Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.1, Feb. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.1.19

SIDH 기반 암호 구현에 대한 홀수 차수 아이소제니 적용*

김 수 리,[1†] 윤 기 순,[2] 박 영 호[3‡]
[1]고려대학교 (박사후연구원), [2]NSHC (연구소장), [3]세종사이버대학교 (교수)

On the Use of Odd-Degree Isogenies for Implementing SIDH-Based Cryptography*

Suhri Kim,[1†] Kisoon Yoon,[2] Young-Ho Park[3‡]
[1]Korea University (Post-doctoral Resercher), [2]NSHC (Chief Cryptographer),
[3]Sejong Cyber University (Professor)

요 약

본 논문에서는 3차, 5차 아이소제니만 이용해 SIDH를 구현할 경우 몽고메리, 에드워드, 허프 곡선 중 어느 곡선에서 더 효율적인지 분석한다. 본 논문에서는 각 타원곡선의 형태에 대해 SIDH 암호를 구성하는 단위연산에 대한 연산량을 비교한 뒤, 홀수 차수만 활용해 SIDH를 구현하기 위해 소수와 파라미터를 설정하는 방법에 관해 설명한다. 본 논문의 결과 몽고메리와 허프 곡선에서 연산량은 유사하며, 에드워드 곡선보다 0.8% 효율적임을 알 수 있다. SIDH 기반 암호에 대한 다양한 파라미터 사용 가능성으로 인해 5차 아이소제니 구현은 필수적이므로, 본 논문은 이러한 SIDH 기반 암호에 대해 어느 타원곡선을 선택해야 하는지에 대해 가이드라인을 제공할 수 있다.

ABSTRACT

In this paper, when SIDH is instantiated using only 3- and 5-isogeny, we demonstrate which curve is more efficient among the Montgomery, Edwards, and Huff curves. To this end, we present the computational cost of the building blocks of SIDH on Montgomery, Edwards, and Huff curves. We also present the prime we used and parameter settings for implementation. The result of our work shows that the performance of SIDH on Montgomery and Huff curves is almost the same and they are 0.8% faster than Edwards curves. With the possibility of using isogeny of degree other than 3 and 4, the performance of 5-isogeny became even more essential. In this regard, this paper can provide guidelines on the selection of the form of elliptic curves for implementation.

Keywords: Post-quantum cryptography, isogeny-based cryptography, SIDH

I. 서 론

대부분 공개키 암호의 안전성은 수학적 난제에 기반을 두고 있다. 현재 사용하는 RSA, DSA, ECC의 경우 각각 인수분해의 어려움, 이산대수의 어려움, 타원곡선 이산대수 문제에 기반을 두고 있다. 일반적인 컴퓨팅 환경에서 해당 문제들은 하지수시간 및 지수시간의 복잡도를 가지고 있어 안전하나, 만약 양자컴퓨터가 개발되어 Shor 알고리즘이 구현될 경우 해당 문제는 다항시간의 복잡도로 감소하여 더 이상 공개키가 안전하지 않게 된다 [16]. 한편, Shor 알고리즘이 구현 가능한 양자 컴퓨터 개발이 가시화되면서 이에 대응하기 위한 후 양자 암호 (Post-quantum cryptography, PQC)에 관한 연구가 활발히 이루어지고 있

Received(12. 21. 2020), Modified(01. 25. 2021), Accepted(01. 25. 2021)
* 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2017R1A2B4011599)
† 주저자, suhrikim@gmail.com
‡ 교신저자, youngho@sjcu.ac.kr(Corresponding author)

다. PQC란 일반 컴퓨터에서 구현할 수 있지만, 양자 컴퓨팅 환경에서도 안전한 암호를 의미하며, 기반문제에 따라 다변수 기반, 래티스 기반, 코드 기반, 해시 기반, 아이소제니 기반의 5가지로 나눌 수 있다. NIST는 2016년에 PQC 암호에 관한 표준화 공모를 시작한다는 발표를 하였으며, 현재는 Round 3에 진입하였다. Round 3은 7개의 최종 후보와, 8개의 예비 후보 (alternate candidate)로 구성되어있다.

한편, 아이소제니 기반 암호는 2006년 Couveignes [3] 와 Stolbunov에 의해 처음 제안되었으며, 해당 기법은 오늘날 CRS라 칭한다. 하지만 제안한 암호에 대한 하지수시간 공격이 존재할 뿐만 아니라, 현실적으로 사용하기에는 속도가 매우 느려서 더 이상 연구되지 않았다. 2011년 De Feo와 Jao 에 의해서 Supersingular Isogeny Diffie-Hellman (SIDH)가 소개되면서, 아이소제니 기반 암호는 다시 주목을 받기 시작하였다 [10]. SIDH는 supersingular 타원곡선의 endomorphism ring이 가환이 아니라는 성질을 사용하여서, CRS 스킴 공격에 사용되었던 공격 기법이 적용되지 않아 가장 효율적인 양자 공격 기법이 지수시간 복잡도를 가지게 되어 안전하다. 또한, 기존 CRS 기반 암호보다 효율적이고 다른 PQC 암호들보다 키 사이즈가 작다는 점으로 SIDH 기반으로 설계된 key encapsulation mechanism 인 SIKE는 현재 Round 3의 예비 후보로 선정되었다 [1].

아이소제니 기반 암호의 구현은 크게 아이소제니 함수 구현과 타원곡선 연산 구현으로 나뉠 수 있다. 아이소제니 함수 구현 측면에서 바라보면, 해당 암호에서 사용하는 아이소제니 차수는 암호가 구현되는 유한체와 연관이 있다. SIDH 기반 암호의 경우 $p=\ell_A^{e_A}\ell_B^{e_B}f\pm 1$ 형태의 소수를 사용하고, ℓ_A,ℓ_B가 사용되는 아이소제니 차수에 해당한다. 타원곡선 연산 측면에서 살펴보자면, 아이소제니 기반 암호의 기존 타원곡선 암호와 달리 고정된 하나의 타원곡선을 사용하지 않기 때문에, 타원곡선 연산을 타원곡선의 특성에 맞춰 최적화할 수 없다. 따라서 임의의 타원곡선에서 가장 효율적인 타원곡선 연산을 제공해주는 타원곡선의 형태를 사용해야 한다. 종합해보자면, 아이소제니 기반 암호를 구현할 때에는, 해당 알고리즘에 사용되는 아이소제니 차수의 구현이 효율적이면서 타원곡선 연산을 효율적으로 제공해주는 타원곡선의 형태를 사용해야 한다. 현재는 몽고메리 타원곡선을 이용해 대부분 아이소제니 기반 암호가 구현되어있다. 또한, SIDH 에 대한 state-of-the-art 구현도 몽고메리 타원곡선을 이용하고 있다.

