Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.3, Jun. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.3.497

CSIDH 기반 암호에 대한 뒤틀린 몽고메리 곡선 사용*

김 수 리†‡
성신여자대학교 (조교수)

On the Use of Twisted Montgomery Curves for CSIDH-Based Cryptography*

Suhri Kim†‡
Sungshin Women's University (Assistant Professor)

요 약

본 논문에서는 뒤틀린 몽고메리 곡선을 사용하는 대표적인 암호인 CSURF의 최적화 구현에 대해 분석한다. Projective 형태의 타원곡선 연산은 몽고메리 곡선보다 뒤틀린 몽고메리 곡선이 더 느려서, CSURF는 hybrid 형태의 CSIDH 보다 성능이 느리다. 하지만, square-root Velu 공식을 사용할 경우 타원곡선 연산량을 줄일 수 있으므로 최적화할 여지가 있다. 본 논문에서는 처음으로 뒤틀린 몽고메리 곡선에서의 square-root Velu 공식을 제안하고, 2-isogeny 공식을 최적화하였다. 본 논문의 결과, 제안하는 CSURF는 기존보다 23.3% 빠르고, CSIDH 보다는 10.8% 느리다. 또한, 제안하는 constant-time CSURF의 경우 constant-time CSIDH 보다 6.8% 느리다. 제안하는 결과 CSURF는 CSIDH 보다 느리지만, 기존 뒤틀린 몽고메리를 이용한 구현과 비교하면 상당한 향상으로, 향후 뒤틀린 몽고메리 곡선에 적합한 구현에 본 논문의 결과를 이용할 수 있을 것으로 전망한다.

ABSTRACT

In this paper, we focus on optimizing the performance of CSURF, which uses the tweaked Montgomery curves. The projective version of elliptic curve arithmetic is slower on tweaked Montgomery curves than on Montgomery curves, so that CSURF is slower than the hybrid version of CSIDH. However, as the square-root Velu formula uses less number of ellitpic curve arithmetic than the standard Velu formula, there is room for optimization We optimize the square-root Velu formula and 2-isogeny formula on tweaked Montgomery curves. Our CSURFis 14% faster than the standard CSURF, and 10.8% slower than the CSIDH using the square-root Velu formula. The constant-time CSURF is 6.8% slower than constant-time CSIDH. Compared to the previous implementations, this is a remarkable result.

Keywords: Post-quantum cryptography, isogeny-based cryptography, CSIDH, twisted Montgomery curves

I. 서 론

아이소제니 기반 암호는 [10]에서 Couveignes에 의해 처음으로 제안되었다. Couveignes는 유한체 위에 정의된 ordinary 타원곡선을 이용한 키 교환 알고리즘을 제안하였다. 유한체 위에서의 ordinary 타원곡선의 endomorphism ring은 이차 수체에서 order $\mathcal{O}$를 형성한다. 이 $Cl(\mathcal{O})$의 가환성을 이용하면 Diffie-Hellman 방식의 키 교환 알고리즘을 설계할 수 있다. 이 연구는 후에 Rostovtsev와 Stolbunov에 의해 확장되었고, 현재는 이 알고리즘을 CRS 라 부른다 [23]. 하지만, CRS 알고리즘에 대해 양자 하지수시간의 공격이 존재할 뿐만 아니라 [7], 이 CRS 알고리즘의 가장 큰 문제는 속도가 느리다는 점이다. 이 후, 아이소제니 기반 암호는 De Feo와 Jao의 Supersingular Isogeny Diffie-Hellman

Received(04. 26. 2021), Accepted(05, 13, 2021)
* 이 논문은 2021년도 성신여자대학교 학술연구비 조성 지원에 의하여 연구되었음
† 주저자, suhrikim@gmail.com
‡ 교신저자, suhrikim@gmail.com(Corresponding author)

(SIDH)에 의해 다시 주목받게 된다 [15]. SIDH는 supersingular 타원곡선을 쓰기 때문에, [7]에서와 같은 공격을 사용할 수 없어 현재까지도 양자 지수시간의 공격 복잡도를 가진다. SIDH의 안전성은 유한체 위에 정의된 두 타원곡선 사이의 아이소제니를 찾는 어려움에 기반을 둔다. 현재, SIDH 기반 key encapsulation mechanism인 Supersingular Isogeny Key Encapsulation (SIKE)는 NIST 표준화 공모전의 Round 3의 대체후보로 선정되었다.

최근에, CRS 알고리즘은 De Feo, Keiffer, Smith와 Castryck 등에 의해 다시 연구되었다 [5]. CRS 기반 암호의 장점은 효율적이고 안전한 공개키 검증방식을 제공하기 때문에 non-interactive 키 교환 알고리즘을 설계하는데 적합하다 [11]. [11] 에서는 CRS 기반 암호를 구현하는데 있어서 group action 연산과 파라미터 설정을 최적화하는 방식을 제안하였다. CRS 기반 암호는 Castryck 등에 의해 더 최적화가 이루어졌다 [5]. 그들은 CSIDH (Commutative SIDH)를 제안하였는데, 제안하는 방식은 ordinary 곡선을 사용함에 있어서 발생하는 파라미터 선정과 관련된 문제를 supersingular 곡선을 사용함으로 해결하였다. CSIDH 키 교환 알고리즘의 평균 속도는 35ms 정도로, 다른 CRS 기반 암호와 비교하면 몇 배나 빠르다. 현재, CSIDH 기반 암호는 SIDH 기반 암호보다 느리지만, 다양한 암호학적 스킴 설계 가능성으로 많은 학자의 관심을 받고 있다 [3,16].

