Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.6, Dec. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.6.1103

Blacklist를 활용한 선택적 평문 충돌 쌍 공격*

김 은 희,† 김 태 원, 홍 석 희‡
고려대학교

Chosen Plaintext Collision Attack Using the Blacklist*

Eun-hee Kim,† Tae-won Kim, Seok-hie Hong‡
Korea University

요 약

부채널을 이용한 충돌 쌍 공격은 부채널 신호를 통해 동일한 비밀 중간 값을 확인하고, 이를 이용하여 암호 알고리즘의 비밀정보를 복원한다. CHES 2011에서 Clavier 등은 Blacklist를 활용하여 기존보다 적은 수의 평문으로 충돌 쌍 공격을 수행하였다. 하지만 Blacklist를 활용한 구체적인 방법 또는 수행 알고리즘에 대한 언급이 없고 단지 사용된 평문의 수치만 소개하였다. 따라서 본 논문에서는 효율적인 충돌 쌍 공격을 위한 구체적인 방법을 소개한다. 우선 Blacklist를 활용할 수 있도록 기본적인 개념 및 용어, 표기법을 정의했으며, 이를 바탕으로 여러 가지 기법들을 제안했다. 또한 설계 시 공격 성능에 가장 큰 영향을 주는 사항에 대해 중점적으로 기술하였고, 분석을 통해 좀 더 효율적인 알고리즘을 설계함으로써 성능 향상을 꾀하였다.

ABSTRACT

Collision attacks using side channel analysis confirm same intermediate value and restore sensitive data of algorithm using this point. In CHES 2011 Clavier and other authors implemented the improved attack using Blacklist so they carried out the attack successfully using less plaintext than before. However they did not refer the details of Blacklist method and just performed algorithms with the number of used plaintext. Therefore in this paper, we propose the specific method to carry out efficient collision attack. At first we define basic concepts, terms, and notations. And using these, we propose various methods. Also we describe facts that greatly influence on attack performance in priority, and then we try to improve the performance of this attack by analyzing the algorithm and structuring more efficient one.

Keywords: Side Channel Attack, Collision Attack, Masked AES

I. 서 론

부채널 공격(Side channel attack)[1]은 디바이스가 암호 알고리즘을 구동하는 동안 발생하는 연산 시간, 소비전력, 전자파 등의 다양한 부가정보를 활용하여 비밀정보를 획득하는 공격이다. 이러한 부채널 공격에는 디바이스에 오류를 주입하여 오작동을 일으킨 결과 값을 이용하여 비밀정보를 복원해 내는 오류 주입 공격 (Fault Attack)[2,3], 디바이스에서 연산되는 시간 정보를 이용하여 비밀정보를 복원해 내는 시간 공격 (Timing Attack)[4,5], 디바이스로부터 누설된 전력 소비 신호의 통계적 특성을 이용하여 비밀정보를 복원해 내는 전력 분석 공격 (Power Analysis Attack) 등[6]이 있다.

접수일(2014년 10월 17일), 수정일(2014년 11월 25일), 게재확정일(2014년 11월 25일)
* 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터육성 지원사업의 연구결과로 수행되었음 (NIPA-2014-H0301-14-1004)
† 주저자, hahehohu3875@korea.ac.kr
‡ 교신저자, shhong@korea.ac.kr(Corresponding author)

전력 분석 공격이 소개되면서 동시에 공격에 안전하게 설계하기 위한 다양한 대응 기법들이 제안되었다. 이 중 가장 널리 사용되는 대응기법은 마스킹 기법(Masking Method)[7,8,9,10,11,12]이 있다. 마스킹 기법은 암호 알고리즘 수행 시 소비되는 전력량을 공격자가 추정할 수 없도록 중간 값을 마스킹 난수(랜덤 값)에 의해 랜덤화시키는 방법이다. 하지만, 이차 전력 분석[13] 및 고차 전력 분석에서 마스킹 기법이 안전 하지 않다는 것이 밝혀졌다.

충돌 쌍 공격[14,15,16]은 평문을 조절하여 임의의 두 중간 값이 동일한 값을 가질 때 충돌이 발생함을 전력 분석을 통해 확인하고, 이를 이용하여 알고리즘의 비밀 키를 복원하는 방법이다. 두 바이트가 충돌했을 때, 두 바이트 간의 평문 차분 값과 키 차분 값이 같음을 이용한다. 충돌 정보를 모두 종합하여 16바이트가 모두 서로 연결되면, 키 정보량은 16바이트에서 1바이트로 감소된다.

Moradi 등은 하드웨어 구현을 통하여 평문 16바이트 간 서로 다른 마스킹 난수가 사용된 AES에서 다수의 평문을 사용했을 경우 마스킹 효과가 감소하는 실험 결과를 내었다. 이를 기반으로 충돌 쌍 공격을 통해 AES 키를 찾는데 성공하였다[17]. 하지만, 충돌 쌍 공격을 성공하는데 평문이 굉장히 많이 필요하다는 단점이 존재하였다.

Clavier 등은 평문 16바이트 간 서로 같은 마스킹 난수가 사용된 AES에서 충돌 쌍 공격을 통해 AES 키를 찾는데 성공하였다. 보다 적은 파형을 사용하여 평문 16바이트에서 발생할 수 있는 충돌 쌍 120개에 부채널 분석을 이용하여 충돌 쌍 공격을 하였다. 상관계수를 통해 두 바이트 간 중간 값이 동일한지 판단하였다. 또한 평문을 사용하였을 경우, 충돌이 발생하지 않은 정보들인 Blacklist를 언급하였다. 평문에서 충돌이 발생하지 않는 경우가 충돌이 발생하는 경우보다 훨씬 많기 때문에 이 정보들을 공격에 이용한다면 사용되는 평문의 개수를 더 줄일 수 있으며, 공격 성능을 높일 수 있다. Blacklist 방법을 통해 평문의 개수가 기존보다 적게 사용된 결과를 보여주었다[18]. 하지만, 실험방법 뿐만 아니라 사용된 알고리즘에 대한 구체적인 설명이 나와 있지는 않다.

본 논문에서는 Clavier 등이 언급한 Blacklist 정의를 통해 충돌 쌍 정보를 이용하여 평문을 구성하는데 사용되는 알고리즘과 유효 충돌 쌍을 찾는 알고리즘을 정의한다. 정의한 알고리즘을 통해 충돌 쌍 공격 알고리즘을 제안하고 이에 대한 효율성을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 AES 마스킹 기법 및 충돌 쌍 공격에 대해 살펴본다. 그리고 3장에서는 충돌 쌍 공격에 사용되는 용어들에 대한 표기법을 살펴보며 이에 대해 정리해본다. 4장에서는 이를 바탕으로 효율적 충돌 쌍 공격 알고리즘을 제안하고 효율성을 검증한다. 마지막으로 5장에서 결론을 맺는다.

II. 마스킹 기법이 적용된 AES에 대한 충돌 쌍 공격

본 장에서는 분석대상 알고리즘인 마스킹 기법이 적용된 AES와 충돌 쌍 공격에 대한 기본적인 내용을 살펴본다.

2.1 AES 마스킹

AES(Advanced Encryption Standard)는 블록 암호화 방식 중 하나이다. 비밀 키의 길이는 128, 192, 256비트로 3가지이며 각각의 길이에 따라 10, 12, 14 라운드가 수행된다. 각 라운드는 AddRoundKey, SubBytes, ShiftRows, MixColumns의 4가지 연산으로 구성되어 있고 마지막 라운드에만 MixColumns 연산을 수행하지 않는다.

