정 보 보 호 학 회 지
제 33 권 제3호, 2023. 6

LWE-like KEM/전자서명에 대한 부채널 대응기법 동향

이 정 환*, 정 상 윤***, 신 원 근**, 박 수 진*, 김 희 석**

요 약

LWE-like 암호는 지난 몇 년간 표준화 선정 대상 유력 후보로 주목받았다. 실제로 NIST가 선정한 양자내성암호 표준화 진행 대상 4종 중 2종이 LWE 기반 암호인만큼, LWE-like 암호는 많은 중요성을 가지고 있으며 이에 따라 LWE-like 암호에 대한 부채널 분석/대응 기술 연구가 활발히 진행되었다. 특히 전력/전자파를 이용한 분석과 이에 대한 마스킹 대응기법 연구가 많은 결실을 맺었다. 따라서 본 논문은 LWE-like KEM/전자서명에 대한 부채널 대응기술 중 마스킹 기법 동향을 서술한다. LWE-like KEM에 대한 1차 및 고차 마스킹 기법 연구 동향을 정리하고, LWE-like 전자서명에 대한 고차 마스킹 기법 동향을 정리한다.

Ⅰ. 서 론

양자 컴퓨팅 환경에서 인수분해 문제와 이산대수 문제가 다항시간 내로 해결될 수 있음이 Shor 알고리즘을 통해 증명됨에 따라 이산대수 문제 및 인수분해 문제를 기반으로 하는 RSA, ECC와 같은 상용 공개키 암호 시스템 안전성이 큰 위협을 받고 있다. 이에 대비하기 위하여 미국 국립표준기술연구소(NIST)는 2016년 PQCrypto 컨퍼런스에서 양자내성암호 표준화 작업을 발표하였고 여러 라운드를 거쳐 2022년 표준화 작업 대상으로 KEM 1종(CRYSTALS-Kyber), 전자서명 3종(CRYSTALS-Dilithium,Falcon,SPHINCS+)이 선정되었다. NIST는 표준화 작업 시작부터 선정 평가 기준에 부채널 분석 및 오류 주입에 대한 안전성을 강조하였으며 시간이 경과함에 따라 다양한 부채널 분석 시나리오와 대응기법이 등장하였다.

부채널 분석은 1996년 P.Kocher에 의해 소개된 암호 분석 기법으로, 실제 장비 위에서 동작하는 암호화 알고리즘의 연산에 인해 발생하는 전력, 전자파, 시간 등의 부가적인 정보를 이용하여 비밀정보를 추출하는 물리적 분석 기법이다. 부채널 분석에는 단순전력 분석, 상관전력 분석, 프로파일링 공격 등 다양한 기법이 존재한다.

이러한 부채널 공격으로부터 암호 알고리즘의 안전성을 보호하기 위해 부채널 대응기법이 개발되고 있다. 이 중에서 대표적인 기법으로는 하이딩 기법과 마스킹 기법이 있다. 하이딩 기법은 신호대비잡음비(Signal-to-Noise,SNR)를 감소시켜 장비에서 나오는 부채널 정보와 암호화 연산의 중간값 사이에 관계성을 제거한다. 이를 통해 공격자가 부채널 정보를 이용하여 중간값을 찾아내는 것을 어렵게 만든다. 마스킹 기법은 난수를 통해 비밀정보에 민감한 암호 연산 중간값을 분할하여 부채널 정보와 중간값 사이의 연관성을 제거한다. 이를 통해 공격자가 부채널 정보를 이용하여 중간값을 추론하는 것을 방지한다. 한편, 2015년 ring-LWE 공개키 알고리즘 복호화 과정에 대한 1차 마스킹 기법이 최초로 제안[3]된 이후 최근 CRYSTALS-Kyber에 대한 고차 마스킹 기법[7]까지 LWE-like KEM/전자서명에 대한 마스킹 대응기법 연구가 활발하게 진행되고 있다. 따라서 본 논문은 LWE 계열 암호의 마스킹 대응기법에 대한 동향을 조사한다.

논문은 다음과 같이 구성된다. 2장에서 LWE-like KEM/전자서명과 마스킹 기법의 배경지식을 소개한다. 3장에서 LWE-like KEM에 대한 마스킹 대응기법 동향을 소개하고 4장에서 LWE-like 전자서명 마스킹 대응기법 동향을 소개한다. 마지막으로 결론 및 향후 연구 방향을 제시한다.

* 고려대학교 정보보호대학원 정보보호학과 (대학원생, hwani0814@korea.ac.kr, lemontrees33@korea.ac.kr)
** 고려대학교 인공지능사이버보안학과 (학부생, 부교수, persshins99@gmail.com, 80khs@korea.ac.kr)
*** 고려대학교 사이버보안학과(사이버보안) (대학원생, jung2217@korea.ac.kr)

Ⅱ. 배경지식

2.1. LWE-like KEM/전자서명 알고리즘

2.1.1. Saber

Saber는 LWE(Learning With Errors) 문제에서 파생된 MLWR(Module Learning With Rounding) 문제를 기반으로 설계된 KEM이다. 모듈 위에서 정의되며 개인키는 R_q^k로 구성되며 이는 CRYSTALS-Kyber와 CRYSTALS-Dilithium 또한 동일하다. 이때, R_q는 $Z_q[X]/ < X^{256}+1 >$인 다항식환이며 k는 보안강도를 나타낸다. IND-CCA2를 만족하기 위해 IND-CPA 안전성이 보장된 Saber Public Key Encryption(PKE)를 Fujisaki-Okamoto(FO) 변환하여 Saber Key Encapsulation Mechanism(KEM)을 설계하였다.

[표 1]은 Saber의 보안 강도에 따른 파라미터이다. Saber는 LWE 문제를 기반으로 하는 다른 격자 암호와는 다르게 모듈러로 사용되는 q,p와 T가 모두 2의 거듭제곱으로 구성되어있기 때문에 감산 연산 시 시프트 연산 외 비용이 발생하지 않는다.

〔표 1〕 Saber 파라미터

secuirty level	k	q	p	T
Light-Saber	2	2^{13}	2^{10}	2^3
Saber	3	2^{13}	2^{10}	2^4
Fire-Saber	4	2^{13}	2^{10}	2^6

2.1.2. CRYSTALS-Kyber(Kyber)

Kyber는 MLWE(Module Learning With Errors) 문제를 기반으로 하는 KEM이다. 소수 3329를 모듈러 q로 사용하는 다항식환에서 정의된다. IND-CCA2를 만족하기 위해 IND-CPA 안전성이 보장된 Kyber PKE를 FO 변환하여 Kyber KEM를 설계하였다.

[표 2]는 Kyber의 보안 강도 k에 따른 파라미터를 나타낸 것이다. (d_u,d_v)는 암호문의 크기를 줄이기 위해 사용되는 $compress$와 $decompress$ 함수의 파라미터이며 δ는 복호화 실패 확률을 나타낸다.

