NCC-Sign

송민호

유튜브 주소: https://youtu.be/VkbuGhO5thE

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

NCC-Sign

- 격자 기반 양자내성 전자서명 알고리즘
  - KpqC 공모전 1라운드 후보

- 비사이클로토믹 다항식(Non-Cyclotomic Polynomials) 사용
  - $\Phi(X) = X^p - X + 1$
  - NTT 대신 Toom-Cook 및 Karatsuba 사용
  - 모든 연산이 상수 시간으로 수행되어 타이밍 공격에 안전

- CRYSTALS-Dilithium의 설계방식을 따름
  - 비사이클로토믹 다항식 사용으로 MLWE 대신 RLWE에 기반
  - 두 개의 다항식을 활용한 새로운 최적화된 해싱 기법 사용
    - CRYSTALS-Dilithium보다 더 강한 보안성을 제공하면서 유사한 수준의 키 크기 및 서명 크기 유지

Cyclotomic Polynomials

- 기존의 효율적인 격자 기반 서명 알고리즘은 대부분 2의 거듭제곱 사이클로토믹 다항식(Cyclotomic Polynomials) 사용
  - $\Phi(X) = X^n + 1$ (ex. CRYSTALS-Dilithium)

- 효율성 측면에서 장점이 존재하지만 구조적 위협이 있음
  - 높은 속도와 작은 서명 및 키 크기

- 불필요한 대수 구조를 이용한 공격
  - 수체 $\mathbb{Q}[X]/\phi(X)$가 특정 $\phi(X)$에 대해 많은 부분체를 가짐[1]
  - 수체 $\mathbb{Q}[X]/\phi(X)$가 작은 갈루아 군을 갖고 있음[2]

[1] Bauch, J., Bernstein, D.J., Valence, H.d., Lange, T., Vredendaal, C.v.: Short gen- erators without quantum computers: the case of multiquadratics. In: Annual Inter- national Conference on the Theory and Applications of Cryptographic Techniques. pp. 27–59. Springer (2017)
[2] Campbell, P., Groves, M., Shepherd, D.: Soliloquy: A cautionary tale. In: ETSI 2nd Quantum-Safe Crypto Workshop. vol. 3, pp. 1–9 (2014)

Cyclotomic Polynomials

- 서브필드 공격
  - 수체 $\mathbb{Q}[X]/\phi(X)$의 부분체를 통해 Ring-LWE 문제를 약한 하위 문제로 환원
    - 보안 강도가 실제보다 낮아질 수 있음

- Ring Homomorphism 기반 공격
  - 링 동형사상을 이용하여 원래의 링 $\mathbb{Z}_q[X]/\phi(X)$에서 더 작은 링으로 만들어서 암호문 분석을 시도하는 공격[1, 2]

- 완전동형암호(FHE) 관련 공격 사례[3]
  - Gentry의 초기 FHE 및 soliloquy 등 사이클로토믹 기반 FHE들은 구조적 약점으로 인해 양자 또는 다항 시간 공격에 무너짐
    - STOC 2009에서 공개된 공격은 이러한 약점이 실제로 악용 가능함을 보여줌

[1] Eisentra ̈ger, K., Hallgren, S., Lauter, K.: Weak instances of PLWE. In: Interna- tional Conference on Selected Areas in Cryptography. pp. 183–194. Springer (2014)
[2] Elias, Y., Lauter, K.E., Ozman, E., Stange, K.E.: Provably weak instances of ring- LWE. In: Annual Cryptology Conference. pp. 63–92. Springer (2015)
[3] Biasse,J.F.,Song,F.:Efficientquantumalgorithmsforcomputingclassgroupsand solving the principal ideal problem in arbitrary degree number fields. In: Proceed- ings of the twenty-seventh annual ACM-SIAM sym posium on Discrete algorithms. pp. 893–902. SIAM (2016)

NCC-Sign 대응

- 비사이클로토믹 다항식 사용
  - $\Phi(X) = X^p - X + 1$

- 불필요한 대수 구조 제거
  - 갈루아 군이 큰 소수 차수 다항식 선택

- 기존 공격 기법들이 적용될 수 없는 환경을 만들었음

- 비구조적 격자 선택으로 인해 보안적으로는 더 강해졌으나 키 크기 및 서명 크기 등이 커짐

NCC-Sign 내부 함수

- Expand
  - ExpandA: 랜덤 시드를 바탕으로 공개 다항식 $a \in R_q$ 생성
  - ExpandS: 랜덤 시드를 바탕으로 비밀 다항식 $(s_1, s_2) \in S_\eta \times S_\eta$ 생성
  - ExpandMask: 랜덤 시드를 바탕으로 무작위 벡터 $\mathrm{y} \in \tilde{S}_\eta$ 생성