마스킹 기법은 부채널 분석에 사용되는 중간 값을 공격자로부터 노출시키지 않는 방법으로, 현재 가장 널리 사용되고 있다. 제안된 마스킹 기법에는 Boolean 마스킹[19,20], Multiplicative 마스킹[21,22], SBox의 역원 계산을 변경하는 방법 등[23]이 있다. Multiplicative 마스킹의 경우 Zero-Value 공격에 취약한 것으로 알려져 있다. 또한 SBox 역원 계산을 변경하는 방법은 하드웨어 구현에 적합한 방법으로 소프트웨어 구현에는 적합하지 않다. 따라서 이 중에서는 Boolean 마스킹 기법이 가장 많이 사용되고 있다. Boolean 마스킹은 알고리즘 수행 시 중간 값과 마스킹 난수의 차분 연산을 통해 공격자가 소비되는 전력량을 추정할 수 없도록 하는 기법이다. 이 때, 마스킹 난수는 매번 수행 마다 균등하게 다른 값이어야 한다.

본 논문에서는 Boolean 마스킹 기법이 적용된 AES-128(평문 m_i, 비밀 키 k_i : $0 \le i \le 15$, i는 평문 바이트)을 분석대상으로 한다. 각 16개 바이트에 대해 서로 동일한 마스킹 난수 u_i, v_i

($u_i = u$, $v_i = v$)를 사용한다. 마스킹 기법이 적용된 AES의 수행순서는 다음과 같다.

단계 1. AddRoundKey 연산

마스킹 된 평문과 마스킹 된 라운드 키의 차분 값을 구한다. AddRoundKey 연산을 수행한 후, 중간 값은 마스킹 난수 u로 마스킹 된다.

단계 2. SubBytes 연산 : 마스킹 SBox S' 생성, ShiftRows 연산

2.1 마스킹 난수 u, v를 생성한다.

2.2 $S'(m \oplus k \oplus u) = S(m \oplus k) \oplus v$ 을 만족하는 S'을 생성한다.

SubBytes 후의 결과 값은 마스킹 난수 v로 마스킹 된다.

단계 3. Remasking 연산

Remasking 함수를 통해 마스킹 난수 v로 마스킹 된 중간 값들을 서로 상이하게 바꿔준다. 각각의 행마다 다른 마스킹 난수 m_1, m_2, m_3, m_4으로 마스킹 되도록 한다.

단계 4. MixColumns 연산

Remasking 연산에 의해 추가된 네 개의 마스킹 난수 m_1, m_2, m_3, m_4가 MixColumns 연산을 통해 새로운 마스킹 난수로 바뀐다. 이 때 마스킹 난수의 값은 다음과 같다.

$$m'_1 = 02(m_1) \oplus 03(m_2) \oplus m_3 \oplus m_4$$
$$m'_2 = m_1 \oplus 02(m_2) \oplus 03(m_3) \oplus m_4$$
$$m'_3 = m_1 \oplus m_2 \oplus 02(m_3) \oplus 03(m_4)$$
$$m'_4 = 03(m_1) \oplus m_2 \oplus m_3 \oplus 02(m_4)$$

2.2 충돌 쌍 공격

부채널을 이용한 충돌 쌍 공격은 Hans Dobbertin에 의해 제안되었다. 그 후 Schramm 등은 DES(Data Encryption Standard)의 SBox 단계의 출력 값에 대한 부채널 분석을 이용하여 충돌 쌍 공격을 수행하였다[14]. SBox 치환과정을 통해 각 SBox는 입력 값 6비트에서 출력 값 4비트로 출력된다. 각 SBox에서 동일한 출력 값에 대하여 4개의 입력 값을 얻을 수 있으므로 하나의 입력 값과 나머지 3개의 입력 값과의 차분 값을 구하여, 이 3개의 차분 정보에 대한 전체 차분 정보를 이용하여 키 값을 추측한다.

또한 Schramm 등은 AES(Advanced Encryption Standard)의 MixColumns 단계에서도 출력 값에 대한 부채널 분석을 이용하여 충돌 쌍 공격을 수행하였다[15]. MixColumns은 16개 바이트들의 State로 구성되어 있으며, 각 MixColumns 바이트의 출력 값은 MixColumns의 State 열 바이트들의 입력 값 4개와 연산된다. 열 바이트들의 입력 값 4개에 사용되는 첫 번째와 두 번째 평문을 0으로 세 번째와 네 번째 평문을 동일하게 세팅한다. 이러한 조건을 고려한 2개의 다른 평문을 사용하여 MixColumns 출력 값이 같은지 확인하고 이 정보를 활용하여 키 값을 추측한다.

충돌 쌍 공격은 일반적인 차분 전력 분석 공격보다 적은 수의 파형으로도 비밀정보를 복원해낼 수 있다. Clavier 등은 평문 59개를 사용하여 평문 16바이트 간 서로 같은 마스킹 난수가 사용된 AES에서 충돌 쌍 공격에 성공하였다. 또한 언급한 Blacklist를 활용하여 사용되는 평문의 개수를 27.5개로 감소시켰다.

공격 방법에 대한 구체적인 내용은 다음과 같다. 우리는 분석대상이 되는 마스킹 기법이 적용된 AES에 대해 설명한다.

단계 1. 동일한 평문에 대해 N개의 파형 $(T^n)_{0 \le n \le N-1}$ (n : 파형 수)을 수집한다.

단계 2. 파형들의 $(T^n)_{0 \le n \le N-1}$ 두 바이트 간 중간 값의 상관계수(*Correlation*) 120개를 계산한다. 상관계수 식(1)은 다음과 같다.

단계 3. 상관계수 값이 기준치 이상이면 충돌 *Collision* (θ_0, θ_1)이 일어났다고 할 수 있다.

충돌이 발생했다는 것은 평문 두 바이트 중간 값이 동일하다고 할 수 있다. 따라서 두 바이트 간의 평문 차분 값을 알기 때문에 그 해당 바이트 간의 비밀 키 차분 값도 알 수 있다. 공격을 계속 수행하면서 충돌 쌍들의 정보를 축적한다. 이를 통해 바이트 간 서로 유기적으로 연결되면, 최소 15개의 충돌 쌍을 통해 1바이트 비밀 키의 추측만으로 16바이트 비밀 키를 전부 복원할 수 있다.

$$\widehat{\rho}_{\theta_0,\theta_1}(t) = \frac{Cov(\theta_0(t),\theta_1(t))}{\sigma_{\theta_0(t)}\sigma_{\theta_1(t)}}$$
$$= \frac{N\sum(T^n_{t_0+t}T^n_{t_1+t}) - \sum T^n_{t_0+t}\sum T^n_{t_1+t}}{\sqrt{N\sum(T^n_{t_0+t})^2-(\sum T^n_{t_0+t})^2}\sqrt{N\sum(T^n_{t_1+t})^2-(\sum T^n_{t_1+t})^2}} \quad (1)$$

$$S(m_{i_1}\oplus k_{i_1})\oplus v = S(m_{i_2}\oplus k_{i_2})\oplus v$$
$$S'(m_{i_1}\oplus k_{i_1}\oplus u) = S'(m_{i_2}\oplus k_{i_2}\oplus u)$$
$$m_{i_1}\oplus k_{i_1}\oplus u = m_{i_2}\oplus k_{i_2}\oplus u$$
$$m_{i_1}\oplus k_{i_1} = m_{i_2}\oplus k_{i_2}$$
$$m_{i_1}\oplus m_{i_2} = k_{i_1}\oplus k_{i_2}$$

III. 표기법 및 용어정리

본 장은 제안하는 개념을 명확하게 설명하기 위해 필요한 용어 및 표기법을 소개한다.

3.1 충돌 쌍 그룹

충돌 쌍 그룹이란 충돌 쌍들이 연결되어 1바이트 정보량을 나타내고 있는 충돌 쌍들의 집합이다. 예를 들어 공격을 통해 1번째 바이트와 2번째 바이트의 충돌 쌍 (1,2)와 2번째 바이트와 3번째 바이트의 충돌 쌍 (2,3)을 얻었다고 하자. 두 충돌 쌍은 공통된 2번째 바이트로 인해 (1,3)의 차분정보도 알아낼 수 있다. 따라서 두 충돌 쌍은 1바이트의 정보량으로 통합될 수 있으며 통합된 두 충돌 쌍은 하나의 충돌 쌍 그룹으로 나타낼 수 있다. 〔Fig.1-(a)〕는 충돌 쌍(1,3),(3,7),(0,7),(0,11),(0,15),(5,9),(5,13),(5,14),(8,14)에 대한 충돌 쌍 그룹을 나타낸다.

