정 보 보 호 학 회 지
제 31 권 제1호, 2021. 2

NIST PQC Round 3 FALCON 전자서명 알고리즘의 전력 분석 취약점 연구

김 규 상*, 박 동 준*, 홍 석 희**

요 약

기존의 공개키 암호가 양자 알고리즘에 취약함이 밝혀지고 양자컴퓨터의 개발이 현실화됨에 따라 NIST는 미연방표준 양자 내성 암호 공모전을 실시하고 있다. FALCON은 공모전 Round 3까지 통과한 전자서명 알고리즘으로 서명 및 검증 속도가 빠르고 공개키 및 서명 길이가 짧다는 장점이 있다. 하지만 FALCON은 부동소수점 연산 등 특별한 구조로 설계되어 새로운 형태의 부채널 공격이 존재할 수 있다. 본 논문에서는 FALCON에 대한 세 가지 전력 분석 공격의 가능성을 제시한다. 또한 주어진 공격을 활용하여 개인키를 복원하는 방안에 대해서 제시한다.

Ⅰ. 서 론

전자서명은 안전하지 않은 통신채널을 통해 본인을 인증할 수 있는 수단이다. 전자서명 알고리즘으로는 미연방표준인 DSA, ECDSA 등이 널리 사용되고 있다[1]. 하지만 기존에 사용하던 전자서명 알고리즘들은 쇼어 알고리즘에 의해 다항 시간 내에 공격당하므로[2], 양자컴퓨터가 개발되는 가까운 미래에는 더이상 사용할 수 없다. 이에 NIST에서는 양자컴퓨팅 환경에서도 안전한 전자서명 알고리즘을 표준화하기 위해 양자 내성 암호 공모전을 실시하고 있다. 양자 내성 암호 공모전은 현재 Round 3까지 진행됐으며 그 후보는 공개키 암호 알고리즘 4종, 전자서명 알고리즘 3종이다.

양자 내성 암호는 기반 문제에 따라 크게 격자(lattice), 부호(code), 다변수(multivariate), 해시(hash), 아이소제니(isogeny) 기반 암호로 분류된다. 이 중에서 격자 기반 암호는 공모전 최종후보에 올라온 알고리즘 7종 가운데 5종에 달할 정도로 관심과 기대가 높으며 연구 가치가 크다. 특히 FALCON은 격자 기반 전자서명 알고리즘으로서 서명 및 검증 속도가 빠르고 공개키 및 서명 길이가 짧다는 장점이 있다[3].

한편, 부채널 공격은 암호 알고리즘을 처리하는 동안 기기에서 발생하는 물리적 신호를 통해 비밀정보를 알아내는 기법이다. 부채널 공격은 암호 알고리즘의 수학적 안전성과는 별개로 현실적인 위협이므로 양자 내성 암호 또한 부채널 공격에 대한 안전성이 확보되어야 한다. 특히 FALCON의 경우에는 Fast Fourier Transform(FFT), discrete Gaussian 샘플링 뿐만 아니라 다른 알고리즘에서 볼 수 없던 부동소수점 연산을 사용하는 등 특별한 구조로 설계되어 새로운 형태의 부채널 공격이 존재하는지 연구할 필요가 있다[4].

본 논문에서는 격자 기반 전자서명 알고리즘인 FALCON에 대한 세 가지 전력 분석 공격의 가능성과 비밀정보 복원 방안을 논하고자 한다. 첫 번째는 다수의 서명 과정에서 발생하는 FFT 곱셈에서 부동소수점 곱 연산을 대상으로 하는 공격이다. 다른 두 가지는 한 번의 서명 과정에서 발생하는 샘플링을 대상으로 하는 공격이다. 구체적으로는 discrete Gaussian 샘플링을 담당하는 *SamplerZ* 내부에서 half Gaussian 분포로 정수를 추출하는 *BaseSampler*를 대상으로 하는 공격과, *SamplerZ* 내부에서 *BaseSampler*의 결과를 discrete Gaussian 분포로 변환하는 과정을 대상으로 하는 공격이다.

본 논문의 구성은 다음과 같다. 먼저 2장과 3장에서는 각각 배경지식과 FALCON 전자서명 알고리즘을

본 연구는 삼성전자의 지원(과제번호 IO201209-07857-01)을 받아 수행된 결과임.
* 고려대학교 정보보호학과(대학원생, ks9509@korea.ac.kr), (대학원생, djpark@korea.ac.kr)
** 고려대학교 (교수, shhong@korea.ac.kr)

소개한다. 4장과 5장에서는 각각 FALCON 전자서명 알고리즘에 대한 세 가지 전력 분석 공격 가능성과 비밀정보 복원 방안을 제시한다. 마지막으로 6장에서는 결론을 맺는다.

