MQ-Sign 개선 사항

NIMS 암호기술연구팀

심경아, 권혁동

설계 방향/원칙/특징

- 단일 레이어 UOV 기반 구조의 최소화/짧은 길이의 전자서명
  - 기존 다변수 이차식 기반 Multiple-layer 구조의 위험성 제거 => Single-layer 구조 사용
  - 짧은 서명 길이: 안전도 1/3/5에서 150/216/276 byte(Falcon 666/1280 byte 대비 20/16%)
- UOV 보다 짧은 비밀키
  - MQ-Sign-RR: 랜덤 Vinegar*Vinegar 이차항
  - MQ-Sign-LR: 선형식 기반 Vinegar*Vinegar 이차항: 비밀키 길이 42% 이상 축소
- 우수한 성능: 안전도 1, 3에서 서명 생성, 검증 Dilithium 보다 빠름
- 공개키 변형을 이용한 잠재적 대한 안전성 보장
  - H(M) 대신 H(M||H(P)) 사용함으로써 공개키 변형을 통한 잠재적 위조 공격 방어
- 빠른 서명 생성과 사전 계산이 용이한 구조
  - 블록 행렬을 이용한 고속화 기법으로 서명 생성 효율성 향상
  - MQ-Sign-LR: RR 보다 키 생성 34%-42% 향상,서명 생성 27%-37% 향상
  - 서명 생성에서 대부분의 계산이 메시지와 독립적으로 이루어지도록 설계
    - ✓ on-line 서명 생성 계산과 행렬과 벡터 계산 한번, 4-6배 이상 서명 속도 향상

MQ-Sign

- 서명 생성에 공개키와 서명을 묶는 binding technique 추가
  - 두 개의 공개키 $\mathcal{P} = \mathcal{F} \circ T,\ \mathcal{P}' = (\mathcal{F} \circ T) \circ T'$
  - 메시지 M, 공개키 $\mathcal{P}$ 에 대한 서명 $\sigma = (\mathbf{z}, r)$ 이용

    -> $\mathcal{P}'$ 에 대한 서명 $\sigma' = (\mathbf{z}', r)$, $\mathbf{z}' = (T')^{-1}(\mathbf{z})$ 생성 가능
  - 서명 생성/검증 메시지와 공개키의 해시 값 추가: $H(M||r||H(\mathcal{P}))$
    - ✓ 공개키와 메시지에 정확하게 연결된 서명 생성
- $H(\mathcal{P})$ 계산: 큰 공개키 사이즈로 비효율적
  - MQ-Sign-RR: 키생성에서 $H(\mathcal{P})$ 계산 후 공개키 비밀키에 추가
  - $H(\mathcal{P})$대신 $H(\mathcal{P}_2||\mathcal{P}_3),\ \mathcal{P}_2 = \{\mathcal{P}_2^{(k)}\}_{k=1}^{o}, \mathcal{P}_3 = \{\mathcal{P}_3^{(k)}\}_{k=1}^{o}$ 계산
    - ✓ 위 공격에서 $\mathcal{P}_1^{(k)} = P_1'^{(k)}\ (k = 1, \cdots, o)$, linear map에 의존하지 않음
    - ✓ 두 공개키의 central map $\mathcal{F}$ 로 동일, 서로 다른 linear map $\mathcal{T}$ 와 $\mathcal{T} \circ \mathcal{T}'$

Security Level	1	3	5
MQ-Sign-RR(s)	9,454,708	40,250,626	102,775,550
MQ-Sign-RR(m)	8,291,246	36,168,836	91,291,653
MQ-Sign-RR(o)	6,633,743	30,024,722	77,684,841

사전 계산

- 서명 생성에서 메시지에 의존하지 않는 부분 사전 계산 가능
  - Intel Xeon(R) Gold 6234 CPU, 3.3GHz
  - 키생성: 1,000번 평균, 서명 생성/검증: 10,000번 중간 값
  - 사전 계산이 없는 경우 대비 안전도 1, 3, 5에서 각각 4.6배, 5.2배, 6.3배 빠름.