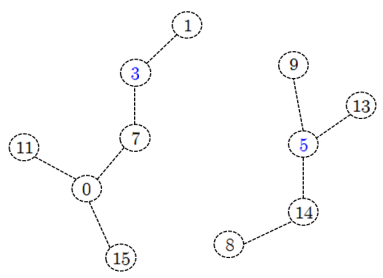
3.2 유효 충돌 쌍(Vaild Collision Pair, VCP)

공격자는 여러 번의 충돌 쌍 공격으로 키에 대한 부분정보를 가지고 있다고 하자. 공격자는 계속 진행된 충돌 쌍 공격으로 충돌 쌍을 얻었다고 가정하자. 이때, 충돌의 발생으로 얻은 두 바이트의 키 차분 정보를 기존에 공격자가 얻은 키의 부분 정보로부터 얻을 수 없다면 이러한 충돌 쌍을 유효 충돌 쌍(Valid Collision Pair)이라고 정의한다.

예를 들어 기존의 충돌 쌍 공격으로 〔Fig.1〕와 같이 충돌 쌍 그룹이 형성되었다고 하자.

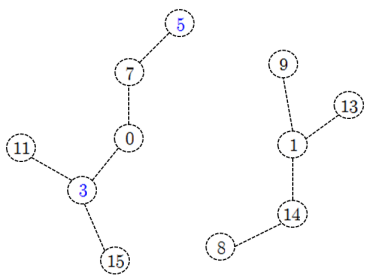
공격자는 새로운 충돌 쌍 공격을 통해 (3,5)을 얻었다고 가정하자. 〔Fig.1-(a)〕의 경우 (충돌 쌍 그룹 1)과 (충돌 쌍 그룹 2)를 통해 (3,5)의 차분 정보를 얻을 수 없으므로 유효 충돌 쌍이 된다. 하지만 〔Fig.1-(b)〕의 경우, (충돌 쌍 그룹 1)에서 (3,5)의 차분정보를 뽑아낼 수 있으므로 유효 충돌 쌍이 될 수 없다.

따라서 공격자는 충돌 쌍 공격을 통해 얻은 충돌 쌍으로 충돌 쌍 그룹을 만들고 이를 기반으로 새로운 충돌 쌍의 유효성을 검증한다. 또한 찾아낸 유효 충돌



Group (1) Group (2)

(a) Valid Collision Pair



Group (3) Group (4)

(b) Non-Valid Collision Pair

Fig. 1. Valid Collision Pair Example

쌍으로 기존 구성되어 있는 충돌 쌍 그룹을 업데이트 시킨다.

3.3 Blacklist

공격자는 한 번의 충돌 쌍 공격으로 120개($_{16}C_2$)의 바이트 간 키 차분정보를 확인할 수 있다. 이 중 키 차분 값이 아닌 것들의 집합을 Blacklist라 하자. Blacklist에 속해 있는 값들은 두 바이트 간 옳지 않은 키 차분 값이며 다음과 같이 표기한다.

$D^{N}_{(i_1, i_2)}$, (N : 평문 수 , i : 평문 바이트 번호)

공격자는 다음 충돌 쌍 공격 시 Blacklist를 이용하여 평문을 선택할 수 있다. 즉, 공격자는 Blacklist에 있는 옳지 않은 키 차분 값이 또 다시 관찰되지 않도록 평문을 선택할 수 있다.

또한 Blacklist는 두 바이트 간의 평문 차분 값이므로 하나의 바이트 값이 결정되어야 나머지 한 바이트의 값이 결정됨을 명심해야 한다. 따라서 공격자는 평문 한 바이트 값을 선택 후 Blacklist 값을 이용하여 나머지 바이트의 선택되면 안 되는 값을 구할 수 있다. 이 때, 평문 선택 시 제외되는 값을 Black Plaintext Value(BP-Value)라 정의하고 $b^{N}_{(i,j)}$ 이라 표기한다. 여기서 N은 평문 수를 의미하며 i 번 째 바이트가 선택되었을 때 선택되면 안 되는 j 번 째 바이트를 나타낸다.

3.4 평문 바이트 순서 (PB-Order)

공격자는 Blacklist를 활용할 때 평문의 한 바이트씩 선택하여 수행해야 한다. 이 때, 선택되는 평문의 순서는 공격 성능에 매우 중요한 요소이다. 왜냐하면 먼저 선택 된 평문 바이트에 따라 BP-Value가 작성되며 이를 통해 나머지 바이트들의 값이 결정되기 때문이다.(이것에 대한 구체적인 설명은 4.2에서 찾아볼 수 있다.) 따라서 평문 바이트 순서는 Blacklist를 이용하여 BP-Value 작성 시 선택되는 평문 바이트의 순서를 의미한다.

IV. Blacklist를 활용한 충돌 쌍 공격

본 장에서는 3장에서 정의된 용어 및 표기법을 바탕으로 충돌 쌍 공격 시 Blacklist를 활용한 다양한 평문 선택 방법에 대해 설명한다.

기존의 랜덤하게 평문을 생성하는 방법과는 달리 공격자가 필요한 평문을 선택하여 공격을 진행한다. 따라서 공격자는 선택 평문 공격(Chosen Plaintext Attack)환경을 가정한다.

제안하는 공격의 가장 핵심적인 아이디어는 충돌 쌍 공격으로 얻은 옳지 않은 키 차분 정보를 다음 공격에 관찰되지 않도록 평문선택을 조작하는 것이다. 따라서 공격자는 매 수행마다 옳지 않은 키 정보(Blacklist)를 작성하여 다음 공격 때 이를 바탕으로 평문을 선택한다.

최적화 된 기법을 찾기 위하여 다양한 평문 선택 방법으로 접근을 시도하였으며 이에 대한 충돌 쌍 공격 실험을 진행하였다. 각 기법에 대하여 발생하는 유효 충돌 쌍 개수 비교 분석을 통해 성능뿐만 아니라 한계 및 개선방향도 기술하였다. 또한 이해를 돕기 위해 예를 중심으로 설명한다.

4.1 공격 흐름도

제안하는 공격 방법은 [Fig.2]와 같이 진행된다. 최초로 공격자는 구성되어 있는 Blacklist를 활용하여 평문을 선택한다. 이 후, 충돌 쌍 공격을 수행하여 충돌 발생 여부를 확인한다. 만약 충돌이 발생했다면 유효 충돌 여부를 검사한 후 기존에 구성되어 있는 충돌 쌍 그룹을 업데이트해 준다.

유효 충돌 쌍 수에 따라 공격을 종료할 것인지

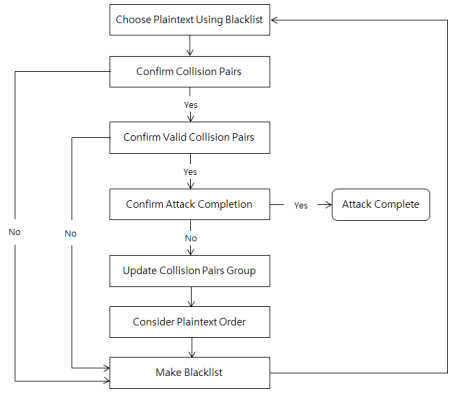


Fig. 2. Flow of Collision Attack

Blacklist를 업데이트 하여 위의 과정을 수행할 것인지 결정한다. 만약 유효한 15개의 충돌 쌍이 있다면 공격자는 알고 있는 평문-암호문 쌍을 이용하여 2^8개의 키 후보군 중 최종적으로 옳은 키를 선택한다.