Ⅱ. 배경지식

본 장에서는 논문 내용의 이해를 돕기 위한 배경지식에 대해 살펴본다.

2.1. Fast Fourier Transform(FFT)

이산 푸리에 변환은 $x=(x_0,x_1,\dots,x_{n-1})$에 대해서 다음과 같이 정의된다. ($n=2^m$ 일 때, nth root of unity인 $\zeta_n=e^{2\pi i/n}$은 $\zeta_n^n=1$을 만족한다.)

$$X_k=\sum_{m=0}^{n-1}x_k e^{-\frac{2\pi i}{n}km} \quad (k=0,1,...,n-1)$$

역변환은 다음과 같이 정의된다.

$$x_k=\sum_{m=0}^{n-1}X_k e^{\frac{2\pi i}{n}km} \quad (k=0,1,...,n-1)$$

두 과정의 시간복잡도는 $O(n^2)$이다. 이산 푸리에 변환을 거치면 다항식의 곱셈, 나눗셈을 계수별로 처리하여 $O(n)$만에 연산할 수 있다는 장점이 있다.

FFT는 이산 푸리에 변환과 그 역변환을 빠르게 수행하는 알고리즘으로 수치해석, 신호처리 등 다양한 분야에서 사용되고 있다[5]. FFT는 butterfly operation이라 불리는 연산을 사용하는데, 이 연산은 짝수 지수와 홀수 지수를 분리하여 계산하며 이를 재귀적으로 진행한다. 결과적으로 FFT는 위의 두 변환을 $O(n\log n)$만에 수행한다.

한편, NTT는 유한체 $\mathbb{Z}_p$에서 이산 푸리에 변환을 진행한다. $p=1 \bmod 2n$을 만족하는 소수 p에 대하여, x^n+1은 $\mathbb{Z}_p$에서 n개의 근을 갖게 되며 이 n개의 근으로 $\mathbb{Z}_p[x]/(x^n+1)$의 모든 원소를 표현할 수 있다. NTT도 FFT와 마찬가지로 시간복잡도를 $O(n\log n)$까지 낮출 수 있으며, 곱셈과 나눗셈을 $O(n)$만에 연산할 수 있다는 장점을 갖고 있다.

2.2. 부동소수점 연산

컴퓨터는 모든 데이터를 이진수로 저장하기 때문에 정수 및 유리수를 표기하는 규칙이 필요하다. IEEE-754는 유리수를 32bit 또는 64bit로 표기하는 부동소수점에 관한 표준이다[6]. IEEE-754의 64bit 부동소수점 표기법에서는 최상위 1bit가 부호를, 다음 11bit가 지수를, 나머지 52bit가 가수를 각각 의미한다. 부호, 지수, 가수를 각각 s, e, m라 하면, 64bit 부동소수점 데이터의 값은 $(-1)^s\times 2^{e-1023}\times 1.m$로 해석된다.

Ⅲ. FALCON 전자서명 알고리즘

본 장에서는 격자 기반 전자서명 알고리즘인 FALCON의 개요 및 내부 알고리즘에 관하여 서술한다.

3.1. FALCON 전자서명 알고리즘의 개요

FALCON[3]은 fast Fourier lattice-based compact signatures over NTRU의 줄임말로, NTRU 격자에서 Short Integer Solution 문제가 양자알고리즘으로도 효율적으로 풀리지 않는다는 점에 안전성을 두고 있다. FALCON은 GPV framework, NTRU, fast Fourier 샘플링을 융합하여 설계되었다. 구체적으로 FALCON은 안전한 격자 기반 전자서명을 만들기 위한 방법론인 GPV framework[7]에 따라 설계되었으며, fast Fourier 샘플링이라는 트랩도어 샘플러를 사용한 NTRU를 기반으로 한다.

FALCON은 $\mathbb{Q}[x]/(\phi)$인 체에서의 연산을 사용하며 이때, $n=2^\kappa$(주로 512, 1024)이고 $\phi=x^n+1$이다. FALCON의 공개키 A와 개인키 B는 같은 격자의 기저를 가진다. 먼저 공개키는 $A=\begin{bmatrix}-h & I_n\\ qI_n & O_n\end{bmatrix}$의 형태이며 각 원소는 $n\times n$행렬이기 때문에 전체 행렬의 크기는 $2n\times 2n$이다. 이때 $q=12289$이고, 다항식 h의 계수는 0과 $q-1$사이의 수로 구성되어있다. ($n-1$차 다항식 h는 $x^ih \bmod \phi$가 각 행이 되어 $n\times n$행렬로 나타낼 수 있다.)