〔표 2〕 CRYSTALS-Kyber 파라미터

secuirty level	k	(d_u,d_v)	δ
Kyber 512	2	(10,4)	2^{-139}
Kyber 768	3	(10,4)	2^{-164}
Kyber 1024	4	(11,5)	2^{-174}

2.1.3. CRYSTALS-Dilithium(Dilithium)

Dilithium은 MLWE 문제를 기반으로 하는 전자서명 알고리즘이다. 소수 8,380,417를 모듈로 q로 사용하는 다항식환에서 정의되며 SUF-CMA를 만족한다.

[표 3]는 보안 강도에 따른 파라미터를 나타낸 것이다. k와 l은 개인키의 모듈 길이이며 η는 개인키 계수의 최댓값이다. Dilithium은 서명 과정에서 계수가 γ_1보다 작은 다항식환으로 이루어진 마스킹 벡터 y를 생성한다. 이때 γ_1이 너무 크면 서명 위조가 쉬워지고 반대로 작으면 생성된 서명에서 개인키에 대한 정보가 드러나기 때문에 적절한 γ_1값을 설정하는 것이 중요하다.

〔표 3〕 CRYSTALS-Dilithium 파라미터

secuirty level	(k,l)	η	γ_1
1	(3,2)	7	523776
2	(4,3)	6	523776
3	(5,4)	5	523776
4	(6,5)	3	523776

2.2. 마스킹 기법

마스킹 기법은 전력/전자파 분석 공격으로부터 암호를 보호하기 위한 대표적인 부채널 대응기법 중 하나이다. 난수를 통해 암호화 또는 복호화 연산의 중간값을 분할하여 중간값과 소비 전력/전자파 간의 연관성을 제거한다. 이때 t개의 난수를 사용하여 중간값을 $t+1$개로 분할하는 것을 t차 마스킹이라 부르고 각 분할된 부분들을 몫(share)이라 한다. t차 마스킹은 t차 DPA 공격에 안전하다.

마스킹 연산은 암호 연산의 종류에 따라 산술 마스킹과 부울 마스킹으로 구분할 수 있다. 산술 마스킹은

모듈로 q 상의 유한체에서 생성한 난수 t개를 사용하여 중간값 x를 $t+1$개의 $x_i(1 \leq i \leq t+1)$로 분할하며 $x = x_1 + \ldots + x_{t+1} \ mod \ q$을 만족한다. 한편 부울 마스킹은 유한체 F_2 상에서 생성한 난수 t개를 사용하여 중간값 $x \in F_2$를 $t+1$개의 $x_i(1 \leq i \leq t+1)$로 분할하며 xor 연산 $\oplus$에 대해 $x = \bigoplus_{i=1}^{t+1} x_i = x_1 \oplus x_2 \oplus \ldots \oplus x_{t+1}$를 만족한다.

마스킹 기법의 안전성은 Ishai 등이 제안한 t-프로빙 모델을 통해 처음으로 정형화되었다[1]. 이 모델에서 공격자는 암호가 동작하는 회로의 최대 t개 도선에 접근할 수 있다. 이때 도선을 프로브라고 한다. 앞서 설명한 바와 같이 비밀정보가 속해있는 중간값 s는 t개 난수를 통해 $t+1$개의 몫 $s_i(1 \leq i \leq t+1)$로 마스킹되며 모든 $t+1$개 몫 s_i을 조합해야한 s를 복원할 수 있다. 회로 연산 중 어떠한 t개의 중간값 집합을 생성하더라도 $s_i(1 \leq i \leq t+1)$ 중 적어도 하나와 독립이 되도록 보장하기 위해 s_i는 각각 따로 연산되며 비선형 연산을 수행해야하는 경우 추가적인 난수를 사용해야한다. 이러한 비선형 연산을 위한 마스킹 게이트를 가젯(gadget)이라고 부르며 t-프로빙 모델에서 가젯들이 조합될 때 안전성을 증명하려면 아래의 두 정의 중 하나를 만족해야 한다.

정의 1. t-NI

가젯 G 입력의 최대 t개 몫을 사용하여 모든 t개의 프로브 집합을 시뮬레이션할 수 있으면 가젯 G가 t-Non-Interfering(t-NI)를 만족한다고 정의한다.

정의 2. t-SNI

$t_1 + t_2 \leq t$를 만족하는 t_1, t_2에 대해 가젯 G의 입력 중 최대 t_1개 몫을 사용하여 t_1개의 중간값 프로브와 t_2개의 출력값 프로브의 모든 집합을 시뮬레이션할 수 있으면 가젯 G가 t-Strong-Non-Interfering(t-SNI)를 만족한다고 정의한다.

Ⅲ. LWE-like KEM에 대한 마스킹 대응기법 동향

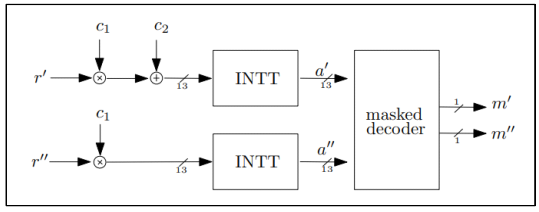
LWE-like 암호의 경우 다수가 다항식환 R_q의 선형연산 곧 덧셈과 곱셈을 사용하기 때문에 R_q 상에서 산술 마스킹 적용이 가능하다. 반면 그 외 암호에서 사용되는 샘플러나 메시지 디코딩/인코딩 등의 경우 대부분 비트연산으로 표현된다. 따라서 암호 알고리즘 전체에 마스킹 기법을 적용할 경우 산술-부울 마스킹 변환(A2B)과 부울-산술 마스킹 변환(B2A)이 필요하며 각 함수에 대한 새로운 마스킹 알고리즘이 필요한 경우도 존재한다.

LWE-like KEM의 디캡슐화 과정은 개인키가 연산에 사용되기 때문에 마스킹 대응기법이 적용되어야한다. 초기 디캡슐화 과정의 마스킹 연구는 1차 마스킹 알고리즘 개발을 위주로 진행되었다. 1차 마스킹은 알고리즘 별로 최적화가 용이하나 범용적이지 않고 고차 마스킹으로 확장이 어렵다는 단점이 존재한다. 이후 t-프로빙 모델을 이용한 고차 마스킹 연구가 현재까지 진행되었으며 고차 마스킹 연구가 발전함에 따라 시간 효율적이고, 범용적인 알고리즘이 계속해 등장하고 있다.

3.1. LWE-like KEM에 대한 1차 마스킹 기법 동향

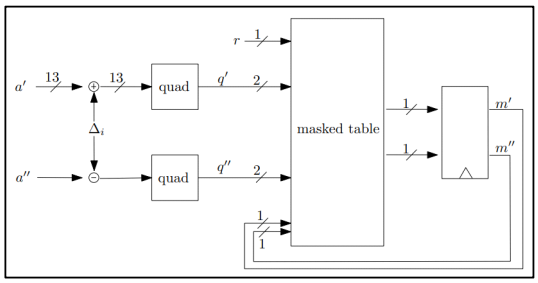
2015년 ring-LWE 공개키 알고리즘 복호화 과정에 대한 1차 마스킹 기법이 소개되었다[3]. 먼저 R_q에서 난수를 생성하여 개인키 $r \in R_q$를 r'과 r''으로 나누어 독립적으로 연산을 진행한다. 이때 암호문 c_2의 경우 [그림 1]와 같이 r'의 프로브에만 더해지는데 이는 $c_2 + c_1 r = c_2 + c_1(r' + r'') = (c_2 + c_1 r') + c_1 r''$를 만족하기 때문이다. 각 프로브는 INTT를 통해 a', a''이 되고 이를 계수 별로 마스킹 디코더에 입력하여 부울 마스킹된 메시지 비트를 출력한다.