4.2 Blacklist를 활용한 평문 선택방법 1

공격자는 한 번의 충돌 쌍 공격을 수행하였고 이 때 발생 된 충돌 쌍은 없다고 가정하자. 그러면 공격자는 120개의 Blacklist 정보를 얻게 된다. 구성된 Blacklist 정보를 이용하여 BP-Value값을 구한 후 공격자는 다음 공격에 선택될 평문을 BP-Value 값을 고려하여 결정한다. 즉, 공격자는 Blacklist에 존재하는 값이 다음 공격에서 발생되지 않도록 평문을 선택해야 한다. 따라서 차분 값으로 존재하는 $D^N_{(i_1,\,i_2)}$에서 먼저 한 바이트를 선택하여 고정시킨 후, 고정 된 값과 $D^N_{(i_1,\,i_2)}$을 이용하여 다른 한 바이트의 BP-Value를 계산한다. 〔Table 1〕은 BP-Value를 계산해 놓은 Table이다. 공격자는 첫 번째로 P_0^{N+1}값을 랜덤하게 선택하여 고정시킨다. 이 후 0번째 바이트와 관계된 Blacklist를 고려하여 P_0^{N+1}와 차분 연산을 통해 BP-Value, $b^N_{(0,j)}$ ($j=1,\cdots,15$)을 계산한다.

$$
\begin{aligned}
b^N_{(0,1)} &= p_0^{N+1} \oplus D^N_{(0,1)} \\
b^N_{(0,2)} &= p_0^{N+1} \oplus D^N_{(0,2)} \\
&\vdots \\
b^N_{(0,15)} &= p_0^{N+1} \oplus D^N_{(0,15)}
\end{aligned}
$$

마찬가지로 공격자는 랜덤하게 P_1^{N+1}을 선택하여 고정한다. 이 때, P_1^N의 값으로 $b^N_{(0,1)}$가 선택된다면 Blacklist에 존재하는 키 차분 값과 같게 되므로 공격자는 반드시 $b^N_{(0,1)}$와 다른 값으로 P_1^{N+1}을 선택해야한다. P_1^{N+1}을 선택한 공격자는 BP-Value, $b^N_{(1,j)}$ ($j=2,\cdots,15$)을 계산한다.

$$
\begin{aligned}
b^N_{(1,2)} &= p_1^{N+1} \oplus D^N_{(1,2)} \\
b^N_{(1,3)} &= p_1^{N+1} \oplus D^N_{(1,3)} \\
&\vdots \\
b^N_{(1,15)} &= p_1^{N+1} \oplus D^N_{(1,15)}
\end{aligned}
$$

나머지 평문 바이트 $P_2^{N+1},\ldots,P_{15}^{N+1}$에 대하여, 위와 동일한 방식으로 $D^N_{(i,j)}$를 고려하여 평문을 선택하고 이에 대한 BP-Value를 작성한다. 〔Table 1〕에서처럼 평문을 선택하고 BP-Value를 작성하는 순서는 Table의 왼쪽에서 오른쪽으로 진행된다. 따라서 가장 오른쪽에 있는(15번째 바이트) 바이트는 평문 선택 시 15개의 BP-Value를 고려하여 선택된다. 만약 공격자가 2번의 충돌 쌍 공격으로 (충돌은 발생하지 않았다고 가정한다.)얻은 Blacklist를 이용하여 BP-Value를 계산하면 평문 1번째 바이트에는 2개의 BP-Value, 평문 2번째 바이트에는 4개의 BP-Value, 제일 오른편에 있는 평문 15번째 바이트에는 30개의 BP-Value이 채워지게 된다. 〔Table 2〕는 20개의 평문을 이용하고 작성된 Blacklist를 통해 계산된 BP-Value 이다. (단, 공

Table 1. Building BP-Value Table Using the Blacklist

0	1	2	3	…	14	15
$P_0^{N_2}$	$b^{N_1}_{(0,1)}=p_0^{N_2}\oplus D^{N_1}_{(0,1)}$	$b^{N_1}_{(0,2)}=p_0^{N_2}\oplus D^{N_1}_{(0,2)}$	$b^{N_1}_{(0,3)}=p_0^{N_2}\oplus D^{N_1}_{(0,3)}$	…	$b^{N_1}_{(0,14)}=p_0^{N_2}\oplus D^{N_1}_{(0,14)}$	$b^{N_1}_{(0,15)}=p_0^{N_2}\oplus D^{N_1}_{(0,15)}$
	$P_1^{N_2}$	$b^{N_1}_{(1,2)}=p_1^{N_2}\oplus D^{N_1}_{(1,2)}$	$b^{N_1}_{(1,3)}=p_1^{N_2}\oplus D^{N_1}_{(1,3)}$	…	$b^{N_1}_{(1,14)}=p_1^{N_2}\oplus D^{N_1}_{(1,14)}$	$b^{N_1}_{(1,15)}=p_1^{N_2}\oplus D^{N_1}_{(1,15)}$
		$P_2^{N_2}$	$b^{N_1}_{(2,3)}=p_2^{N_2}\oplus D^{N_1}_{(2,3)}$	…	$b^{N_1}_{(2,14)}=p_2^{N_2}\oplus D^{N_1}_{(2,14)}$	$b^{N_1}_{(2,15)}=p_2^{N_2}\oplus D^{N_1}_{(2,15)}$
			$P_3^{N_2}$	…	…	…
				…	$b^{N_1}_{(13,14)}=p_{13}^{N_2}\oplus D^{N_1}_{(13,14)}$	$b^{N_1}_{(13,15)}=p_{13}^{N_2}\oplus D^{N_1}_{(13,15)}$
					$P_{14}^{N_2}$	$b^{N_1}_{(14,15)}=p_{14}^{N_2}\oplus D^{N_1}_{(14,15)}$
						$P_{15}^{N_2}$

격에 사용된 20개의 평문에서 발생한 충돌은 없다고 가정한다.)

4.2.1 기대효과

우리는 가장 기본적인 아이디어에 충실하여 평문을 선택하였다. 즉, 충돌 쌍이 발생하지 않는 평문을 선택하지 않기 때문에 다음 공격 시 충돌 발생 확률을 높이는 것이다. 〔Table 1〕에서처럼 평문선택 시 제외시키는 값이 많을수록 충돌 확률이 높아질 것으로 기대한다. 이러한 원리로, 왼쪽 바이트부터 고정시켜 나가기 때문에 가장 오른쪽에 있는 바이트(15번째 바이트)가 충돌 발생 확률이 다른 바이트들보다 높을 것이며, BP-Value가 255개가 된다면 무조건 해당 바이트에서 충돌 쌍을 확인할 것이라 예상한다.

Table 2. BP-Value of 15th Byte Using Blacklist about 20 Plaintext

$b^{0}_{(0,15)} = p^{1}_{0} \oplus D^{0}_{(0,15)}$
$b^{1}_{(0,15)} = p^{2}_{0} \oplus D^{1}_{(0,15)}$
⋮
$b^{19}_{(0,15)} = p^{20}_{0} \oplus D^{19}_{(0,15)}$
$b^{0}_{(1,15)} = p^{1}_{1} \oplus D^{0}_{(1,15)}$
$b^{1}_{(1,15)} = p^{2}_{1} \oplus D^{1}_{(1,15)}$
⋮
$b^{19}_{(1,15)} = p^{20}_{1} \oplus D^{19}_{(1,15)}$
⋮
$b^{0}_{(14,15)} = p^{1}_{14} \oplus D^{0}_{(14,15)}$
$b^{1}_{(14,15)} = p^{2}_{14} \oplus D^{1}_{(14,15)}$
⋮
$b^{19}_{(14,15)} = p^{20}_{14} \oplus D^{19}_{(14,15)}$

4.3 성능측정

우리는 위의 충돌 쌍 공격 방법을 수행하여 성능을 측정하였다. 〔Fig.3〕은 30개의 평문을 사용하는 동안 관찰된 충돌 쌍(유효 충돌 쌍을 고려하지 않은)과 유효 충돌 쌍의 수를 5개의 평문 단위로 나누어서 측정한 것이다. 결과를 측정하기 위하여 서로 다른 30개의 평문을 10만 번씩 반복적으로 수행한 후 평균을 내었다. 30개의 평문으로 얻은 충돌 쌍 수와 유효 충돌 쌍의 수는 대략 각각 13.5개, 12개이다.