SIDH 기반 암호와 CSIDH 기반 암호의 공통적인 장점으로는 다른 PQC 암호와 비교하면 키 사이즈가 작다는 점이다. 하지만, 효율적인 행렬-벡터 곱 연산이 주를 이루는 다른 PQC 암호와 달리, 아이소제니 기반 암호는 400비트 이상의 유한체 위에 정의된 타원곡선 연산과 아이소제니 연산이 필요하므로, 다른 PQC 암호에 비해 몇 배나 느린 속도를 가지고 있다. 따라서 아이소제니 기반 암호의 최적화를 위해서 많은 연구가 진행되고 있다. 그중 하나의 갈래는, 아이소제니 연산 자체를 최적화하는 연구이다. 이 방법은 다른 타원곡선을 사용하거나, 아이소제니 공식 자체를 최적화한다는 것을 의미한다. [17,19,20]에서는 몽고메리 곡선과 에드워드 곡선을 혼합하여 사용하는 hybrid 방식이 제안되었다. 아이소제니 연산 자체를 최적화는 데에는 Bernstein 등이 최근에 ℓ차 아이소제니를 연산하는데 기존에 $O(\ell)$의 연산량을 드는 방식을 $O(\sqrt{\ell})$의 연산량이 들도록 최적화하는 방법을 제안하였다 [2]. 다른 연구 방향으로는, 효율적 구현을 위해 기존 아이소제니 기반 알고리즘을 변형하는 방법이다. 이 중 하나가 BSIDH로, Alice는 $(p+1)$-torsion 부분그룹에서 연산을 진행하고 Bob은 $(p-1)$-torsion 부분그룹에서 연산을 하는 방식을 제안하였다 [8]. BSIDH는 SIDH의 변형으로 볼 수 있으며, 기존 SIDH 와 다르게 Alice (사용자) 쪽에서 Montgomery reduction friendly 유한체를 사용하여 효율성을 가져온다. CSIDH 기반 쪽에서는 뒤틀린 몽고메리 곡선을 이용하여 CSIDH에 수평적 2차 아이소제니 사용으을 제안한 CSURF 가 있다 [4].

CSURF는 $p \equiv 7 \bmod 8$인 소수 p에 대해 endomorphism ring이 $\mathbb{Z}[(1+\sqrt{p})/2]$인 supersingular 타원곡선을 사용한다. 그들은 이러한 타원곡선들은 '뒤틀린 몽고메리 곡선 (몽고메리$^-$ 곡선)으로 대응시킬 수 있다는 것을 증명했으며, 몽고메리$^-$ 곡선은 몽고메리 곡선 (몽고메리$^+$ 곡선)과 타원곡선 연산이 유사하다는 점을 확인했다. 이러한 유한체 위에서, 소수 2는 $Q(\sqrt{-p})$에서 분해할 수 있고, 이는 수평적 2차 아이소제니를 사용할 수 있게 만들어준다. 2차 아이소제니 연산은 F_p에서의 단순한 지수연산으로 대체할 수 있으며, 2차 아이소제니 사용으로 보안강도에 적합하면서 효율적인 개인키를 선택할 수 있게 한다. 하지만 [12]에서 분석했듯이, 몽고메리$^-$곡선이 몽고메리$^+$ 곡선과 affine 형태에서는 연산량이 동일하지만, projective 형태에서는 몽고메리$^+$ 곡선이 더 효율적이며, projective 형태를 사용해 구현하는 아이소제니 암호 특성상, 몽고메리$^-$곡선은 더 비효율적이라는 결과를 제시했다. [12]의 구현결과에 의하면, CSURF는 [19]에서 제안된 hybrid 형태의 CSIDH보다 28%나 느리다. CSURF의 성능 저하를 가져오는 가장 큰 원인은, 몽고메리$^-$곡선에서의 타원곡선 연산량이 몽고메리$^+$ 곡선에서의 타원곡선 연산량보다 느리다는 점이다. 하지만, square-root Velu 공식은 타원곡선 연산을 기존보다 적게 사용하므로, 이를 CSURF 구현에 사용하면 더 효율적인 결과를 나을 수 있을 것이라 전망한다.

본 논문은 CSURF의 성능 최적화를 목표로 한다. 타원곡선 연산이 몽고메리$^-$곡선에서 느리기 때문에 CSURF는 CSIDH 보다 성능이 느리다. 따라서 본 논문에서는 몽고메리$^-$곡선에서의 아이소제니 연산 최적화에 초점을 맞춘다. 본 논문의 기여를 정리하면 다음과 같다

- CSURF에 사용되는 홀수 차수 아이소제니에 대한 최적화를 위해, 본 논문에서는 처음으로 몽고메리$^-$곡선에서 square-root Velu 공식을 적용하였다. Square-root Velu 공식을 적용하기 위해서는 타원곡선 위의 점 $P, Q, P-Q, P+Q$의 관계를 표현하는 2차 다항식을 정의해야 한다. 본 논문에서는 몽고메리$^-$곡선에서 2차 다항식을 정의했으며, 기존에 제안된 몽고메리$^+$ 곡선에 대한 2차다항식도 추가 최적화를 진행하였다. 이에 대한 자세한 내용은 본문의 3장에 정리되어있다.

- 본 논문에서는 몽고메리$^-$ 곡선에서의 2^e차 아이소제니 곡선의 최적화를 제안하였다. 다른 차수의 아이소제니와 달리 2^e차 아이소제니 구현시에는 affine 좌표계를 사용하는 것이 더 효율적이다. 본 논문에서는 최적화된 2^e 아이소제니 공식을 제안하으며, 해당 연산의 핵심 부분인 제곱근 연산을 sliding window를 이용해 최적화하였다.
- 추가적으로 CSURF 의 최적화를 위해 본 논문에서는 최적화된 파라미터를 제안한다. 제안하는 파라미터는 기존 파라미터보다 같은 보안강도에서 5.4% 빠른 속도를 보인다. 본 논문에 제안된 파라미터와 방법을 이용한 결과 본 논문의 CSURF는 기존 CSURF보다 추가적으로 23.3% 빠른 속도를 보이며, constant-time CSURF는 constant-time CSIDH에 비해 6.8%의 느린 속도를 보인다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 본 논문 구현 시 사용될 프로토콜인 CSURF와 몽고메리$^-$곡선에 대해 소개한다. 3장과 4장에서는 CSURF의 속도 최적화에 초점을 맞춘다. 3장에서는 몽고메리$^-$곡선에서 홀수 차수 아이소제니를 최적화할 수 있는 square-root Velu 공식에 대해 소개하고 이를 몽고메리$^-$곡선에 적용한다. 5장에서는 2^e차 아이소제니의 최적화와 구현 결과를 제시하고, 6장의 결론으로 마무리한다.

II. 배경 지식

본 장에서는 논문에 필요한 배경 지식을 소개한다. 먼저 CSURF에 대해 소개한 뒤, 몽고메리$^-$ 곡선과 곡선에서의 연산에 대해 소개한다.

