Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.4, Aug. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.4.591

M-SIDH 구현 및 성능 평가를 통한 효율성 연구*

김 수 리,[1†] 서 민 혜[2‡]
[1]성신여자대학교 (교수), [2]덕성여자대학교 (교수)

Implementing M-SIDH: Performance and Efficiency Evaluation*

Suhri Kim,[1†] Minhye Seo[2‡]
[1]Sungshin Women's University (Assistant Professor),
[2]Duksung Women's University (Assistant Professor)

요 약

최근 Castryck-Decru에 의해 SIDH 기반 암호가 다항시간 안에 개인키를 복구할 수 있음에 따라 이에 대응하기 위한 여러 방법이 제안되었다. 이 중, Fouotsa 등이 제안한 M-SIDH는 상대방에게 전달하는 torsion point 정보를 마스킹하여 Castryck-Decru 공격에 대응한다. 본 논문에서는 처음으로 C를 이용해 M-SIDH를 구현하였으며, 최적화를 통해 효율성을 평가한다. 본 논문은 M-SIDH의 성능을 확인하기 위해 1024비트 소수를 사용하여 파라미터를 선택하는 방법을 제시하였으며, square-root Velu 공식의 확장체에서의 구현을 통해 M-SIDH를 최적화하였다. 본 논문의 결과 고전 64비트 보안강도를 가지는 MSIDH-1024의 경우 키 교환하는데 대략 1129ms 정도가 필요하다.

ABSTRACT

Due to the recent attack by Castryck-Decru, the private key of SIDH can be recovered in polynomial time so several methods have been proposed to prevent the attack. Among them, M-SIDH proposed by Fouotsa et al, counteracts the attack by masking the torsion point information during the key exchange. In this paper, we implement M-SIDH and evaluate its performance. To the best of our knowledge, this is the first implementation of M-SIDH in C language. Toward that end, we propose a method to select parameters for M-SIDH instantiation and propose a 1024-bit prime for implementation. We implemented the square-root Velu formula over the extension field for further optimization. As a result, 1129 ms is required for a key exchange in the case of MSIDH-1024, providing the classic 64-bit security level.

Keywords: PQC, Isogeny-based cryptography, CSIDH, SIDH, MSIDH

I. 서 론

2006년 Couvegines에 의해 isogeny 기반 암호는 처음으로 제안되었으며, 오늘날에 CRS 암호라 한다 [1]. CRS는 유한체 위에 정의된 ordinary 타원곡선을 사용하여 Diffie-Hellman 형태의 키 교환 알고리즘이다. 하지만 ordinary 곡선은 endomorphism ring이 가환적인 성질을 가져 Childs가 제안한 공격으로 양자 하지수시간 공격이 존재할 뿐만 아니라, ordinary 곡선의 사용으로 속도가 매우 느리다는 단점이 있다 [2]. Isogeny 기반 암호는 이후 2011년 De Feo와 Jao 가 제안한 Supersingular Isogeny Diffie-Hellman (SIDH)에 의해 다시 주목받게 된다 [3]. SIDH는 supersingular 곡선의 사용으로 endomorphism

Received(03. 17. 2023), Modified(05. 09. 2023), Accepted(05. 22. 2023)
* 본 연구는 삼성전자의 지원 (IO221213-04119-01)을 받아 수행된 결과임
† 주저자, suhrikim@sungshin.ac.kr
‡ 교신저자, mhseo@duksung.kr(Corresponding author)

ring이 가환적이지 않아 Childs의 공격에 대응하였으며, 2016년 Costello 등에 의해 제안된 최적화 기법으로 효율적인 구현이 가능해져 SIKE라는 key encapsulation 알고리즘으로 NIST PQC 표준화 공모전에 제안되었다.

하지만, 2022년 제안된 Castryck-Decru의 공격에 의해 SIDH 기반 암호는 다항시간 안에 해결되면서 더 이상 사용할 수 없다[4]. Castryck-Decru의 공격의 핵심은 torsion point 정보를 이용하는 것이다. SIDH 기반 암호의 경우 supersingular 곡선의 사용으로 비 가환적인 성질을 가지고 있어, 기존 Diffie-Hellman과 다르게 키 교환을 진행할 때 상대방에게 자신의 연산값만 전달하면 되는 것이 아니라, 같은 공유된 비밀키를 연산하기 위해서는 자신의 개인키로 상대방의 공개키에 대한 값을 연산해 전달하는 부분이 있다. 이를 isogeny 기반 암호에서는 torsion point 정보를 이용한다. Castryck-Decru 공격에는 torsion point 정보가 핵심으로 사용되기 때문에, SIDH 기반 암호를 비롯해 torsion point 정보를 사용하는 BSIDH, SETA 등의 암호가 다항시간안에 해결된다. 한편, isogeny 기반 암호 중 가환적인 성질로 torsion point 정보를 제공하지 않는 CSIDH 기반 암호는 해당 공격에 영향을 받지 않는다.