각 구간별로 살펴보면 평문 5개당 약 2.3개의 충돌 쌍을 관찰할 수 있다. 또한 유효 충돌 쌍의 수는 공격이 진행됨에 따라 줄어드는 것을 확인할 수 있는데 이

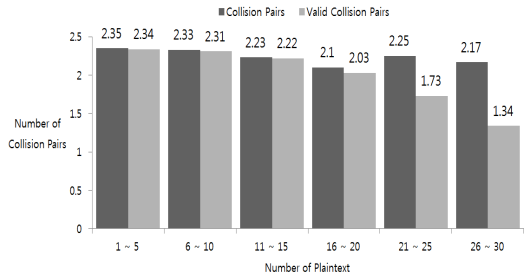


Fig. 3. Result of using the method 1

것은 공격인 진행됨에 따라 충돌 쌍이 축적되므로 그만큼 새로 발견된 충돌 쌍이 유효하게 될 확률은 줄어들기 때문이다. 이러한 이유로 유효 충돌 쌍을 고려하지 않은 것은 공격이 진행되어도 발생 빈도가 고른 것을 알 수 있다.

4.4 성능분석

위의 공격은 기존 Clavier 등이 명시한 수치에 미치는 못하는 결과이다. 따라서 우리는 성능 향상을 위하여 위의 공격에 대하여 분석을 진행하였다. 분석 결과 3가지 성능 저하 요인을 관찰하였으며 이러한 요인이 발생한 이유 및 해결방안에 대해 설명한다.

4.4.1 중복된 BP-Value

j번째 바이트의 평문을 선택할 때 고려되어야 하는 BP-Value을 살펴본 결과 서로 동일한 값이 존재하였다. 즉, 동일한 j에 대하여 서로 다른 i를 갖는 $b^{N}_{(i,j)}$에서 중복된 값이 존재하는 것이다. 이것은 BP-Value 작성이 두 바이트의 차분 값을 기준으로 되기 때문이다. j번째 바이트의 BP-Value을 작성할 때 공격자는 i번째 바이트를 고려한다. 다시 말해 서로 다른 $i_1 \neq i_2$에 대하여 $b^{N}_{(i_1,j)}$ 과 $b^{N}_{(i_2,j)}$는 서로 고려 대상이 아니기 때문에 서로에게 어떠한 영향도 미치지 않는다. 이와 같은 이유로 서로 다른 평문에 대하여 동일한 인덱스를 가지는 BP-Value(예. $b^{1}_{(1,2)}, b^{2}_{(1,2)}, b^{3}_{(1,2)}$)들은 서로 중복되는 값이 존재하지 않는다.

중복된 값으로 인해 BP-Value가 줄어들기 때문에 그만큼 선택될 수 있는 평문의 수도 증가한다. 그 결과 충돌 쌍의 확률도 그 만큼 낮아진다.

4.4.2 Trade-Off 발생

제안한 공격방식에 대하여 4.2.1에서 예상했던 것과는 달리 평문 바이트 선택 시 제외시키는 값(BP-Value)이 많을수록 충돌 발생확률이 크게 높아지지 않는 것을 확인할 수 있었다. 이것은 BP-Value가 많아지는 만큼 충돌 발생확률을 저하시키는 요소도 증가하기 때문이다. 즉, BP-Value이 많아짐에 따라 충돌 확률의 Trade-Off가 존재한다. 이러한 현상이 일어나는 원인을 규명하기 위하여 BP-Value가 작성되는 방식을 다시 한 번 살펴볼 필요가 있다. 예를 들어 〔Table 1〕에서 $b^{N_1}_{(14,15)}$가 계산되기 위해 고려되는 평문은 $P^{N_2}_{14}$ 뿐이다. $P^{N_2}_{14}$를 제외한 $P^{N_2}_0, P^{N_2}_1, \cdots, P^{N_2}_{13}$는 전혀 고려되지 않았다. 이와 같은 방식은 다음과 같은 문제점을 갖는다. 극단적으로 0-14번째 바이트의 키 값이 $b^{N_1}_{(14,15)}$와 각각의 $P^{N_2}_1, P^{N_2}_2, \ldots, P^{N_2}_{15}$의 차분 값과 같다면 15번째 바이트에서는 절대로 충돌이 발생하지 않을 것이다. 왜냐하면 15번째 바이트 평문은 $b^{N_1}_{(14,15)}$를 제외시키고 선택되었기 때문이다. 이처럼 $b^{N_1}_{(i,j)}$는 $P^{N_2}_i$를 제외한 나머지 평문 바이트와의 차분 값이 각각의 키 바이트들의 차분 값일 경우가 존재하므로 충돌 발생 확률이 떨어진다. 또한 충돌 쌍에 사용된 평문이 많을수록 작성되는 BP-Value은 증가하므로 충돌 발생을 증가시키는 확률뿐만 아니라 감소시키는 확률도 증가하기 때문에 전체적인 성능 향상에 저하를 가져온다. 따라서 우리는 이러한 문제를 해결하고 $b^{N_1}_{(i,j)}$가 계산되기 위해서 같은 행에 위치한 평문만을 고려(이하 '가로고려')할 뿐만 아니라 그것을 제외한 다른 평문들까지도(이하 '세로고려') 반영하여 BP-Value를 작성할 수 있도록 설계하였다. 여기에서 '가로고려'라고 하는 것은 j번째 바이트에 대하여 작성된 BP-Value $b^{N}_{(i,j)}$에 있어서 평문 i, j번째 바이트만을 고려하는 것을 말하며, '세로고려'라고 하는 것은 평문 i번째 바이트를 제외하고 $i < j$을 만족하는 나머지 평문 i, j번째 바이트들을 고려하는 것을 말한다.('가로고려'와 '세로고려'에 대한 구체적인 설명은 4.5.1에서 찾아볼 수 있다.)

4.4.3 평문 바이트 순서

BP-Value은 가장 왼쪽의 평문 바이트부터 순차적으로 고정시키면서 계산된다. 그 결과 가장 오른쪽에 위치한 바이트가 가장 많은 BP-Value를 갖게 된다. 따라서 가장 충돌 발생을 기대하는 바이트 순으로 가장 오른쪽 바이트부터 위치하도록 평문의 순서를 재배열 시킨다. 유효 충돌 쌍이 관찰된 평문 바이트는 더 이상 충돌 발생을 기대하지 않아도 되기 때문에 왼편으로 위치시키고 아직 충돌이 발생하지 않은 바이트의 번호 중에 하나를 골라 오른쪽 바이트에 위치시킨다. 우리는 위와 같은 과정을 BP-Vaule를 계산하기 전에 수행함으로써 공격자가 원하는 충돌 발생 확률을 높일 수 있다.

4.5 Blacklist를 활용한 평문 선택방법 2

위에서 분석한 성능 저하 원인을 극복하기 위하여 새로운 기법을 적용하여 성능 향상을 고안하였다.

첫째, 중복되어 발생하는 BP-Value은 설계한 기법에서 필연적으로 발생하는 현상임을 확인하였기 때문에 이것에 대한 수정사항은 없다.

둘째, 4.4.3에서 언급했듯이 작성된 유효충돌 쌍을 이용하여 가장 오른쪽 바이트부터 순차적으로 위치시켜야할 바이트를 재배열시켰다.

