블록 암호 PIPO의 음함수 기반 화이트박스 구현에 대한 연구

고정빈[2], 이강현[3], 이준영[3], 염용진[1,2]*, 강주성[1,2]
국민대학교 정보보안암호수학과[1] / 국민대학교 금융정보보안학과[2] / NSHC[3]
wjdqls1025@kookmin.ac.kr, {khlee, jylee}@nshc.net, {*salt, jskang}@kookmin.ac.kr

Toward Implicit White-box Implementation of Block Cipher PIPO

Jeongbeen Ko[2], Kanghyun Lee[3], Junyeong Lee[3], Yongjin Yeom[1,2]*, Ju-sung Kang[1,2]
Dept. of Information Security, Cryptology, and Mathematics, Kookmin Univ.[1] /
Dept. of Financial information security, Kookmin Univ.[2] / NSHC[3]

요 약

2002년 Chow 등은 DRM, 모바일뱅킹과 같이 민감한 암호학적 계산을 수행하는 디바이스에 대해 모든 제어가 가능한 공격자 모델을 정의하였다. 디바이스에 대해 모든 제어가 가능한 공격자는 알고리즘의 입출력뿐만 아니라 중간값을 관측할 수 있으며, 심지어 수정도 가능하다. 따라서 화이트박스암호의 안전성은 화이트박스 공격자에 대해 설계가 알려진 암호 알고리즘에서도 키를 추출해낼 수 없는 것을 목표로 한다. 본 논문에서는 모바일 환경에서 효율적으로 연산이 가능한 경량 블록 암호 알고리즘인 PIPO를 화이트박스암호로 구현하고, 필요한 테이블의 개수와 크기를 계산한다. 또한, 인코딩 공간을 축소하는 공격에 대한 계산량을 제시하고, 안전하지 않음을 설명한다. 또한, 향후 안전한 화이트박스 설계를 위해 연구 중인 음함수 기반 화이트박스 디자인에 대한 방향성을 제시한다.

Ⅰ. 서 론

2002년 Chow 등은 민감한 암호학적 계산을 수행하는 디바이스의 모든 제어를 가지고 있는 공격자 모델을 제안하였다. 이 공격자는 암호 알고리즘의 입출력뿐만 아니라 암호학적 계산의 모든 중간값을 관측할 수 있으며, 관측한 값들을 위변조할 수 있는 능력을 가진다. 이러한 공격자 모델에 대해 하드웨어 또는 설계의 기밀성에 의존하지 않고 암호키를 보호하는 기술을 화이트박스암호라 하며, Chow 등이 제안한 화이트박스암호 설계 방식을 CEJO 프레임워크라 부른다[4]. 그러나 현재까지 설계가 알려진 화이트박스는 모두 안전하지 않음이 증명되었다. 따라서 안전한 화이트박스를 설계하기 위한 지속적인 연구가 필요하다.

본 논문에서는 2020 ICISC에 제안된 경량 블록 암호 알고리즘 PIPO를 CEJO 프레임워크에 적용하고 공격량을 계산하여 CEJO 프레임워크 기반 화이트박스 PIPO-128가 안전하지 않음을 보인다. 따라서, 화이트박스 PIPO를 새롭게 제안된 화이트박스암호 설계 기법인 음함수 기반 화이트박스 디자인의 적용 가능성을 설명한다[2].

Ⅱ. 경량 블록 암호 PIPO[3]

대칭 키 알고리즘인 PIPO는 저성능, 저용량 기기를 위해 2020년에 제안된 경량 블록 암호 알고리즘이다. PIPO는 S-box의 비선형 연산 개수를 최소화하고, 모든 연산은 비트별 AND, OR, XOR, NOT 그리고 Rotation으로만 구성하여 연산의 병렬처리가 가능한 효율적인 구조로 설계되었다[3]. PIPO는 64비트 입출력을 통해 암·복호화를 수행하며, 키 길이는 128, 256비트 두 가지로 나뉜다. 키 길이에 따라 라운드 수가 결정되며, 128비트 키를 사용하면 13라운드, 256비트 키를 사용하면 17라운드를 진행한다. 각 라운드는 치환 계층인 S-layer, 전치 계층인 R-layer와 Addroundkey 연산으로 구성된다.