- 비트 분해 및 추출
  - Power2Round: 정수 또는 다항식을 상위 비트와 하위 비트로 분해하여 압축 가능한 형태로 만드는 함수
  - HighBits, LowBits: 입력 값의 상위 비트와 하위 비트를 추출하는 함수

- 해시
  - SampleInBall: 해시 값을 기반으로 일정 기준 이하의 챌린지 다항식 $c \in B_\tau$를 샘플링하는 함수

- 힌트
  - MakeHint: $r \in \mathbb{Z}_q$와 z를 기반으로 검증자가 상위 비트를 정확히 복원할 수 있도록 정보를 생성하는 함수
  - UseHint: Hint 값을 이용해 $\mathrm{r} + z$의 상위 비트를 복원하는 함수

NCC-Sign KeyGen

- 두 개의 256비트 시드 생성
  - ζ: 공개키 생성에 사용
  - ζ': 비밀키 구성 요소 생성에 사용
- ζ'를 해시하여 세 개의 256비트 시드 ζ_1, ζ_2, K생성
- ζ로부터 R_q에 속하는 다항식 a 생성
- ζ_1, ζ_2로부터 비밀 다항식 (s_1, s_2) 생성
  - 비밀 키 역할을 하는 핵심 요소
- (s_1, s_2)를 이용해 t 생성
- t를 고정 비트 수 d에 맞춰 상위 비트 t_1와 하위 비트 t_0로 나눔
- ζ와 t_1을 해시하여 tr 생성

Algorithm 8: KeyGen

$(\zeta, \zeta') \leftarrow \{0,1\}^{256} \times \{0,1\}^{256}$
$(\xi_1, \xi_2, K) \in \{0,1\}^{256} \times \{0,1\}^{256} \times \{0,1\}^{256} := \mathrm{H}(\zeta')$
$\mathbf{a} \in R_q := \mathsf{ExpandA}(\zeta)$
$(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta \times S_\eta := \mathsf{ExpandS}(\xi_1, \xi_2)$
$\mathbf{t} := \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$
$(\mathbf{t}_1, \mathbf{t}_0) := \mathsf{Power2Round}_q(\mathbf{t}, d)$
$tr \in \{0,1\}^{256} := \mathrm{H}(\zeta \parallel \mathbf{t}_1)$
return $(pk = (\zeta, \mathbf{t}_1), sk = (\zeta, tr, K, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$

NCC-Sign Sign

- tr과 M 값을 해시하여 메시지 바인딩 값 μ 생성
- 마스킹 시드 ρ를 이용하여 무작위 벡터 y 샘플링
  - y는 일종의 노이즈로 서명 값의 랜덤성을 책임짐
- 서명에서 챌린지 값 생성을 위해 w로부터 w_1 값 추출
- 메시지와 w_1를 해싱하여 챌린지 다항식 $\tilde{c}$ 생성
  - 해시 결과를 바탕으로 c 샘플링
- 서명 값 z 와 하위 비트 r_0 생성
  - 조건에 만족하지 않으면 서명 실패로 판단 후 서명 재시도
- 힌트 h 생성
  - 조건에 만족하지 않으면 실패 처리 후 다시 시도

Algorithm 9: Sign(sk, M)

$\mathbf{a} \in R_q := \mathsf{ExpandA}(\zeta)$
$\mu \in \{0,1\}^{512} := \mathrm{H}(tr \parallel M)$
$\kappa := 0,\ (\mathbf{z}, \mathbf{h}) := \perp$
$\rho \in \{0,1\}^{512} := \mathrm{H}(K \parallel \mu)$ (or $\rho \leftarrow \{0,1\}^{512}$ for randomized signing)
while $(\mathbf{z}, \mathbf{h}) = \perp$ **do**
  $\mathbf{y} \in \tilde{S}_{\gamma_1} := \mathsf{ExpandMask}(\rho, \kappa)$
  $\mathbf{w} := \mathbf{ay}$
  $\mathbf{w}_1 := \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
  $\tilde{c} \in \{0,1\}^{256} := \mathrm{H}(\mu \parallel \mathbf{w}_1)$
  $\mathbf{c} \in B_\tau := \mathsf{SampleInBall}_{p,\tau}(\tilde{c})$
  $\mathbf{z} := \mathbf{y} + \mathbf{cs}_1$
  $\mathbf{r}_0 := \mathsf{LowBits}_q(\mathbf{w} - \mathbf{cs}_2, 2\gamma_2)$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**
    $(\mathbf{z}, \mathbf{h}) := \perp$
  else
    $\mathbf{h} := \mathsf{MakeHint}_q(-\mathbf{ct}_0, \mathbf{w} - \mathbf{cs}_2 + \mathbf{ct}_0, 2\gamma_2)$
    if $\|\mathbf{ct}_0\|_\infty \geq \gamma_2$ **or** the # of 1's in $\mathbf{h}$ is greater than ω **then**
      $(\mathbf{z}, \mathbf{h}) := \perp$
    $\kappa := \kappa + 1$
return $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