개인키는 $B=\begin{bmatrix}g & -f\\ G & -F\end{bmatrix}$의 형태이다. 이때 다항식 f,g,F,G는 작은 정수를 계수로 가지며 NTRU equation인 $fG-gF=q \bmod \phi$를 만족해야 한다. 공

개키와 개인키가 같은 기저를 가지려면 $h = g/f \bmod \phi$를 만족해야 한다. 한편 FALCON 트리 T는 B를 통해 만들어지며 샘플링 과정에서 사용되는 비밀정보이다.

3.2. FALCON 전자서명 내부 알고리즘

FALCON 전자서명 알고리즘은 공개키 및 개인키를 만드는 *keygen*, 서명 값을 만드는 *sign*, 검증하는 단계인 *verify*로 크게 세 가지의 내부 알고리즘으로 구성되어있다.

*sign*은 메시지 m, 개인키 sk를 입력으로 받은 후, 솔트 값 r과 서명 값 s를 출력한다. *sign*을 격자에 맞춰 설명하면, 격자 위에 있지 않은 t라는 점을 *ffSampling*을 통해 근처 격자 위의 점으로 옮긴 후, 그 차이 값을 압축하여 서명 값으로 쓴다. *sign*에 나오는 *HashToPoint*는 메시지의 해시값을 정수 계수 다항식으로 바꾸는 역할이며 *Compress*는 서명 값을 압축하는 역할이다. *ffSampling*은 FFT 상태에서 수행하기 좋기 때문에 *ffSampling*이란 이름이 붙었으며 구조 또한 실제 FFT의 butterfly operation과 유사함을 알 수 있다. *ffSampling* 내부의 *splitfft*, *mergefft*는 도메인을 변환하는 과정이며, *SamplerZ*는 discrete Gaussian 분포와 유사한 분포에서 입력과 유사한 정수를 뽑는 과정이다. 평균과 표준편차를 입력으로 받으며 *SamplerZ* 내부는 기각 샘플링을 기반으로 작동한다. *SamplerZ* 내부의 *BerExp*는 $ccs \cdot \exp(-x)$의 확률로 1을 반환한다. *BaseSampler*는 half Gaussian 분포를 따르며

알고리즘 1. FALCON *sign*

입력: 메시지 m, 개인키 sk, bound β^2
출력: 메시지 m의 서명 값 sig

1. $r \leftarrow 0,1^{320}$ uniformly
2. $c \leftarrow HashToPoint(r \| m, q, n)$
3. $t \leftarrow \left(-\frac{1}{q} FFT(c) \odot FFT(F), \frac{1}{q} FFT(c) \odot FFT(f)\right)$
4. *do*
4.1 $z \leftarrow ffSampling_n(t, T)$
4.2 $s' = (t - z)\hat{B}$
5. *while* ($\| s' \|^2 > \lfloor \beta^2 \rfloor$)
6. $(s_1, s_2) \leftarrow \mathrm{invFFT(s')}$
7. $s \leftarrow Compress(s_2)$
8. *return* $sig = (r, s)$

알고리즘 2. FALCON *ffSampling*

입력: $\boldsymbol{t} = (t_0, t_1) \in FFT(\mathbb{Q}[x]/(x^n+1))^2$
FALCON 트리 T
출력: $\boldsymbol{z} = (z_0, z_1) \in FFT(\mathbb{Z}[x]/(x^n+1))^2$

1. if $n = 1$ *then*
1.1 $\sigma' \leftarrow T.value$
1.2 $z_0 \leftarrow SamplerZ(t_0, \sigma')$
1.3 $z_1 \leftarrow SamplerZ(t_1, \sigma')$
1.4 *return* $z = (z_0, z_1)$
2. $(l, T_0, T_1) \leftarrow (T.value, T.leftchild, T.rightchild)\}$
3. $\boldsymbol{t_1} \leftarrow splitfft(t_1)$
4. $\boldsymbol{z_1} \leftarrow ffsampling_{n/2}(\boldsymbol{t_1}, T_1)$
5. $z_1 \leftarrow mergefft(\boldsymbol{z_1})$
6. $t_0' \leftarrow t_0 + (t_1 - z_1) \odot l$
7. $\boldsymbol{t_0'} \leftarrow splitfft(t_0')$
8. $\boldsymbol{z_0} \leftarrow ffsampling_{n/2}(\boldsymbol{t_0}, T_0)$
9. $z_0 \leftarrow mergefft(\boldsymbol{z_0})$
10. *return* $\boldsymbol{z} = (z_0, z_1)$

*SamplerZ*는 *BaseSampler* 결괏값을 이용하여 Gaussian 분포에 맞게 수를 뽑을 수 있다. *BaseSampler*에서 $RCDT$는 2^{72}에서 CDT배열의 각 요소를 뺀 배열이며 상수 배열이다.