하지만 최근에는 몽고메리 타원곡선보다 효율적인 다른 타원곡선의 형태가 존재하지 않을까에 관한 연구가 활발히 진행되고 있다. 첫 번째 후보로는 에드워드 곡선으로, 에드워드 곡선과 몽고메리 곡선이 birationally equivalent 하다는 점과, 두 타원곡선 사이의 변환이 효율적이라는 점으로 많은 연구가 진행되었다. 에드워드 곡선을 아이소제니 기반 암호의 사용은 Meyer 등에 의해 처음으로 제안되었다 [13]. Meyer 등은 [13]에서 SIDH 구현 시 아이소제니 연산은 몽고메리 곡선으로, 타원곡선 연산은 에드워드 곡선으로 구현하는 방법을 제안하였다. 이 후, [12]에서는 아이소제니 연산이 에드워드 곡선에서 더 효율적이라는 점에 착안해, 타원곡선 연산은 몽고메리 곡선에서, 아이소제니 연산은 에드워드 곡선에서 수행하는 방법을 제안하였다. 특히 [11]에서 제안된 에드워드에서 홀수 차수 연산 공식이 몽고메리와 유사할 뿐만 아니라, 이미지 곡선의 계수를 복원하는 연산이 몽고메리보다 더 효율적이어서 에드워드 곡선은 더 주목받고 있다.

한편, 최근에는 허프 곡선을 이용한 연구가 활발히 진행되고 있다. 허프 곡선에서의 아이소제니는 [15]에서 처음으로 제안되었다. 하지만 타원곡선 연산과 아이소제니 연산이 비효율적인 관계로 주목받지 못하다가 최근 [6]와 [8]에서 효율적인 타원곡선 및 아이소제니 연산을 위한 좌표계가 제안되면서 다시 주목받고 있다. 특히, 해당 좌표계를 이용하면 3차와 4차 아이소제니 공식의 연산량이 몽고메리 곡선과 같아서 허프 곡선을 이용해 효율적인 SIDH 구현이 가능하다는 것을 보여준다 [8].

본 논문에서는 홀수 차수 아이소제니만 이용하여 SIDH를 몽고메리, 에드워드, 허프 곡선에서 구현한 결과를 제시한다. 일반적으로 SIDH는 3차 4차 아이소제니만 이용하는 파라미터만 제시되었다. 하지만 아이소제니 기반 그룹 키 합의를 구성할 때 [2] 사용되는 소수의 형태는 $p=\ell_A^{e_A}\ell_B^{e_B}\ell_C^{e_C}f\pm 1$로 서로 다른 세 개의 소수를 사용하므로 5차 아이소제니가 필요하며, BSIDH [5]를 구성할 때에는 작은 차수의 아이소제니를 적용 가능할수록 효율적이기 때문에, 낮은 홀수 차수 아이소제니 연산에 대한 효율성 분석도 필요하다. 따라서 본 논문에서는 홀수 차수 아이소제니 연산을 사용했을 때, 어느 형태의 타원곡선이 효율적인지 분석한다. 본 논문의 분석결과 몽고메리와 허프 곡선에서 연

산량은 거의 유사하며, 몽고메리와 허프 곡선이 에드워드 곡선보다 0.8% 빠르다는 것을 알 수 있다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 본 논문 구현 시 사용될 프로토콜인 SIDH에 대해 소개하고, 본 논문에서 사용될 타원곡선 형태에 대해 알아본다. 3장에서는 2장에서 소개된 타원곡선에 대해 SIDH를 구성하는 각 함수의 연산량을 분석한다. 4장에서는 홀수 차수 아이소제니만 사용한 SIDH 구현을 위한 파라미터 세팅 및 구현결과를 제시하고, 5장에서 본 논문을 마무리한다.

II. 배경 지식

본 장에서는 구현에서 사용될 타원곡선에 대한 소개를 간략하게 한 뒤, SIDH 프로토콜에 관해 설명한다.

2.1 타원곡선의 종류

2.1.1 몽고메리 곡선과 그 연산

유한체 K위에서의 몽고메리 곡선은 다음과 같이 정의된다.

$$M_{a,b}: by^2 = x^3 + ax^2 + x. \quad (1)$$

위 식에서 $b(a^2-4) \neq 0$을 만족한다. 특히, $b=1$일 경우 간단히 M_a로 나타낸다. SIDH에 사용되는 유한체에서는 몽고메리 곡선 $M_{a,b}$는 $b=1$인 몽고메리 곡선 M_a로 변환이 가능하므로, 본 논문에서는 M_a인 형태를 사용한다.

Velu의 공식을 이용해서 아이소제니 연산을 수행할 때에는 projective coordinate와 projective curve coefficient를 사용하는 것이 더 효율적으로 연산할 수 있다. $C \in \overline{K}^{\times}$에 대하여 $a=A/C$, $b=B/C$라 하면, 식 (1)은 다음과 같이 표현할 수 있다.

$$M_{A:B:C}: By^2 = Cx^3 + Ax^2 + Cx.$$

Projective coordinate의 경우 다양한 방법이 제안되어있으나, 몽고메리 곡선은 x좌표만 이용해 효율적인 타원곡선 연산이 가능하다는 특징이 있으며, 이에 대한 projective coordinate를 XZ-coordinate라 정의한다 [14]. 다시 말해, 몽고메리 곡선 M_a위의 점 $P=(x_1,y_1)$에 대해 함수 $x(\bullet)$를 점에 대한 x좌표를 의미한다. 위의 경우 $x(P)=x_1$을 의미하며, P를 XZ-coordinate로 변환하면 $P=(X_1:Z_1)$으로 표현할 수 있다. 이때, $x_1=X_1/Z_1$을 만족한다.

2.1.2 에드워드 곡선과 그 연산

유한체 K위에서의 에드워드 곡선은 다음과 같이 정의된다.

$$E_d: x^2 + y^2 = 1 + dx^2y^2. \quad (2)$$

위 식에서 $d \neq 0,1$ 이다. 에드워드 곡선에서의 항등원은 $(0,1)$이며 점 $(0,-1)$은 위수가 2인 점이다. 또한, $(1,0),(-1,0)$은 위수가 4인 점이다. E_d위의 두 점 $P=(x_1,y_1), Q=(x_2,y_2)$에 대해 $P+Q$는 다음과 같이 연산한다.

$$P+Q=\left(\frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \frac{y_1y_2-x_1x_2}{1-dx_1x_2y_1y_2}\right).$$

타원곡선 점 P에 대한 doubling 연산 $2P$도 같은 공식을 이용해 구할 수 있다. 마찬가지로 에드워드 곡선도 projective curve coefficient를 이용해 표현한다. $C \in \overline{K}^{\times}$에 대하여 $d=D/C$라 하면, 식 (2)는 다음과 같이 표현할 수 있다.

$$E_{C:D}: Cx^2 + Cy^2 = C + Dx^2y^2.$$

에드워드 곡선에서의 projective coordinate로는 [7]에서 제안된 w-coordinate를 사용한다. [7]에서는 에드워드 위의 점 $P=(x,y)$에 대하여 $w(P)=x^2y^2$로 정의한다. 따라서 P를 WZ-coordinate로 변환하면 $P=(W:Z)$로 표현할 수 있다.