Scheme	Security Level	1	3	5
MQ-Sign-RR (MQ-Sign-LR)	Sign	19,532	49,256	84,465
	Memory	2,266	5,400	9,492

공개키 구조 변경

- MQ-Sign v2.1 공개키 구조 변경
  - ➢ 공개키에서 사용되지 않는 부분 발생

    해당 부분이 코드 상에서 제거되지 않아 '0'으로 저장되는 현상
  - ➢ 2.1에서 해당 부분을 제거하며 키 크기가 다소 축소

Security Level	MQ-Sign-RR 2.0	MQ-Sign-RR 2.1	MQ-Sign-LR 2.0	MQ-Sign-LR 2.1
1	328,505	323,031	328,441	322,967
3	1,238,825	1,225,505	1,238,761	1,225,441
5	2,893,025	2,869,505	2,892,961	2,869,441

AVX2 최적 구현

- AVX2 최적 구현: 키생성 향상
  - 공개키 적정 부분을 랜덤하게 선택한 후 계산을 통해 비밀키 생성, RR에만 적용 가능
  - Intel® Core i7-1360P, 100회 구현 후 중간 값, clock cycles
  - 괄호 안 수치는 2.0 대비 개선된 정도

MQ-Sign-RR 2.0	1	3	5
Keygen (s)	9,454,708	40,250,626	102,775,550
Keygen (m)	8,291,246	36,168,836	91,291,653
Keygen (o)	6,633,743	30,024,722	77,684,841
MQ-Sign-RR 2.1	**1**	**3**	**5**
Keygen (s)	8,574,372(9.77%)	34,464,908(15.49%)	99,201,093(3.54%)
Keygen (m)	7,595,005(8.77%)	31,538,016(13.68%)	90,031,140(1.39%)
Keygen (o)	5,718,159(14.82%)	25,427,415(16.58%)	75,055,554(3.44%)

MQ-Sign 활용

- MQ-Sign: 짧은 전자서명 길이, 상대적으로 큰 공개키 크기
- Root CA 전자서명으로 적합
  - ➢ Root CA의 인증서는 전송하지 않고 저장, 공개키 크기가 전송량에 영향을 주지 않음. 서명 생성, 검증 성능 우수
  - ➢ 인증서 크기 최소화 가능: CA 전자서명 안전도 3, 사용자 전자서명 안전도 1 적용
    - **(CA, 사용자)=(MQ-Sign, Falcon)의 경우, 인증서 크기: 1,116 bytes**
    - (CA, 사용자)=(Falcon, Falcon)의 경우, 인증서 크기: 2,177 bytes
    - (CA, 사용자)=(RSA, RSA)의 경우, 인증서 크기: 1,344 bytes

NCC-Sign 개선 사항

NIMS 암호기술연구팀 심경아,
김정수, 권혁동

설계 방향/원칙/특징

- **[효율성에 초점을 맞춘 설계]** 대부분 효율적인 격자 기반 암호 power-of-2 cyclotomic polynomial
  - 상대적으로 짧은 키길이, 서명 길이, 우수한 성능
  - 공격의 원인이 되는 부가적 구조, 양자 알고리즘
- **[안전성 강화]** 최초의 non-cyclotomic/trinomial 격자 기반 전자서명 알고리즘
  - Intermediate Security Guarantee, 구조의 최소화: non-cyclotomic
  - 보수적인 안전성 유지하면서 효율성 증대: Trinomial
- **[보수적인 파라미터 선택]**
  - Core-SVP 복잡도 128, 192, 256에 밀접하거나 초과하도록 선택
  - Dilithium 보다 더 높은 복잡도 제공: 진화되는 공격에 대한 충분한 안전성 마진 제공
- **[빠른 성능]**
  - NCC-Sign-T(Trinomial)은 Dilithium 보다 성능 우위 (Reference 구현 기준)
  - AVX2 최적 구현의 경우 현재 Dilithium 보다 빠르거나 비슷한 수준

NCC-Sign 파라미터

- NCC-Sign trinomial version
  - ➢ 보수적인 파라미터 선택
    - ✓ 안전도 마진 확보
    - ✓ NTT friendly ring

