通信情報保護學會論文誌
第 8 卷 第 4 號, 1998. 12

16라운드 SKIPJACK에 대한 입출력 변화 공격

박 성 모*, 이 상 진*

Differential Cryptanalysis of 16 round SKIPJACK

Park Sung Mo*, Lee Sang Jin*

요 약

본 논문은 그동안 알려지지 않았던 SKIPJACK 알고리즘의 암호화 방식을 소개하고 입출력 변화 공격 관점에서의 공격가능성을 검토한다. 기본적인 특징으로 키 설정시간이 거의 없고, 적은 양의 메모리를 이용하여 라운드 함수의 비선형 논리를 설계하였으며, 전체적으로 8비트 프로세서로 구현하는 것이 용이하다. 라운드 함수 G의 DC특성으로부터 16라운드 SKIPJACK은 2^{24}개의 선택 평문이 있으면 2^{37}의 계산량으로 키를 찾을 수 있다.

Abstract

SKIPJACK is a secret key encryption algorithm used by the US government in the Clipper chip, which had been calssified since its introduction in 1993. Recently, its structure was made public in 1998. It was designed to be evaluated easily in 8-bit processors with low memory usage. We review the encryption method and summerize its security against differential cryptanalysis. First, we introduce the attack by Biham for 16-round SKIPJACK which uses the differential characteristic of G function. And then, we give more improved attack using only 2^{24} chosen plaintexts.

Ⅰ. 개 요

SKIPJACK은 미국 NSA(National Security Agency)에서 개발한 Clipper Chip[1]에 내장되는 블럭 알고리즘으로 알고리즘 형태 및 구조는 비밀로 분류되어 있다가 최근에 공개되었다[2].

SKIPJACK은 Fortezza Card에 칩 형태로 구현되었으며, 소트프웨어로 구현되는 것을 의도적으로 막아왔다. NSA가 1985년에 개발에 착수하여 1990년에 평가를 마쳐 공표한 SKIPJACK의 사양은 다음과 같다.

* 한국전자통신연구원

- 반복 구조를 갖는 블럭 암호
- 입출력 크기 : 64비트
- 키 크기 : 80비트
- 알고리즘 동작 : ECB, CBC, 64비트 OFB, 8, 16, 32, 64 비트 CFB
- 라운드 수 : 32라운드

본 고는 최근에 발표된 SKIPJACK의 구조를 소개하고,입출력 변화 공격(Differential Cryptanalysis, DC)으로 분석을 시도한다. Biham[3]은 12라운드 입출력 변화 특성을 이용하여 2^{55}개의 선택 평문 쌍을 가지고 2^{32}의 계산량으로 키를 찾았는데, 본 고에서는 이를 개선하여 4라운드 입출력 변화 특성을 이용하여2^{24}개의 선택평문을 이용하여 2^{32}의 계산량으로 키를 찾을 수 있음을 보인다.

2. 기본 구조

SKIPJACK은 4개의 워드(1워드는 16비트)를 두개의 변환 규칙(변환규칙 A, 변환규칙 B)에 의해 암호화한다. 변환 규칙 A와 B는 16비트를 입출력으로 하는 전단사 함수 G를 비선형 논리로 사용하며, G는 4라운드 Feistal 구조를 가지며, 내부에 8비트 입출력을 가지는 총 4개의 F를 사용한다.

2.1 변환 규칙

단계 k에서의 4개 워드를 $(w_1^k, w_2^k, w_3^k, w_4^k)$하면 단계 k에서 $k+1$로의 변환과정은 다음과 같다. 이때 입력 변수로 counter가 추가되고, 키 요소는 G 함수에서 입력된다.