2.1.3 허프 곡선과 그 연산

허프 곡선은 Joye, Tibouchi, Vergnaud에 의해서 [9]에서 처음으로 제안되었다. 유한체 K위에서

의 허프 곡선은 다음과 같이 정의된다.

$$H_{a,b}: ax(y^2-1)=by(x^2-1).$$

위 식에서 $a^2 \neq b^2$이고 $a,b \neq 0$이다. 허프 곡선에서 항등원은 $(0,0)$이다. 위의 허프 곡선 $H_{a,b}$는 다음과 같이 간단히 나타낼 수 있다.

$$H_c: cx(y^2-1)=y(x^2-1). \quad (3)$$

이 때, $c=a/b$이고 $c \neq 1$을 만족한다. $H_{a,b}$위의 두 점 $P=(x_1,y_1), Q=(x_2,y_2)$에 대해 $P+Q$는 다음과 같이 연산한다.

$$P+Q = \left(\frac{(x_1+x_2)(1+y_1y_2)}{(1+x_1x_2)(1-y_1y_2)}, \frac{(y_1+y_2)(1+x_1x_2)}{(1-x_1x_2)(1+y_1y_2)} \right).$$

에드워드 곡선과 마찬가지로 타원곡선 점 P에 대한 doubling 연산 $2P$도 같은 공식을 이용해 구할 수 있다.

허프 곡선도 projective curve coefficient를 이용해 표현한다. $C \in \overline{K}^{\times}$에 대하여 $C=C/D$라 하면, 식 (3)은 다음과 같이 표현할 수 있다.

$$H_{C:D}: Cx(y^2-1)=Dy(x^2-1).$$

허프 곡선에서의 projective coordinate로는 [8]에서 제안된 w-coordinate를 사용한다. [8]에서는 허프 곡선 위의 점 $P=(x,y)$에 대하여 $w(P)=1/xy$로 정의한다. 따라서 P를 WZ-coordinate로 변환하면 $P=(W:Z)$로 표현할 수 있다.

2.2 SIDH

서로 소인 두 수 ℓ_A, ℓ_B에 대해 $p=\ell_A^{e_A}\ell_B^{e_B}f \pm 1$ 형태의 소수를 생성한다. 이때, $\ell_A^{e_A} \cong \ell_B^{e_B}$가 되도록 e_A, e_B를 선택한다. 이 소수를 사용해 형성한 유한체 F_{p^2}위에서 위수가 $(\ell_A^{e_A}\ell_B^{e_B}f)^2$인 supersingular 타원곡선 E를 선택한다. 이 경우, $\ell \in \{\ell_A,\ell_B\}$와 $e \in \{e_A,e_B\}$에 대해 ℓ^e-torsion subgroup이 생성되며, 기저 $\{P_A,Q_A\}$와 $\{P_B,Q_B\}$를 각각 $\ell_A^{e_A}$와 $\ell_B^{e_B}$ torsion subgroup에서 선택한다.

Alice와 Bob이 서로 비밀 공유키를 교환한다고 가정하자. $\{P_A,Q_A\}$를 Alice의 기저라 하고 $\{P_B,Q_B\}$를 Bob의 기저라 하자. 키 생성 단계에서, Alice는 동시에 ℓ_A로 나누어지지 않는 수 m_A, n_A를 $Z/\ell_A^{e_A}Z$에서 선택하고 부분군 $\langle R_A \rangle = \langle [m_A]P_A+[n_A]Q_A \rangle$를 계산한다. 그 뒤 Velu의 공식을 이용해서 $\langle R_A \rangle$를 커널로 하는 아이소제니 $\phi_A: E \rightarrow E_A$를 연산한다. 여기에서 $E_A=E/\langle R_A \rangle$를 만족한다. Alice는 이 아이소제니를 이용해서 연산한 $(E_A, \phi_A(P_B), \phi_B(Q_B))$를 Bob에게 전달한다. Bob도 비슷한 과정을 통해 Alice에게 $(E_B, \phi_B(P_A), \phi_B(Q_A))$를 연산해 전달한다.

Share secret key를 연산하는 과정에서는, Alice는 Bob에게 받은 점을 이용하여 부분그룹 $\langle R_A' \rangle = \langle [m_A]\phi_B(P_A)+[n_A]\phi_B(Q_A) \rangle$를 연산한 뒤, Velu의 공식을 이용하여 $\langle R_A' \rangle$를 커널로 하는 아이소제니로 생성된 타원곡선 $E_{AB}=E_B/\langle R_A' \rangle$를 연산한다. Bob도 Alice와 비슷한 과정을 통해 $E_{BA}=E_A/\langle R_B' \rangle$를 생성한다. Alice와 Bob 사이의 비밀 공유키는 생성된 타원곡선의 j-invariant인 $j(E_{AB})=j(E_{BA})$이다.

III. Building blocks for SIDH

본 장에서는 SIDH를 구성하는 각 단위연산에 대해 아이소제니 구현 환경에서의 연산량을 알아본다. 다시 말해, projective coordinate와 projective curve coefficient를 사용했을 때의 연산량을 비교하며, 몽고메리의 경우 XZ-coordinate, 에드워드와 허프 곡선의 경우 WZ-coordinate를 사용한다. 또한, 본 논문에서는 홀수인 소수 – 구체적으로는 3, 5 – 만 사용한 구현이 목표이기 때문에, doubling 연산, differential addition, 3차, 5차 아이소제니 연산에 대해 알아본다. 이 장 이후로 $\boldsymbol{M}$은 유한체 위에서의 곱셈연산, $\boldsymbol{S}$는 유한체 위에서의 제곱연산 $\boldsymbol{I}$는 유한체 위에서의 역원연산을 의미한다.