Castryck-Decru 공격에 대응하기 위해 Fouotsa 등은 2023년 M-SIDH와 MD-SIDH를 제안하였다 [5]. Castryk-Decru 공격은 SIDH 기반 암호가 torsion point 정보를 제공한다는 점과, 두 타원곡선 사이를 연결하는 isogeny의 차수가 알려져 있고, 연산의 효율성을 위해 해당 차수는 power-smooth 라는 형태를 이용한다. 만약, isogeny 기반 암호에서 torsion point 정보나 isogeny 차수 중 하나라도 알아내기 힘들 경우 Castryck-Decru의 공격은 어려워지게 된다. 따라서 [5]에서 제안하는 M-SIDH는 masked torsion point SIDH의 약자로, torsion point 정보를 전달할 때 랜덤한 값으로 마스킹을 하여 공격에 대응하는 방법이고, MD-SIDH는 masked-degree SIDH의 약자로, 사용하는 isogeny의 차수를 마스킹하여 공격에 대응하는 방법이다.

본 논문에서는 Fouotsa 등이 제안하는 방법 중 같은 보안강도에서 사용하는 유한체의 크기가 비교적 적은 M-SIDH에 대한 구현 결과를 제시한다. 본 논문에서는 처음으로 M-SIDH를 C로 구현하였으며 이를 위한 본 논문의 기여를 정리하면 다음과 같다.

- 본 논문에서는 처음으로 확장체에서의 square-root Velu 공식을 구현하였다. M-SIDH는 CSIDH 기반 암호처럼 홀수 차수 isogeny를 사용하기 때문에 기존 Velu의 공식을 최적화한 square-root Velu 공식의 적용이 필수적이다. 또한, SIDH 에서는 여러 타원곡선 위의 점에 대해 isogeny 함수값을 연산해야하는 만큼, 이에 맞게 square-root Velu 공식을 추가로 최적화하였다.
- 본 논문에서는 M-SIDH의 구현을 위해서 파라미터 선택 방법을 제안하였으며, 성능 확인을 위해 64비트의 고전 보안강도를 갖는 MSIDH-1024 파라미터를 제안한다. 이에 대한 자세한 내용과 구현 방법은 본 논문의 3장에 정리되어 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 필요한 SIDH와 MSIDH를 소개한다. 3장에서는 구현에 필요한 파라미터 설정 및 알고리즘을 소개하고 4장에서는 구현 결과를 제시하고, 5장의 결론으로 마무리한다.

II. 배경지식

본 장에서는 논문에 필요한 배경 지식을 소개한다. 먼저 SIDH와 Castryk-Decru 공격을 소개한 뒤, M-SIDH에 대해 소개한다.

2.1 SIDH 와 Castryck-Decru 공격

SIDH는 유한체 F_{p^2}에 정의된 supersingular 타원곡선을 활용한 isogeny 기반 암호이다. 먼저 서로 소인 두 소수 ℓ_A, ℓ_B에 대해서 $p = \ell_A^{e_A}\ell_B^{e_B}f \pm 1$ 형태의 소수가 되도록 선택한다. 이 때, $\ell_A^{e_A} \cong \ell_B^{e_B}$가 되도록 e_A, e_B를 선택한다. 실제 구현에서는 보통 $\ell_A = 2, \ell_B = 3$을 선택하여 $p = 2^{e_A}3^{e_B} - 1$ 형태가 되도록 한다. 이 소수를 사용해 형성한 유한체 F_{p^2}

위에서 위수가 $(\ell_A^{e_A}\ell_B^{e_B})^2$인 supersingular 타원곡선 E를 선택한다. 그 후, $E[\ell_A^{e_A}]$ torsion subgroup의 기저인 $\langle P_A, Q_A\rangle$를 선택하고, $E[\ell_B^{e_B}]$ torsion subgroup의 기저인 $\langle P_B, Q_B\rangle$를 선택한다.

Alice는 기저 $\langle P_A, Q_A\rangle$에 대해 개인키 n_A를 이용하여 그룹 $\langle R_A\rangle = \langle P_A + [n_A]Q_A\rangle$를 계산한다. 그 뒤 Velu의 공식을 이용해 $\langle R_A\rangle$를 커널로 하는 isogeny $\phi_A : E \rightarrow E_A$를 연산한다. 여기에서 $E_A = E/\langle R_A\rangle$를 만족한다. Alice는 연산된 E_A와 추가로 $\phi_A(P_B), \phi_A(Q_B)$를 연산해 Bob에게 전달한다. Bob도 비슷한 과정을 통해 연산한 뒤, Alice에게 $E_B, \phi_B(P_A), \phi_B(Q_A)$를 Alice에게 전달한다. Alice는 Bob에게 받은 점을 이용하여 다시 그룹 $\langle R_A'\rangle = \langle \phi_B(P_A) + [n_A]\phi_B(Q_A)\rangle$를 연산한 뒤, Velu의 공식을 이용하여 $\langle R_A'\rangle$를 커널로 하는 isogeny의 codomain 곡선인 $E_{AB} = E_B/\langle R_A'\rangle$를 연산한다. Bob도 Alice와 비슷한 과정을 통해 $E_{BA} = E_A/\langle R_B'\rangle$를 연산한다. Alice와 Bob 사이의 비밀 공유키는 생성된 타원곡선의 j-invariant에 해당되는 $j(E_{AB}) = j(E_{BA})$ 이다.