변환 규칙 A : 그림 1

1. $w_1^{k-1} = G^k(w_1^k) \oplus w_4^k \oplus counter^k$
2. $w_2^{k+1} = G^k(w_1^k)$
3. $w_3^{k+1} = w_2^k$
4. $w_4^{k+1} = w_3^k$
5. $counter^{k+1} = counter^k + 1$

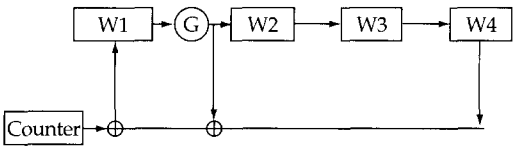


그림 1. SKIPJACK의 변환규칙 A

변환 규칙 B : 그림 2

1. $w_1^{k+1} = w_4^k$
2. $w_2^{k+1} = G^k(w_1^k)$
3. $w_3^{k+1} = w_1^k \oplus w_2^k \oplus counter^k$
4. $w_4^{k+1} = w_3^k$
5. $counter^{k+1} = counter^k + 1$

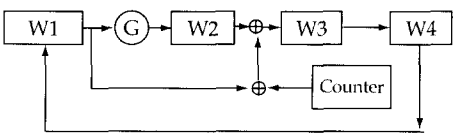


그림 2. SKIPJACK의 변환규칙 B

2.2 암호화 방식

SKIPJACK의 입력 4워드(64비트)는 변환규칙 A, B를 이용하여 다음과 같이 암호화 한다.

1. 4 워드 입력을 $(w_1^0, w_2^0, w_3^0, w_4^0)$라 하자.
2. $counter^0 = 1$
3. 변환 규칙 A로 8번 변환
4. 변환 규칙 B로 8번 변환
5. 변환 규칙 A로 8번 변환
6. 변환 규칙 B로 8번 변환
7. 4 워드 $w_1^{32}, w_2^{32}, w_3^{32}, w_4^{32}$를 출력

변환 규칙에서 입력 4워드는 라운드마다 1 워드만큼 순환하여 매 4라운드에는 원래의 자리에 돌아오므로 실제 구현시에는 그림 4와 같이 실행하는 것이 수행 속도를 빠르게 한다. 이 그림은 SKIPJACK의 16라운드까지를 블럭형태로 표시한 것으로 라운드 키는 표시하지 않았으며, 그림의 오른쪽으로 부터 입력되는 값은 변환규칙 A, B에서 사용된 counter 값이다.

2.3 G 함수

G 함수는 입력 16비트를 16비트로 출력하는 전단사 함수로 구체적인 꼴은 4라운드의 Feistal 구조이며 내부의 F함수는 8비트를 8비트로 출력하는 전단사 함수로 256 바이트로 구성된 테이블로 표시한다(표 1). 이 표는 입력 8비트가 16진법으로 xy라 하면 출력 비트는 x행 y열의 값을 의미한다.

F함수는 어떠한 대수적인 방정식에 의해 생성되었는지는 알 수 없지만, 유한체 $GF(2^8)$위에서 정의된 g^x 형태와 유사한 특징이 있으며, 고정점은 존재하지 않는다. SKIPJACK의 유일한 비선형 요소로 안전성을 좌우하는 바 이에 대한 다양한 검토가 필요하다.

표 1 . F함수 테이블, F(69) = 00

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	a3	d7	09	83	f8	48	f6	f4	b3	21	15	78	99	b1	af	f9
1x	e7	2d	4d	8a	ce	4c	ca	2e	52	95	d9	1e	4e	38	44	28
2x	0a	df	02	a0	17	f1	60	68	12	b7	7a	c3	e9	fa	3d	53
3x	96	84	6b	ba	f2	63	9a	19	7c	ae	e5	f5	f7	16	6a	a2
4x	39	b6	7b	0f	c1	93	81	1b	ee	b4	1a	ea	d0	91	2f	b8
5x	55	b9	da	85	3f	41	bf	e0	5a	58	80	5f	66	0b	d8	90
6x	35	d5	c0	a7	33	06	65	69	45	00	94	56	6d	98	9b	76
7x	97	fc	b2	c2	b0	fe	db	20	e1	eb	d6	e4	dd	47	4a	1d
8x	42	ed	9e	6e	49	3c	cd	43	27	d2	07	d4	de	c7	67	18
9x	89	cb	30	1f	8d	c6	8f	aa	c8	74	dc	c9	5d	5c	31	a4
ax	70	88	61	2c	9f	0d	2b	87	50	82	54	64	26	7d	03	40
bx	34	4b	1c	73	d1	c4	fd	3b	cc	fb	7f	ab	e6	3e	5b	a5
cx	ad	04	23	9c	14	51	22	f0	29	79	71	7e	ff	8c	0e	e2
dx	0c	ef	bc	72	75	6f	37	a1	ec	d3	8e	62	8b	86	10	e8
ex	08	77	11	be	92	4f	24	c5	32	36	9d	cf	f3	a6	bb	ac
fx	5e	6c	a9	13	57	25	b5	e3	bd	a8	3a	01	05	59	2a	46