$Z_q[X]/(X^n - X^{n/2} - 1)$

에서의 NTT 사용

Parameter/Security Level	1	3	5′	5
n	1152	1536	2048	2304
q	8401537	8397313	8380417	8404993
d [dropped bits from t] ($2^d\tau < \gamma_2$)	12	12	11	13
τ [# of ± 1's in c]	25	29	32	32
challenge entropy [$\log \binom{p}{\tau} + \tau$]	195	232	265	271
γ_1 [y coefficient range]	2^{18}	2^{18}	2^{18}	2^{19}
γ_2 [low-order rounding range]	131274	131208	130944	262656
η [secret key range]	1	1	1	1
β	50	58	64	64
ω [max # of 1's in hint]	80	80	80	80
Exp. reps. [$\approx e^{n\beta(1/\gamma_1+1/\gamma_2)}$]	1.93	2.76	4.49	2.32
Key/Signature Size				
pk size	1760	2336	3104	3200
sk size	2400	3168	3936	4992
sig size	2912	3872	5152	6080
SIS Hardness (Core-SVP)				
BKZ block-size b to break SIS	462	671	963	1005
Best Known Classical bit-cost	135	196	281	293
Best Known Quantum bit-cost	122	177	255	266
LWE Hardness (Core-SVP)				
BKZ block-size b to break LWE	451	652	934	1078
Best Known Classical bit-cost	131	190	273	315
Best Known Quantum bit-cost	119	172	247	285
Lattice estimator (Core-SVP)				
BKZ block-size b to break LWE	452	652	930	1072
Classical bit-cost	132	190.7	271.7	313.3
(method)	(usvp)	(dual hybrid)	(dual hybrid)	(dual hybrid)

NCC-Sign vs. Dilithium

- 복잡도, 사이즈 비교
  - ➢ Dilithium 대비 안전도 1,3,5에서
    최소 8비트 최대 41비트
    안전성 마진 제공

Scheme	Core-SVP/Size	1	3	5 (5′)
Dilithium	SIS	123	186	265
	LWE	123	182	252
	Repetitions	4.25	5.1	3.85
	Public key size	1312	1952	2592
	Signature size	2420	3293	4595
NCC-Sign Non-Cyclototmic ($\eta = 2$)	SIS	135	194	261
	LWE	143	207	279
	Repetitions	2.27	2.7	3.43
	Public key size	1984	2443	3091
	Signature size	3186	4251	5385
NCC-Sign Non-Cyclototmic ($\eta = 1$)	SIS	135	194	261
	LWE	131	191	258
	Repetitions	1.58	1.74	1.98
	Public key size	1984	2443	3091
	Signature size	3936	5255	6659
NCC-Sign Trinomial	SIS	135	196	293 (281)
	LWE	131	190	315 (273)
	Repetitions	1.93	2.71	2.32 (4.49)
	Public key size	1760	2336	3200 (3104)
	Signature size	2912	3872	6080 (5152)

최적 구현 분석

- NCC-Sign-T의 Dilithium 대비 성능 우위
  - Ring-LWE vs. mod-LWE
    - ✓ NCC-Sign-T 공개키 생성을 위한 계수의 개수가 Dilithium 개수의 1/2-1/3 정도
  - Rejection sampling 반복 횟수가 Dilithium의 ½
    - ✓ Dilithium의 경우 MLWE 병렬화 매우 용이, SHAKE 더 많은 부분 차지
    - ✓ SHAKE 병렬화로 성능 반전
- Avx2 최적 구현
  - 모든 함수 병렬화, ASM 병렬화
  - NTT 구현 최적화: Signed Montgomery 사용을 통한 8 packing 병렬화
  - Merging 이용
  - 파이프라인 구조 활용
  - Sparse multiplication 이용

레퍼런스 구현

- 레퍼런스 코드 성능 개선
  - Intel® Core i7-1360P
  - 10,000회 반복한 중간 값, clock cycles
    (괄호 안은 2.0 버전과 2.1 버전의 성능 차이)

Scheme	Algorithm	1	3	5
NCC-Sign(2.0)^R	Keygen	141,483	177,386	276,296
	Sign	317,367	591,754	877,231
	Verify	177,142	207,295	339,468
NCC-Sign(2.1)^R	Keygen	**127,415(10.46%)**	**162,315(8.87%)**	**259,499(6.27%)**
	Sign	**250,128(23.70%)**	**421,926(33.51%)**	**704,288(21.87%)**
	Verify	**128,538(31.80%)**	**155,346(28.65%)**	**252,686(29.31%)**