SIDH의 키 교환 과정을 살펴보면, supersingular 곡선의 사용으로 비가환적인 성질 때문에 자신의 개인키로 상대방의 공개키를 연산해 전달하는 과정 (Alice의 경우 $\phi_A(P_B), \phi_A(Q_B)$와 Bob의 경우 $\phi_B(P_A), \phi_B(Q_A)$) 이 있다. 이러한 특성은 일반적인 키 교환 알고리즘에서는 볼 수 없고 SIDH 기반 암호만 가지는 독특한 특성으로 이 부분을 취약점으로 의심하여 연구가 진행되고 있었다.

Isogeny 기반 암호의 안전성은 두 타원곡선 사이의 isogeny를 구하는 어려움에 있다. 한편, Deuring correspondence에 의해 두 타원곡선 사이의 isogeny와 각 타원곡선의 endomorphism ring을 연결하는 ideal과 동치이며, 해당 ideal은 KLPT 알고리즘을 활용해서 연산할 수 있다 [6]. 그러나 KLPT 알고리즘을 활용해서 얻은 ideal은 실제 SIDH 기반 암호에 사용되는 power-smooth 한 형태가 아니다. 하지만 2021년 Fouotsa 등의 연구를 통해서 torsion point 정보를 활용하면 power-smooth 한 ideal을 얻음으로써 공격 복잡도를 낮출 수 있었다. [7].

Torsion point 정보가 본격적으로 활용된 공격은 2023년의 Castryck-Decru의 공격으로, 이 정보와 Kani의 glue-and-split 알고리즘을 활용해 다항시간 안에 SIDH 기반 암호를 분석하게 되었다. 타원곡선 E_0의 위수가 서로 소인 두 그룹을 H_0, H_1이라 하자. $\phi : E_0 \rightarrow E_1 = E_0/H_1$이라 하고 $\gamma : E_0 \rightarrow E_2 = E_0/H_2$를 두 isogeny라 하고, $\phi' : E_2 \rightarrow E_3$을 커널이 $\gamma(H_1)$인 isogeny, $\gamma' : E_1 \rightarrow E_3$를 커널이 $\phi(H_2)$인 isogeny라 하자. $N = \#H_1 + \#H_2$라 하고, P_0, Q_0를 $E_0[N]$의 기저로 하여 $P_1, Q_1, P_2, Q_2\ P_3, Q_3$를 다음과 같이 정의한다.

$$(P_1, Q_1) = (\phi(P_0), \phi(Q_0)) \tag{1}$$

$$(P_2, Q_2) = (\gamma(P_0), \gamma(Q_0)) \tag{2}$$

이를 이용해 다음과 같은 함수를 정의하자.

$$\rho : E_2 \times E_1 \rightarrow E_0 \times E_3 \tag{3}$$

Kani는 ρ가 커널이

$$H = \langle (P_2, P_1), (Q_2, Q_1)\rangle \tag{4}$$

인 isogeny 라는 것을 증명했으며, 이러한 Abelian surface 사이의 map을 (N, N)-isogeny라 한다. Castryck-Decru의 공격은 이를 활용해 기존에 중간에서 E_0에서 E_3로 가는 isogeny path 중간에서 만나는 meet-in-the-middle 방식의 공격을 변형하였다. 해당 공격은 E_0에서 E_3로 가는 isgoeny degree가 smooth 하다는 점을 이용해서 작은 b에 대한 3^b-isogeny를 guessing을 한 뒤, ρ 를 구성해서 mapping 의 결과를 확인한다. 만약 결과가 타원곡선의 곱 형태로 주어지면 옳게 추측한 것이고, 그렇지 않고 랜덤한 genus 2 curve 이면 틀리게 추측한 것이라는 것이며, 이를 통해 공격복잡도를 크게

낮출 수 있었다.

2.2 M-SIDH

Castryck-Decru의 공격에 성공하려면 1) 두 타원곡선 사이의 isogeny ϕ의 차수가 알려져 있어야 하고, 2) ϕ로 연산된 torsion basis P, Q값인 $\phi(P), \phi(Q)$가 알려져있어야 하며, 각각이 알려져 있지 않을 경우 공격이 어렵게 된다. 따라서, Fouotsa 등은 각 정보를 마스킹해서 Castryck-Decru 공격에 대응하는 방법을 제안하였으며, 1)에 대응하여 isogeny 차수를 마스킹한 MD-SIDH와 2)에 대응하여 torsion point 정보를 마스킹한 M-SIDH를 제안하였다. 본 논문에서는 동일한 보안강도에서 비교적 작은 유한체의 크기를 사용하는 M-SIDH에 대해서만 설명한다.