S-layer는 상태 행렬의 열 단위로 동작하며, 8개의 열에 대해 8개의 S-box가 병렬로 테이블 참조 혹은 비트별 연산을 수행한다. R-layer는 상태 행렬의 행 단위로 동작하며, S-layer와 마찬가지로 8개의 행에 대한 병렬로 테이블 참조 혹은 비트별 연산을 통해 수행된다.

Ⅲ. CEJO 프레임워크 기반 화이트박스 PIPO-128

ⅰ. CEJO 프레임워크[1]

정의1. 고정된 키 k에 대한 n비트 블록 암호의 암호 알고리즘을 $E_k = E^{(n_r)} \circ \cdots \circ E^{(1)}$이라 하자. E_k의 인코딩된 함수는 인코딩된 라운드 함수들로 구성된다. 즉, 다음과 같다.

$$\overline{E_k} = \overline{E^{(n_r)}} \circ \cdots \circ \overline{E^{(1)}} = (O^{(n_r)} \circ E^{(n_r)} \circ I^{(n_r)}) \circ \cdots \circ (O^{(1)} \circ E^{(1)} \circ I^{(1)}).$$

이때, n비트 치환 $(I^{(i)}, O^{(i)})$은 $I^{(i+1)} = (O^{(i)})^{-1}$을 만족한다.

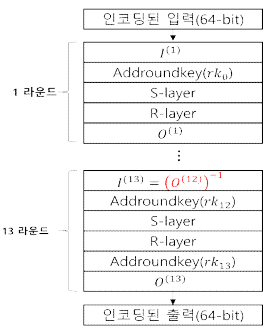
$(i = 1, 2, \ldots, n_r - 1)$

다시 말해, $\overline{E_k}$는 라운드 간의 인코딩이 상쇄되는 구조이다.

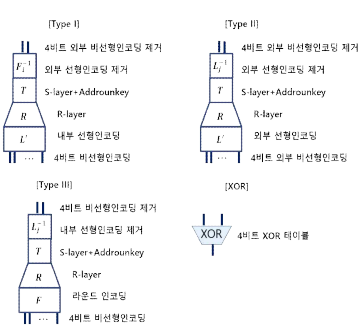
ⅱ. CEJO 프레임워크 기반 화이트박스 PIPO-128

PIPO-128은 입출력이 64비트이고 키 길이가 128비트이며, 이에 대한 CEJO 화이트박스 알고리즘은 [그림 1]과 같이 설계한다. 입출력은 외부인코딩이 적용된 64비트이며, 한 라운드는 addroundkey단계와 S-layer와 R-layer의 인코딩된 테이블이다. 마지막 13라운드는 addroundkey단계를 한번 더 시행한다. 또한, CEJO 프레임워크 구조에 기반하므로 이전 라운드의 출력인코딩은 다음 라운드의 입력 인코딩과 서로 상쇄된다.

다음 [그림 2]는 화이트박스 PIPO-128에 사용되는 테이블을 도식화한 것이며, [표 2]는 구현을 위해 필요한 테이블의 개수와 메모리 크기이다.