마스킹 되지 않은 디코더의 경우 a의 계수 a_i $(0 \leq i \leq 255)$를 입력으로 하여 $0 \leq a_i < \frac{q}{2}$ 또

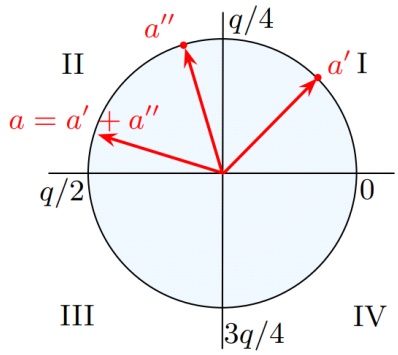


〔그림 1〕 ring-LWE 공개키 암호 복호화 과정 마스킹

는 $\frac{3}{4}q \leqq a_i < 0$일 때 0을 메시지 비트로 출력하고 $\frac{q}{2} \leqq a < \frac{3}{4}q$일 때 1을 메시지 비트로 출력한다. 마스킹 디코더의 경우 산술 마스킹된 입력으로부터 같은 역할의 동작을 수행해야 하며 결과값은 부울 마스킹된 상태로 출력되어야한다. 이를 수행하기 위해 논문에서는 마스킹 디코더를 [그림 2]와 같이 설계하였다. $quad$는 모듈로 q 상의 입력값 $\hat{a_i}$가 가질 수 있는 범위를 [그림 3]과 같이 4분할 하여 $\hat{a_i}$가 속한 구역을 출력하는 함수이다. Ⅰ, Ⅱ, Ⅲ, Ⅳ는 각각 0,1,2,3을 값으로 출력한다. 따라서 $quad$의 출력 크기는 2bit이다. 마스킹 디코더는 먼저 입력값 a'_i,a''_i을 각각 $quad$에 입력하여 각 값이 속해있는 모듈러 q 상의 구역을 출력으로 받는다. 이때 출력된 구역의 범위만으로 일부 메시지 디코딩이 가능해지는데 예를 들어 a'_i가 Ⅰ에 속해있고 a''_i가 Ⅱ에 속해있다면 a_i가 가질 수 있는 범위는 Ⅱ,Ⅲ이기 때문에 메시지가 1로 복호화됨을 알 수 있다. 반면에 a'_i가 Ⅰ에 속해있고 a''_i가 Ⅰ에 속



〔그림 2〕 마스킹 디코더



〔그림 3〕 quad의 입력 범위에 따른 출력값

해있다면 a_i가 가질 수 있는 범위는 Ⅰ, Ⅱ이기 때문에 구역 정보만으로 메시지 디코딩을 할 수 없는 경우 또한 존재한다. 마스킹 테이블은 이러한 케이스를 테이블화하여 메시지 디코딩이 가능할 경우 부울 마스킹된 메시지로 디코딩하고 메시지 디코딩이 가능하지 않을 경우 고정된 상수 Δ_i에 대해 $a' + \Delta_i$, $a'' - \Delta_i$를 입력으로 마스킹 디코더를 다시 실행시킨다. 이 과정을 다항식환의 모든 계수에 대해 메시지 디코딩이 완료될 때까지 반복한다.

ring-LWE 공개키 알고리즘 복호화 과정의 마스킹 구현은 Xilinx Virtex-Ⅱxc2vp7 FPGA에서 2.7배 정도의 오버헤드가 발생하였다.

2018년 IND-CCA2를 만족하는 ring-LWE 기반 암호와 해당 암호의 복호화 과정에 대한 1차 마스킹 기법이 소개되었다[4]. 논문에서 제시한 IND-CCA2 암호의 복호화 과정은 KEM과 마찬가지로 PKE 복호화, PKE 재암호화, 비교연산으로 구성되어 있다. 또한 [3]과 같은 방식으로 난수를 사용해 개인키를 2개로 분할하여 복호화 과정을 동작시키는 산술 마스킹을 적용하였으며 산술 마스킹이 불가능한 PKE 복호화 과정의 메시지 디코딩과 PKE 재암호화 과정의 인코딩, 이항 샘플러, 비교 연산에 대해 1차 마스킹 알고리즘을 제시하였다.

논문에서는 [1]에서 제시한 마스킹 디코더보다 효율적인 마스킹 디코더 MDecode를 제안한다. MDecode의 알고리즘은 다음과 같다. 먼저 산술 마스킹된 입력값에 $-\frac{q}{4}$를 더해준 뒤 모듈로 q에서 모듈로 $2^{\lceil \log_2 q \rceil}$로 스위칭한다. 이때 스위칭은 추상적으로 중간값이 가질 수 있는 범위가 바뀌는 것일 뿐 실제적인 연산은 취하지 않는다. 이후 $-\frac{q}{2}$를 더해주고 $A2B$를 적용하면 MSB의 값에 따라 메시지 비트 결정할 수 있다. 이때 $-\frac{q}{4}$, $-\frac{q}{2}$와 같은 상수를 더해주거나 스위칭을 하는 이유는 모듈로 2의 거듭제곱에서의 산술 마스킹에서 부울 마스킹으로 변환하는$A2B$ 알고리즘을 모듈로 q가 소수인 환경에서 사용해야하기 때문이다.

PKE 암호문은 c_1과 c_2로 구성되며 $c_1 = ae_1 + e_2$, $c_2 = pe_1 + e_3 + Encode(m)$ 를

만족한다. c_1은 R_q에서 선형연산으로 구성되어 있으므로 $c'_1 = ae'_1 + e'_2$과 $c''_1 = ae''_1 + e''_2$으로 산술 마스킹 될 수 있다. 반면, c_2의 경우 메시지 인코딩이 선형연산이 아니므로 마스킹을 위한 추가적인 처리가 필요하다. $m = m' \oplus m''$를 만족하는 부울 마스킹된 메시지 m', m''와 홀수 모듈러 q에 대해 아래의 식이 만족한다.

$$Encode(m') + Encode(m'') \bmod q = \begin{cases} 0 & if\ m' = 0 \text{ and } m'' = 0 \\ \lfloor \frac{q}{2} \rfloor & if\ m' = 1 \text{ and } m'' = 0 \\ \lfloor \frac{q}{2} \rfloor & if\ m' = 0 \text{ and } m'' = 1 \\ 2\lfloor \frac{q}{2} \rfloor \neq q & if\ m' = 1 \text{ and } m'' = 1 \end{cases}$$