검증단계인 *verify*는 굉장히 간단하며 연산량이 적어 FALCON은 검증 시간이 짧다는 장점도 가지고 있다. *verify*는 *HashToPoint*, *Decompress*를 통해 *sign*에서 사용한 c, s_2를 복구하고 이를 이용해서 검증과정을 거친다. *Decompress*는 *Compress*의 올바른 출력값을 입력으로 넣는다면, 원래의 값으로 복원할 수 있는 성질을 갖고 있다.

알고리즘 3. FALCON *SamplerZ*

입력: μ, σ' $(\sigma' \in [\sigma_{\min}, \sigma_{\max}])$
출력: $z \in \mathbb{Z}$

1. $r \leftarrow \mu - \lfloor \mu \rfloor$
2. $ccs \leftarrow \sigma_{\min}/\sigma'$
3. *while*(1) *do*
3.1 $z_0 \leftarrow BaseSamper()$
3.2 $b \leftarrow UniformBits(8) \bmod 2$
3.3 $z \leftarrow b + (2 \cdot b - 1) z_0$
3.4 $x \leftarrow \frac{(z-r)^2}{2\sigma'^2} - \frac{z_0^2}{2\sigma_{\max}^2}$
3.5 if$(BerExp(x, ccs) = 1)$ *then*
3.5.1 *return* $z + \lfloor \mu \rfloor$

알고리즘 4. FALCON *BaseSampler*

입력: 없음
출력: $z_0 \in 0,...,18$

1. $u \leftarrow UniformBits(72)$
2. $z_0 \leftarrow 0$
3. for $i = 0,...,17$ *do*
3.1 $z_0 \leftarrow z_0 + [[u < RCDT[i]]]$
4. *return* z_0

알고리즘 5. FALCON *verify*

입력: $m,\ sig,\ pk$
출력: *Accept* or *reject*

1. $c \leftarrow HashToPoint(r \| m,q,n)$
2. $s_2 \leftarrow Decompress(s)$
3. $s_1 \leftarrow c - s_2 h \bmod q$
4. *return* $\| (s_1,s_2) \|^2 \leq \lfloor \beta^2 \rfloor$

Ⅳ. FALCON의 전력 분석 공격 방식 제시

본 장에서는 FALCON의 전력 분석 공격 방법을 세 가지 제시한다.

부채널 공격은 분석 방법이 다양한데, 시간 분석 공격(Timing Attack)[8], 전력 분석 공격(Power Analysis Attack)[9], 전자기파 분석 공격(Electromagnetic Analysis Attack)[10], 오류 주입 공격(Fault Injection Attack)[11] 등이 있다. 이들 중 전력 분석 공격은 기기의 전력소비량을 이용하여 중간값을 추측하고 비밀정보를 획득하는 공격이다. 상관 전력 분석(Correlation Power Analysis)[12]은 전력 분석 공격의 일종으로 알고리즘 내부의 중간값의 해밍 웨이트와 그때의 전력소비량이 상관관계가 있음을 이용한 공격이다. 추측한 중간값과 측정된 전력소비량 간의 상관관계가 가장 높게 나온 경우, 이때의 추측 값을 실제 중간값으로 간주한다.

4.1. 부동소수점 곱 연산을 이용한 상관 전력 분석

FALCON에서는 부동소수점 연산을 자주 사용하는데, 이때 나타나는 곱셈을 이용한 상관 전력 분석 공격을 알아본다. *sign*의 3번째 줄 연산에서는 $FFT(c) \odot FFT(F)$와 $FFT(c) \odot FFT(f)$ 연산이 중간과정으로 실행된다. 두 연산은 FFT 곱셈 연산으로 실제로는 요소별 부동소수점 곱셈 연산을 수행한다.

알고리즘 6. Mantissa multiplication part attack

입력: 32bit 부호 없는 정수 배열 a, 곱셈 부분 파형 k개
출력: 32bit 부호 없는 정수 b $(c_i = a_i \times b)$

1. a_i와 b의 하위 8bit 곱이 c_i의 하위 8bit 값을 생성하므로 예상 c의 하위 8bit의 해밍 웨이트와의 상관계수가 가장 높게 나타나는 b의 후보군 m쌍 획득

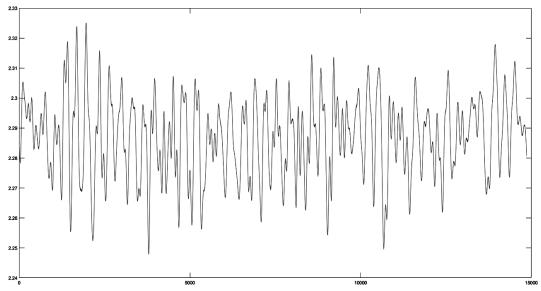
2. a_i와 b의 하위 16bit 곱이 c_i의 하위 16bit 값을 생성하므로 b값은 1번 줄에서의 후보군 m쌍을 하위 8bit로 하며 그 앞 8bit를 추측하여 예상 c_i의 하위 16bit의 해밍 웨이트와의 상관계수가 가장 높게 나타나는 b의 후보군 m쌍 획득