2.1 CSIDH

CSIDH는 Castryck 등에 의해 제안된 아이소제니 기반 Diffie-Hellman 스타일의 키 교환 프로토콜이다 [5]. CSIDH는 유한체 F_p위에 정의된 supersingular 타원곡선에서 가환성을 가지는 group action 연산을 이용한 알고리즘이다. $\mathcal{O}$를 이차 수체에서의 order라 하자. $E\ell\ell_p(\mathcal{O})$를 endormophism ring을 $\mathcal{O}$로 하는 F_p위에 정의된 타원곡선들의 집합이라 하자. 그러면 class group $Cl(\mathcal{O})$가 $E\ell\ell_p(\mathcal{O})$에서 자유롭고 전이적으로 작용한다는 것은 잘 알려져 있다. 이러한 group action은 $[a]E$로 표현할 수 있는데, 여기서 $E \in E\ell\ell_p(\mathcal{O})$이고, $[a] \in Cl(\mathcal{O})$이다.

소수 p는 $p = 4\ell_1\ell_2 \cdots \ell_n - 1$의 형태를 가지는 소수라 정의하자. 여기에서 $\ell_1, \cdots, \ell_n$은 서로 다른 홀수인 작은 소수이다. E를 $End_p(E) = \mathbb{Z}[\pi]$를 만족하는 F_p위에서 정의된 supersingular 타원곡선이라 하자. 여기서 $End_p(E)$는 F_p에서 정의된 E의 endomorphism ring을 의미한다. 또한, $End_p(E)$는 quaternion order $End(E)$의 가환적인 subring 이다. 따라서 Frobenious 의 trace는 0이 되고, $E(F_p) = p+1$을 만족한다. $\pi^2 - 1 \equiv 0 \bmod \ell_i$를 만족하기 때문에, 아이디얼 $\ell_i\mathcal{O}$는 $\ell_i\mathcal{O} = \mathfrak{l}_i\bar{\mathfrak{l}_i}$의 형태로 분리될 수 있고, 여기에서 $\mathfrak{l}_i = (\ell_i, \pi - 1)$, $\bar{\mathfrak{l}_i} = (\ell_i, \pi + 1)$이다. 그러면 group action $[\mathfrak{l}_i]E$는 Velu의 공식을 이용해서 F_p위의 아이소제니 $\phi_{\mathfrak{l}_i}$로 연산될 수 있다.

Alice와 Bob가 서로 키를 교환한다고 하자. Alice는 개인 벡터 $(e_1, \cdots, e_n) \in \mathbb{Z}^n$에서 선택하는데, 이때 각각의 e_i는 양의 정수 m에 대해서 $e_i \in [-m, m]$의 범위를 가진다. 이 벡터가 아이디얼 클래스 $[a] = [\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}]$에 대해 group action과 연관된 아이소제니를 나타낸다. Alice는 공개키 $E_A = [a]E$를 연산하고 E_A를 Bob에게 전달한다. Bob도 Alice와 동일한 과정을 반복한 뒤, 공개키 $E_B = [b]E$를 전달한다. Bob의 공개키를 받은 뒤, Alice는 $[a]E_B$를 연산하고, Bob도 마찬가지로 $[b]E_A$를 연산한다. 가환성에 의해 $[a]E_B = [b]E_A$를 만족하게 된다.

2.2 CSURF

Castryck 와 Decru는 [4]에서 몽고메리$^-$곡선을 이용한 새로운 hard homogeneous space를 제안한다. CSURF는 $p \equiv 7 \bmod 8$형태의 소수와 F_p에서 정의된 몽고메리$^-$곡선을 사용한다. 이 경우, 몽고메리$^-$곡선의 F_p-endormophism ring은 $Z[(1+\sqrt{-p})/2]$를 만족하고, 이러한 설정에서 모든 곡선은 3개의 F_p에서 정의된 2-torsion 점을 가진다. 따라서, CSURF의 경우 아이디얼 $\iota_0 = (2, \frac{\sqrt{-p}-1}{2})$를 이용해 수평적 2차 아이소제니를 사용할 수 있다.

CSIDH에서는 $p \equiv 3 \bmod 8$형태의 소수를 사용해서 supersingular 몽고메리 곡선과, 몽고메리 곡선의 F_p-isomorphism class간의 일대일 대응이 존재했다. 이와 유사하게, CSURF의 환경에서도 몽고메리$^-$곡선의 계수와 F_p-isomorphism class간의 일대일 대응이 존재한다. 따라서 CSURF 에서도 CSIDH와 유사하게 잘 정의된 자유롭고 전이적으로 작용하는 group action이 존재한다. CSURF의 키 교환 프로토콜은 CSIDH와 유사하며, 가장 큰 차이는 CSURF는 2차 아이소제니를 사용함으로 인해 개인키의 공간이 원하는 보안강도에 보다 더 잘 대응할 수 있다.

2.3 Square-root Velu formula

일반적인 Velu의 공식의 경우 ℓ차 아이소제니를 연산하는데 $O(\ell)$ 유한체 연산이 필요하다. 최근, [2]에서 새로운 아이소제니 연산 방법이 제안되었으며, 해당하는 방법을 사용하면 ℓ차 아이소제니를 연산하는데 $O(\sqrt{\ell})$ 유한체 연산이 필요하다. 아이소제니 연산을 단순화한다면, Velu 공식은 어느 유한체 K에서 정의된 다항식의 함수값을 계산하는 것으로 생각할 수 있는데, 특징적인 점은 이 다항식의 해는 K에서 정의된 cyclic group 의 원소라는 것이다. G를 P로 인해 생성된 cyclic group 이라 하고, S를 Z의 유한부분집합이라고 하고, 다음과 같이 다항식을 정의하자.

$$h_S(X) = \prod_{s \in S} (X - f([s]P)) \quad (1)$$

위 식에서 $[s]P$는 P를 s번 연산한다는 것을 의미한다. $E(K)$를 타원곡선이라 하고, $P \in E(K)$, $G = \langle P \rangle$를 ℓ차 아이소제니 $\phi : E \to E'$의 커널이라 하자. 아이소제니 기반 암호에서의 S 는 커널 G라고 생각할 수 있으며, $f([s]P)$에서의 함수 f는 $[s]P$의 x 좌표를 읽어오는 함수라고 생각할 수 있다. 다음 Proposition 1은 식 (1)을 사용해 ℓ차 아이소제니를 연산하는 방법을 나타낸 것이다.