이때 $m' = 1$, $m'' = 1$인 경우 모듈러 q상에서 0이 되지 않으므로 이 연산에 대해 산술 마스킹을 곧바로 적용할 수 없다. 논문은 아래의 수식을 이용하여 추가적인 난수를 통해 부울 마스킹된 메시지에 대한 마스킹된 메시지 인코딩을 설계하였다.

$$Encode(m') + Encode(m'') + m' \,\&\, m'' \bmod q = Encode(m)$$

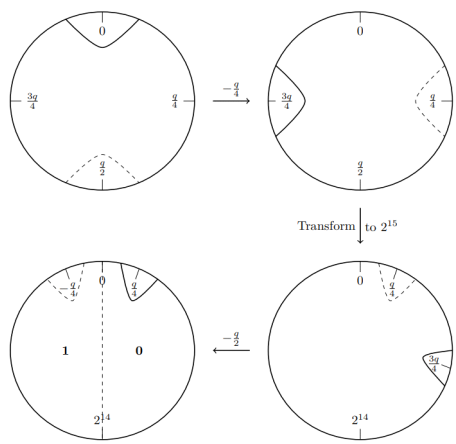
또한 논문은 PKE 암호화 과정 중 e_1, e_2를 생성하기 위한 이항 샘플러에 대한 1차 마스킹 기법을 제안하였다. 이항 샘플러는 두 k 비트 크기의 입력값 α와 β 각각의 해밍무게를 계산하고 그 차를 출력하여 결과값이 이항 분포를 따르도록 만들어진 샘플러이다. $\alpha[i], \beta[i] (0 \leq i < k)$를 각각 α, β의 원소라 하고 $\alpha_1[i], \alpha_2[i]$가 $\alpha[i]$의 부울 마스킹값이라 했을 때 아래의 수식을 만족하므로 이를 이용하여 마스킹 이항 샘플러를 설계할 수 있다. 이때 마스킹된 값이 한 연산에서 두 번 사용되는 것을 방지하기 위해 추가적인 난수 생성이 필요하다.

$$\begin{aligned} out &= \sum_{i=0}^{k-1} (\alpha_1[i] + \alpha_2[i] - 2\alpha_1[i]\alpha_2[i]) \\ &\quad - \sum_{i=0}^{k-1} (\beta_1[i] + \beta_2[i] - 2\beta_1[i]\beta_2[i]) \\ &= \sum_{i=0}^{k-1} (\alpha_1[i] - \beta_1[i]) + \sum_{i=0}^{k-1} (\alpha_2[i] - \beta_2[i]) \\ &\quad - 2\sum_{i=0}^{k-1} (2\alpha_1[i]\alpha_2[i]) + 2\sum_{i=0}^{k-1} (2\beta_1[i]\beta_2[i]) \end{aligned}$$

비교 연산의 부채널 정보가 유출될 경우 공격자는 해당 정보를 이용하여 선택 암호문 공격을 실행할 수 있다. 따라서 비교 연산 또한 마스킹되어야한다. 논문은 복호화 과정의 입력 암호문 c_1과 재암호화로부터 산술 마스킹된 상태로 출력된 암호문 $c^{*\prime}_1$, $c^{*\prime\prime}_1$과의 마스킹 비교 연산을 해시함수의 충돌저항성을 이용하여 설계하였다. 곧 해시함수 H에 대해 아래의 식이 만족함을 이용하여 비교연산에 대한 1차 마스킹을 수행하였다. 그러나 논문에서 c_1와 c_2에 대한 비교를 한번에 수행하지 않았고 이로 인해 [7]에 의해 선택 암호문 환경에서 부채널 공격이 가능함이 밝혀졌다.

$$compare(H(c_1 - c^{*\prime}_1), H(c^{*\prime\prime}_1)) \Leftrightarrow compare(H(c_1), H(c^{*\prime}_1 + c^{*\prime\prime}_1))$$

제안된 IND-CCA2 ring-LWE 공개키 알고리즘 복호화 과정의 마스킹 구현은 ARM Cortex-M4F에서 약 5.7배의 오버헤드가 발생하였다.



〔그림 4〕 계수 하나에 대한 메시지 디코딩

2020년 Saber의 디캡슐화 과정에 대한 1차 마스킹 기법이 소개되었다[2]. [3]과 마찬가지로 개인키를 산술 마스킹하여 디캡슐화 과정에 적용하였으며 산술 마스킹 되지 않은 메시지 인코딩/디코딩, 비교 연산에 대한 마스킹 알고리즘을 제시하였다. 이항 샘플러의 경우 [5]의 알고리즘을 Saber 파라미터에 최적화하여 비트슬라이스로 구현하였다.

Saber의 메시지 인코딩/디코딩은 시프트 연산으로 진행된다. 이때 시프트 연산은 비트연산이기 때문에 부울 마스킹 상태에서 각 마스킹 값에 대해 독립적인 연산이 가능하다. 그러나 산술 마스킹의 경우 각 마스킹 값에 대해 시프트 연산 실행 시 캐리가 고려되지 않아 원래 값으로 복원되지 않는다. 이를 해결하기 위한 가장 간단한 방법은 A2B 알고리즘을 통해 산술 마스킹을 부울 마스킹으로 변환한 뒤 시프트 연산을 수행하고 이를 B2A 알고리즘을 통해 다시 산술 마스킹 변환하는 것이다. 그러나 이 방법은 디캡슐화 과정에서 큰 오버헤드를 발생시킨다. [2]에서는 시간 복잡도를 감소시키기 위하여 테이블 기반 A2A 알고리즘을 소개한다. 테이블 기반 A2A 알고리즘은 $x = A + R \bmod 2^{m+nk}$로 산술 마스킹된 A와 R을 입력으로 $x \gg (nk) = A + R \bmod 2^{m}$를 만족하는 A와 R을 출력하는 함수이다. 먼저 k비트의 마스킹 캐리 테이블을 사전 계산한 뒤 A에 적절한 난수를 더해준다. 그 후 A와 R의 하위 k비트부터 순차적으로 더해가며 마스킹 캐리 테이블을 참조해 캐리를 계산한다. 이를 n번 반복하면 $x \gg (nk) = A + R \bmod 2^{m}$를 만족하는 A, R을 부채널 민감 정보의 유출 없이 계산할 수 있다.

Saber의 비교 연산을 위해 [4]의 방법이 변형되어 사용되었다. 해시함수의 충돌저항성을 이용하여 마스킹 비교 연산을 설계하였으나 해시함수 H에 대해 암호문 c_1, c_2를 연접하여 아래와 같이 진행함으로써 [4]와 달리 부채널 민감 정보의 유출 없이 비교 연산을 수행하였다.

$compare(H((c_1 - c^{*\prime}_1) \| (c_2 - c^{*\prime}_2)), H(c^{*\prime\prime}_2 \| c^{*\prime\prime}_2))$

제안된 Saber 디캡슐화 과정 마스킹 구현은 ARM Cortex-M4F에서 약 2.5배 정도의 오버헤드가 발생하였다.