M-SIDH의 핵심은 torsion point 정보를 전달할 때 $\phi(P), \phi(Q)$를 전달하지 않고 랜덤값 α를 이용해 $[\alpha]\phi(P), [\alpha]\phi(Q)$를 전달하는 방법이다. 상대방의 torsion group의 위수가 B라 가정하자. 사용하는 isogeny의 차수는 고정되므로, Weil pairing을 계산하면 $e_B([\alpha]P,[\alpha]Q) = e_B(P,Q)^{\alpha^2 \deg\phi}$을 만족하게 되고, $\alpha^2 \bmod B$를 연산할 수 있게 된다. 만약 $\alpha^2 \equiv 1 \bmod B$인 α를 활용해서 마스킹을 진행한다면, 상대방은 Weil paring을 활용하여 올바르게 마스킹이 되어있는지 안되어있는지 알 수 있게 된다. 마스킹 값은 공격자에게 알려지지 않도록 $\alpha^2 \equiv 1 \bmod B$를 만족하는 α의 수가 많도록 torsion point의 위수를 선택한다. M-SIDH의 알고리즘은 다음과 같다.

① 파라미터 생성

보안강도 λ에 대해서 $t = t(\lambda)$로 보안강도에 의해 결정되는 자연수라 하자. 소수 $p = ABf - 1$형태로 선택한다. 여기에서 $A = \prod_{i=1}^{t} \ell_i$, $B = \prod_{i=1}^{t} q_i$이며, ℓ_i, q_i는 서로 다른 소수에 해당되고 A, B는 소로 소이며 $A \approx B \approx \sqrt{p}$를 만족하고 f는 cofactor에 해당된다. $E_0[A] = \langle P_A, Q_A \rangle$라 하고 $E_0[B] = \langle P_B, Q_B \rangle$라 한다. $\mu_2(N)$을 다음과 같이 정의한다. 아래 식에서 Z는 정수를 의미한다.

$$\mu_2(N) = \{x \in Z/NZ \mid x^2 \equiv 1 \bmod N\} \quad (5)$$

② 공개키 생성

Alice는 $\mu_2(B)$와 Z/AZ 에서 랜덤한 정수 α와 a를 각각 선택한다. 그 후, $\langle P_A + [a]Q_A \rangle$를 커널로 하는 isogeny $\phi_A : E_0 \rightarrow E_A = E_0/\langle P_A + [a]Q_A \rangle$를 연산하고, $E_A, [\alpha]\phi_A(P_B), [\alpha]\phi_A(Q_B)$를 Bob에게 전달한다.

Bob도 Alice와 동일하게 $\mu_2(A)$와 Z/BZ 에서 랜덤한 정수 β와 b를 각각 선택한다. 그 후, $\langle P_B + [b]Q_B \rangle$를 커널로 하는 isogeny $\phi_B : E_0 \rightarrow E_B = E_0/\langle P_B + [b]Q_B \rangle$를 연산하고, $E_B, [\beta]\phi_B(P_A), [\beta]\phi_B(Q_A)$를 Alice에게 전달한다.

③ 공유된 비밀키 생성

Alice는 Bob에게 받은 공개키로 $e_A(R_A, S_A) = e_A(P_A, Q_A)^B$임을 확인하고, 그러지 않을 경우 종료한다. 그 후 isogeny $\phi_A' : E_B \rightarrow E_{AB} = E_B/\langle R_A + [a]S_A \rangle$ 를 연산한다. Bob도 Alice에게 받은 공개키로 $e_B(R_B, S_B) = e_B(P_B, Q_B)^A$임을 확인하고, 그러지 않을 경우 종료한다. 그 후 isogeny $\phi_B' : E_A \rightarrow E_{BA} = E_A/\langle R_B + [b]S_B \rangle$ 를 연산한다. Alice와 Bob의 공유된 비밀키는 $j(E_{AB}) = j(E_{BA})$이다.

III. M-SIDH 구현

본 장에서는 M-SIDH 구현을 위한 파라미터 선택과, 구현에 필요한 알고리즘에 관해 설명한다. 제안하는 파라미터는 CSIDH 와 비교하여 M-SIDH의 대략적인 성능을 평가하기 위해 고전 64비트 보안강도에 해당하는 파라미터를 선택하였으며, 실제 양자 128비트의 보안강도를 위해서는 적어도 5911비트의 소수 사용이 필요하다.