Proposition 1 (몽고메리$^+$ 곡선에서 square-root Velu 공식) M_a를 몽고메리$^+$ 곡선이라 하고, $P \in M_a$를 위수가 소수 $\ell \neq 2$인 점이라 하자. 커널 $\langle P \rangle$로 하는 아이소제니 $\phi : M_a \to M_{a'}$는 식 (1)을 이용해 다음과 같이 표현할 수 있다.

$$\phi(X) = \frac{X^\ell h_S(1/X)^2}{h_S(X)^2}$$

위 식에서 $S = \{1, 3, ..., \ell - 2\}$ 이다.

아이소제니 연산 시에는 이미지 곡선의 타원계수를 복원하는 연산도 필요한데, 몽고메리$^+$ 곡선에서는 에드워드 곡선을 이용하여 효율적으로 계산할 수 있다. 먼저, 몽고메리$^+$ 곡선과 대응하는 에드워드 곡선을 연산하는 것은 유한체 덧셈 뺄셈 연산으로 연산량의 거의 들지 않는다는 장점이 있다. 추가적으로, 에드워드 곡선을 사용하면 타원곡선 계수를 복원하는 연산도 square-root Velu 공식을 이용하여 최적화 할 수 있다.

$P = (x_1, y_1)$, $[i]P = (x_i, y_i)$라 하자. [21]에 제안된 에드워드에서의 아이소제니 공식을 사용하면, $M_{a'}$의 계수 a'는 $a' = 2(1+d)/(1-d)$로 연산이 된다. 여기에서 d는

$$d = (a-2)^\ell (\prod x_i - 1)^8 / (a+2)^\ell (\prod x_i + 1)^8$$

을 의미한다. 이 연산도 square-root Velu 형태를 이용해서 최적화 할 수 있으며, 따라서 d를 구하는 연산은 다음과 같다 [2]:

$$d = \left(\frac{a-2}{a+2}\right)^\ell \left(\frac{h_S(1)}{h_S(-1)}\right)^8$$

따라서, h_S가 $O(\sqrt{\ell})$ 유한체 연산으로 연산되면, a'도 $O(\sqrt{\ell})$ 유한체 연산으로 연산할 수 있다.

2.3.1 M_a에서의 이차다항식

결론적으로 square-root Velu 공식이 $O(\sqrt{\ell})$ 유한체 연산으로 연산할 수 있으려면, h_S가 $O(\sqrt{\ell})$ 유한체 연산으로 연산되어야 한다. h_S의 연산량이 $O(\sqrt{\ell})$이 되도록 하는 핵심 아이디어는 집합 S를 크기가 $\sqrt{S}$와 유사한 작은 집합 I, J로 나누는 것이다. [2]에서는 I, J는 S원소 대부분이 $(I+J)\cup(I-J)$에 존재하도록 선택한다. 이에 대한 자세한 설명은 [2]를 참조할 것을 권장한다. 따라서, 근이 $[s]P$인 다항식의 함수값을 계산하는 것은, 근이 $[i]P,[j]P$인 함수값을 계산하는 것으로 생각할 수 있다. 여기에서 $i\in I, j\in J$이다. 집합 I와 J 연관된 다항식의 resultant를 계산하는 것으로 h_S를 구할 수 있다. 이를 위해서는, 타원곡선 위의 점 $[i]P$, $[j]P$, $[i+j]P$, $[i-j]P$의 x좌표에 대한 관계식이 필요하다. 다음 Lemma 1은 타원곡선 E와 타원곡선 위의 점 $P, Q\in E$에 대해서 $P, Q, P+Q, P-Q$의 관계식을 나타내는 이차 다항식의 존재성을 제시한다.

Lemma 1 (x좌표들 사이의 관계[2]) q를 소수의 지수형태라 가정하고, $E(F_q)$를 타원곡선이라 하자. 그러면 모든 $P, Q\in E$에 대해 다음과 같이 이차다항식 F_0, F_1, F_2 이 존재한다.

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

위 식에서 $x(P)$는 P의 x좌표를 의미한다.

E가 affine 몽고메리 곡선 $By^2=x^3+Ax^{2+x}$로 정의되면, F_0, F_1, F_2는 다음과 같이 정의된다 [2]:

$$F_0(X_1,X_2)=(X_1-X_2)^2$$
$$F_1(X_1,X_2)=-2((X_1X_2+1)(X_1+X_2)+2AX_1X_2)$$
$$F_2(X_1,X_2)=(X_1X_2-1)^2$$

III. 몽고메리 곡선과 뒤틀린 몽고메리 곡선

이 장에서는 두 형태의 몽고메리 곡선에 대해 소개한다. 그 뒤, 각각의 곡선에 대한 타원곡선 연산과 아이소제니 연산을 비교한다. 해당 연산은 아이소제니 기반 암호를 구현하는데 가장 기본이 되는 연산이다.

K를 characteristic이 2나 3이 아닌 유한체라고 가정하자. K위에서 정의된 몽고메리 곡선은 다음과 같은 형태이다.

$$M_{a,b}: by^2=x^3+ax^2+x$$

위 식에서 $b(a^2-4)\neq 0$이다. 본 논문에서는 위와 같은 형태의 타원곡선을 몽고메리$^+$ 곡선이라 정의하며, $b=1$일 경우 간단히 M_a로 나타낸다. K에서 정의된 뒤틀린 몽고메리 곡선은 다음과 같이 정의된다.

$$M^-_{a,b}: by^2=x^3+ax^2-x$$

위 식에서 $b(a^2+4)\neq 0$이다. 본 논문에서는 위와 같은 형태의 타원곡선을 몽고메리$^-$곡선을 이라 한다. $b=1$일 경우에 간단히 M^-_a로 나타낸다.