마지막으로 우리는 가로고려뿐만 아니라 세로고려까지 생각하여 기법을 설계하였다. 다음의 Cross-Check 기법은 이를 위한 것이다.

4.5.1 Cross Check

Cross-Check는 작성된 $b^{N_1}_{(i,j)}$와 $P^{N_2}_i$를 제외한 다른 평문과의 차분 값과 키 값들의 차분 값이 같지 않도록 필터링 시켜준다. 따라서 공격자는 $j < i$ 를 만족하는 $P^{N_2}_j$에 대하여 $b^{N_1}_{(i,j)}$와의 차분 값이 해당 바이트에 대한 Blacklist에 존재한다면 BP-Value로 남겨주고 그렇지 않으면 BP-Value Table에서 삭제시켜 평문 선택에 영향을 주지 않도록 한다. 예를 들어 설명하자. 평문 20개를 사용하여 Blacklist를 작성하였다고 가정한다. 또한 충돌은 하나도 발생하지 않았다고 가정한다. 그러면 0번째와 2번째 바이트에 존재하는 Blacklist는 $D^1_{(0,2)}, D^2_{(0,2)}, \cdots, D^{20}_{(0,2)}$ 이고, 1번째와 2번째 바

이트에 존재하는 Blacklist는 $D^1_{(1,2)}, D^2_{(1,2)}, \cdots, D^{20}_{(1,2)}$ 이다. 공격자는 $P_0^{N_{21}}$을 랜덤하게 선택하여 고정시킨 후 BP-Value를 작성한다. 이 후, $P_1^{N_{21}}$을 랜덤하게 선택하여 고정시킨 다음 BP-Value를 작성한다. 이 때, 2번째 바이트의 BP-Value는 2종류의 집합으로 나눈다. 첫 번째는 0번째와 1번째 바이트에 관한 것 $(b^1_{(0,2)}, b^2_{(0,2)}, \cdots, b^{20}_{(0,2)})$이고 두 번째는 1번째와 2번째 바이트 관한 것 $(b^1_{(1,2)}, b^2_{(1,2)}, \cdots, b^{20}_{(1,2)})$이다. 우선 $b^i_{(0,2)}$ $(i=1,2,\cdots,20)$에 대하여 $P_1^{N_{21}}$과 차분 연산을 한다. 이 후, 연산된 20개의 결과 값을 1번째와 2번째 바이트의 Blacklist 값 $D^1_{(1,2)}, D^2_{(1,2)}, \cdots, D^{20}_{(1,2)}$과 비교하여 동일한 값이 존재하면 유지시키고 그렇지 않으면 삭제한다. 이와 동일하게 $b^i_{(1,2)}$ $(i=1,2,\cdots,20)$에 대하여 $P_0^{N_{21}}$과 차분 연산을 한다. 이 후, 연산된 20개의 결과 값이 0번째와 2번째 바이트의 Blacklist 값 $D^1_{(0,2)}, D^2_{(0,2)}, \cdots, D^{20}_{(0,2)}$과 비교하여 동일한 값이 있다면 유지시키고 그렇지 않으면 삭제한다. 이렇게 Cross-Check를 통해 필터 된 BP-Value는 같은 열을 제외한 평문과의 차분 값이 키 값들의 차분 값과 같지 않게 되기 때문에 충돌 발생 확률을 높일 수 있다.

〔Table 3〕은 공격에 사용한 평문 수에 따른 가장 오른쪽 바이트에 작성 된 BP-Value의 수를 나타낸다. 필터적용 결과 절반이 훨씬 넘는 수의 BP-Value가 필터되어 삭제된 것을 확인할 수 있었다.

Table 3. Number of Blacklist Cross Check

Number of Plaintext	Number of Blacklist
20	89.5
25	97.3
30	103.7

4.5.2 성능분석

〔Fig.4〕는 평문 20개 사용 이후, Blacklist Cross Check를 적용했을 때의 결과이다. 평문 21-25개에 대한 구간에서 기존 1.73개의 평균 충돌 쌍 개수가 1.79개로 증가된 것을 확인할 수 있다. 또한 평문 26-30개에 대한 구간은 1.34개에서 1.41개로 증가하였다. 이러한 향상된 수치는 불필요한 BP-Value를 제거한 효과이다. 하지만 〔Table 3〕에서처럼 Cross-Check 후 BP-Value가 급격하게 감소하기 때문에 충돌 쌍 감소효과도 동시에 발생한다. 따라서 높은 성능 향상을 보이지 못한 것으로 해석된다.

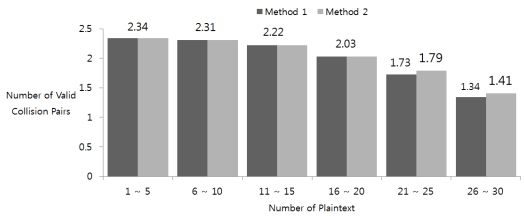


Fig. 4. Result of using the method 2

4.6 Blacklist를 활용한 평문 선택방법 3

위의 실험 결과 모두 20개 이상의 평문을 사용할 때 눈에 띄게 유효 충돌 쌍의 수가 줄어드는 것을 확인할 수 있다. 이것은 기존에 발생한 유효 충돌 쌍까지 고려하기 때문에 그 만큼 확률이 줄어든다. 따라서 20개의 평문을 사용해서 얻지 못한 약 5.8개의 유효 충돌 쌍에 좀 더 집중하기로 한다.

우선 가장 오른편(15번째 바이트)부터 충돌 발생을 기대하는 평문을 배치시킨다. 이 후, 15번째 바이트의 평문을 랜덤하게 선택하여 고정시킨다. 선택된 평문 값 정보를 이용하여 (0,15),(1,15), …,(14,15) 쌍에 대한 BP-Value를 작성한다. 각 바이트에는 15번째 바이트와 관계된 BP-Value만이 존재할 것이다. 공격자는 이것의 정보만을 이용하여 나머지 15개의 평문을 선택한다.

위의 방식은 15번째 바이트에 대한 Trade-Off가 발생하지 않기 때문에 그만큼 15번째 바이트에 대한 충돌 발생 확률의 증가를 기대한다. 우리는 가장 오른쪽 바이트, 가장 오른쪽 2개의 바이트, 가장 오른쪽 3개의 바이트, 가장 오른쪽 4개의 바이트에 대한 BP-Value를 작성하여 각각 실험을 진행하였다.

4.6.1 성능분석

〔Fig.5〕는 평문 20개 사용 이후에 위의 기법을 적용시킨 결과이다. 가장 오른쪽 1개의 바이트(15번째)만을 고정시킨 결과, 가장 오른쪽 2개의 바이트(14,15번째)만을 고정시킨 결과, 가장 오른쪽 3개의 바이트(13,14,15번째)만을 고정시킨 결과, 가장 오른

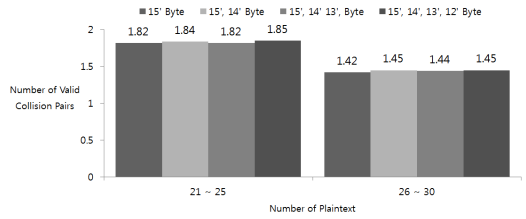


Fig. 5. Result of using the method 3 fixing 1,2,3,4 Bytes

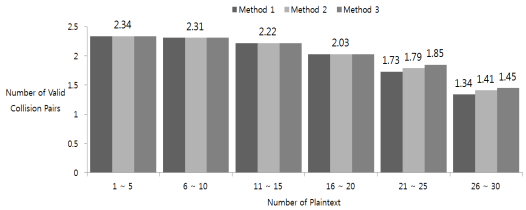


Fig. 6. Result of using the method 3

쪽 4개의 바이트(12,13,14,15번째)만을 고정시킨 결과이다. 각 4가지 경우에 대하여 눈에 띄게 성능의 차이가 생기는 것은 찾아볼 수 없었다. 하지만 방법 1, 방법 2를 사용했을 때 보다 성능이 향상됨을 확인할 수 있었다. 특히 평문 21-25개에 대한 구간에서 더 높은 성능 향상이 이루어졌다. 본 방법은 충돌 쌍의 가장 큰 영향을 주는 가로고려와 세로고려 중 극단적으로 세로고려만을 적용하여 수행하였다. 따라서 위의 결과로 Trade-Off를 구성하는 2가지 요인 중 불필요한 BP-Value의 값을 제거하는 것이 성능 향상에 좀 더 중요한 작용을 하는 것이라 판단된다.