3.1 파라미터 선택

M-SIDH 구현을 위해 사용한 992비트 소수는 다음과 같다.

$$p = 2^2 \ell_1 \ell_2 \cdots \ell_{127} - 1 \quad (6)$$

여기에서 $\ell_1, \ldots \ell_{126}$ 은 처음 126개의 홀수 소수를 의미하고 $\ell_{127} = 809$에 해당된다. Alice는 $A = 2^2 \cdot \ell_1 \ell_3 \cdots \ell_{127}$를 위수로 사용하고 Bob은 $B = \ell_2 \ell_4 \cdots \ell_{126}$를 위수로 사용한다. 이러한 형태의 소수를 사용하는 이유는 Alice와 Bob의 위수의 크기가 유사하고, 사용하는 isogeny 크기를 맞추기 위함이다. 해당 파라미터가 제공하는 고전 보안강도는 64비트이며, 양자 보안강도는 32비트이다.

위 소수를 이용한 유한체 $F_{p^2} = F_p(i)$위에 정의된 supersingular 몽고메리 곡선은 다음을 사용하였다. 초기 조건의 선택은 SIKE specification을 따랐으며, 임의의 몽고메리 곡선을 사용해도 상관없다.

$$M : y^2 = x^3 + 6x^2 + x \quad (7)$$

F_{p^2}에서 M-SIDH 구현을 위해서는 generator point P_A, Q_A, P_B, Q_B 를 선택해야하는데, 여기에서 $P_A, Q_A \in E[A]$에 해당되고 두 점 모두 위수가 A를 가진다. 또한, $P_B, Q_B \in E[B]$이며, 마찬가지로 두 점 모두 위수가 B에 해당된다. Generator point 선택 방법은 다음과 같다. 먼저, Alice의 경우 양의 정수 $t = 0$에 대해서 $i + t$ 를 타원곡선의 x좌표로 하여 $x^3 + 6x^2 + x$ 의 연산결과가 F_{p^2}에서 제곱근을 가지는지 확인한다. 제곱근을 가지면, 해당 타원곡선 위의 점 P에 대해서 $[B]P$를 연산한 뒤, 위수가 A가 되는지 확인한다. 위수가 A가 아니라면 t를 1씩 증가시키며 위의 과정을 반복한다. Bob의 경우도 동일하게 수행한다. Alice의 generator point P_A를 선택했으면 Q_A도 동일한 방법으로 선택하는데, Weil pairing을 활용하여 P_A와 Q_A가 서로 독립이 되도록 Q_A를 선택한다. Generator point 생성이 완료되면 효율적인 M-SIDH 연산을 위해 $R_A = P_A - Q_A$, $R_B = P_B - Q_B$도 연산해 저장한다.

3.2 F_{p^2}에서의 square-root Velu

M-SIDH는 CSIDH 와 유사하게 큰 홀수 차수 isogeny 연산이 필요하다. 일반적인 Velu의 경우 ℓ 차 isogeny를 연산하는데 $O(\ell)$ 유한체 연산이 필요하다. [8]에서는 이 연산량을 $O(\sqrt{\ell})$에 연산하는 방법을 제안하였으며, isogeny 차수가 커질수록 square-root Velu 방법이 효율적이기 때문에 본 논문에서는 square-root Velu 공식을 M-SIDH에 맞게 확장체에서 구현하였다. [8]에서 제시되어있듯이, 최대 isogeny 차수가 587인 CSIDH-512는 square-root Velu와 기존 Velu 공식과의 유의미한 차이가 나지 않았지만, 그 이상의 차수부터는 유의미한 차이가 있었다. 본 논문의 경우 M-SIDH의 대략적인 성능을 위해 992 비트 소수를 사용하였지만, 향후 양자 128비트 보안강도의 경우 더 큰 소수를 사용함이 확실해짐에 따라 square-root Velu 공식의 적용은 필수적이라 볼 수 있다.

Square-root Velu 공식의 핵심은 커널 P에 대해서 P를 크기가 $\sqrt{P}$와 유사한 작은 집합 I, J로 나누는 것이다. 이 때, I, J는 P의 원소 대부분이 $(I+J) \cup (I-J)$ 안에 존재하도록 선택한다. 이를 이용하여 Velu의 공식에서 $[s]P$를 계산하는 연산을 $[i]P, [j]P$, $[i+j]P$, $[i-j]P$ 의 연산을 활용해 구하는 것으로 변형할 수 있으며, 특히 $[i]P, [j]P$, $[i+j]P$, $[i-j]P$ 의 연산은 타원곡선의 differential addition을 활용하면 효율적으로 계산할 수 있다. 다음 Lemma는 타원곡선의 점 사이의 관계식을 나타내는 이차 다항식의 존재성을 제시한다.