3.1 M_a와 M^-_a 에서의 연산

M_a에서의 타원곡선 연산은 x좌표만 이용해 효율적으로 진행할 수 있는 사실이 알려졌다. $P=(x_p,y_p)$, $Q=(x_q,y_q)$를 $x_p\neq x_q$인 M_a위에 정의된 점이라 하자. $P-Q=(x_{p-q},y_{p-q})$. $P+Q$에 대한 x좌표를 x_{p+q}라 하고, $[2]P$의 x좌표를 $x_{[2]P}$라 하면, x_{p+q}와 $x_{[2]P}$ 은 다음식을 이용해 구할 수 있다.

$$x_{p+q}=(x_px_q-1)^2/(x_{p-q}(x_p-x_q)^2)$$
$$x_{2[P]}=(x_p^2-1)^2/(4x_p(x_p^2+ax_p+1))$$

이 식을 이용해 M_a에 대한 홀수 차수 아이소제니 공식도 정의할 수 있다 [9]. $P=(x_1,y_1)$을 몽고메리$^+$ 곡선 M_a에서 위수가 $\ell=2d+1$인 점이라 하자. $[i]P=(x_i,y_i)$라 하고, $[1]P=P$라 하자. 이 경우 $\langle P\rangle$를 커널로 하는 ℓ아이소제니

$\phi : M_a \rightarrow M_{a'} = M_a / \langle P \rangle$ 는 다음과 같이 정의된다.

$$\phi : (x,y) \rightarrow (f(x), y f'(x))$$

위 식에서 $f(x)$는 다음으로 정의한다.

$$f(x) = x \prod_{i=1}^{d} \left(\frac{x x_i - 1}{x - x_i} \right)^2$$

위 식에서, $f'(x)$는 $f(x)$를 미분한 결과이다. 타원곡선 계수 a'는

$$a' = (6\sigma + a)\pi^2$$

을 이용해 구할 수 있으며,

$$\sigma = \sum_{i=1}^{d} (x_{[i]P} - 1)/x_{[i]P}$$
$$\pi = \prod_{i=1}^{d} x_{[i]P}$$

이다. 이와 유사하게 몽고메리$^-$ 곡선에서도 타원곡선 연산을 정의할 수 있다 [4]. $P=(x_p,y_p)$, $Q=(x_q,y_q)$를 $x_p \neq x_q$인 M_a^-위에 정의된 점이라 하자. $P-Q=(x_{p-q},y_{p-q})$. $P+Q$에 대한 x좌표를 x_{p+q}라 하고, $[2]P$의 x좌표를 $x_{[2]P}$라 하면, x_{p+q}와 $x_{[2]P}$ 은 다음식을 이용해 구할 수 있다.

$$x_{p+q} = (x_p x_q + 1)^2 / (x_{p-q}(x_p - x_q)^2)$$
$$x_{2[P]} = (x_p^2 + 1)^2 / (4x_p(x_p^2 + a x_p + 1))$$

마찬가지로, 위 식을 이용해 M_a^-에 대한 홀수 차수 아이소제니 공식도 정의할 수 있다 [4]. $P=(x_1,y_1)$을 몽고메리$^-$ 곡선 M_a^-에서 위수가 $\ell = 2d+1$인 점이라 하자. $[i]P=(x_i,y_i)$라 하고, $[1]P=P$라 하자. 이 경우 $\langle P \rangle$를 커널로 하는 ℓ아이소제니 $\phi : M_a^- \rightarrow M_{a'}^- = M_a^- / \langle P \rangle$ 는 다음과 같이 정의된다.

$$\phi : (x,y) \rightarrow (f(x), c_0 y f'(x))$$

위 식에서 $f(x)$는 다음으로 정의한다

$$f(x) = x \prod_{i=1}^{d} \left(\frac{x x_i + 1}{x - x_i} \right)^2$$

위 식에서, $f'(x)$는 $f(x)$를 미분한 결과이다. 타원곡선 계수 a'는

$$a' = \pi'(a - 3\sigma')$$

을 이용해 계산할 수 있으며,

$$\sigma' = \sum_{i=1}^{d} (x_{[i]P} + 1)/x_{[i]P}$$
$$\pi' = \prod_{i=1}^{d} x_{[i]P}$$
$$c_0^2 = \pi$$

이다.

위 식에서 볼 수 있듯이 M_a^-에서의 타원곡선 및 아이소제니 연산은 분자의 부호만 다르고 M_a에서 타원곡선 및 아이소제니 연산과 유사하다는 것을 알 수 있다. 하지만 projective 좌표계를 사용할 경우 이 부호가 연산량에 큰 영향을 미치게 된다. 다음 〔Table 1〕은 아이소제니 암호를 구현하는 기반 함수에 대한 연산량을 비교한 표이다. 〔Table 1〕에서 Hybrid는 타원곡선 연산은 몽고메리 곡선을 이용하고 아이소제니 연산은 에드워드 곡선을 이용하는 기법을 의미한다 [19]. 〔Table 1〕에서 $w(\ell) = (h-1)\mathbf{M} + (t-1)\mathbf{S}$를 의미하는데, 여기에서 h는 ℓ의 hamming weight을, t는 ℓ의 비트 길이를 의미한다. 또한 $\mathbf{M}$은 유한체 위에서의 곱셈 연산을, $\mathbf{S}$은 유한체 위에서의 제곱 연산s을 의미한다. 〔Table 1〕에서 알 수 있듯이, 몽고메리$^-$ 곡선에

Table 1. Computational cost of lower-level functions on Montgomery$^+$ curves and Montgomery$^-$ curves

	Mont.$^+$	Hybrid	Mont.$^-$
DBLADD	$8\boldsymbol{M}+4\boldsymbol{S}$	$8\boldsymbol{M}+4\boldsymbol{S}$	$11\boldsymbol{M}+8\boldsymbol{S}$
DBL	$4\boldsymbol{M}+2\boldsymbol{S}$	$4\boldsymbol{M}+2\boldsymbol{S}$	$5\boldsymbol{M}+3\boldsymbol{S}$
ℓ-isog. eval	$(4d)\boldsymbol{M}+2\boldsymbol{S}$	$(4d)\boldsymbol{M}+2\boldsymbol{S}$	$6d\boldsymbol{M}+2\boldsymbol{S}$
ℓ-isog. coeff	$(6d\text{-}2)\boldsymbol{M}+3\boldsymbol{S}$	$2d\boldsymbol{M}+6\boldsymbol{S}+2w(\ell)$	$(6d\text{-}2)\boldsymbol{M}+3\boldsymbol{S}$

서 연산이 몽고메리$^+$ 곡선보다 비효율적이라는 것을 알 수 있다.