3. a_i와 b의 하위 24bit 곱이 c_i의 하위 24bit 값을 생성하므로 b값은 2번 줄에서의 후보군 m쌍을 하위 16bit로 하며 그 앞 8bit를 추측하여 예상 c_i의 하위 24bit의 해밍 웨이트와의 상관계수가 가장 높게 나타나는 b의 후보군 m쌍 획득

4. a_i와 b의 곱이 c_i값을 생성하므로 b값은 2번 줄에서의 후보군 m쌍을 하위 24bit로 하며 그 앞 8bit를 추측하여 예상 c_i의 해밍 웨이트와의 상관계수가 가장 높게 나타나는 b 반환

만약 $FFT(c)$값을 안다고 가정하면, 고정값이지만 모르는 값인 $FFT(F)$, $FFT(f)$와 매 연산 우리가 알 수 있는 값인 $FFT(c)$의 부동소수점 곱 연산만이 남는다.

IEEE-754 표기식 부동소수점 32bit 수 두 개의 곱은 상관 전력 분석을 이용해서 구할 수 있음이 알려졌지만[13], FALCON reference code에서는 부동소수점을 새로운 64bit 부호 없는 정수에 저장하여 연산을 재정의한다. 이 중 곱셈 연산은 가수의 곱셈, 지수의 덧셈, 부호의 배타적 논리합 순서로 진행한다. 그림 1은 이 곱셈 연산 전체 파형을 나타낸 것이다.



〔그림 1〕 FALCON 부동소수점 곱셈 파형

알고리즘 7. FALCON *BaseSampler* attack

입력: *BaseSampler*의 파형 m개
출력: z_0값 배열

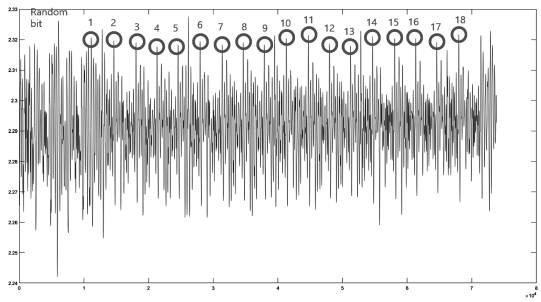
1. *BaseSampler* 파형 m개와 상수배열 $RCDT$를 사용하여 z_0값이 더해지는 위치를 찾음
2. z_0값이 더해지는 위치에서 파형이 변화하는 부분을 찾아 z_0값을 복원

가수 a, b의 곱셈을 하기 위해 a의 하위 25bit, 상위 27bit 각각을 a_0, a_1, b의 하위 25bit, 상위 27bit 각각을 b_0, b_1로 분리한다. 먼저 실제로는 $1.a$, $1.b$간의 곱셈이므로, a_1, b_1값 앞에 1을 붙여 28bit의 부호 없는 정수로 만든다. 그 후 정수의 곱셈을 이용하여 a_0b_0, a_0b_1, a_1b_0, a_1b_1값들을 계산한다. 마지막으로, $a_1b_1 2^{50} + (a_0b_1 + a_1b_0)2^{25} + a_0b_0$ 값을 구해서 2^{105}이상은 지수의 덧셈 부분으로 넘기며 나머지 값 중 상위 52bit를 새로운 가수로 만든다. 그러므로 32bit 부호 없는 정수 곱셈을 상관 전력 분석을 통해 복원할 수 있다면 가수의 곱셈 부분은 복원할 수 있다.

지수와 부호는 곱셈 연산이 아니기 때문에 위의 방법으로는 공격할 수 없다.

4.2. *BaseSampler*의 상관 전력 분석 공격

FALCON에서 직접적인 샘플링을 담당하는 *BaseSampler*를 살펴보자. 입력이 없고 출력은 *SamplerZ*에서 입력의 정수 부분과 출력 간에 거리를 얼마나 두는가를 의미한다. *BaseSampler*는 $RCDT$값을 이용한 알고리즘이기 때문에 $RCDT$ 상수 값을 이용한다면 파형에서 해당 연산의 위치를 알 수 있으며 $RCDT$는 감소수열이기 때문에 반복문을 통해 z_0값에 더해지는 값은 $1,\cdots,1,0,\cdots,0$이 되며 이때의 1의 개수가 z_0값을 결정한다. 해당 값은 $RCDT$의 값을 포함하므로 해당 연산을 파형에서 쉽게 찾을 수 있으며 파형이 나뉘는 부분이 반복문이 진행됨에 따라 연속적으로 일어나므로 z_0값은 쉽게 구할 수 있다. *BaseSampler*는 *sign* 1번 동안 최소 $2n$번 나타나므로 단일 서명 과정으로 충분하다. 그림 2는 *BaseSampler* 하나의 파형을 나타낸 파형이며 동그라미로 표시한 부분을 통해 반복문이 18번 반복됨을 확인할 수 있다.