2.4. 키스케줄 방식

키 10 바이트 (80비트)를 4바이트씩 차례로 사용한다. 즉, 10바이트를 $K(0)$, $K(1)$, …, $K(9)$라고 하면 k-라운드에서 사용되는 라운드 키는 $K(4k)$, $K(4k+1)$, $K(4k+2)$, $K(4k+3)$이며 그림 3. 괄호안의 숫자는 mod 10 연산으로 계산한다.

키 스케줄이 단순히 키 바이트를 4바이트씩 순환적으로 선택하기 때문에 키 설정 시간은 필요없다. 한편 고정된 10바이트의 키를 4바이트씩 나누어 반복적으로 사용하기 때문에 1, 6, 11, 16, 21, 27, 32 라운드에서의 라운드 키가 같아지는 특징이 있다.

3. 입출력 변화 공격

SKIPJACK의 비선형 요소를 좌우하는 F 함수에 대한 입출력 변화 특성은 다음과 같은 특성이 있다.

- 최대값은 12이며 약 40%가 0아닌 값을 가진다.
- 01(hex) → 01(hex)로의 확률이 2/256이다. 이는 고정된 1비트 변화에 대한 확률인데, 이 확률이 크면 SKIPJACK과 같이 비트위치 변환이 없는 알고리즘에서는 쉽게 입출력 변화특성이 구해지는 취약점이 있다.

3.1 G 함수의 DC 특성

a와 b를 바이트 변수라 하고 $a \rightarrow b$가 F 함수에서 0아닌 확률을 가진다고 하자. G의 입출력 변화 특성을 입력변화가 $(a,0)$, $(0,b)$ 그리고 (a,b) 형태의 경우에 대해 출력 변화 특성을 살펴보자.

그림 3에서 보는 바와 같이 G 함수는 4개의 F 함수를 반복하여 사용하는데, 이러한 F 함수의 출력변화를 중심으로 G의 입출력변화를 살펴보자. 예를들어, 입력 변화가 $(a,0)$, 출력변화가 $(0,b)$인 경우, F함수의 출력변화를 나타내는 중간 변수들을 표시해 보면

$$(a,0) \rightarrow (a,0) \rightarrow (a,b) \rightarrow (0,b) \rightarrow (0,b)$$

가 된다.

이를 쉽게 나타나기 위해 G함수에 있는 4개의 F 함수를 각각 $F1$, $F2$, $F3$ 그리고 $F4$라 하자. 그리고 입력변화가 a이고 출력변화가 b인 F 함수의 입출력 변화를 $a \rightarrow b$ 혹은 $b \leftarrow a$로 표시하자.

입력변화 $(a,0)$, $(0,b)$, (a,b)에 대한 G함수의 입출력변화를 기술해 보면 (표 2)와 같다.

a, b가 다음과 같을 경우

$(a,0)(0,b)$가 $48/2^{16} = 2^{-10.42}$로 가장 큰 확률을 가짐을 알 수 있다(괄호안은 $a \rightarrow b$의 확률).

- $a = 52$, $b = f5$, (8/256)
- $a = f5$, $b = 52$, (6/256)
- $a = 77$, $b = 92$, (6/256)
- $a = 92$, $b = 77$, (8/256)

표 2. G의 입출력 변화 특성

$(a,0) \rightarrow (0,b)$					
P	a		0		
F1	a	←	0		
F2	a	→	b	a	→ b
F3	0	←	b	a	← b
F4	0	→	b		
C	0		b		
$(0,b) \rightarrow (a,b)$					
P	0		b		
F1	a	←	b	a	← b
F2	a	→	0	a	→ b
F3	a	←	0		
F4	a	→	b	a	→ b
C	a		b		
$(a,b) \rightarrow (a,0)$					
P	a		b		
F1	0	←	b	a	← b
F2	0	→	b		
F3	a	←	b	a	← b
F4	a	→	0	a	→ b
C	a		0		