3.2 M_a^-의 홀수 차수 아이소제니 최적화

〔Table 1〕에 제시된 바와 같이 몽고메리$^+$ 곡선과 비교했을 때 몽고메리$^-$ 곡선의 단점은 타원곡선 연산이 몽고메리$^+$ 곡선보다 느리다는 점이다. 따라서 본 논문에서는 몽고메리$^-$ 곡선에서 아이소제니 연산 최적화에 초점을 맞춘다. 이 장에서는, square-root Velu 공식이 활용이 M_a^-에서의 아이소제니 연산의 효율성을 가져올 수 있는지 분석한다. 이를 위해 먼저 Proposition 1처럼 몽고메리$^-$ 곡선에서 이차 다항식을 정의한다.

Proposition 1 (몽고메리$^-$ 곡선에서 square-root Velu 공식) M_a^-를 몽고메리$^-$ 곡선이라 하고, $P\in M_a^-$를 위수가 소수 $\ell\neq 2$인 점이라 하자. 커널 $\langle P\rangle$로 하는 아이소제니 $\phi: M_a^-\rightarrow M_{a'}^-$는 다음과 같이 표현할 수 있다.

$$\phi(X)=\frac{X^{\ell}h_S(-1/X)^2}{h_S(X)^2}$$

위 식에서 $S=\{1,3,\ldots,\ell-2\}$ 이다.

3.2.1 M_a^-에서 타원곡선 계수 복원 연산

아이소제니 연산의 중요한 다른 한 부분은, 이미지 곡선의 계수를 복원하는 연산이다. M_a와 달리, M_a^-에서 에드워드 곡선을 사용하는 방법은 복잡하다.

$$M_a^-\xrightarrow{\iota}M_A\xrightarrow{\phi}E_d\xrightarrow{\psi}E_{d'}\xrightarrow{\psi^{-1}}M_A{}'\xrightarrow{\iota^{-1}}M_{a'}^-$$

위 다이어그램에서 ι는 M_a^-에서 M_A로의 변환, ι^{-1} 그의 역변환을 의미한다. ϕ는 몽고메리$^+$ 곡선에서 에드워드 곡선으로의 변환,ϕ^{-1}은 에드워드 곡선에서 몽고메리$^+$ 곡선으로 변환을 의미한다. ψ는 에드워드에 대응되는 ℓ차 아이소제니 연산을 의미한다. 위 의 함수들 합성을 통해 $M_{a'}^-$의 계수 a'는 $a'=(2i(1+d))/(1-d)$를 이용해 계산할 수 있고, 여기에서 d는

$$d=\left(\frac{-ia-2}{-ia+2}\right)^{\ell}\left(\prod\frac{-ix_i-1}{-ix_i+1}\right)^8$$

로 정의된다.

위 결과가 F_p에서 정의되긴하지만, 몽고메리$^+$ 곡선과 달리 몽고메리$^-$ 곡선에서 에드워드 곡선 사용은 비효율적이라는 사실을 알 수 있다. 따라서 본 논문에서는 〔9〕에서 제시된 2-torsion 방법을 이용하여 타원곡선 계수 a'를 복원하는 방법을 사용하였다. 이 방법을 사용하면, 타원곡선 계수를 복원하는데의 공식도 square-root Velu 공식의 형태를 따르기 때문에 효율적인 연산이 가능하다.

$P_\alpha=(X_\alpha:Z_\alpha)$를 projective 좌표계로 나타낸 M_a^-에서의 2-torsion 점이라 하자. ϕ를 ℓ차 아이소제니라 하자. ϕ는 홀수 차수이기 때문에, 2-torsion 점에 대한 함수값을 계산하여도 2-torsion을 보존한다. 따라서 $\phi(P)=(X_\alpha':Z_\alpha')$도 M_a^-에서 2-torsion 점이며, 타원곡선 식을 사용하면

$$a'=(Z_\alpha'^2-X_\alpha'^2)/X_\alpha'Z_\alpha'$$

를 만족한다. 따라서 2-torsion 점에 대한 함수값을 연산한 뒤, 타원곡선을 복원하는 데에는 2M 연산량이 필요하다.

3.2.2 M_a^-에서 이차다항식

Square-root Velu 공식을 사용하기 위해서, M_a에서와 마찬가지로, M_a^-에서도 이차다항식을새롭게 정의해야한다. 몽고메리$^-$ 곡선위의 점 P,Q에 대해서 $P,Q,P+Q,P-Q$에 대한 x좌표는 다음과 같은 관계를 가진다.

$$\begin{aligned}&(X-x(P+Q))(X-x(P-Q))\\&\quad=X^2+\frac{G_1(x(P),x(Q))}{G_0(x(P),x(Q))}X+\frac{G_2(x(P),x(Q))}{G_0(x(P),x(Q))}\end{aligned}$$

위 식에서 G_0, G_1, G_2는 다음과 같이 정의된다

$$G_0(X_1,X_2)=(X_1-X_2)^2$$
$$G_1(X_1,X_2)=-2((X_1X_2-1)(X_1+X_2)+2aX_1X_2)$$
$$G_2(X_1,X_2)=(X_1X_2+1)^2$$

3.2.3 M_a^+와 M_a^-의 이차다항식 연산량

Square-root Velu 공식을 이용해 아이소제니 연산을 할 때 h_S를 연산하기 위해 알고리즘은 F_0,F_1,F_2 (resp, G_0,G_1,G_2)를 커널의 생성자와 M_a (resp M_a^-) 위의 점에 대해 연산한다. 따라서 이차다항식을 구현하는데 있어서도 projective 형태의 좌표와 타원곡선 계수를 사용한다.

$P=(X_p:Z_p)$를 커널의 생성자라 하고, $Q=(X_q:Z_q)$를 타원곡선 M_a 위의 다른 점이라 하자. 이 경우, 몽고메리$^+$ 곡선에서 F_0,F_1,F_2 은 다음과 같이 연산된다.

$$F_0=C(X_pZ_q-X_qZ_p)^2$$
$$F_1=-2C(X_pX_q+Z_pZ_q)(X_pZ_q+X_qZ_p)-4AX_pX_qZ_pZ_q$$
$$F_2=C(X_pX_q-Z_pZ_q)^2$$

위 식에서 $a=A/C$를 만족한다. 또한 $h_S(1/Q)$도 연산이 되어야 하는데, $1/Q=(Z_q:X_q)$이기 때문에 F_0,F_1,F_2에서 연산된 값을 다시 사용할 수 있다 -- F_0와 F_2의 값만 바꿔주면 된다. [2]의 구현된 결과에 의하면 F_0,F_1,F_2의 연산량은 $10\mathbf{M}+2\mathbf{S}$ 이다. [1]에서는 $7\mathbf{M}+3\mathbf{S}+12\mathbf{a}$로 최적화하였으며, 여기에서 $\mathbf{a}$는 유한체에서의 덧셈을 의미한다. 본 논문에서는 이를 $7\mathbf{M}+2\mathbf{S}+4\mathbf{a}$의 연산량으로 최적화하였다.