3.2. LWE-like KEM에 대한 고차 마스킹 기법 동향

〔표 4〕 LWE-like KEM에 대한 고차 마스킹 기법 동향

마스킹 대상	
$Compress_q$	9, 10
이항 샘플러	5
비교 연산	6, 7, 8, 9, 10

3.2.1. $Compress_q$에 대한 고차 마스킹 기법 동향

$Compress_q$는 Kyber에서 사용되는 함수이며 아래와 같이 정의된다.

$$Compress_q(x,d) = \lfloor \frac{2^d x}{q} \rceil \bmod 2^d$$

$Compress_q$는 두가지의 역할을 가지고 있다. 첫째는, 복호화 과정에서 메시지 디코딩의 역할을 하며 이때 $d=1$로 하여 $Compress_q(x,1)$이 사용된다. 둘째는, 암호화 과정에서 복호화 실패 확률에 영향을 미치지 않게 암호문의 하위 비트를 제거하여 암호문의 사이즈를 줄인다.

2021년 일반적인 모듈러 q에 대한 고차 마스킹 $Compress_q(x,1)$ 알고리즘이 제안되었다[9]. $Compress_q(x,1)$의 입력값 x에 $\lfloor \frac{q}{4} \rceil$를 더하면 메시지 디코딩의 범위가 $x < \frac{q}{2}$일 때 0, 나머지는 1로 바뀌게 된다. 논문에서는 해당 범위에 맞춰 아래와 같이 새로운 $Compress^s_q$ 함수를 정의한다.

$$Compress^s_q(x) = \begin{cases} 0 & \text{if } x < \frac{q}{2} \\ 1 & otherwise. \end{cases}$$

이때 $Compress_q(x,1)$과 $Compress^s_q$는 아래와 같은 식을 만족한다.

$$Compress_q(x,1) = Compress^s_q(x + \lfloor \frac{q}{4} \rceil \bmod q)$$

또한 $Compress_q^s$는 q에 따라 입력값 x의 비트들의 비트연산 조합으로 표현이 가능하다. 예를 들어 Kyber와 같이 $q=3329$인 경우 아래와 같이 표현된다.

$$Compress_{3329}^s(x) = x_{11} \oplus (\neg x_{11} \bullet x_{10} \bullet x_9 \bullet (x_8 \oplus (\neg x_8 \bullet x_7)))$$

이러한 특징을 활용하여 설계한 고차 마스킹 $Compress_q(x,1)$ 알고리즘은 다음과 같다. 산술 마스킹된 입력 x에 $\lfloor \frac{q}{4} \rfloor$를 더하고 $A2B$를 통해 부울 마스킹으로 변환한다. $Compress_q^s$는 입력의 비트들의 비트연산으로 표현되므로 부울 마스킹된 몫들에 대해 각각 연산을 수행하면 결과로 부울 마스킹된 메시지가 출력된다. 논문은 t-프로빙 모델을 통해 t-SNI를 만족한다는 것을 증명하였다.

2023년 일반적인 q,d에 대한 고차 마스킹 $Compress_q(x,d)$가 제안되었다[10]. 모듈러 q에서 모듈러 2^d로 스위칭 해주었던 $Compress_q(x,d)$ 함수와 달리 제안된 고차 마스킹 알고리즘은 양의 정수 α에 대해 모듈러 $2^{d+\alpha}$로 스위칭 함으로써 더 큰 정밀도를 사용하여 모듈러 스위칭 시 발생되는 오류를 시프트 연산으로 완전히 없앨 수 있게 만들어준다. 알고리즘은 다음과 같다. 산술 마스킹된 x의 몫 $x_i(1 \leq i \leq n)$을 입력으로 아래와 같이 중간값 $z_i(1 \leq i \leq n)$을 계산한다.

$$z_1 = \lfloor x_1 \frac{2^{d+\alpha}}{q} \rceil + 2^{\alpha-1} \bmod 2$$
$$= \lfloor (x_1 2^{d+\alpha+1} + q)/(2q) \rfloor \bmod 2^{d+\alpha}$$
$$z_i = \lfloor x_i \frac{2^{d+\alpha}}{q} \rceil + 2^{\alpha-1} \bmod 2^{d+\alpha}$$
$$= \lfloor (x_i 2^{d+\alpha+1} + q)/(2q) \rfloor \bmod 2^{d+\alpha} (2 \leq i \leq n)$$

이후 $z_i(1 \leq i \leq n)$를 $A2B$를 통해 부울 마스킹 $(c_1,c_2,...,c_n)$로 변환한다. 마지막으로 각 c_i를 오른쪽으로 α만큼 시프트 연산해주면 부울 마스킹된 $Compress_q(x)$이 결과로 출력된다. 논문에서는 $2^\alpha > qn$를 만족하는 모든 양의 정수 α에 대해 알고리즘이 성립함을 보였으며 t-프로빙 모델을 통해 t-NI를 만족한다는 것을 증명하였다.

3.2.2. 이항 샘플러에 대한 고차 마스킹 기법 동향

2019년 고차 마스킹 이항 샘플러가 소개되었다[5]. 논문에서 제안한 모든 함수가 t-SNI를 만족하도록 하여 상위 알고리즘의 가젯으로 활용할 수 있도록 개발하였으며 이를 통해 논문은 $B2A_{q-Bit}$ 등과 같은 작은 가젯부터 시작해서 고차 이항 샘플러까지 t-프로빙 모델에서 안전성을 증명하였다.

논문은 먼저 부울 마스킹에서 일반적인 모듈러 q에 대한 산술 마스킹으로의 변환 $B2A_q$을 개발하고 이를 사용해서 2가지의 고차 마스킹 이항 샘플러를 제안한다. 첫 번째 고차 마스킹 이항 샘플러는 [4]의 1차 마스킹 이항 샘플러를 확장한 것이다. 알고리즘은 그림 5의 $SecSampler_1$와 같다. 부울 마스킹된 k비트의 x와 y를 입력으로 받고 아래의 수식이 성립함을 이용하여 하위 비트부터 순차적으로 모든 k비트에 $B2A_q$를 적용하여 x의 비트는 더하고 y의 비트는 빼면 최종적으로 $A = \sum_i A_i = HW(x) - HW(y) \bmod q$를 출력한다.

$$A = \sum_{j=0}^{k-1} ((B2A_q(x \gg j) \wedge 1) - (B2A_q(y \gg j) \wedge 1))$$
$$\sum_i A_i = \sum_i \sum_{j=0}^{k-1} ((B2A_q(x_i \gg j) \wedge 1) - (B2A_q(y_i \gg j) \wedge 1))$$
$$= HW(x) - HW(y) \bmod q$$

두 번째 고차 마스킹 이항 샘플러는 비트슬라이스로 구현 이항 샘플러이다. 알고리즘은 그림 5의 $SecSampler_2$와 같다. 첫 번째 방법과 달리 $B2A_q$를 한번만 호출하여 높은 효율성을 가진다. 그러나 비트슬라이스 상태에서 $B2A_q$ 변환을 적용할 때 음수가 잘못 변환되는 경우가 있어 이를 보정하기 위해 변환 전 상수 k를 더해야 한다. 논문은 이러한 보정을 위해 부울 마스킹 상태에서 상수를 더해주는 알고리즘