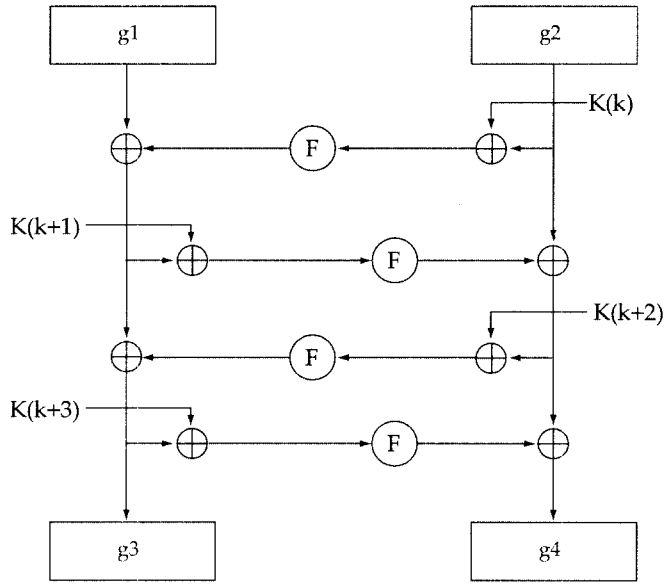


그림 3. SKIPJACK의 함수

3.2 16라운드 SKIPJACK에 대한 Biham의 입출력 변화 공격

앞 절에서 기술한 G 함수의 입출력변화 특성(표 2)을 이용하여 16라운드 SKIPJACK에 대한 입출력변화를 살펴보자. 우선 그림 4에서 입력변수인 16비트 4개의 변수를 좌측으로 부터 $w1$, $w2$, $w3$, $w4$라 하자. 가능한 모든 조합을 검색하여 이중에서 최적인 것을 구하면 입력 변화가 $(a,0)$, $(a,0)$, $(0,0)$, $(0,b)$일때 이고, 이 때 출력의 변화는 $(0,b)$, $(0,b)$, $(a,0)$, $(0,0)$이다. 중간값을 모두 표시하면 (표 3)와 같다. 이 중에서 G의 입력변화가 $(0,0)$이 아닌것은 1, 4, 8, 9, 13, 16 라운드 6개이며, 이 경우의 최적의 확률을 구해보면 $2^{-32.9}$이다.

위에서 구한 16라운드 입출력 변화 특성을 그대로 사용할 경우 필요한 선택평문쌍은약 2^{73}이며 현실적으로 불가능하다. 그러나 확률 $2^{-52.1}$로 성립하는 12라운드 입출력 변화 특성을 사용하면 공격이 가능하다. 이 방법은 12라운드의 입출력 변화 특성과 1라운드와 16라운드의 키가 같은 사실을 이용하여 32비트 키를 먼저 구한 다음 나머지 48비트는 전수조사를 통하여 구한다. 이를 구체적으로 살펴보면 다음과 같다.

우선 입력변화 $(a,0),(a,0),(0,0),(0,b)$에 대해 12라운드 이 후의 출력변화는 $(a,0),(a,0),(0,0),(0,0)$이다(표 3). 이 경우 1, 3, 8, 9 라운드에 있는 G의 입출력 변화 특성에 의해 전체 확률이 결정되며 이 확률을 구하면 $2^{-52.1}=48\times48\times384\times384/2^{80}$됨을 알 수 있다.

Biham의 공격 방법

1. 임의의 선택 평문쌍 $P\oplus P^*=(a,0),(a,0),(0,0),(0,b)$에 대응되는 암호문쌍$(C, C^*)$ 2^{55}개 획득
2. $C\oplus C^*=(u,v),(w,x),(a,0),(0,0)$의 형태가 아니면 이 쌍은 wrong pair로 이를 버리면2^{55}개의 선택평문 중 약 $2^{23}=2^{55}/2^{32}$개만 남게된다.