몽고메리$^-$ 곡선에서 $P=(X_p:Z_p)$를 커널의 생성자라 하고, $Q=(X_q:Z_q)$를 타원곡선 M_a^- 위의 다른 점이라 하자. 이 경우, 몽고메리$^-$ 곡선에서 G_0,G_1,G_2 은 다음과 같이 연산된다.

$$G_0=C(X_pZ_q-X_qZ_p)^2$$
$$G_1=-2C(X_pX_q-Z_pZ_q)(X_pZ_q+X_qZ_p)-4AX_pX_qZ_pZ_q$$
$$G_2=C(X_pX_q+Z_pZ_q)^2$$

몽고메리$^-$ 곡선에서는 $h_S(-1/Q)$도 연산이 되어야 하는데, $1/Q=(Z_q:X_q)$이기 때문에, 마찬가지로 G_0,G_1,G_2의 값을 다시 사용할 수 있다 -- G_0과 G_2의 값은 바꿔주고, G_1은 음수를 취해주면 된다. 직관적으로 구현하면 G_0,G_1,G_2를 연산하는데 $10\mathbf{M}+2\mathbf{S}$가 필요하다. 2-torsion 점에 대한 함수값도 계산해야 하므로, 이를 합하면 $20\mathbf{M}+4\mathbf{S}$의 연산량이 필요하다. 본 논문에서는 이를 $13\mathbf{M}+4\mathbf{S}$로 최적화하였다.

[Table 2]는 몽고메리$^+$ 곡선과 몽고메리$^-$곡선에서 h_S의 연산량을 정리한 표이다. [Table 2]에서 Q_2는 몽고메리$^-$ 곡선에서 2-torsion 점을 의미한다.

Table 2. Computational cost of biquardatic polynomials

		Mont$^+$	Mont$^-$
ℓ eval	*Computation*	$h_S(Q)$, $h_S(1/Q)$	$h_S(Q)$, $h_S(-1/Q)$
	Cost	$7M+2S+4a$	$7M+2S$
ℓ coeff	*Computation*	$h_S(1)$, $h_S(-1)$	$h_S(Q_2)$, $h_S(-1/Q_2)$
	Cost	$3M+2S$	$6M+2S$

IV. 구현결과

본 장에서는 square-root Velu 공식을 사용한 CSURF 구현결과와 constant-time CSURF 구현결과를 제시한다. 먼저, constant-time CSURF의 추가 최적화를 위해 2^e차 아이소제니 최적화 방안을 제시한다.

4.1 2차 아이소제니 최적화 구현

[4]에서는 몽고메리$^+$ 곡선에서 2차 아이소제니 연산과 몽고메리$^+$ 곡선과 몽고메리$^-$ 곡선변환을 합성하여, 몽고메리$^-$ 곡선에서의 연속적인 2차 아이소제니를 연산하는 방법을 제안했다.

Algorithm 1을 이용하여 연속적인 2차 아이소제니를 구현할 때, 기존 홀수 차수 아이소제니 연산과는 다르게 affine 형태의 타원곡선 계수를 사용하는 것이 효율적이다. Algorithm 1의 Step 4는 다음

Algorithm 1. Computing 2^e-isogeny on M_a^- over F_p for $p \equiv 7 \bmod 8$ [4]

1. If $e=0$, then return a
2. else
3. $\quad a \leftarrow sign(e) \bullet a$
4. $\quad a \leftarrow 2\dfrac{a-3\sqrt{a^2+4}}{a+\sqrt{a^2+4}}$
5. For i from 2 to e do
6. $\quad a \leftarrow 2(3+a(\sqrt{a^2-4}-a))$
7. $a \leftarrow \dfrac{a+3\sqrt{a^2-4}}{\sqrt{2\sqrt{a^2-4}(a+\sqrt{a^2-4})}}$
8. return $sign(e) \bullet a$

Fig. 1. Method of computing consecutive 2-isogenies on M_a^-

과 같이 다시 표현할 수 있다.

$$a \leftarrow 2(a\sqrt{a^2+4}-(a^2+3))$$

[4]에서 제시된 것처럼 Algorithm 1을 그대로 구현했을 때 보다, 위 식을 사용하면 역원연산을 1개 줄일 수 있다. 위 방법을 사용한 2^e차 아이소제니의 연산량은 $3\mathbf{M}+2\mathbf{S}+3\mathbf{E}+(e\text{-}1)(1\mathbf{M}+1\mathbf{S}+1\mathbf{E})$이다. 여기에서 E는 유한체 위에서 지수연산을 의미한다. $p \equiv 3 \bmod 4$인 소수 p에 대한 유한체 F_p에서는, $a \in F_p$에 대해 a의 제곱근은 $a^{(p+1)/4}$로 연산되고, a의 역원은 a^{p-2}로 연산할 수 있다. 또한, 연산한 뒤 제곱근을 구할 때에는 $a^{(p+1)(p-2)/4 \bmod p-1}$로 연산할 수 있다. 위 세 연산이 2^e차 아이소제니 연산의 효율성에 큰 영향을 미치기 때문에, window size가 6인 sliding window 방식을 이용해 구현하였다. 해당 window size는 실험을 통해 얻어진 값이다.

4.2 CSURF 구현 결과

본 단원에서는 논문에서 제시된 방법을 이용해서 CSURF 의 최적화 구현과, 이를 사용한 constant-time CSURF 구현 결과를 제시한다. Constant-time CSURF의 경우는 [22]에서 제시된 방식을 사용하였다. 측정에 사용한 CPU는 3.40GHz의 동작 주파수를 가지는 Intel Core i7-6700를 사용했으며, Ubuntu 18.04.2 LTS 운영체제상에서 최적화 옵션 -O3과 clang-6.0.0 컴파일러를 이용했다.