3.1 몽고메리 곡선

3.1.1 타원곡선 연산

- Doubling: 몽고메리 곡선 M_a에서의 점 $P=(X:Z)$에 대해 doubling $[2]P=(X':Z')$는 다음과 같이 연산된다.

$$X' = C(X+Z)^2(X-Z)^2,$$
$$Z' = 4XZ(C(X+Z)^2+(A-2C)XZ).$$

Doubling의 연산량은 $4\boldsymbol{M}+2\boldsymbol{S}$이다

- Differential addition: 몽고메리 곡선 위의 두 점 $P_1=(X_1:Z_1)$ $P_2=(X_2:Z_2)$에 대해 $x_0=x(P_1-P_2)$, $x_3=x(P_1+P_2)$라하고, $x_0=X_0/Z_0$, $x_3=X_3/Z_3$라 하면, 다음과 같이 계산할 수 있다.

$$X_3 = Z_0(X_1X_2-Z_1Z_2)^2,$$
$$Z_3 = X_0(X_1Z_2-Z_1X_2)^2.$$

Doubling과 differential addition을 동시에 수행하는데 연산량은 $6\boldsymbol{M}+4\boldsymbol{S}$이다.

3.1.2 아이소제니 연산

- 3-isogeny: 몽고메리 곡선 M_a위의 위수가 3인 점을 $P=(X_3:Z_3)$라 하자. $\langle P\rangle$를 커널로 하는 3-isogeny ϕ를 $\phi:M_a\rightarrow M_{a'}=M_a/\langle P\rangle$라 정의하면, M_a의 다른 점 $Q=(X:Z)$에 대해서 $x(\phi(Q))=(X':Z')$는 다음과 같다.

$$X' = X(XX_3-ZZ_3)^2,$$
$$Z' = Z(XZ_3-ZX_3)^2.$$

이때 타원곡선의 계수는 이후에 진행되는 tripling 연산을 효율적으로 하기 위해 a대신에 $(a-2)/4$를 연산하며, 이에 대한 projective coefficient $(A_{24}:C_{24})=(A-2C:4C)$는 다음과 같이 연산한다.

$$A_{24} = (X_3+Z_3)(Z_3-3X_3)^3,$$
$$C_{24} = 16X_3Z_3^3.$$

3-isogeny의 함수값 $(X':Z')$을 계산하는 과정의 연산량은 4M+2S이며, 이미지 타원곡선의 계수 $(A_{24}:C_{24})$를 계산하는 과정은 2M+3S의 연산량이 필요하다.

- ℓ-isogeny: 몽고메리 곡선 M_a위의 위수가 $\ell=2s+1$인 점을 $P=(W_1:Z_1)$라하고 $[i]P=(X_i:Z_i)$라 하자. $\langle P\rangle$를 커널로 하는 ℓ-isogeny ϕ를 $\phi:M_a\rightarrow M_{a'}=M_a/\langle P\rangle$라 정의하면, M_a의 다른 점 $Q=(X:Z)$에 대해서 $x(\phi(Q))=(X':Z')$는 다음과 같다.

$$X' = X\prod_{i=1}^{s}(XX_i-ZZ_i)^2,$$
$$Z' = Z\prod_{i=1}^{s}(XZ_i-ZX_i)^2.$$

이 때 a'는 다음과 같이 정의한다.

$$a' = (6\tilde{\sigma}-6\sigma+a)\pi^2 \quad (4)$$

위 식에서

$$\sigma=\sum_{i=1}^{s}x(P_i),\ \tilde{\sigma}=\sum_{i=1}^{s}1/x(P_i),\ \pi=\prod_{i=1}^{s}x(P_i).$$

로 정의한다.
ℓ-isogeny의 함수값 $(W':Z')$을 계산하는 과정의 연산량은 4s$\boldsymbol{M}+2\boldsymbol{S}$이며, 이미지 타원곡선의 계수 a'에 대해서 $a'=A'/C'$라 할 때, $(A':C')$를 계산하는 과정은 (6s-2)$\boldsymbol{M}+3\boldsymbol{S}$의 연산량이 필요하다.

3.2 에드워드 곡선

3.2.1 타원곡선 연산

- Doubling: 에드워드 곡선 E_d에서의 점 $P=(W:Z)$에 대해 doubling

$[2]P=(W':Z')$는 다음과 같이 연산된다.

$$W' = 4WZ(D(W+Z)^2 - 4CWZ),$$
$$Z' = D(W+Z)^2(W-Z)^2.$$

Doubling의 연산량은 $4\mathbf{M}+2\mathbf{S}$이다

- Differential addition: 에드워드 곡선 위의 두 점 $P_1=(W_1:Z_1)$ $P_2=(W_2:Z_2)$에 대해 $w_0=w(P_1-P_2)$, $w_3=w(P_1+P_2)$라하고, $w_0=W_0/Z_0$, $w_3=W_3/Z_3$라 하면, 다음과 같이 계산할 수 있다.

$$W_3 = Z_0(W_1Z_2 - W_2Z_1)^2,$$
$$Z_3 = W_0(W_1W_2 - Z_1Z_2)^2.$$

Doubling과 differential addition을 동시에 수행하는데 연산량은 $6\mathbf{M}+4\mathbf{S}$이다.

3.2.2 아이소제니 연산

- 3-isogeny: 에드워드 곡선 E_d위의 위수가 3인 점을 $P=(W_3:Z_3)$라 하자. $\langle P\rangle$를 커널로 하는 3-isogeny ϕ를 $\phi: E_d \to E_{d'}=E_d/\langle P\rangle$라 정의하면, E_d의 다른 점 $Q=(W:Z)$에 대해서 $w(\phi(Q))=(W':Z')$는 다음과 같다.

$$W' = W(WZ_3 - ZW_3)^2,$$
$$Z' = Z(WW_3 - ZZ_3)^2.$$

이때, $d'=D'/C'$는 다음과 같이 정의한다.

$$D' = -16W_3^2Z_3,$$
$$C' = (W_3^2 - 2W_3Z_3 - 3Z_3^2)(W_3^2 - 6W_3Z_3 + 9Z_3)^2.$$

3-isogeny의 함수값 $(W':Z')$을 계산하는 과정의 연산량은 $4\mathbf{M}+2\mathbf{S}$이며, 이미지 타원곡선의 계수 $(D':C')$를 계산하는 과정은 $2\mathbf{M}+3\mathbf{S}$의 연산량이 필요하다.

- ℓ-isogeny: 에드워드 곡선 E_d위의 위수가 $\ell=2s+1$인 점을 $P=(W_1:Z_1)$라 하고 $[i]P=(W_i:Z_i)$라 하자. $\langle P\rangle$를 커널로 하는 ℓ-isogeny ϕ를 $\phi: E_d \to E_{d'}=E_d/\langle P\rangle$라 정의하면, E_d의 다른 점 $Q=(W:Z)$에 대해서 $w(\phi(Q))=(W':Z')$는 다음과 같다.

$$W' = W\prod_{i=1}^{s}(WZ_i - ZW_i)^2,$$
$$Z' = Z\prod_{i=1}^{s}(WW_i - ZZ_i)^2.$$

이때 $d'=D'/C'$는 다음과 같이 정의한다.

$$D' = D^{\ell}\prod_{i=1}^{s}(W_i+Z_i)^8,$$
$$C' = C^{\ell}\prod_{i=1}^{s}(2Z_i)^8.$$

ℓ-isogeny의 함수값 $(W':Z')$을 계산하는 과정의 연산량은 $4s\mathbf{M}+2\mathbf{S}$이며, 이미지 타원곡선의 계수$(C':D')$를 계산하는 과정은 $2s\mathbf{M}+6\mathbf{S}+2w(\ell)$의 연산량이 필요하다. 여기에서 $w(\ell)=(h-1)M+(t-1)S$를 의미하며, h는 ℓ의 hamming weight를 의미하고 t는 ℓ의 비트길이를 의미한다.