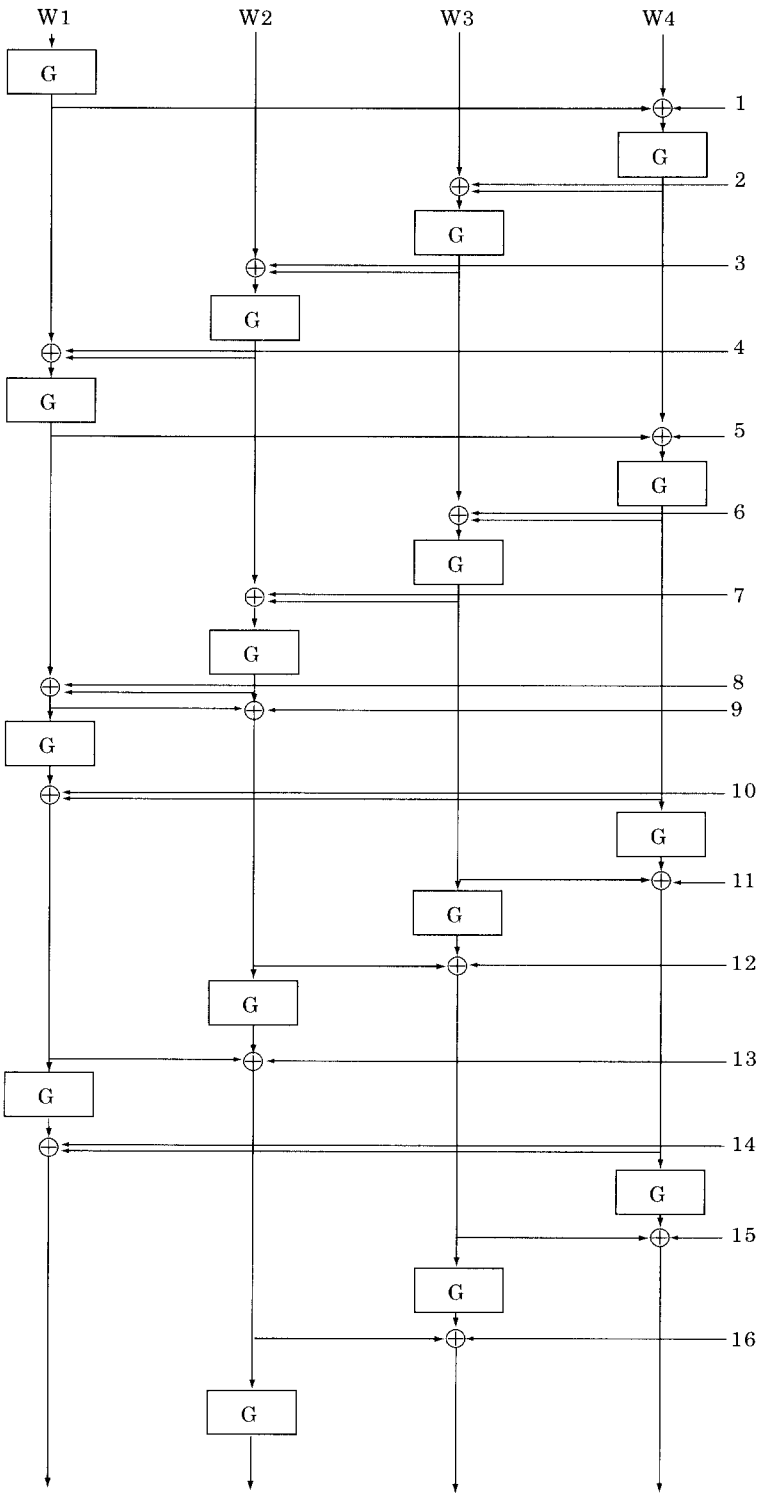


그림 4. SKIPJACK 16라운드

3. 1라운드와 16라운드의 키가 같으므로 나머지 2^{23}개의 평문쌍에 대해 다음과 같은 방법으로 라운드 키 32비트를 결정

표 3. 16라운드 SKIPJACK의 입출력 변화

16비트 워드	w_1	w_2	w_3	w_4
입력 변화	$(a,0)$	$(a,0)$	$(0,0)$	$(0,b)$
1 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
2 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
3 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
4 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
5 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
6 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
7 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
8 라운드	(a,b)	(a,b)	$(0,0)$	$(0,0)$
9 라운드	$(a,0)$	$(0,0)$	$(0,0)$	$(0,0)$
10 라운드	$(a,0)$	$(0,0)$	$(0,0)$	$(0,0)$
11 라운드	$(a,0)$	$(0,0)$	$(0,0)$	$(0,0)$
12 라운드	$(a,0)$	$(0,0)$	$(0,0)$	$(0,0)$
13 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
14 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
15 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
16 라운드	$(0,b)$	$(0,b)$	$(a,0)$	$(0,0)$
출력 변화	$(0,b)$	$(0,b)$	$(a,0)$	$(0,0)$

(a) 임의의 32비트 키에 대해 1라운드 평문쌍의 w_1을 각각 암호화하여 출력변화가 $(0,b)$인지 확인

(b) 암호문 쌍의 w_2를 복호화 하여 $(a,0)$인지 확인

(c) 이러한 특성을 만족하는 것의 개수가 4 이상이면 키로 출력

올바른 키가 아닐 경우 위의 테스트를 통과할 확률은 $2^{-26.4}=2^{-16}\times 2^{-10.4}$인데 평문쌍이 2^{23}개 밖에 없으므로 확률적으로 0이라 할 수 있다.