Lemma 1. q를 소수 p에 대해서 $q = p^n$ 형태라 가정하고 $E(F_q)$를 F_q에서 정의된 타원곡선이라 하자. 그러면 모든 $P, Q \in E$에 대해서 다음을 만족하는 이차다항식 F_0, F_1, F_2 가 존재한다.

$$(X - x(P+Q)(X - x(P-Q)) = X^2 + \frac{F_1(x(P), x(Q))}{F_0(x(P), x(Q))} X + \frac{F_2(x(P), x(Q))}{F_0(x(P), x(Q))} \quad (8)$$

위 식에서 $x(P)$는 P의 x좌표를 의미한다. E가 몽고메리 곡선 $y = x^3 + ax^2 + x$일 경우 F_0, F_1, F_2는 다음과 같이 정의된다.

$$\begin{aligned} F_0(X_1, X_2) &= (X_1 - X_2)^2 \\ F_1(X_1, X_2) &= -2((X_1X_2 + 1)(X_1 + X_2) \\ &\quad + 2aX_1X_2) \\ F_2(X_1, X_2) &= (X_1X_2 - 1)^2 \end{aligned} \quad (9)$$

3.2.1 Square-root Velu 공식의 최적화

실제 구현에서는 연산의 효율성을 위해 projective coordinate와 projective curve coefficient를 이용하므로 다항식 F_0, F_1, F_2 도 이에 맞는 형태로 변환해야 한다. 다항식의 입력은 두 점 P, Q 로 여기에서 P는 isogeny 함수값을 계산하고 싶은 점을 의미하고 Q는 커널의 점을 의미한다. P, Q x좌표를 각각 x_P, x_Q 라 하고, $x_P = P_x/P_z$, $x_Q = Q_x/Q_z$라 하자. 또, $a = A/C$라 하자. 위의 식을 변형하면 다음과 같다.

$$\begin{aligned} F_0(P, Q) &= C(P_xQ_z - P_zQ_x)^2 \\ F_1(P, Q) &= -2(C(P_xQ_x + P_zQ_z)(P_xQ_z + P_zQ_x)) \\ &\quad - 4AP_xP_zQ_xQ_z \\ F_2(P, Q) &= C(P_xQ_x - P_zQ_z)^2 \end{aligned} \quad (10)$$

특히, F_0, F_2를 연산할 때 다음을 이용하면 효율적으로 구할 수 있다.

$$\begin{aligned} \alpha &= (P_x + P_z)(Q_x - Q_z) \\ \beta &= (P_x - P_z)(Q_x + Q_z) \end{aligned} \quad (11)$$

이 경우 $\alpha + \beta = 2(P_xQ_x - P_zQ_z)$, $\beta - \alpha = 2(P_xQ_z - P_zQ_x)$ 로, 곱셈 2 번과 제곱연산 2번으로 F_0, F_1을 계산할 수 있다.

한편, image 곡선의 계수를 계산할 때에도 다항식 F_0, F_1, F_2 를 활용할 수 있는데, 이 경우 입력값은 커널과 1, 커널과 -1에 해당된다. 커널과 1이 입력일 경우 연산되는 값은 다음과 같다.

$$\begin{aligned} F_0(1, Q) &= C(Q_z - Q_x)^2 \\ F_1(1, Q) &= -2(C(Q_x + Q_z)^2 + 2AQ_xQ_z) \\ F_2(1, Q) &= C(Q_x - Q_z)^2 \end{aligned} \quad (12)$$

기존 square-root Velu 공식의 적용은 주로 CSIDH 기반 암호로, 이 경우에는 한 값에 대한 isogeny 함수값만 연산하면 된다. 따라서, 함수값을 사용할 때 구하는 F_0, F_1, F_2 값들과, image 곡선의 계수를 구하는 F_0, F_1, F_2 값 사이에는 큰 연관이 없어 순서에 상관을 받지 않는다. 그러나, M-SIDH에는 isogeny evaluation이 총 4번이 일어나게 되는데, 이 때 사용하는 커널은 동일하다. 따라서 커널에 대한 값을 미리 연산하면 전체적인 square-root Velu의 연산량을 줄일 수 있다. 본 논문에서는 먼저 image 곡선의 계수를 구하기 위해 F_0, F_1, F_2 를 연산하는 과정에서 중간값

$$\begin{aligned} &Q_x + Q_z, Q_x - Q_z, 2C(Q_x^2 + Q_z^2), \\ &4AQ_xQ_z, 4CQ_xQ_z \end{aligned}$$

들을 저장하고, 이를 함수값을 계산할 때 사용하여 연산량을 감소했다.

3.2.2 Table look-up을 통한 isogeny 연산 최적화

SIDH 기반 암호에서는 ℓ^e-order를 가지는 generator $R = P + [n]Q$의 점에 대해서 $[\ell^{e-1}]R$ 연산을 통해 위수를 ℓ로 만든 뒤, 이를 커널로 하여 isogeny 연산을 진행하고 $\phi(R)$을 계산한다. $\phi(R)$은 위수가 ℓ^{e-1}로, $[\ell^{e-2}]\phi(R)$을 연산하여 다시 위수를 ℓ로 만든 뒤, 이를 커널로 하여 isogeny 연산을 진행하며, 이 과정을 e번 반복하게 되며 효율을 위해 strategy를 설계하여 연산을 진행한다.