3.3 허프 곡선

3.3.1 타원곡선 연산

- Doubling: 허프 곡선 H_c에서의 점 $P=(W:Z)$에 대해 doubling $[2]P=(W':Z')$는 다음과 같이 연산된다.

$$W' = \hat{D}(W+Z)^2(W-Z)^2,$$
$$Z' = 4WZ(\hat{D}(W+Z)^2 + \hat{C}\bullet 4WZ).$$

위 식에서 $\hat{c}=\hat{C}/\hat{D}$이고, $\hat{c}=\frac{1}{4}\left(c+\frac{1}{c}-2\right)$를 만족한다. $\hat{C},\hat{D}$가 주어졌을 때, Doubling의 연산량은 $4\mathbf{M}+2\mathbf{S}$이다

- Differential addition: 허프 곡선 위의 두 점 $P_1=(W_1:Z_1)$ $P_2=(W_2:Z_2)$에 대해 $w_0=w(P_1-P_2)$, $w_3=w(P_1+P_2)$라 하고, $w_0=W_0/Z_0$, $w_3=W_3/Z_3$라 하면, 다음과 같이 계산할 수 있다.

$$W_3=Z_0(W_1W_2-Z_1Z_2)^2,$$
$$Z_3=W_0(W_1Z_2-Z_1W_2)^2.$$

Doubling과 differential addition을 동시에 수행하는데 연산량은 $6\boldsymbol{M}+4\boldsymbol{S}$이다.

3.3.2 아이소제니 연산

- 3-isogeny: 허프 곡선 H_c위의 위수가 3인 점을 $P=(W_3:Z_3)$라 하자. $\langle P\rangle$를 커널로 하는 3-isogeny ϕ를 $\phi:H_c\rightarrow H_{c'}=H_c/\langle P\rangle$라 정의하면, H_c의 다른 점 $Q=(W:Z)$에 대해서 $w(\phi(Q))=(W':Z')$는 다음과 같다.

$$W'=W(WW_3-ZZ_3)^2,$$
$$Z'=Z(WZ_3-ZW_3)^2.$$

이때, $c'=C'/D'$는 다음과 같이 정의한다.

$$C'=C(DZ_3+CW_3)^2,$$
$$D'=D(CZ_3+DW_3)^2.$$

3-isogeny 연산 후 얻은 타원곡선에서 tripling 연산을 쉽게 하도록 c'대신에 $\hat{c}=\frac{1}{4}\left(c'+\frac{1}{c'}-2\right)=\hat{C}/\hat{D}$를 연산하는 것이 더 효율적이며, 이때 $\hat{C},\hat{D}$는 다음과 같이 연산한다.

$$\hat{C}=(W_3+Z_3)(Z_3-3W_3)^3,$$
$$\hat{D}=16W_3Z_3^3.$$

3-isogeny의 함수값 $(W':Z')$을 계산하는 과정의 연산량은 $4\boldsymbol{M}+2\boldsymbol{S}$이며, 이미지 타원곡선의 계수 $(\hat{C}:\hat{D})$를 계산하는 과정은 $2\boldsymbol{M}+3\boldsymbol{S}$의 연산량이 필요하다.

- ℓ-isogeny: 허프 곡선 H_c위의 위수가 $\ell=2s+1$인 점을 $P=(W_1:Z_1)$라하고 $[i]P=(W_i:Z_i)$라 하자. $\langle P\rangle$를 커널로 하는 ℓ-isogeny ϕ를 $\phi:H_c\rightarrow H_{c'}=H_c/\langle P\rangle$라 정의하면, H_c의 다른 점 $Q=(W:Z)$에 대해서 $w(\phi(Q))=(W':Z')$는 다음과 같다.

$$W'=W\prod_{i=1}^{s}(WW_i-ZZ_i)^2,$$
$$Z'=Z\prod_{i=1}^{s}(WZ_i-ZW_i)^2.$$

이때, $c'=C'/D'$는 다음과 같이 정의한다.

$$C'=C\prod_{i=1}^{s}(DZ_i+CW_i)^2,$$
$$D'=D\prod_{i=1}^{s}(CZ_i+DW_i)^2.$$

ℓ-isogeny의 함수값 $(W':Z')$을 계산하는 과정의 연산량은 $4s\boldsymbol{M}+2\boldsymbol{S}$이며, 이미지 타원곡선의 계수$(C':D')$를 계산하는 과정은 $4s\boldsymbol{M}+2\boldsymbol{S}$의 연산량이 필요하다.

Remark 1. ℓ-isogeny도 마찬가지로 이미지 타원곡선 위에서 효율적인 타원곡선 연산을 수행하기 위해서는 c'가 아닌 $\hat{c}=\frac{1}{4}\left(c'+\frac{1}{c'}-2\right)$를 구하는 것이 더 편리하다. 하지만 3-isogeny 공식처럼 타원곡선의 계수 구하는 공식을 커널의 원소로만 표현하기 위해서는 division polynomial이 필요하며, 5차 이상 아이소제니에 대해서 이를 이용해 식을 일반화하기란 어렵다. 따라서 일반적으로 5차 이상 아이소제니 연산을 수행할 때에는 c'를 계산한 뒤에, $\hat{c}$로 변환하는 과정을 거쳐야 한다. 다시 말해 $c'=C'/D'$에 대해 $\hat{C}=(C'-D')^2$, $\hat{D}=4C'D'$를 구하는 추가적인 연산이 필요하며, 이 연산은 $2\boldsymbol{S}$가 필요하다.