그러나 올바른 키일 경우 확률 $2^{-52.1}$로서 12라운드에서의 입출력 변화 특성이 성립하므로 2^{23}개 중 약 $8=2^{25-52.1}$개 정도가 테스트를 통과한다.

3.3 4라운드 입출력 변화 특성을 이용한 공격

Biham이 선택한 입력 변화에 대해 4라운드 출력 변화 특성만을 가지고 각 라운드별 출력 변화 특성을 추적하면 표 4와 같다.

표 4. 4 라운드 SKIPJACK의 입출력 변화

16비트 워드	w_1	w_2	w_3	w_4
입력 변화	$(a,0)$	$(a,0)$	$(0,0)$	$(0,b)$
1 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
2 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
3 라운드	$(0,b)$	$(a,0)$	$(0,0)$	$(0,0)$
4 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
5 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
6 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
7 라운드	$(0,0)$	$(0,b)$	$(0,0)$	$(0,0)$
8 라운드	(c,d)	(c,d)	$(0,0)$	$(0,0)$
9 라운드	(e,f)	$(0,0)$	$(0,0)$	$(0,0)$
10 라운드	(e,f)	$(0,0)$	$(0,0)$	$(0,0)$
11 라운드	(e,f)	$(0,0)$	$(0,0)$	$(0,0)$
12 라운드	(e,f)	$(0,0)$	$(0,0)$	$(0,0)$
13 라운드	(g,h)	(e,f)	$(0,0)$	$(0,0)$
14 라운드	(g,h)	(e,f)	$(0,0)$	$(0,0)$
15 라운드	(g,h)	(e,f)	$(0,0)$	$(0,0)$
16 라운드	(g,h)	(i,j)	(e,f)	$(0,0)$
출력 변화	(g,h)	(i,j)	(e,f)	$(0,0)$

표 4 에서 알 수 있듯이 4라운드의 출력 변화가 $(0,0),(0,b),(0,0),(0,0)$이면 w_2에 해당하는 15라운드 출력 변화와 w_3에 해당하는 16라운드

출력 변화가 같다. 그런데 4라운드 입출력 변화 특성의 확률은 $2^{-20.82}$가 되기 때문에 선택 평문쌍의 개수가 2^{24}개 정도 있으면 다음과 같은 공격이 가능하다.