NCC-Sign Verify

- 서명자와 동일한 a 생성
- 메시지 M과 ζ, t_1을 해시하여 메시지 바인딩 값 μ 생성
  - 서명자가 생성한 μ 값과 일치해야 함
- 전달받은 $\tilde{c}$를 바탕으로 c 값 복원
  - 서명자가 생성한 c 값과 일치해야 함
- 힌트 h를 이용하여 상위 비트 w'_1 복원
  - 서명자가 생성한 w'_1 값과 일치해야함
- 최종 조건 확인 및 결과 반환
  - 조건에 맞지 않으면 유효하지 않은 서명으로 간주하고 실패 반환

Algorithm 10: Verify$(pk, M, \sigma) = (\tilde{c}, \mathbf{z}, \mathbf{h})$

1 $\mathbf{a} \in R_q := \mathsf{ExpandA}(\zeta)$
2 $\mu \in \{0,1\}^{512} := \mathrm{H}(\mathrm{H}(\zeta \parallel \mathbf{t}_1) \parallel M)$
3 $\mathbf{c} := \mathsf{SampleInBall}(\tilde{c})$
4 $\mathbf{w}'_1 := \mathsf{UseHint}_q(\mathbf{h}, \mathbf{az} - \mathbf{ct}_1 \cdot 2^d, 2\gamma_2)$
5 **return** $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ **and** $[\![\tilde{c} = \mathrm{H}(\mu \parallel \mathbf{w}'_1)]\!]$ **and**
$[\![\#\text{ of 1's in } \mathbf{h} \text{ is } \leq \omega]\!]$

NCC-Sign 성능 비교

- KPQClean

Type	Algorithm	Keygen(Med.)	Signature(Med.)	Verification(Med.)	Keygen(Avr.)	Signature(Avr.)	Verification(Avr.)
Lattice	GCKSign-II	179,771	601,707	176,987	181,822	848,504	178,229
Lattice	GCKSign-III	186,673	649,049	183,367	198,852	899,646	185,793
Lattice	GCKSign-V	252,822	917,415	277,733	255,206	1,099,271	284,217
Lattice	HAETAE-I	798,312	4,605,461	147,494	1,091,637	5,704,780	148,078
Lattice	HAETAE-III	1,533,941	11,474,155	257,926	2,127,683	12,060,749	259,846
Lattice	HAETAE-V	846,713	3,902,298	305,428	1,104,472	5,214,861	306,973
Lattice	NCCSign-I(original)	1,869,079	23,762,252	3,681,057	1,882,892	23,763,293	3,684,640
Lattice	NCCSign-III(original)	3,655,334	39,587,190	7,241,808	3,675,996	39,635,337	7,246,465
Lattice	NCCSign-V(original)	6,263,739	179,281,596	12,418,902	6,268,503	179,337,534	12,422,702
Lattice	NCCSign-I(conserparam)	2,650,542	10,404,301	5,232,079	2,670,083	10,419,012	5,244,741
Lattice	NCCSign-III(conserparam)	4,477,513	17,657,839	8,867,243	4,497,436	17,666,605	8,869,094
Lattice	NCCSign-V(conserparam)	7,240,343	64,377,767	14,358,074	7,257,655	64,387,183	14,375,040
Lattice	Peregrine-512	12,401,256	329,933	37,294	12,609,569	332,600	37,505
Lattice	Peregrine-1024	39,405,505	709,848	80,243	42,160,344	722,426	81,200
Lattice	SOLMAE-512	23,848,774	378,392	43,935	29,181,985	385,719	44,109
Lattice	SOLMAE-1024	55,350,546	760,380	141,375	70,141,847	764,304	142,357

Benchmark result

Testing Environment1

- OS: Ubuntu 22.04
- CPU: Ryzen 7 4800H (2.90 GHz)
- RAM: 16GB
- Compiler: gcc 11.3.0
- Optimization Level: -O2. -O3

Q & A