Algorithm 10 $\texttt{SecSampler}_1$

Input: $\mathbf{x}=(x_i)_{1\le i\le n}\in\mathbb{F}_{2^\kappa}$, $\mathbf{y}=(y_i)_{1\le i\le n}\in\mathbb{F}_{2^\kappa}$ such that $\bigoplus_i x_i=x$, $\bigoplus_i y_i=y$
Output: $\mathbf{A}=(A_i)_{1\le i\le n}\in\mathbb{F}_q$ such that $\sum_i A_i=\texttt{HW}(x)-\texttt{HW}(y) \mod q$

$(A_i)_{1\le i\le n}\leftarrow 0$
for $j=0$ to $\kappa-1$ **do**
  $\mathbf{B}\leftarrow \texttt{B2A}_q((\mathbf{x}>>j)\wedge 1)$
  $\mathbf{C}\leftarrow \texttt{B2A}_q((\mathbf{y}>>j)\wedge 1)$
  $\mathbf{A}\leftarrow \mathbf{A}+\mathbf{B} \mod q$
  $\mathbf{A}\leftarrow \mathbf{A}-\mathbf{C} \mod q$
end for

Algorithm 15 $\texttt{SecSampler}_2$

Input: $\mathbf{x}=(x_i)_{1\le i\le n}\in\mathbb{F}_{2^\kappa}$, $\mathbf{y}=(y_i)_{1\le i\le n}\in\mathbb{F}_{2^\kappa}$, κ, such that $\bigoplus_i x_i=x$, $\bigoplus_i y_i=y$
Output: $\mathbf{A}=(A_i)_{1\le i\le n}\in\mathbb{F}_q$ such that $\sum_i A_i=\texttt{HW}(x)-\texttt{HW}(y) \mod q$

$\mathbf{z}\leftarrow \texttt{SecBitAdd}(\mathbf{x})$
$\mathbf{z}\leftarrow \texttt{SecBitSub}(\mathbf{z},\mathbf{y})$
$\mathbf{z}\leftarrow \texttt{SecConstAdd}(\mathbf{z},\kappa)$
$\mathbf{A}\leftarrow \texttt{B2A}_q(\mathbf{z})$
$A_1\leftarrow A_1-\kappa \mod q$

〔그림 5〕 [5]에서 제안하는 고차 마스킹 이항 샘플러

SecConstAdd 함수를 제안하였다. 이후 $B2A_q$ 변환한 뒤 더해주었던 상수 k를 다시 빼주면 결과적으로 첫 번째 고차 마스킹 이항 샘플러와 같은 결과를 출력하게 된다.

[그림 6]는 제안된 이항 샘플러 마스킹 구현에 대한 ARM-Cortex-M4에서의 cycle counts를 나타낸 것이다. (A)의 경우 $SecB2A_q$ 알고리즘을 (C)의 경우 *SecBoolArithModp*(quadratic)을 사용하였다. n은 마스킹 몫의 개수를 나타낸다. $SecB2A_2$를 사용한 $SecSampler_2$의 경우 고차 마스킹 구현임에도 불구하고 $n=2$일 때 [4]의 구현물보다 성능이 좋은 것을 확인할 수 있다.

	n	2	3	4	5
	[25]	3,637	-	-	-
$\texttt{SecSampler}_1$	(C)	271,423	638,315	1,076,155	1,758,184
	(A)	6,145	13,913	24,397	37,880
$\texttt{SecSampler}_2$	(C)	17,564	40,977	68,914	112,402
	(A)	2,649	5,573	9,462	14,587

〔그림 6〕 마스킹 이항 샘플러에 대한 cycle counts

3.2.3. 비교 연산에 대한 고차 마스킹 기법 동향

2020과 2021년 난수합 방식을 사용한 고차 마스킹 비교 연산 알고리즘이 소개되었다[6][7]. $n-1$차 다항식과 $w|n$을 만족하는 양의 정수 w와 $h=\frac{n}{w}$에 대해 인덱스 집합을 $I_h=\{\mathrm{h}\frac{\mathrm{n}}{\mathrm{w}},\ldots,(\mathrm{h}+1)\frac{\mathrm{n}}{\mathrm{w}}-1\}$ $(0\le h<w)$라 정의하고 디캡슐화 과정 중 t차 산술 마스킹되어있는 재암호화의 출력 암호문을 $A^{(j)}(0\le j\le t)$라 할 때 난수 $r_{1,i}$와 $r_{2,j}$를 생성하여 아래 식과 같이 B_h를 구성한다.

$$B_h=\sum_{j=0}^{t}(B_h^{(j)}=\sum_{i\in I_h}(A_i^j+r_{1,i})r_{2,i})$$
$$=\sum_{i\in I_h}((\sum_{j=0}^{t}A_i^j+(t+1)r_{1,i})r_{2,i})$$
$$=\sum_{i\in I_h}((A_i+(t+1)r_{1,i})r_{2,i})$$

이는 마스킹되어 있지 않은 디캡슐화 과정의 입력 암호문을 $\hat{A}$에 대해서에 대해서도 아래와 같이 계산할 수 있다.

$$\widehat{B_h}=\sum_{i\in I_h}(\widehat{A_i}+(t+1)r_{1,i})r_{2,i})$$

이 과정은 난수 $r_{1,i}$와 $r_{2,i}$를 통하여 비밀정보의 유출 없이 진행된다. 마지막으로 모든 h에 대해 B_h와 $\widehat{B_h}$을 계산하여 같으면 비교 연산이 맞다고 판정하고 다르면 비교 연산이 틀리다고 판정한다. 이 알고리즘은 계수 각각을 비교하는 것이 아니라 단위를 정해 난수와 함께 하나의 값으로 합쳐 비교하는 과정이기 때문에 거짓 양성이 존재할 수 있다. 다시 말해 다른 두 다항식에 대해 같다고 판정내릴 수 있는 확률이 존재한다. 논문은 해당 확률이 $\frac{1}{q^w}$에 근사한다고 주장하였으나 모든 h에 대해 B_h와 $\widehat{B_h}$의 비교 연산을 각각 진행하기 때문에 [7]에 의해 선택 암호문 환경에서 부채널 공격 및 충돌쌍 공격이 존재함이 밝혀졌다.

[7]은 [6]의 취약점을 보완하여 새로운 난수합 비교 연산 알고리즘을 제안하였다. 기존 인덱스 집합 I_h에 대해 B_h와 $\widehat{B_h}$를 계산했던 것과 달리 새로운 알고리즘은 난수 R을 사용하여 다항식 $A^{(j)}(0\le j\le t)$와 $\hat{A}$의 모든 항을 각각 B_{i_B},$\widehat{B}_{i(B)}$ 에 합치고 둘을 비교하여 같으면 비교 연산이 맞다고 판정하고 다르면 비교 연산이 틀리다고 판정한다. 이때 모든 항을 하나의 변수에 담아 비교하는 것이기 때문에 거짓 양성의 확률이 커지므로 해당 알고리즘을 충분히 반복해주어야

한다. 논문에서는 알고리즘의 반복 횟수를 l_B라 할 때 거짓 양성의 확률이 $\frac{1}{q^{l_B}}$임을 보였으며 t-프로빙 모델을 통해 t-NI를 만족한다는 것을 증명하였다.