4.7 Blacklist를 활용한 평문 선택방법 4

본 절은 평문 바이트 순서를 결정하는 새로운 규칙에 대하여 설명한다.

기존의 방식은 유효 충돌 쌍 그룹 중에서 그룹 내에 가장 많은 평문 바이트를 가지고 있는 순서로 가장 오른쪽부터 배치시켰다. 하지만 다음의 예를 살펴보면 이러한 방식의 문제점을 확인할 수 있다.

〔Fig.7〕은 2개의 VCP 그룹을 나타낸다. 그룹 1은 10개의 유효 충돌 쌍이 존재하고 그룹 2는 4개의 유효 충돌 쌍이 존재한다. 만약 공격자는 그룹 1에서의 한 바이트와 그룹 2에서의 한 바이트의 충돌 쌍을 찾는다면 공격은 종료될 것이다. 기존 방식에 의하면 그룹 1의 평문 바이트를 15번째 바이트부터 위치시켰다. 하지만 그룹 2의 평문 바이트를 15번째

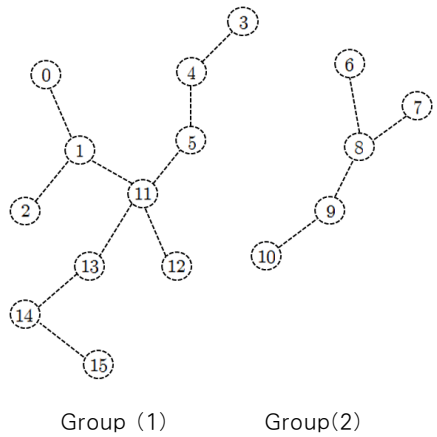


Fig. 7. Two groups of Valid Collision Pairs

바이트부터 위치시키는 것이 충돌 발생 확률을 높이는 것이다. 왜냐하면 그룹 1의 평문 바이트가 충돌 발생을 기대하는 것은 그룹 2의 5개의 평문 바이트이고, 반대로 그룹 2의 평문 바이트가 충돌 발생을 기대하는 것은 그룹 1의 11개의 평문 바이트이기 때문에 5개로부터 충돌 발생을 기대하는 것 보다 11개로부터 충돌 발생을 기대하는 것이 더 높은 확률을 갖기 때문이다. 그러므로 우리는 유효 충돌 쌍을 고려하여 가장 많은 바이트와 충돌 발생을 기대할 수 있는 바이트를 가장 오른쪽에 배치되도록 규칙을 변경하였다.

4.7.1 성능분석

〔Fig.8〕은 방법 4를 방법 1,2,3에 적용시켜 얻은 결과이다. 두 개의 인접한 막대그래프 중 왼쪽의 막대그래프는 방법 4를 적용시키지 않은 것이고, 오른쪽의 막대그래프는 방법 4를 적용시킨 것이다. 전반적으로 적용했을 때와 하지 않았을 때의 차이는 매우 작지만 성능은 향상되었음을 관찰할 수 있었다. 이러한 결과는 앞에서 예상한 것과 같이 충돌을 가장 기

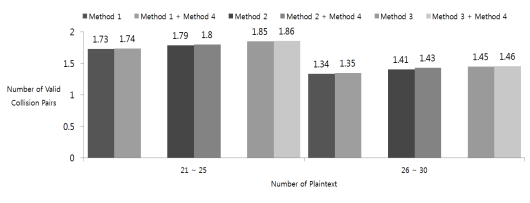


Fig. 8. Result of using the method 4

대하는 바이트를 가장 오른쪽에 놓아 확률을 높였기 때문이다.

4.8 Blacklist를 활용한 평문 선택방법 5

본 절에서 제안하는 방법은 평문 바이트에 작성된 불필요한 BP-Value를 줄이는 기법이다. 여기서 가장 핵심적인 아이디어는 같은 그룹 내에 존재하는 평문 바이트 사이에 발생하는 충돌은 더 이상 의미가 없다는 것이다. 따라서 평문이 선택되어 고정된 후 BP-Value 작성 시 같은 충돌 쌍 그룹에 존재하는 평문 바이트는 제외시킨다. 또한 충돌 쌍 그룹 당 하나의 평문 바이트만을 선택하여 BP-Value를 작성한다.

예를 들어 〔Table 4〕의 그룹 1에서 최초로 0번째 바이트의 평문을 선택하여 고정 시킨 후 같은 충돌 쌍 그룹의 평문 바이트(1,2,3,4번째)는 BP-Value을 작성하지 않고 나머지 평문 바이트(5-15번째)에 대해서만 작성한다. 이 후, 그룹 2의 5번째 평문 바이트에 대하여 평문을 선택한 후 고정시킨다. 그러면 그룹 2의 다른 평문 바이트(6,7번째)를 제외한 나머지 바이트에 대하여 BP-Value를 작성한다.

의미 없는 평문 바이트들 간 충돌은 고려할 필요가 없다. 따라서 선택된 평문바이트가 속한 충돌 쌍 그룹 내에 존재하는 다른 평문 바이트는 제외시키고 BP-Value을 작성하는 것이 가능하다. 결과적으로 불필요한 BP-Value이 줄어들었기 때문에 Trade-Off는 그만큼 감소할 것이며 충돌 발생 확률이 증가할 것으로 예상한다.

4.8.1 성능분석

〔Fig.9〕은 방법 1,2,3,5에 대한 결과 값이다. 방법 5의 기법이 다른 기법들 보다 가장 성능이 좋음을 확인할 수 있다. 이것은 불필요한 BP-Value를 제거함으로써 Trade-Off현상으로 인한 성능 저하 효과가 감소되었음을 알 수 있다. 방법 5는 하나의 충돌 쌍 그룹에 존재하는 충돌 쌍들이 많으면 많아질수록 더욱 효과적이다. 따라서 평문이 많이 사용될수록 더 높은 성능을 기대할 수 있다.

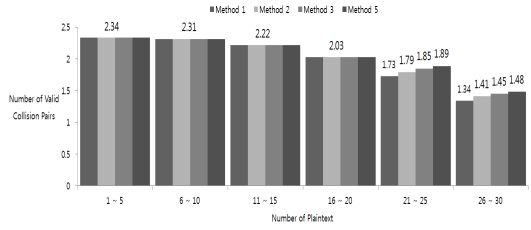


Fig. 9. Result of using the method 5

V. 결론 및 향후연구

본 논문은 효율적인 충돌 쌍 공격을 위한 다양한 기법을 소개하였다. 충돌 쌍이 일어나지 않은 평문 차분정보를 List로 작성하여 다음 공격에 이를 Blacklist를 활용할 수 있도록 기본적인 개념 및 용어를 정의하였으며 이를 바탕으로 여러 가지 활용방안을 기술하였다. 또한 각 방법에 대한 성능 분석을 통해 문제점 도출하고 성능 향상을 꾀하였다.

목표한 결과 수치와 이보다 더 좋은 성능을 얻기 위하여 다음과 같은 향후 연구가 필요하다.