〔그림 2〕 *BaseSampler* 파형

4.3. *SamplerZ*의 전력 분석 공격

4.2절에서의 결과 z_0를 이용하여 실제 z를 전력 분석 공격을 통해서 어떻게 추출할지 살펴보자. 실제 z를 구한다면 입력의 정수 부분과 *SamplerZ*의 출력값의 차이를 구할 수 있다. 4.2절에서 구한 z_0와 임의의 수 b ($b=0$ *or* 1)를 이용하여 $z = b + (2b-1)z_0$식을

〔표 1〕 z_0와 b에 따른 z값과 해밍 웨이트

z_0	z ($b=0$)	해밍 웨이트 ($b=0$)	z ($b=1$)	해밍 웨이트 ($b=1$)	z_0	z ($b=0$)	해밍 웨이트 ($b=0$)	z ($b=1$)	해밍 웨이트 ($b=1$)
0	0	0	1	1	10	-10	30	11	3
1	-1	32	2	1	11	-11	30	12	2
2	-2	31	3	2	12	-12	29	13	3
3	-3	31	4	1	13	-13	30	14	3
4	-4	30	5	2	14	-14	29	15	4
5	-5	31	6	2	15	-15	29	16	1
6	-6	30	7	3	16	-16	28	17	2
7	-7	30	8	1	17	-17	31	18	2
8	-8	29	9	2	18	-18	30	19	3
9	-9	31	10	2					

알고리즘 8. FALCON SamplerZ attack

입력: *SamplerZ*의 파형 m개, z_0값 배열
출력: z값 배열

1. z_0값이 같은 파형들을 모아 파형을 하나로 합침
2. 고점이 높은 파형은 b값이 0, 고점이 낮은 파형은 b값이 1로 b값 복원
3. 구한 b값들을 이용하여 z값들을 복원

통하여 z를 구한다. 위 식을 통해 구한 z는 z_0+1 또는 $-z_0$값을 갖게 되는데 *BaseSampler*의 결괏값은 $0 \le z_0 \le 18$ 이므로 $-18 \le z \le 19$의 범위를 가짐을 알 수 있다. 다음 표 1은 z_0값에 따라 z값과 z의 해밍 웨이트(32bit 기준)를 적은 표이다. 표1에서 z_0값이 0이 아니라면 b값에 따른 해밍 웨이트 값의 차이가 큼을 알 수 있다. *SamplerZ*는 *ffSampling*이 한 번 실행될 때 $2n$번 만큼 실행되므로 $n=512$를 기준으로 각 z_0당 평균 $1024/19=53.89$번 정도의 파형이 나옴을 알 수 있으므로 4.2절의 공격을 통해 얻은 z_0값을 이용하여 같은 z_0끼리 그룹으로 묶으면 b값에 따라 파형이 두 개의 소그룹으로 나뉘게 되고 이후에는 해밍 웨이트를 이용하여 b값을 쉽게 구할 수 있게 된다.

그림 3은 알고리즘 8을 $z_0=5$인 파형을 모아 나타낸 파형이며 굵은 선을 기점으로 위쪽을 0, 아래쪽을 1로 잡으면 실제 값과 동일한 b배열을 얻을 수 있다.