〔그림 1〕 화이트박스 PIPO-128



〔그림 2〕 필요한 테이블 종류

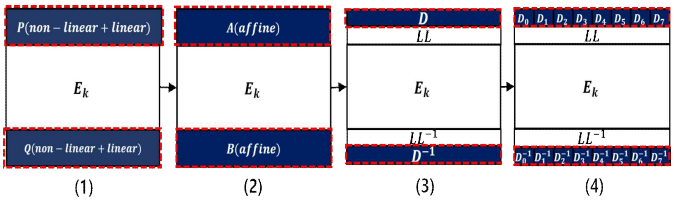
첫 라운드와 마지막 라운드에 사용되는 Type Ⅰ, Type Ⅱ 테이블에는 code-lifting 공격을 방지하기 위해 외부 인코딩이 적용되며, 2라운드부터 12라운드까지 사용되는 Type Ⅲ 테이블은 내부 인코딩이 적용된다. XOR 테이블은 연산의 결과를 합하는 데 필요하며, 실용적인 크기의 구현과 테이블이 거대해지는 것을 방지하기 위해 각 테이블에 4비트 비선형 인코딩을 사용한다.

	테이블 개수	메모리
Type Ⅰ	8	$16.384\,KB = 2^8 \cdot 8 \cdot 8\,bytes$
Type Ⅱ	8	$16.384\,KB = 2^8 \cdot 8 \cdot 8\,bytes$
Type Ⅲ	$88 = 8 \cdot 11$	$180.224\,KB = 2^8 \cdot 8 \cdot 88\,bytes$
XOR	$1456 = (64/4) \cdot 7 \cdot 13$	$186.368\,KB = 2^8 \cdot (1/2) \cdot 1456\,bytes$

〔표 2〕 필요한 테이블의 개수와 메모리

Ⅳ. CEJO 프레임워크 기반 화이트박스 PIPO-128 안전성 분석

ⅰ.안전성 분석



〔그림 3〕 인코딩 공간이 축소되는 과정

[그림 3]은 2020년 Ranea가 제시한 인코딩 축소 공격[1]을 도식화한 것이다. 공격자는 인코딩된 함수를 블랙박스로써 접근할 수 있고, 선형레이어인 LL 와 비선형레이어인 SL은 알려졌다고 가정한다. 다시 말해, 인코딩 함수와 키에 대한 정보를 제외한 모든 정보에 공격자가 접근할 수 있다고 가정한다.

(1) 공격자에게 세 개의 연속된 인코딩된 라운드 함수가 주어진다. 인코딩은 비선형 인코딩과 선형 인코딩으로 구성되어있다.

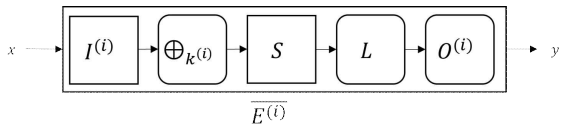
(2) BGE 공격을 통하여 비선형 출력 인코딩을 선형 출력 인코딩으로 변환한다[5]. 출력 인코딩과 다음 라운드의 입력 인코딩이 상쇄되는 CEJO 프레임워크의 특성에 의해 세 개의 선형 출력 인코딩으로 다음 두 라운드의 선형 입력 인코딩을 얻는다. 공격자는 얻어진 인코딩이 선형 인코딩이라는 사실만을 알고 있고, 정확한 인코딩 함수는 알 수 없다.

(3) 3장에서 소개한 CEJO 프레임워크 화이트박스 PIPO-128에 대하여 $O((64/4) \cdot 2^{12} + 2^8 \cdot 64^3 + 64^4/8 + 2^{16} \cdot 8 \cdot 64^2) = O(2^{31})$의 복잡도로 인코딩 공간이 비선형레이어의 자기동치 군에 의존하는 새로운 인코딩된 함수를 얻을 수 있다[6]. 이때, D는 $SL = D \circ SL \circ C$를 만족하는 아핀 치환이다.

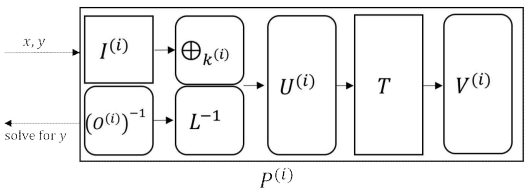
(4) (3)에서 새롭게 얻은 함수의 인코딩 공간은 비선형레이어의 자기동치 군의 크기만큼의 복잡도를 가지게 되는데, CEJO 프레임워크는 작은 S-box의 연접으로 비선형레이어를 구성하고 있기 때문에 $|SE(SL)| = (n/m)! \times |SE(S)|^{(n/m)} = 8! \times |SE(S)|^8 < 2^{16} \times |SE(S)|^8$ 이다[1].