M-SIDH 기반도 이와 유사하지만, 차이점은 generator R의 위수는 소수들의 곱 $\prod p_i$의 형태를 가진다. 소수의 개수를 t개라 하자. R을 커널로 하는 $\prod p_i$-isogeny 연산을 위해 마찬가지로 $\left[\prod_{i=1}^{t-1} p_i\right]R$ 연산을 진행 한 뒤, 위수를 p_t로 만들고, 이를 커널로 하는 isogeny ϕ를 연산하여 $\phi(R)$을 구한다, 그 후,

$\left[\prod_{i=1}^{t-2} p_i\right]\phi(R)$ 연산을 진행하여 위수를 p_{t-1}로 만든 뒤, 이를 커널로 하는 isogeny를 연산하고, 이 과정을 모든 소수를 사용할때까지 반복한다. CSIDH와 달리 M-SIDH는 torsion point를 랜덤하게 선택할 필요가 없기 때문에, 커널로 주어진 R은 항상 full order를 가지고, 따라서 $k=1,\ldots,t-1$에 대한 $\prod_{i=1}^{k} p_i$ 값들을 미리 저장하면 효율적으로 연산을 진행할 수 있다.

3.3 랜덤 마스킹 선택

M-SIDH 에서는 다음과 같은 집합에서 랜덤한 원소를 선택해 scalar multiplication을 통해 마스킹하는 과정이있다.

$$\mu_2(N)=\left\{x \in Z/NZ \mid x^2 \equiv 1 \bmod N\right\} \quad (13)$$

여기에서 $N=\prod p_i$ 형태의 정수이다. N이 서로 다른 소수의 곱으로 이루어져 있기 때문에, $x^2 \equiv 1 \bmod N$을 만족하는 x를 선택하는 것은 다음을 만족하는 x를 선택하는 것과 동치이다.

$$\begin{aligned} &x \equiv 1 \text{or } -1 \bmod p_1 \\ &x \equiv 1 \text{or } -1 \bmod p_2 \\ &\cdots \\ &x \equiv 1 \text{or } -1 \bmod p_n \end{aligned} \quad (14)$$

따라서, $\mu_2(N)$에서 x를 랜덤하게 선택한다는 것의 의미는, 각 p_i에 대해 1인지 -1인지 랜덤하게 선택한 후, CRT (Chinese Remainder Theorem)을 이용해 x를 복원하는 것을 의미한다. 여기에서 x값이 알려지게 되면, 마스킹의 의미가 사라지게 되고 기존 SIDH 기반 암호처럼 torsion point 정보가 그대로 노출된다. 따라서 $x^2 \equiv 1 \bmod N$을 만족하는 x의 개수가 원하는 보안강도 수준만큼 많은 것이 중요하다. 본 논문에서 사용한 MSIDH-1024의 경우 Alice 와 Bob 각각 64개 정도의 소수를 사용하기 때문에 x를 복원하는데 고전 환경에서는 64비트, 양자 환경에서는 32비트의 복잡도를 가진다.

IV. 구현 결과

본 장에서는 Montgomery 곡선을 활용해 M-SIDH를 구현한 결과를 제시한다. 측정에 사용한 CPU는 3.6 GHz의 동작 주파수를 가지는 Intel Core i7-7700을 사용했으며, Ubuntu 20.04 LTS 운영체제상에서 최적화 옵션 -O3와 gcc 9.4.0 컴파일러를 이용하였다.

4.1 실험결과

3장에서 제시한 MSIDH-1024 파라미터를 사용해서 키 교환을 한 결과는 아래 〔Table 1〕, 〔Table 2〕와 같다. 성능 평가를 위해 CSIDH-512와 비교하였으며, M-SIDH가 constant-time으로 구현된 만큼 CSIDH도 Onuki가 제안한 방식의 constant-time 방법을 활용하였다 〔9〕. 표에서의 〔Key exchange〕 는 Alice와 Bob이 키를 교환하는데의 총 시간을 의미하며, 구체적으로는 1) Alice가 자신의 비밀키로 Bob에게 전달해 줄 공개값 연산, 2) Bob이 자신의 비밀키로 Alice에게 전달해 줄 공개값 연산, 3) Alice가 Bob에게 받은 값으로 공유하는 비밀키 연산, 4) Bob이 Alice에게 받은 값으로 공유하는 비밀키 연산의 4 과정에 관한 총 시간을 의미한다.

먼저, 〔Table 1〕은 일반적인 Velu 공식을 활용해서 M-SIDH를 구현할 때와, square-root Velu 공식을 활용해서 M-SIDH를 비교한 표이다. 아래 표에서 확인할 수 있듯이, square-root Velu를 활용할 경우 기존 대비 약 9.3%의 성능 향상이 있다.

〔Table 2〕는 CSIDH-512와 비교하여 MSIDH-1024의 성능을 나타낸 표이다.

위 표에서도 확인할 수 있듯이, CSIDH-512 보다 보안강도가 1/2임에도 불구하고 확장체를 사용하기 때문에 속도가 더 느리다는 것을 확인할 수 있다.