이후 비교 연산에 대한 고차 마스킹 기법 연구는 재암호화의 압축 연산(예: Kyber의 $Compress_q(x,d)$)과 비교 연산을 하나의 마스킹 알고리즘으로 개발하려는 방향으로 진행되었다.

2021년 Kyber에 대한 고차 마스킹 비교 연산 알고리즘이 소개되었다[9]. 디캡슐화의 입력 암호문 c와 압축 함수를 제거한 재암호화의 출력값 (u',v')에 대해 아래 관계가 성립한다.

$$compare((Compress_q(u',d_u),Compress_q(v',d_v)),c) \Leftrightarrow compare((u',v'),Decompress_q(c))$$

알고리즘은 c와 산술 마스킹된 (u',v')를 입력으로 디캡슐화 과정 비교 연산을 수행한다. (u',v')를 압축하는 대신 디캡슐화의 입력 암호문 c를 $Decompress_q$ 함수에 입력하여 범위를 구하고 그 범위 안에 (u',v')가 존재한다면 비교 연산이 맞다고 판정하고 존재하지 않는다면 비교 연산이 틀리다고 판정한다.

이때 입력값에 b에 따른 $Decompress_q(b)$의 범위 $[S(b),E(b)-1]$는 사전 계산하여 테이블 형태로 사용한다. 논문에서는 t-프로빙 모델을 통해 해당 알고리즘이 t-SNI를 만족한다는 것을 증명하였다. 한편 [9]가 제안한 고차 마스킹 $Compress_q(x,1)$과 비교 연산을 모두 적용한 Kyber 디캡슐화 과정 마스킹 구현은 ARM Cortex-M4F에서 1차 마스킹 약 2.5배, 2차 마스킹 약 50배, 3차 마스킹 약 130배 정도의 오버헤드가 발생하였다.

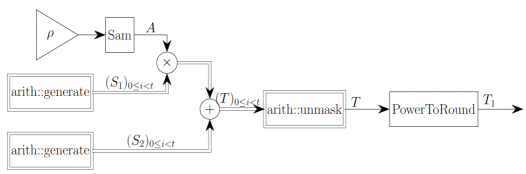
2022년 개선된 난수합 고차 마스킹 비교 연산 알고리즘이 제안되었다[8]. 알고리즘은 두 개의 입력을 받는데, 첫 번째 입력은 디캡슐화된 암호문이고, 두 번째 입력은 압축 함수가 제거된 재암호화의 출력값이다. 이때, 재암호화의 출력값은 산술 마스킹 형태로 표현된다. 알고리즘은 크게 3 단계로 구분 가능하다. 첫 번째 단계에서는 입력값을 모듈러 q에서 충분히 큰 모듈러 $p2^{s-1}$으로 변환한다. 여기서 s는 거짓 양성 확률을 결정하는 정밀도이며, 해당 확률은 2^{-s}이다. 두 번째 단계에서는 하나의 항으로 난수합을 수행한다. 세 번째 단계에서는 난수합된 항의 값을 조사하여 0이면 비교 연산이 맞다고 판정하고 0이 아니라면 비교 연산이 틀리다고 판정한다. 논문에서는 t-프로빙 모델을 통해 해당 알고리즘이 t-SNI를 만족한다는 것을 증명하였다.

2023년 [8],[9]의 고차 마스킹 비교 연산 알고리즘의 장점을 합친 하이브리드 알고리즘이 소개되었다[10]. Kyber 암호화 시 암호문 (u,v)의 첫 번째 부분 u은 비교적 작은 압축을 거치는 방면 두 번째 방법 v는 큰 압축을 거치기 때문에 u에는 [9]의 방법을 v에는 [8]의 방법을 적용하여 알고리즘을 수행하였다. 해당 알고리즘을 적용한 Kyber 디캡슐화 과정 마스킹 구현은 ARM Cortex-M3에서 1차 마스킹 약 6.8배, 2차 마스킹 약 12.5배, 3차 마스킹 약 21.86배 정도의 오버헤드가 발생하였다.

Ⅳ. LWE-like 전자서명에 대한 마스킹 대응기법 동향

2019년 Dilithium에 대한 고차 마스킹 기법이 제안되었다[12]. Dilithium의 경우 개인키 정보가 포함되어 있는 키 생성 과정과 서명 과정에 대한 마스킹이 이루어져야 한다.

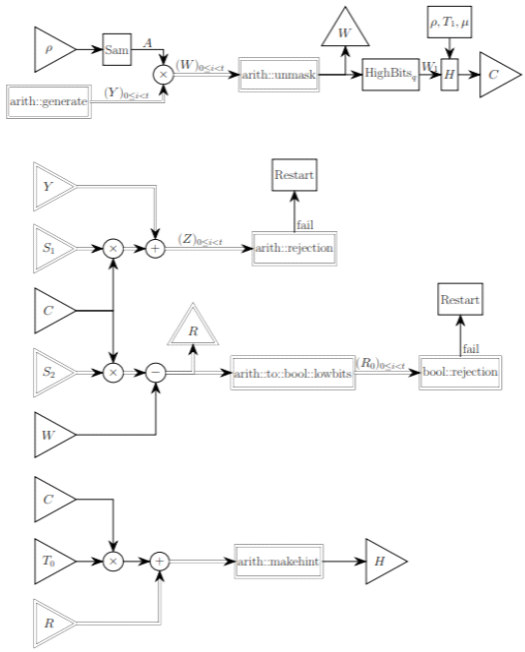
[그림 7]은 Dilithium의 키 생성 과정을 나타낸 것으로 크게 A,S_1,S_2 생성 과정, $T=AS_1+S_2$ 연산 과정, $PowerToRound$함수를 사용하여 T의 상위 비트를 계산하는 과정 세 단계로 구분할 수 있다. 이때 개인키 S_1,S_2가 생성될 때부터 공개키 T가 계산될 때까지 마스킹이 되어야 한다. 논문에서는 개인키 S_1,S_2 산술 마스킹하였으며 마스킹 개인키 생성 알고리즘 $arith::generate$를 제안하였다. $arith::generate$는 $[-\beta,\beta]$ 구간의 정수를 샘플링하는 알고리즘이다. 부울 마스킹된 x를 생성한 후 $2\beta-1$을 뺀 뒤 MSB를 확인하여 양수일 경우 다시 생성한다. 이를 통해 $[0,2\beta]$에서 x가 생성되었음을 확인할 수 있다. 이후 x



〔그림 7〕 Dilithium 키 생성 과정 마스킹

가 $[0,\beta]$인지 $(\beta,2\beta]$인지 확인하고 $(\beta,2\beta]$라면 2β을 빼 $(-\beta,0]$이 되도록 만든다. 마지막으로 B2A를 통해 부울 마스킹되어있는 x를 산술 마스킹으로 바꿔준 후 출력한다.