Ⅴ. FALCON의 전력 분석 공격의 활용 방안

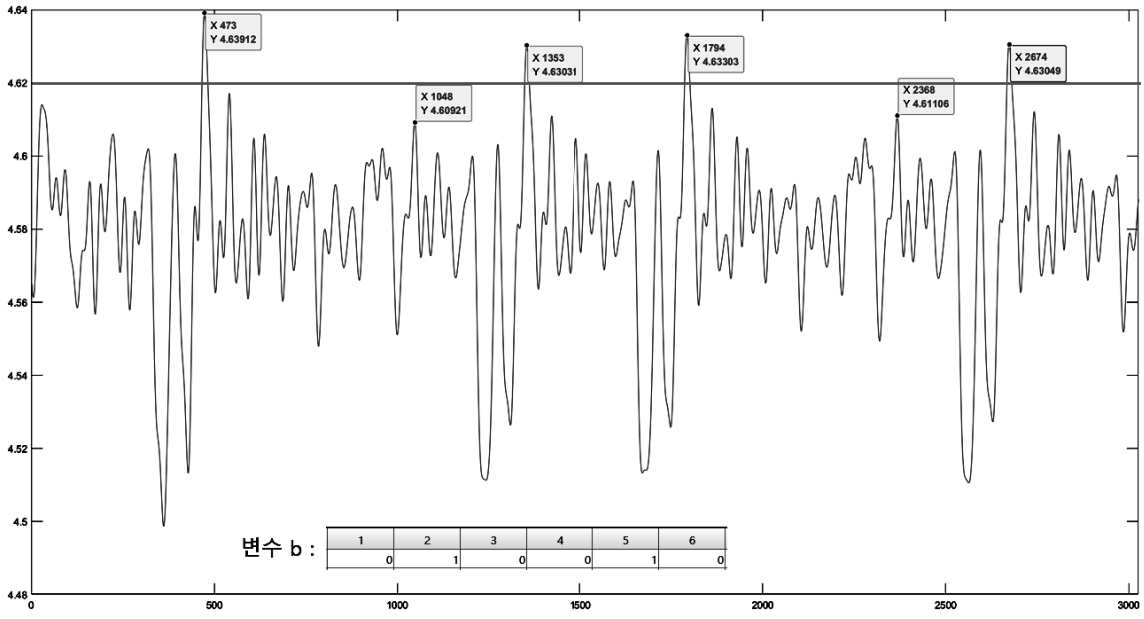
본 장에서는 4장에서의 공격 결과를 통해 사용자의 비밀정보를 복원하는 방법을 서술한다.

5.1. 부동소수점 곱 연산을 통한 개인키 복원

4.1절에서의 공격 가정에 따르면 공격자는 *sign*의 입출력을 얻을 수 있으며 같은 키로 여러 번 서명할 수 있다. 공격자는 r,m값을 이용하여 c값을 구할 수 있으므로 $FFT(c)$값을 알 수 있다. 이후, 부동소수점 곱 연산 공격이 성공한다면 $FFT(F)$, $FFT(f)$값을 구할 수 있으므로 *IFFT*를 통해 각각 F, f로 얻을 수 있다. 마지막으로 $g=fh \bmod q$와 $G=f^{-1}(q+gF) \bmod \phi$의 관계식으로부터 개인키의 나머지 부분을 구해 개인키 sk를 복원할 수 있다. 실제로는 F와 f 중 하나의 값만 얻을 수 있어도 *keygen* 내부의 알고리즘을 사용하면 개인키 전부를 복원할 수 있다.

5.2. *sign* 역추적

*sign*의 출력은 솔트 값인 r과 서명 값인 s이다. *verify*에서는 r과 m으로 *HashToPoint*를 계산하여 c



〔그림 3〕 SamplerZ 3.2, 3.3줄 부분 파형($z_0=5$)

를 얻은 후, 서명 값을 *Decompress*해서 s_2를 복원할 수 있다. 이후, $s_1 = c - s_2 h \bmod q$를 이용해서 s_1을 복원하며 $s' = FFT(s_1, s_2)$를 이용해서 s'을 복원한다. 하지만 $t, z, \hat{B}$값은 이 과정만으론 복원할 수 없다.

4.2, 4.3절에서 수행한 공격을 이용하면 *SamplerZ*의 결괏값과 입력값 μ의 차이의 정수 부분을 알 수 있다. 이를 *sign*의 4.2줄까지 확장 가능하며 솔트 값을 일정 값으로 유지하고 *sign*을 여러 번 수행한다 가정하자. *sign*의 4.2줄의 $s', t-z$ 값을 두 번 수행하여 얻은 후 그 차이인 Δ_s, Δ_z를 이용하여 $\Delta_s = -\Delta_z \hat{B}$를 나타낸다. 이를 이용하여 $\begin{pmatrix} \Delta_s^1 \\ \Delta_s^2 \\ \vdots \\ \Delta_s^n \end{pmatrix} = -\begin{pmatrix} \Delta_z^1 \\ \Delta_z^2 \\ \vdots \\ \Delta_z^n \end{pmatrix} \hat{B}$를 만든 후, $-\begin{pmatrix} \Delta_z^1 \\ \Delta_z^2 \\ \vdots \\ \Delta_z^n \end{pmatrix}$의 역행렬을 구한다면 $\hat{B}$를 복원할 수 있다.