다음 Table 1.은 몽고메리, 에드워드, 허프 곡선 위에서 SIDH를 구성하는 함수들의 연산량을 비교한 표이다. 아래 표에서 **Mont.**는 몽고메리 곡선을, **Edwards**는 에드워드 곡선, **Huff**는 허프 곡선을

의미한다. **DBLADD**는 doubling 연산과 differential addition 연산을 의미하고 **DBL**은 doubling 연산을 의미한다. **3-isogeny**는 3차 아이소제니 함수값을 계산하는 연산과 타원곡선 계수를 계산하는 연산을 합한 연산량을 의미한다. 또한, **ℓ-eval**은 ℓ차 아이소제니 함수값을 계산하는 연산량을 의미하며, **ℓ-coeff**은 ℓ차 아이소제니의 이미지 곡선의 계수를 복원하는 연산량을 의미한다. **coeffTrans**는 타원곡선의 계수를 타원곡선 연산하기 쉬운 형태로 변환하는 과정을 의미하며, 이는 허프 곡선에서만 사용됨을 알 수 있다. 특히, 본 논문에서는 3차, 5차 아이소제니를 활용 해 구현을 할 것이므로, 5차 아이소제니에서의 함수값을 계산하는 연산량과 타원곡선의 계수를 복원하는 연산량을 비교하였다.

Table 1. Computational cost of the building blocks of SIDH

	Mont.	Edwards	Huff
DBLADD	$6M+4S$	$6M+4S$	$6M+4S$
DBL	$4M+2S$	$4M+2S$	$4M+2S$
3-isogeny	$6M+5S$	$6M+5S$	$6M+5S$
5-eval	$8M+2S$	$8M+2S$	$8M+2S$
5-coeff	$10M+3S$	$6M+10S$	$8M+2S$
ℓ-eval	$4sM+2S$	$4sM+2S$	$4sM+2S$
ℓ-coeff	$(6s-2)M$ $+3S$	$2sM+6S$ $+2w(\ell)$	$4sM+2S$
coeffTrans	-	-	$2S$
j-invarant	$3M+4S+1I$	$3M+4S+1I$	$3M+4S+1I$

위의 표에서 확인할 수 있듯이, 몽고메리, 에드워드, 허프 곡선의 타원곡선 연산량 및 3차 아이소제니 연산량이 같음을 알 수 있다. 하지만 아이소제니 차수가 올라갈수록 몽고메리보다 에드워드나 허프 곡선이 더 효율적임을 알 수 있다. 다음 장에서는 추가적인 최적화 기법을 사용해서, 3차 5차 아이소제니를 이용해 SIDH를 구현하는 방법과 구현결과를 제시한다.

IV. 구현결과

본 장에서는 3차, 5차 아이소제니를 활용한 SIDH를 구현결과를 제시하였는데, 해당 차수를 선정한 이유는 다음과 같다.

- 에드워드 곡선의 w-coordinate의 경우 4차 compression 함수로, 효율적인 4차 아이소제니를 구성하기가 힘들다. 허프 곡선의 경우 3차, 4차 아이소제니를 사용한 SIDH 구현은 이미 [8]에 제시되어있다.
- 따라서 현재 아이소제니 기반 암호 구현에 언급되고 있는 몽고메리, 에드워드, 허프 곡선을 동일 선상에서 비교하기 위해서는 홀수 차수 아이소제니를 활용하는 것이 좋다.
- 특히 허프 곡선의 경우 5차 아이소제니 연산 시 타원곡선 계수 변환이 중요한 점으로 작용할 수 있으며, 몽고메리 곡선의 경우 SIDH 구현에서는 효율적인 타원곡선 계수 변환 방법이 존재하기 때문에, 이를 적용해 구현 성능을 알아보는데 5차 아이소제니가 적합하다고 볼 수 있다.

따라서 본 장에서는 홀수 차수 아이소제니를 이용한 SIDH를 구현하기 위해 설정한 파라미터와 추가적인 구현 기법, 그리고 구현결과를 제시한다.

4.1 파라미터 설정

홀수 차수 아이소제니를 이용한 SIDH 구현을 위해 사용한 621-비트 소수는 다음과 같다.

$$p = 2^{67}3^{175}5^{119} - 1.$$

위의 소수에 대해 $3^{175} \cong 2^{277.368}$이며 $5^{119} \cong 2^{276.309}$로, SIDH 프로토콜이 요구하는 $\ell_A^{e_A} \cong \ell_B^{e_B}$를 만족시킴을 알 수 있다. 해당 소수를 사용할 경우 고전 컴퓨팅 환경에서 138비트의 보안강도를 가진다. 위의 소수를 이용한 유한체 $F_{p^2} = F_p(i)$ 위에 정의된 다음 몽고메리, 에드워드, 허프 곡선을 이용하였다.

$$M: y^2 = x^3 + x,$$
$$E_{-1}: x^2 + y = 1 - x^2y^2,$$
$$H_i: ix(y^2-1) = y(x^2-1).$$

F_{p^2}에서 SIDH 구현을 위해서는 generator point P_A, Q_A, P_B, Q_B의 선택이 필요하다. 이때,

$P_A, Q_A \in E[3^{175}]$에 있으며, 두 점 모두 위수가 3^{175}이다. 또한, $P_B, Q_B \in E[5^{119}]$이며 두 점 모두 위수가 5^{119}이다. 해당 점을 선택하는 방법은 SIKE specification [1]에 나와 있는 방법을 따랐으며, 몽고메리 곡선에서 generator point를 선택하고, 해당 점을 에드워드 곡선으로 이동해서 w-coordinate로 변환해서 사용하였다. 허프 곡선의 경우 다음 식과 같은 동형인 Weierstrass curve를 사용하였다.

$$W: y^2 = x^3 - x.$$

위의 곡선에서 generator point를 설정한 이후 허프 곡선 위의 점으로 변환하여서 사용하였다.

4.2 몽고메리 타원곡선 계수 복원 기법

에드워드와 허프 곡선의 경우 타원곡선 계수를 계산하는 방법이 비교적 최적화 되어있으나, 몽고메리 곡선의 경우 5차 이상의 아이소제니의 경우 타원곡선 계수를 복원하는 방법이 다소 복잡하다. 본 논문에서는 5-isogeny 구현 시, [4]에서 제시된 2-torsion을 이용하는 방법으로 타원곡선의 계수를 변경하였고 이 방법에 대해 상세히 설명한다.

몽고메리 곡선 M_a는 항상 다음과 같은 형태의 위수가 2인 점 (2-torsion point)를 가진다.

$$P_0 = (0,0), P_\alpha(\alpha,0), P_{1/\alpha}(1/\alpha,0).$$

이를 projective XZ-coordinate로 변환하면 다음과 같이 표현된다.

$$P_0 = (0:1), P_\alpha = (X_a : Z_a), P_{1/\alpha} = (Z_a : X_a).$$

또한, P_α는 M_a위의 점이므로, $a = -(\alpha^2+1)/\alpha$를 만족한다. 몽고메리 곡선에서는 연산의 편의를 위해 a대신 $(a-2)/4$를 사용하는데, 이를 다시 projective coordinate로 표현하면

$$(A-2C:4C) = ((X_\alpha + Z_\alpha)^2 : -4X_\alpha Z_\alpha).$$

로 나타낼 수 있으며, 해당 연산은 2S가 소요된다. 다시 말해 2-torsion point P에 대해서 $\phi(P)$가 주어지면, 이를 이용하여 이미지 곡선의 계수를 변경할 수 있음을 알 수 있다. 특히, 본 논문에서는 홀수 차수 아이소제니를 사용하기 때문에 2-torsion point P에 대한 아이소제니 연산을 여러 번 하여도, P의 위수는 변하지 않고 2로 유지된다. 따라서, SIDH 프로토콜을 처음 수행할 때 base curve에서의 2-torsion point를 공개값으로 가지고 있고, 이에 대한 아이소제니 연산과 계수 복원연산을 수행하면 이미지 곡선의 계수를 구할 수 있다.