이 결과는 (3)에서 얻은 함수의 인코딩 개수가 8비트 S-box의 자기동치 군의 크기만큼 존재함을 의미하며, CEJO 프레임워크 기반 화이트박스 PIPO-128가 안전하지 않음을 암시한다.

Ⅴ. 음함수 기반 화이트박스



〔그림 4〕 CEJO 프레임워크



〔그림 5〕 음함수 기반 프레임워크

2022년 Cryto 2022에서 Renea 등에 의해 기존의 테이블을 참조 방식이 아닌 음함수 방정식 기반 설계방식이 제안되었다[2]. 이는 한 라운드를 음함수 방정식으로 저장하여 방정식의 해를 구하는 방식이다. 음함수 기반 프레임워크의 장점은 큰 아핀 치환과 큰 비선형레이어를 사용하여도 효율적으로 인코딩이 가능하다는 점이다.

비선형레이어는 안전성에 중요한 역할을 하게 되는데, 음함수 기반 화이트박스의 경우 비선형레이어의 음함수 방정식인 T 함수의 앞, 뒤 x, y를 전체적으로 인코딩하는 과정을 포함하고, 테이블 참조 방식과 달리 방정식의 차수에 크기를 의존하기 때문에 비선형레이어의 크기에 제한이 없다. 또한, 음함수 기반 프레임워크가 현재까지 알려진 일반적인 키 추출 공격을 방지한다. 이 경우가 화이트박스의 안전성을 보장하지는 않지만, 테이블 참조 방식 화이트박스 암호에 적용 가능한 공격 기법이 음함수 기반 프레임워크까지 확장되지는 않기 때문에[2] PIPO의 화이트박스 안전성을 강화할 수 있을 것으로 예상된다.

Ⅵ.결론

본 논문에서는 모바일 환경에 적합한 경량 블록 암호 알고리즘인 PIPO를 테이블 참조 방식인 CEJO 프레임워크에 적용하여 안전성을 분석하였다. 세 개의 라운드 함수만 주어져도 $O(2^{31})$ 복잡도로 인코딩이 더 간단한 새로운 라운드 함수를 얻을 수 있었고, 테이블 참조 방식 특성상 작은 S-box로 구성된 비선형레이어를 사용하기 때문에 안전하지 않음을 설명하였다. 한편, PIPO 알고리즘은 모든 연산이 비트별 AND, OR, XOR, NOT 및 Rotation 연산으로만 이루어져 있기 때문에 부울 함수로 표현하기에 유리하다. 이러한 특성은 음함수 기반 화이트박스 구현에 적합하므로 향후 이러한 음함수 기반 화이트박스 PIPO에 대한 설계와 그에 따른 안전성 분석에 관한 연구를 진행할 예정이다.

ACKNOWLEDGMENT

본 연구는 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2021-0-00046, 국가공공 정보시스템 안전성 및 활용성 제고를 위한 차세대 암호체계 개발)

참 고 문 헌

[1] Ranea A. et al., “On Self-Equivalence Encodings in White-Box Implementations”, SAC 2020

[2] Ranea A. et al., “Implicit White-Box Implementations: White-Boxing ARX Ciphers”, CRYPTO 2022

[3] Hangi K. et al., “A New Method for Designing Lightweight S-Boxes with High Differential and Linear Branch Numbers, and Its Application”, ICISC 2020

[4] Stanley Chow et.al., “White-Box Cryptography and an AES Implementation”, SAC 2002

[5] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi, “Cryptanalysis of a White Box AES Implementation”, SAC 2004

[6] Patrick Derbez et al.,"On Recovering Affine Encodings in White-Box Implementations", IACR 2018