Table 4. Example of building the BP-Value using the method 5

Group 1					Group 2			Group 3		LeftByteNums			
0	1	2	3	4	5	6	7	8	9	10	11	…	15
① P_0^N					②BP-Value of P_0^N								
					$b_{(0,5)}^N$	$b_{(0,6)}^N$	$b_{(0,7)}^N$	$b_{(0,8)}^N$	$b_{(0,9)}^N$	$b_{(0,10)}^N$	$b_{(0,11)}^N$	…	$b_{(0,15)}^N$
④BP-Value of P_5^N					③ P_5^N			④BP-Value of P_5^N					
$b_{(0,5)}^N$	$b_{(1,5)}^N$	$b_{(2,5)}^N$	$b_{(3,5)}^N$	$b_{(4,5)}^N$				$b_{(5,8)}^N$	$b_{(5,9)}^N$	$b_{(5,10)}^N$	$b_{(5,11)}^N$	…	$b_{(5,15)}^N$
⑥BP-Value of P_8^N								⑤ P_8^N		⑥BP-Value of P_8^N			
$b_{(0,8)}^N$	$b_{(1,8)}^N$	$b_{(2,8)}^N$	$b_{(3,8)}^N$	$b_{(4,8)}^N$	$b_{(5,8)}^N$	$b_{(6,8)}^N$	$b_{(7,8)}^N$			$b_{(8,10)}^N$	$b_{(8,11)}^N$	…	$b_{(8,15)}^N$
⑦Choose Left Plaintext Considering BP-Value that Calculated Upon													
	P_1^N	P_2^N	P_3^N	P_4^N		P_6^N	P_7^N		P_9^N	P_{10}^N	P_{11}^N	…	P_{15}^N

1. 성능 저하에 가장 큰 원인은 불필요한 BP-Value에 의한 Trade-Off 증가이다. 따라서 제안된 여러 가지 기법들을 적절하게 조합하여 최적의 BP-Value를 작성해야 할 것이다.

2. 제안된 방법은 평문 20개를 사용하고 난 후에 적용되는 기법들이다. 이것은 평문 20개 이후부터 현저히 줄어드는 유효 충돌의 수를 높이기 위한 의도였다. 따라서 평문1부터 적용시킬 수 있도록 기법을 설계해야한다. 이러한 기법은 기존 20-30의 구간에서만 기대했던 성능 향상을 1-30의 구간 모두에서 기대할 수 있기 때문에 전체적인 성능 향상을 기대할 수 있다.

3. 확률에 기반 하여 기법의 설계 및 성능 분석에 대한 연구가 이루어져야 한다. 충돌 발생에 대한 기댓값을 확률로 표현할 수 있다면 훨씬 더 정교하게 공격에 대한 분석이 용이해질 뿐만 아니라 설계 단계에서도 확률에 기반 하여 공격을 시뮬레이션 할 수 있기 때문에 시행착오를 줄일 수 있다.

현재까지 진행된 연구를 바탕으로 위의 추가적인 연구를 수행함으로써 효율적으로 향상된 충돌 쌍 공격을 설계할 수 있을 것이다.

References

[1] P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis," CRYPTO'99, LNCS 1666, pp. 388-397, 1999.

[2] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," CRYPTO'97, LNCS 1294, pp. 513-525, 1997.

[3] D. Boneh, R. A. DeMillo and R. J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," EUROCRYPT'97, LNCS 1233, pp. 37-51, 1997.

[4] P. Kocher, J. Jaffe, and B. Jun, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Others Systems," CRYPTO'96, LNCS 1109, pp. 104-113, 1996.

[5] P. Kocher, J. Jaffe, and B. Jun, "Introduction to differential power analysis and related attacks," White Paper, Cryptography Research, http://www.cryptography.com/resources/whitepapers/DPATechInfo.pdf. 1998.

[6] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," CHES 2004, LNCS 3156, pp. 16-29, 2004.

[7] M.-L. Akkar and C. Giraud, "An Implementation of DES and AES, Secure against Some Attacks," CHES 2001, LNCS 2162, pp. 309-318, 2001.

[8] J. Blömer, J. Guajardo and V. Krummel, "Provably Secure Masking of AES," SEC 2005, LNCS 3357, pp. 69-83, 2005.

[9] E. Oswald, S. Mangard and N. Pramstaller, "A Side-Channel Analysis Resistant Description of the AES S-Box," FSE 2005, LNCS 3557, pp. 199-228, 2005.

[10] C. Herbst, E. Oswald and S. Mangard, "An AES Smart Card Implementation Resistant to Power Analysis Attacks," ACNS 2006, LNCS 3989, pp. 239-252, 2006.

[11] E. Oswald and K. Schramm, "An Efficient Masking Scheme for AES Software Implementations," WISA 2005, LNCS 3786, pp. 292-305, 2006.

[12] Hee-seok Kim, Tae-hyun Kim, Dong-guk Han, and Seok-hie Hong, "Efficient Masking Methods Appropriate for the Block Ciphers ARIA and AES," ETRI Journal. vol. 32, no. 3. pp. 370-379, Jun. 2010.

[13] T.S. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," CHES 2000, LNCS 1965, pp. 238 - 251. 2000.

[14] K. Schramm, T. Wollinger, C. Paar, "A New Class of Collision Attacks and Its Application to DES," FSE 2003. LNCS 2887, pp. 206-222. 2003.

[15] K. Schramm, G. Leander, P. Felke and

C. Paar, "A Collision Attack on AES: Combining Side Channel and Differential Attack," CHES 2004, LNCS 3156, pp. 163-175, 2004.

[16] A. Bogdanov, "Improved Side-Channel Collision Attacks on AES," SAC 2007, LNCS 4876, pp. 84-95. 2007.

[17] A. Moradi, O. Mischke, and T. Eisenbarth, "Correlation-Enhanced Power Analysis Collision Attack," CHES 2010, LNCS 6225, pp. 125-139, 2010.

[18] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Improved Collision-Correlation Power Analysis on First Order Protected AES," CHES 2011, LNCS 6917, pp. 49-62, 2011.

[19] S. Chari, C.S. Jutla, J.R. Rao and P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks," CRYPTO'99. LNCS 1666, pp. 398 - 412, 1999.

[20] L. Goubin and J. Patarin, "DES and differential power analysis (The "Duplication" Method)," CHES 1999. LNCS 1717, pp. 158 - 172, 1999.

[21] Jovan D. Golic and C. Tymen, "Multiplicative Masking and Power Analysis of AES," CHES 2002, LNCS 2523, pp. 198-212, 2002.

[22] E. Trichina, D. De Seta, and L. Germani, "Simplified Adaptive Multiplicative Masking for AES," CHES 2002, LNCS 2523, pp. 187-197, 2003.

[23] J. Blomer, J. Guajardo, and V. Krummel, "Provably Secure Masking of AES," SAC 2004, LNCS 3357, pp. 69-83, 2005.

〈저자소개〉

김 은 희 (Eun-hee Kim) 학생회원
2012년 2월: 고려대학교 화공생명공학과 학사
2013년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 부채널 공격, 자바카드 구현 및 분석

김 태 원 (Tae-won Kim) 학생회원
2010년 2월: 광운대학교 수학과 학사
2012년 8월: 고려대학교 정보보호대학원 석사
2012년 8월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 부채널 공격, 스마트 카드 보안, 암호시스템 안전성 분석 및 고속구현

홍 석 희 (Seok-hie Hong) 종신회원
1995년 2월: 고려대학교 수학과 학사
1997년 2월: 고려대학교 수학과 석사
2001년 8월: 고려대학교 수학과 박사
1999년 8월~2004년 2월: (주) 시큐리티 테크놀로지스 선임연구원
2003년 8월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U.Leuven, ESAT/SCD-COSIC 박사후연구원
2005년 3월~2013년 8월: 고려대학교 정보보호대학원 부교수
2013년 9월~현재: 고려대학교 정보보호대학원 정교수
〈관심분야〉 대칭키·공개키 암호 분석 및 설계, 컴퓨터 포렌식