Table 1. Performance results of MSIDH-1024 using standard Velu formula and square-root Velu formula

	MSIDH-1024	MSIDH-1024-sqrt
Classic Seruciry	64	64
Key exchange (cc)	4,471,288,492	4,053,781,156

Table 2. Performance results of MSIDH-1024, CSIDH-512, and CSIDH-1024

	CSIDH-512	MSIDH-1024
Classic Seruciry	128	64
Table Generation (cc)	-	60,374
Key exchange (cc)	1,345,150,556	4,053,781,156
Key exchange (ms)	371.7	1129

M-SIDH에서 속도 저하를 일으키는 가장 큰 원인으로는 큰 홀수 차수의 isogeny를 확장체 위에서 연산하는 것에 있다. [5]에서 제안한 MSIDH-5911을 사용할 경우 추가적인 최적화가 필요할 것으로 예상된다.

V. 결 론

본 논문에서는 Castryck-Decru의 공격에 대응하는 M-SIDH를 C로 구현해서 성능을 확인하였다. Castryck-Decru의 공격의 핵심은 SIDH 기반 암호에서 torsion point 정보를 그대로 노출시켰다는 것을 이용한 것으로, M-SIDH에서는 해당 torsion point 정보를 마스킹을 이용해 공격에 활용하지 못하게 하였으며, 이를 위해 사용하는 유한체 소수의 형태도 CSIDH 와 유사하게 작은 소수들의 곱으로 이루어져 있다.

본 논문의 결과 MSIDH-1024에서 키 교환하는데 1129ms 정도의 시간이 소요되며, MSIDH-1024는 고전 환경에서 64비트의 보안강도를 제공하는 만큼 추가적인 최적화가 필요하다. M-SIDH의 경우 당장은 속도가 느리더라도 추가적으로 최적화할 수 있는 부분은 많으며, 구체적으로 홀수 차수 isogeny 연산을 위한 테이블 최적화와, 홀수 차수 isogeny 연산에 적합한 타원곡선의 형태의 사용 등으로 보인다. 또한, 홀수 차수 isogeny 연산을 진행하는데 있어서 [10]에서 제시된 결과와 같이 strategy를 사용할 경우 최적화가 더 가능할 것으로 보인다. 이와 관련해 향후에 연구를 지속할 예정이다.

References

[1] J.M. Couveignes, "Hard homogenous spaces," IACR Cryptology ePrint Archive, 2006:291, 2006

[2] A. Childs et al. "Constructing elliptic curve isogenies in quantum subexponential time," Journal of Mathematical Cryptology, vol. 8, no. 1, pp. 1-29, 2014

[3] D. Jao, L. De Feo, "Towards quantum-resistant cyprosystems from supersingular elliptic curve isogenies," PQCrypto, LNCS 7071, pp. 19-34, 2011

[4] W. Castryck, T. Decru, "An efficient key recovery attack on SIDH," IACR Cryptology ePrint Archive, 2022:975, 2022

[5] T. Fouotsa et al, "M-SIDH and MD-SIDH: countering SIDh attacks by masking information," Cryptology ePrint Archive, 2023:013, 2023

[6] D. Kohel et al. "On the quarternion-isogeny path problem," LMS Journal of COmputation and Mathematics, pp. 418-432, 2014

[7] T. Fouotsa et al, "On the isogeny problem with torsion point information," PKC 2022, LNCS 13177, pp. 142-161, 2022.

[8] D. Bernstein et al. "Faster computation of isogenies of large prime degree," IACR Cryptology ePrint Archive, 2020:341, 2020

[9] H. Onuki et al. "A constant-time algorithm of CSIDH keeping two points." IEICE Transanctions on Fundamentals of Electronics, Communications, and Computer Sciences. vol. E103A, no. 10, pp. 1174-1182, 2020

[10] J. Chaves-Saab et al. "The SQALE of CSIDH: sublinear Velu quantum-resistant

isogeny action with low exponents," Journal of Cryptographic Engineering, vol. 12, no. 3, pp. 349-368, 2022

〈저자소개〉

김 수 리 (Suhri Kim) 정회원
2014년 2월: 고려대학교 수학과 이학사
2016년 8월: 고려대학교 정보보호대학원 공학석사
2020년 2월: 고려대학교 정보보호대학원 공학박사
2020년 3월~2021년 2월: 고려대학교 정보보호대학원 연구교수
2020년 7월~2021년 2월: KU Leuven ESAT/COSIC 박사후연구원
2021년 3월~현재: 성신여자대학교 수리통계데이터사이언스학부 조교수
〈관심분야〉 공개키 암호시스템, 후양자암호학

서 민 혜 (Minhye Seo) 종신회원
2012년 2월: 고려대학교 수학과 이학사
2020년 2월: 고려대학교 정보보호대학원 공학박사
2020년 3월~2020년 8월: 고려대학교 정보보호대학원 연구교수
2020년 9월~현재: 덕성여자대학교 사이버보안전공 조교수
〈관심분야〉 암호, 인증, 프라이버시