그 후 상대방이 연산한 타원곡선에서 공유할 비밀값을 연산하는 과정에서는 몽고메리 곡선에서 이차방정식을 푸는 연산을 이용해 2-torsion point를 직접 계산한 다음, 해당 점을 이용해서 다시 타원곡선 계수 복원을 진행한다. 이차방정식을 풀 때는 square-root 는 한 번 진행되고, $p \equiv 3 \bmod 4$를 만족하므로 간단한 지수연산으로 수행할 수 있다. 또한, 해당 연산은 Bob이 shared secret을 구할 때만 일어나므로, 전체 SIDH 과정에서 한 번만 수행된다.

다음 표는, 몽고메리 곡선에서 식 (4)를 사용해서 5차 아이소제니의 이미지 곡선의 계수를 복원하는 경우와 2-torsion point를 이용했을 때의 연산량을 비교한다. Table 2.에서 **Original**은 식 (4)를 이용한 방법을, **2-torsion**은 2-torsion을 이용해서 복원한 방법을 의미한다.

Table 2. Comparison of the computational costs for recovering the coefficient of the image curve

	Original	2-torsion
cost	$10\boldsymbol{M}+3\boldsymbol{S}$	$8\boldsymbol{M}+4\boldsymbol{S}$

Remark 2. SIDH 기반 암호의 경우, 상대방의 generator $P_i, Q_i\, i \in \{A,B\}$와 자신의 개인키 n을 이용해 $P_i + nQ_i$를 연산하는 부분이 있다. 해당 부분은 몽고메리 래더 (Montgomery ladder) 방식을 이용해 구현하면 효율적으로 구현할 수 있으며, 따라서 SIDH는 P_i, Q_i 뿐만 아니라 $R_i = P_i - Q_i$도 generator point로 여겨 연산한다. [4]에서는 두 점 P, Q와 그의 차 $P-Q$를 이용하면 몽고메리 곡선 계수 a를 복원하는 방법에 대해서 제안하며, 해당 방법을 differential method로 명시하겠다.

Differential method를 이용해 계수를 복원하는 방법은 $8M+5S$의 연산량이 소요된다. 2-torsion point와 비교해서 squaring이 한 번 밖에 더 들지 않으므로, 이 방법이 2-torsion을 이용하는 방법보다 효율적이라고 생각할 수 있다. 하지만 타원곡선의 계수를 복원하는 과정은 매 아이소제니 연산시 일어나야 하므로, 논문에서 제시한 파라미터상으로는 2-torsion 방법보다 $119\times2=238$번의 F_{p^2}에서의 제곱연산이 더 필요하다는 것을 의미한다. (public key generation에서 119번, shared secret key generation에서 119번). 따라서 5-isogeny 연산시에는 2-torsion point 방법이 더 효율적임을 알 수 있다. 하지만 5차 이상의 아이소제니에서는 differential method가 더 효율적이라는 것을 알 수 있다.

4.3 실험 결과

본 장에서는 홀수 차수를 이용한 SIDH를 몽고메리, 에드워드, 허크 곡선에서 구현한 결과를 제시한다. 측정에 사용한 CPU는 3.40GHz의 동작 주파수를 가지는 Intel Core i7-6700를 사용했으며, Ubuntu 18.04.2 LTS 운영체제상에서 clang-6.0.0 컴파일러를 이용했다. Table 3.에서 **A Keygen**은 Alice의 공개키 생성을 의미하며, **B Keygen**은 Bob의 공개키 생성을 의미한다. 마찬가지로 **A Keyshare**는 Alice의 shared secret key 연산, **B Keyshare**는 Bob의 shared secret key 연산을 의미한다. 본 실험에서 Alice가 3 차 아이소제니를 사용하고 Bob이 5차 아이소제니를 사용하는 것으로 한다.

Table 3.에서 확인할 수 있듯이, 몽고메리와 허프 곡선에서의 연산량은 유사하며, 에드워드 곡선에서는 몽고메리나 허프 곡선과 비교했을 때 0.8% 느리다는 것을 알 수 있다.

Table 3. Performance result of SIDH implementation. The results were rounded to the nearest 10^3 clock cycles

	Mont	Ed	Huff
A Keygen	134,333	134,349	134,384
B Keygen	149,012	150,538	148,285
A Keyshare	113,871	113,856	113,892
B Keyshare	132,924	134,918	132,472
Total	530,140	533,661	529,033

먼저 몽고메리와 허프 곡선을 결과를 분석해본다. 3차 아이소제니를 사용하는 Alice 쪽은 몽고메리와 허프 곡선에서 SIDH를 구성하는 모든 단위연산이 동일하여 연산시간이 같다는 것을 쉽게 알 수 있다. 몽고메리와 허프 곡선의 연산량이 달라지는 부분은 Bob 쪽에서 5차 아이소제니의 이미지 곡선의 계수를 복원하는 부분이다.

허프 곡선의 경우 5차 아이소제니의 이미지 곡선 계수를 복원하는데 소요되는 연산량은 $8M+2S$이다. 하지만 생성된 계수를 연산에 효율적으로 바꾸기 위해서는 Table 1.의 **coeffTrans** 연산이 필요하며, 따라서 총 $8M+4S$의 연산량이 필요하다. 몽고메리 곡선의 경우 이미지 곡선의 계수를 복원하는데 2-torsion 방법이 사용될 수 있으며, 이 방법은 $8M+4S$의 연산량이 필요하다. 따라서, **B Keygen**에서도 몽고메리 곡선과 허프 곡선 사이에서 SIDH 구성하는 모든 단위 연산량이 일치하여, 사이클 수도 비슷하게 나온다는 사실을 확인할 수 있다.

