보였다.

기존의 블럭 암호 알고리즘과 제안 블럭 암호 알고리즘에 대해서 차분 특성 확률 비교에 의한 안전성 분석과 암·복호화 처리 속도, 암호 알고리즘 강도를 비교한 결과, 제안 블럭 암호 알고리즘은 AES보다는 성능이 다소 떨어지지만, SEED나 DES보다는 다소 나은 것으로 평가되었다.

시뮬레이션 결과 제안 알고리즘은 자료의 암호화와 복호화 처리시간이 짧고 암호문의 해독이 어려워 전자상거래 보안 등의 다양한 곳의 정보보호에 응용될 수 있으리라 본다.

참 고 문 헌

[1] H. Feistel, W. A. Notz, J. L. Smith, "Some Cryptographic Techniques for Machine-to-Machine Data Communications", Proc. on the IEEE, V. 63, N. 11, pp. 1545-1554, 1975.

[2] J. Daemen, "Cipher and Hash Function Design", Ph.D. thesis, Katholieke Universiteit Leuven, Mar. 1995.

[3] http://www.rsasecurith.com/rsalabs/faq,1998.

[4] H. Heys, S. Tavares, "Substitution-permutation networks resistant to differential and linear cryptanalysis", J. Cryptology, 9(1), 1996.

[5] E.Biham and A.Shamir. "Differential cryptanalysis of the full 16-round DES". In Ernest F.Brickell, editor. proc. CRYPTO 92, pp. 487-496. Springer-Verlag, 1992.

[6] 성재철,이상진,김종수,임종인, "Skipjack 구조에 DC 및 LC의 안전성 증명", 한국정보보호학회 논문지, 제10권 제1호, pp.13-22, 2000.

[7] M. Kanda, Y. Takashima, T. Matsumoto, K. Aoki and K. Ohta, "A Strategy for Constructing Fast Functions with Practical Security against Differential and Linear Cryptanalysis", Proceedings of SAC'98, 1998.

[8] Skipjack and KEA Algorithm Specifications, Version 2.0, 29 May 1998, http://csrc.nist.gov/encryption/skipjack-kea.htm.

[9] Gilles Brassard. "Modern Cryptoloy - A tutorial", Lecture Notes on Computer Science, Vol 325. 1988.

[10] Douglas R. Stinson. "Cryptography Theory and Practice". CRC Press, Inc., Boca Rayton, 1995.

[11] Data Encryption Standard. "Fedral Information Processing Standards (FIPS)" Publication 46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C, January, 1977.

[12] Joan Daemen, Vincent Rijmen, "AES Proposal: Rijndael", 1999

[13] 한국정보보호진흥원, "128 비트 블럭 암호 알고리즘 (SEED) 개발 및 분석 보고서", 1998.

[14] N. Heintze, D. Tygar, "Model Checking Electronic Commerce Protocols", ARPA contract F33615-93-1-1330, Nov. 1995.

[15] 김윤정, 조유근, "블럭 암호 알고리즘을 위한 추적 불가능한 동적 키를 갖는 연산모드", 정보과학회 추계학술 발표 논문집, 26(2), 1999c.

[16] NIST, "2nd AES Conference", AES Round2 Information, 1999.

정 홍 섭

1997년 경남대학교 대학원 컴퓨터 공학과(석사)

2001년 경남대학교 대학원 컴퓨터 공학과(박사과정수료)

1999년~현재 경남대학교 평생교육원 연구원

관심분야 : 멀티미디어, 암호화 설계, 영상처리, VOD, 정보보호

이 창 두

1995년 경남대학교 산업대학원 컴퓨터 공학과(석사)
2002년 경남대학교 대학원 컴퓨터 공학과(박사과정수료)

관심분야 : 암호화 설계, 컴퓨터교육, 멀티미디어, 전자상거래 보안, 정보보호

박 규 석

1980년 중앙대학교 대학원 전자계산학과(석사)
1988년 중앙대학교 대학원 전자계산학과(박사)
1982년~현재 경남대학교 정보통신공학부 교수
1992년~1996년 경남대학교 전산정보원 원장
1995년~1996년 한국 정보과학회 이사, 영남지부장
1999년~2002년 경남대학교 정보통신연구소 소장
2002년~현재 경남대학교 산업대학원 원장
2002년~현재 한국 멀티미디어 학회 회장
관심분야 : 분산처리 시스템, 정보통신 소프트웨어, 멀티미디어 시스템