4라운드 입출력 변화 특성을 이용한 공격

1. 임의의 선택 평문쌍 $P \oplus P^* = (a,0),(a,0),(0,0),(0,b)$에 대응되는 암호문 쌍 (C, C^*) 2^{24}개를 획득한다. ($a = 0x52$, $b = 0xf5$)
2. w_4에 해당하는 출력쌍이 '0'이 아니면 wrong pair가 되어 버린다. 이 작업으로 약 2^8 개의 암호문쌍이 남는다.
3. 2^{32}개의 가능한 키로w_2에 해당하는 암호문 쌍을 복호하여 XOR가 w_3에 해당하는 암호문쌍의 XOR와 일치하는 개수를 계산한다.
4. 위의 작업으로 올바른 키이면 평균 $2^{24-20.82} \approx 9$개 정도 계산되며, 틀린 키이면 거의 0일 것이다. 따라서 4이상 계산되면 올바른 키로 출력한다.

4. 결 론

본 논문은 최근 공개된 블럭 암호 알고리즘인 SKIPJACK에 대한 암호화 방식, 키스케줄 방식 등을 소개하고 입출력 변화 공격 관점에서의 분석가능성을 검토하였다.

키스케줄은 키 설정시간이 거의 없는 단순한 방법을 사용하여 4바이트의 라운드키를 생성한다. 또한 SKIPJACK의 라운드 함수내에 있는 비선형논리 F는 8×8의 테이블을 사용하여 적은 양의 메모리(256바이트)로 고속 구현 가능하도록 설계하는등 전반적으로 8비트 프로세서에 적합하도록 설계되었다.

일반적인 블럭 암호에 비해 특징적인 요소로 counter가 있는데 이 요소는 입출력 변화 공격과 선형근사 공격에 별다른 영향을 미치지 못한다.

한편 Biham[4]은 SKIPJACK에서 1, 16, 17, 32 라운드에서 총 4개의 XOR연산을 제거한 알고리즘(SKIPJACK-4XOR)이 2^{25}개의 선택 평문쌍에 의해 2^{48} 시간에 공격가능함을 보이고[4], 3개의 XOR을 제거한 알고리즘의 경우 2^{20} 시간에 2^9개의 선택 평문쌍에 의해 공격 가능함을 보였다[5].

16라운드 SKIPJACK에 대한 입출력 변화 공격으로 Biham은 12라운드 입출력 변화 특성을 이용하여 2^{55}개의 선택 평문 쌍을 요구하였으나, 본 고에서 제시한 방법을 사용하면 2^{24}개의 선택 평문 쌍만 있어도 충분함을 알 수 있다.

참 고 문 헌

[1] National Institute of Standards and Technology, NIST FIPS PUB 185, "*Escrowed Encryption Standard*", U.S. Department of Commerce, 1994.

[2] National Institute of Standards and Technology, "SKIPJACK and KEA Algorithm Specifications", http://csrc.nist.gov/encryption/skipjack-jea.htm, 1998.

[3] E. Biham, A. Biryukov, O. Dunkelman, and A. Shamir, "Initial Observations on the Skipjack Encryption Algorithm", *http://www.cs.technion.ac.il/ ~biham/Reports/Skipjack*, 1998.

[4] E. Biham, A. Biryukov, O. Dunkelman, and A. Shamir, "Cryptanalysis of Skipjack-4XOR", *http://www.cs.technion. ac.il/ ~biham/Reports/Skipjack*, 1998.

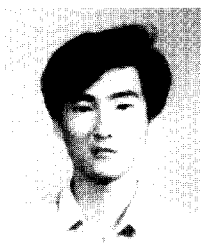
[5] E. Biham, A. Biryukov, O. Dunkelman, and A. Shamir, "Cryptanalysis of Skipjack-3XOR in 2^{50} time using 2^{9} chosen plaintexts", *http://www.cs.technion.ac.il/~biham/Reports/Skipjack*, 1998.

□ 著者紹介

박 성 모

1992년 서울대학교 수학과 졸업(이학학사)
1994년 포항공과대학교 대학원 수학과 졸업(이학석사)
1994년 7월 - 현재 한국 전자통신연구원 재직(연구원)

※ 주관심 분야 : 암호이론, 정보이론



이 상 진

1987년 2월 고려대학교 이과대학 수학과(이학사)
1989년 2월 고려대학교 대학원 수학과(이학석사)
1994년 2월 고려대학교 대학원 수학과(이학박사)
1989년 - 현재 한국전자통신연구원 선임 연구원

※ 주관심 분야 : 응용대수학 및 정수론, 암호론