AVX2 최적 구현

- AVX2 구현 성능 개선: Intrinsic에서 Assembly로 최적 구현
  - ➢ Intel® Core i7-13700K, 10,000회 반복 중간 값, clock cycles
    (괄호 안은 2.1^{A} 버전과 Dilithium^{A}의 성능 차이)
  - ➢ 안전도 3에서는 Dilithium 대비 성능이 더 우수하고, 안전도 1, 5에서는 부분적으로 우세

Scheme	Algorithm	1	3	5
NCC-Sign(2.0)^{A}	Keygen	118,092	147,909	224,222
	Sign	205,023	379,847	570,625
	Verify	115,383	138,513	223,968
NCC-Sign(2.1)^{A}	Keygen	**54,902(-13.88%)**	**67,432(20.47%)**	**109,402(17.86%)**
	Sign	**89,207(20.13%)**	**165,766(8.38%)**	**261,423(-9.32%)**
	Verify	**65,626(-29.95%)**	**79,388(1.83%)**	**128,612(0.23%)**
Dilithium^{A}	Keygen	47,773	82,810	130,863
	Sign	109,176	180,271	238,143
	Verify	48,533	80,851	128,920

Cortex-M4 구현

- Cortex-M4 구현
  - NUCLEO-L4R5ZI, 100회 반복 중간 값, clock cycles, pqm4
  - 모든 파라미터에서 Dilithium 보다 성능 우수: 35%~74% 더 빠름

(괄호 안은 2.1^{A} 버전과 $Dilithium^{A}$의 성능 차이)

Scheme	Algorithm	1	3	5
$NCC\text{-}Sign(2.1)^{R}$	Keygen	2,140k	2,795k	4,236k
	Sign	7,000k	11,015k	16,875k
	Verify	3,110k	3,944k	6,159k
$NCC\text{-}Sign(2.1)^{A}$	Keygen	**1,005k(34.64%)**	**1,308k(63.18%)**	**1,972k(73.77%)**
	Sign	**2,544k(39.97%)**	**4,032k(45.01%)**	**5,647k(40.03%)**
	Verify	**1,263k(11.49%)**	**1,588k(41.16%)**	**2,453k(52.18%)**
$Dilithium^{R}$	Keygen	2,995k	5,229k	8,810k
	Sign	10,034k	16,005k	20,969k
	Verify	3,143k	5,252k	8,935k
$Dilithium^{A}$	Keygen	1,426k	2,516k	4,277k
	Sign	3,815k	6,374k	8,473k
	Verify	1,418k	2,411k	4,185k

Cortex-M4 구현

- Dilithium 보다 거의 대부분에서 stack usage 더 우수: 최대 30%

Scheme	Variant		1		3		5	
			cc	stack	cc	stack	cc	stack
NCC-Sign-T	ref	K	2,140k	31.0k	2,795k	41.1k	4,236k	61.4k
		S	7,000k	49.3k	11,015k	65.4k	16,875k	97.7k
		V	3,110k	36.4k	3,944k	48.3k	6,159k	71.8k
NCC-Sign-T (This work)	speed	K	1,005k	31.0k	1,308k	41.1k	1,972k	61.4k
		S	2,544k	49.3k	4,032k	65.4k	5,647k	97.7k
		V	1,263k	36.4k	1,588k	48.3k	2,453k	71.8k
Dilithium [KKPY24]	m4f	K	1,430k	37.4k	2,521k	59.4k	4,279k	95.4k
		S	3,877k	48.2k	6,754k	67.2k	8,865k	113.2k
		V	1,418k	35.3k	2,413k	56.3k	4,187k	90.6k

- 구현 상세
  - pqm4의 SHAKE를 탑재(XKCP 구현물) Keccak1600.S 포팅
  - NTT merging 사용
  - Modular arithmetic, point-wise multiplication 최적화

최신 Hybrid Dual attack에 대한 안전성 분석

- Bi, Lu, Luo, Wang and Zhang: LWE 문제에 대한 hybrid dual attack 제안
  - 기존 sparse, small secret-> any secret distribution
  - Dilithium 기존 대비 2-4 비트 복잡도 감소
  - NCC-Sign-T은 Dilithium 대비 7-56 비트 높은 복잡도 제공

Scheme	Security Level	Dual	Hybrid Dual [BLL+22]
Dilithium	2	124	122
	3	182	179
	5	251	247
NCC-Sign-T	1	132	129
	3	190.7	186
	5(5′)	313.3(271.7)	306(265)

감사합니다.