B Keyshare에서 몽고메리와 허프 곡선을 비교해보면, 몽고메리의 경우 Alice로부터 받은 타원곡선의 계수를 이용하여 2-torsion 점을 얻어내는 과정이 필요하다. 주어진 타원곡선의 계수를 a라 하면, 이는 x^2+ax+1의 근을 구하는 과정이고, F_{p^2}에서 연산이 진행되기 때문에 해는 존재한다. 해당 해를 구하는데 크게 1번의 제곱근 연산이 필요하다. 한편, 허프 곡선의 경우 H_c에 대해서 $\hat{c}=c+1/c+2$가 전달된다. Alice의 경우 $\hat{c}$를 이용해서 바로 타원곡선 및 아이소제니 연산에 사용할 수 있지만, Bob의 경우 이미지 곡선의 계수를 복원하기 위해서는 c자체가 필요하다. c도 이차방정식의 해를 구하는 과정이 필요하고, 크게 1번의 제곱근 연산이 필요하다. 결론적으로 3차, 5차 아이소제니 사용에 있어서 SIDH를 구성하는 연산량이 몽고메리와 허프 모두 동일하기 때문에, 두 구현결과는 비슷한 사이클 수를 가진다는 것을 해석할 수 있다.

한편, Table 3.에서 살펴보면 몽고메리와 허프 곡선을 이용해 SIDH를 구현한 것보다 에드워드 곡선에서 0.8% 느리다는 것을 확인할 수 있다. Alice 쪽에서의 3차 아이소제니 연산의 경우 몽고메리, 에드워드, 허프 곡선에서의 공식이 동일하여 연산량의 차이가 거의 발생하지 않는다. 하지만 Bob 쪽에서

5차 아이소제니 연산에서는 이미지 곡선의 계수를 복원하는 과정에서 연산량의 차이가 발생한다. 에드워드 곡선에서는 이미지 타원곡선의 계수를 복원하는데 $6M+10S$의 연산량이 필요하며, $1S \simeq 0.8M$ 인 부분을 생각하면, 해당 연산량은 $14M$이 필요하다. 반면 허프 곡선이나 몽고메리 곡선에서는 $8M+4S$가 필요하며, 이는 약 $11.2M$에 해당한다. Bob에서 이미지 타원곡선의 계수를 복원하는 부분이 총 $119 \times 2 = 238$번 (public key generation에서 119번, shared secret key generation에서 119번)이 필요하기 때문에, 이는 에드워드 곡선에서의 속도 저하를 가져온다.

V. 결 론

본 논문에서는 SIDH 프로토콜을 홀수 차수 아이소제니만 이용하여 구현할 경우 몽고메리, 에드워드, 허프 곡선 중 어느 곡선이 효율적인 지에 관해 실험을 진행하였다. 이와 관련하여, 먼저 각 곡선에서 SIDH 구현을 위한 단위연산의 연산량을 비교하였으며 홀수 차수 사용을 위한 유한체를 새롭게 정의하였다. 본 논문의 결과 에드워드 곡선이 몽고메리나 허프 곡선보다 0.8% 느리다는 것을 알 수 있으며, 몽고메리와 허프 곡선에서는 속도가 유사하다는 점을 알 수 있다. 특히, differential method를 사용할 경우 몽고메리와 허프 곡선에서 타원곡선 계수 복원은 아이소제니의 차수와 관계없이 일정하므로, SIDH 기반 암호에서는 차수가 올라갈수록 몽고메리, 허프 곡선과 에드워드 곡선의 성능 차이가 발생할 수 있다.

References

[1] R. Azarderakhsh et al. "Supersingular isogeny key encapsulation", submission to the NIST post-quantum standardization project, 2017

[2] R. Azarderakhsh et al. "Practical supersingular isogeny group key aggrement," IACR Cryptology ePrint Archive, 2019:330, 2019

[3] J. Couveignes, "Hard homogeneous spaces," IACR Cryptology ePrint Archive, 2006:291, 2006

[4] C. Costello and H. Hisil, "A simple and compact algorithm for SIDH with arbitrary degree isogenies," Advances in Cryptology, ASIACRYPT'17, LNCS 10625, pp. 303-329, 2017

[5] Craig Costello "B-SIDH supersingular isogeny Diffie-Hellman using twisted torsion," Advances in Cryptology, ASIACRYPT'20, LNCS 12492, pp. 440-463,2020

[6] R. Drylo et al. "Efficient Montgomery-like formulas for general Huff's and Huff's elliptic curves and their applications to the isogeny-based cryptography," IACR Cryptology ePrint Archive, 2020:526, 2020

[7] R. Farashahi et al. "Differential addition on twisted Edwards curves," ACISP'17, LNCS 10343, pp. 366-378, 2017

[8] Y. Huang et al, "Optimized arithmetic operations for isogeny-based cryptography on Huff curves," ACISP'20, LNCS 12248, pp. 23-40, 2020

[9] M. Joye et al,"Huff's model for elliptic curves," International Algorithmic Number Theory Symposium, ANTS'10, pp. 234-250, 2010

[10] D. Jao, L. De Feo "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," PQCrypto'11, LNCS 7071, pp. 19-34, 2011

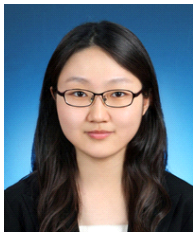
[11] S. Kim et al. "Optimized method for computing odd-degree isogenies on Edwards curves," Advances in Cryptology, ASIACRYPT'19, LNCS 11922, pp. 273-292, 2019

[12] S. Kim et al. "New hybrid method for isogeny-based cryptosystems using Edwards curves," IEEE transactions on Information Theory, vol. 66, no. 3, pp. 1934-1943, 2020

[13] M. Meyer et al. "On hybrid SIDH

schemes using Edwards and Montgomery curve arithmetic," IACR Cryptology ePrint Archive, 2017:1213, 2017
[14] P. Montgomery, 'Speeding the pollard and elliptic curve methods of factorization," Mathematics of computation, vol. 48, no. 177, pp. 243-264, 1971
[15] D. Moody and D. Shumow, "Analogues of Velu's formula for isogenies on alternate models of elliptic curves," Mathematics of Computations, vol. 85, no. 300, pp. 1929-1951, 2016
[16] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM review, vol. 41 no. 2, pp. 303-332, 1999.

〈저자소개〉



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

윤 기 순 (Kisoon Yoon) 정회원
1998년 8월: 경희대학교 수학과 이학사
2007년 8월: 고려대학교 정보보호학과 공학석사
2013년 11월: Universite de Caen 수학과 이학박사
2013년 11월~현재: NSHC 암호기술팀 팀장
〈관심분야〉 정수론, 암호학, 정보보호

박 영 호 (Young-Ho Park) 종신회원
1990년 2월: 고려대학교 수학과 이학사
1993년 2월: 고려대학교 수학과 이학석사
1997년 2월: 고려대학교 수학과 이학박사
2002년 2월~현재: 세종사이보대학교 정보보호학과 교수
〈관심분야〉 공개키 암호, 암호 프로토콜, 부채널 공격, 암호 안전성평가