[그림 8]는 Dilithium의 서명 과정을 나타낸 것이다. 여기서 개인키 S_1,S_2와 개인키를 드러낼 수 있는 Y,Z는 마스킹 되어야 한다. 이때 Z는 공개값 Z가 아닌 서명 과정시 계산되는 Z이며 아직 rejection 여부를 판단하지 않았기 때문에 rejection 시 개인키에 대한 비밀정보를 유출할 가능성이 존재한다. 논문에서는 마스킹을 위한 $arith::to::bool::lowbits$, $arith::rejection$, $bool::rejection$, $arith::makehint$를 제안하였다. 각 함수는 $A2B$와 $B2A$을 통하여 마스킹을 유지한 채 산술 연산과 비트 연산을 수행한다. 논문은 마스킹 구현물에 대한 실제 파형을 통해 실험적으로 안전성을 평가하였다. [그림 9] 제안된 마스킹 Dilithium에 대한 i7-7600U CPU에서의 실행 시간을 나타낸 것이다.



〔그림 8〕 Dilithium 서명 과정 마스킹

	No-masking	Order-1	Order-2	Order-3
DILITHIUM.KeyGen	323 μs (*reference*)	1.83 ms (5.66×)	2.52 ms (7.8×)	4.32 ms (13.4×)
DILITHIUM.Sign	992 μs (*reference*)	5.64 ms (5.68×)	11.68 ms (11.77×)	28.08 ms (28.3×)

〔그림 9〕 마스킹 Dilithium에 대한 실행 시간

V. 결 론

본 논문은 LWE-like 암호에 대한 마스킹 부채널 대응 기법을 조사하였다. LWE-like 암호의 기반 연산은 다항식의 선형 연산으로 구성되어 있으므로 산술 마스킹을 적용하기 용이하다. 그러나 암호에서 사용되는 메시지 디코딩/인코딩, 샘플러, 비교 연산 등에 대해서는 산술 마스킹 적용이 불가능하므로 해당 연산에 대한 효율적이고 안전성이 증명된 마스킹 알고리즘을 개발이 진행되었다.

이렇듯 암호 시스템을 부채널 분석으로부터 안전하게 지키기 위한 대응 기술 개발은 신뢰할 수 있는 암호화 통신 시스템을 구축하는 데 중요한 역할을 한다. 따라서 앞으로의 암호화 기술 개발에는 부채널 분석과 같은 물리적 취약점에 대한 고려가 반드시 포함되어야 하고 이에 대비하기 위한 마스킹 기법과 같은 대응 기술 개발 및 적용이 필수적이다.

참 고 문 헌

[1] Ishai, Yuval, Amit Sahai, and David Wagner. "Private circuits: Securing hardware against probing attacks." In Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23, pp. 463-481. Springer Berlin Heidelberg, 2003.

[2] Beirendonck, Michiel Van, Jan-Pieter D'anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. "A side-channel-resistant implementation of SABER." ACM Journal on Emerging Technologies in Computing Systems (JETC) 17, no. 2 (2021): 1-26.

[3] Reparaz, Oscar, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. "A masked ring-LWE implementation." In Cryptographic Hardware and Embedded Systems--CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings, pp. 683-702. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[4] Oder, Tobias, Tobias Schneider, Thomas

Pöppelmann, and Tim Güneysu. "Practical CCA2-secure and masked ring-LWE implementation." Cryptology ePrint Archive (2016).

[5] Schneider, Tobias, Clara Paglialonga, Tobias Oder, and Tim Güneysu. "Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto." In Public-Key Cryptography–PKC 2019: 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II 22, pp. 534-564. Springer International Publishing, 2019.

[6] Bache, Florian, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. "High-speed masking for polynomial comparison in lattice-based KEMs." IACR Transactions on Cryptographic Hardware and Embedded Systems (2020): 483-507.

[7] Bhasin, Shivam, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. "Attacking and defending masked polynomial comparison for lattice-based cryptography." IACR Transactions on Cryptographic Hardware and Embedded Systems (2021): 334-359.

[8] D'Anvers, Jan-Pieter, Daniel Heinz, Peter Pessl, Michiel Van Beirendonck, and Ingrid Verbauwhede. "Higher-order masked ciphertext comparison for lattice-based cryptography." (2021).

[9] Bos, Joppe W., Marc Gourjon, Joost Renes, Tobias Schneider, and Christine Van Vredendaal. "Masking kyber: First-and higher-order implementations." IACR Transactions on Cryptographic Hardware and Embedded Systems (2021): 173-214.

[10] Coron, Jean-Sébastien, François Gérard, Simon Montoya, and Rina Zeitoun. "High-order polynomial comparison and masking lattice-based encryption." Cryptology ePrint Archive (2021).

[11] D'Anvers, Jan-Pieter, Michiel Van Beirendonck, and Ingrid Verbauwhede. "Revisiting higher-order masked comparison for lattice-based cryptography: algorithms and bit-sliced implementations." IEEE Transactions on Computers 72, no. 2 (2022): 321-332.

[12] Migliore, Vincent, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. "Masking Dilithium: efficient implementation and side-channel evaluation." In Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17, pp. 344-362. Springer International Publishing, 2019.

〈저자소개〉

이 정 환 (JeongHwan Lee)

학생회원

2023년 2월 : 고려대학교 인공지능사이버보안학과 학사

2023년 3월~현재 : 고려대학교 정보보호학과 석사 과정

<관심분야> 부채널 공격, 부채널 대응, 공개키 암호

정 상 윤 (Sang-Yun Jung)

학생회원

2023년 2월 : 고려대학교 인공지능사이버보안학과 학사

2023년 3월~현재 : 고려대학교 일반대학원 사이버보안학과

<관심분야> 부채널 공격, 화이트박스 암호

신 원 근 (Won Geun Shin)

학생회원

2018년~현재 : 고려대학교 인공지능사이버보안학과 학사 과정

<관심분야> 부채널 공격, 부채널 대응

박 수 진 (Sujin Park)

학생회원

2023년 2월 : 고려대학교 인공지능사이버보안학과 학사

2023년 3월~현재 : 고려대학교 정보보호대학원 정보보호학과 석사과정

<관심분야> 부채널 분석, 암호시스템 안전성 분석, 암호칩 설계 기술

김 희 석 (HeeSeok Kim)

종신회원

2006년 : 연세대학교 수학과 학사

2008년 : 고려대학교 정보보호대학원 석사

2011년 : 고려대학교 정보보호대학원 박사

2011년 9월~2012년 12월 : Bristol University 박사후 연구원

2013년~2016년 8월 : 한국과학기술정보연구원(KISTI) 선임연구원

2015년~2016년 8월 : 과학기술연합대학원대학교(UST) 조교수

2016년 9월~현재 : 고려대학교 과학기술대학인공지능사이버보안학과 부교수

<관심분야> 부채널 공격, 암호 시스템 안전성 분석 및 고속구현, 암호 칩 설계 기술, 보안관제, 네트워크 보안