5.3. FALCON 트리 T 복원

2020년 Fouque 등은 FALCON 알고리즘에서 시간 분석 공격을 이용하여 FALCON 트리 T의 각 리프의 값을 알아낸 후, 이를 이용하여 FALCON 트리 T 전체를 복원하여 공격하는 방법을 제시하였다[14]. 비록, 시간 분석 공격을 이용하여 *SamplerZ*를 공격하는 과정에서 σ'의 근삿값만을 복원 가능하므로 키 복원을 할 수 없었지만, 4.2, 4.3절에서 수행한 공격을 이용하여 *SamplerZ*의 결괏값과 입력 μ의 차이의 정수 부분과 연계하여 사용하는 방법이나 *SamplerZ*의 두 번째 줄인 $ccs \leftarrow \sigma_{\min}/\sigma'$ 과정에서 σ'를 전력 분석을 통해 복원한다면, FALCON 트리 T를 복원 후 개인키 전체를 복원할 수 있을 것이다.

Ⅵ. 결 론

본 논문에서는 NIST 양자 내성 암호 공모전 Round 3에 진출한 전자서명 알고리즘 FALCON의 전력 분석 공격 세 가지를 제시하였으며 후에 공격을 통해 사용자의 비밀정보를 복원하는 방안에 관하여 논하였다. NIST 공모전의 최종 공개키 암호 및 전자서명 알고리즘은 미연방에서 표준으로 사용할 뿐만 아니라 국제적으로도 많이 사용할 것이 예상되므로 부채널 안전성 검증이 동행하여야 할 것이다.

참 고 문 헌

[1] Kerry, Cameron F., and Patrick D. Gallagher. "Digital signature standard (DSS)." FIPS PUB (2013): 186-4.

[2] Shor, Peter W. "Algorithms for quantum computation: discrete logarithms and factoring." Proceedings 35th annual symposium on foundations of computer science. Ieee, 1994.

[3] Fouque, Pierre-Alain, et al. "Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU." 2020

[4] Alagic, Gorjan, et al. "Status report on the second round of the NIST post-quantum cryptography standardization process." US Department of Commerce, NIST (2020).

[5] Cooley, James W., and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series." Mathematics of computation 19.90 (1965): 297-301.

[6] Markstein, Peter. "The new IEEE-754 standard for floating point arithmetic." Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

[7] Gentry, Craig, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions." Proceedings of the fortieth annual ACM symposium on Theory of computing. 2008.

[8] Kocher, Paul C. "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1996.

[9] Kocher, Paul, Joshua Jaffe, and Benjamin Jun. "Differential power analysis." Annual international cryptology conference. Springer, Berlin, Heidelberg, 1999.

[10] Quisquater, Jean-Jacques, and David Samyde. "Electromagnetic analysis (ema): Measures and

counter-measures for smart cards." International Conference on Research in Smart Cards. Springer, Berlin, Heidelberg, 2001.

[11] Biham, Eli, and Adi Shamir. “Differential fault analysis of secret key cryptosystems.” Advances in Cryptology—CRYPTO'97, pp. 513-525, 1997.

[12] Brier, Eric, Christophe Clavier, and Francis Olivier. "Correlation power analysis with a leakage model." International workshop on cryptographic hardware and embedded systems. Springer, Berlin, Heidelberg, 2004.

[13] Batina, Lejla, et al. "CSI neural network: Using side-channels to recover your artificial neural network information." arXiv preprint arXiv:1810.09076 (2018).

[14] Fouque, Pierre-Alain, et al. "Key Recovery from Gram–Schmidt Norm Leakage in Hash-and-Sign Signatures over NTRU Lattices." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2020.

〈저 자 소 개〉

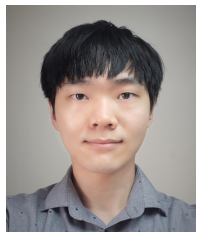
김 규 상 (GyuSang Kim)

학생회원

2020년 2월 : 연세대학교 수학과 학사

2020년 9월~현재 : 고려대학교 정보보호학과 석박사 통합 과정

<관심분야> 공개키 암호, 부채널 공격



박 동 준 (DongJun Park)

학생회원

2018년 8월 : 세종대학교 정보보호학과 학사

2020년 8월 : 고려대학교 정보보호학과 석사

2020년 9월~현재 : 고려대학교 정보보호학과 박사 과정

<관심분야> 부채널 공격

홍 석 희 (SeokHee Hong)

종신회원

1995년 2월 : 고려대학교 수학과 학사

1997년 2월 : 고려대학교 수학과 석사

2001년 8월 : 고려대학교 수학과 박사

2005년 3월~2013년 8월 : 고려대학교 정보보호대학원 부교수

2013년 9월~현재 : 고려대학교 정보보호대학원 정교수

<관심분야> 대칭키 및 공개키 암호 알고리즘, 부채널 공격 및 대응기법, 디지털 포렌식