CSIDH와 CSURF 구현을 위해서 [13]에 제시된 다음 511비트 소수를 사용하였다.

$$p = 2^4 \bullet 3^3 \bullet 5 \bullet 7 \bullet 11^2 \bullet 13 \cdots 373 - 1$$

이 유한체 위해서 supersinguler 몽고메리$^+$ 곡선인 $M_0^+ : y^2 = x^3 + x$와 supersingular 몽고메리$^-$ 곡선인 $M_0^- : y^2 = x^3 - x$를 각각 CSIDH와 CSURF의 시작 곡선으로 사용하였다. 개인키는 다음을 사용하였다.

[Table 3]에서 CSIDH는 [5]에 제시된 CSIDH의 개인키 공간을 의미하고, CSURF는 [4]에 제시된 CSURF의 개인키 공간을 의미한다. Proposed는 본 논문에서 제안하는 CSURF의 개인키 공간을 의미한다.

먼저 [4]에서 제안된 CSURF 파라미터와 본 논문에서 제안된 파라미터를 사용했을 때의 속도를 비교한다. [Table 4]에서 **CSURF Original**은 [4]에 제시된 파라미터를 사용해 구현한 결과를, **CSURF Proposed**는 [Table 3]에 제안된 파라미터를 사용했을 때의 결과를 의미한다. 두 구현 모두 일반적인 Velu 공식을 사용해 구현하였다.

Table 3. Private key exponent range and its securfity

	CSIDH [5]	CSURF [4]	Proposed
Exponent Range	$[-5,5]^{73}$	$[-137,137]$ $\times[-4,4]^3$ $\times[-5,5]^{45}$ $\times[-4,4]^{25}$	$[-58,58]$ $\times[-3,3]$ $\times[-4,4]^2$ $\times[-5,5]^{53}$ $\times[-4,4]^{15}$ $\times[-3,3]^2$
Security	252.54	252.53	252.53

Table 4. Performance result of group action of CSURF when various parameter is used

	CSURF Original	CSURF Proposed
Group action	121,512,191	114,893,636

〔Table 4〕에 제시된 바와 같이, 본 논문에서 제안한 파라미터는 같은 보안강도에서 〔4〕에서 제안된 파라미터보다 5.4% 빠르다. 따라서 이후의 CSURF 구현에서는 논문에서 제안된 파라미터를 사용한다.

다음, 일반적인 Velu 공식을 사용했을 때와, 본 논문에서 최적화한 square-root Velu 공식을 사용했을 때 CSURF 성능을 비교한다. 실험을 통해서 square-root Velu 공식의 사용은 60차 아이소제니 이상을 연산할 때 사용하였다. 〔Table 5〕에서 **CSURF**는 일반적인 Velu 공식을 사용한 구현을 의미하고, **sqrt-CSURF**는 square-root Velu 공식을 사용한 구현을 의미한다.

〔Table 5〕에 제시된 바와 같이, 본 논문에서 제안한 최적화 기법을 사용하면 **sqrt-CSURF**는 **CSURF** 보다 14% 빠르다. 다음, square-root Velu 공식이 사용됐을 때 CSIDH와 CSURF의 성능을 비교한다.

〔Table 6〕에 제시된 바와 같이 **sqrt-CSURF**는 **sqrt-CSIDH** 보다 10.8% 느리다. 하지만 일반 CSURF (square-root Velu 공식을 사용하지 않은 CSURF)와 비교하면 23.3% 빠른 결과로, 상당한 향상이 있다는 것을 알 수 있다. 마지막으로, square-root Velu 공식을 사용한 constant-time CSIDH와 CSURF를 비교한다.

〔Tabl 7〕에 제시된 바와 같이, **sqrt-CSURF**는 **sqrt-CSIDH**에 비해 6.8% 느리다. constant-time CSURF와 CSIDH의 속도 차이가 non-constant time CSURF와 CSIDH의 속도차이보다 작은 이유는, CSURF의 2차 아이소제니 연산이 몽고메리$^-$ 곡선에서 비효율적인 홀수 차수 아이소제니 연산을 줄여주기 때문이다.

Table 5. Performance result of group action of CSURF and sqrt-CSURF

	CSURF	sqrt-CSURF
Group action	114,893,636	98,782,302

Table 6. Performance result of group action when the square-root Velu formula is used for CSIDH and CSURF

	sqrt-CSIDH	sqrt-CSURF
Group action	88,017,277	98,782,302

Table 7. Performance result of constant-time group action

	sqrt-CSIDH	sqrt-CSURF
Group action	366,759,725	393,590,292

Constant-time CSIDH의 경우 $5\times73=365$ 번의 홀수 차수 아이소제니 연산이 필요하다. Constant-time CSURF의 경우 342개의 홀수 차수 아이소제니와 58번의 2차 아이소제니 연산이 필요하다. 2차 아이소제니 연산으로 CSURF는 56번의 유한체위에서의 지수연산이 필요하지만, 아이소제니 연산량이 줄어 몽고메리$^-$ 곡선에서 비효율적인 타원곡선 연산과 아이소제니 연산이 줄어들었다.

V. 결 론

본 논문에서는 CSURF를 최적화하는 방안을 제시하였다. 몽고메리$^-$ 곡선에서의 타원곡선 연산이 몽고메리$^+$ 곡선에 비해 느리기 때문에, 본 논문에서는 홀수 차수 아이소제니 연산 최적화에 초점을 맞췄다. 특히 square-root Velu 공식을 사용하면, 일반 Velu 공식보다 타원곡선 연산량을 줄일 수 있어서 몽고메리$^-$ 곡선에서 square-root Velu 공식을 적용하였다. 이를 위해 몽고메리$^-$ 곡선에서 이차 다항식을 정의하였고, 이를 최적화하였다. 추가적으로 몽고메리$^-$ 곡선에서 2^e차 아이소제니 공식을 최적화하였다.

본 논문의 구현 결과 non-constant time 구현에 있어서, sqrt-CSURF는 sqrt-CSIDH에 비해 10.8% 느리다. 하지만 기존 23.3% 느린 결과에 비하면 이는 큰 향상이다. Constant-time 구현의 경우 sqrt-CSURF는 sqrt-CSIDH에 비해 6.8% 느리다. 하지만 CSURF의 장점은 보안강도에 보다 더 잘 맞게 개인키를 선택하는 점이므로, 적절한 파라미터를 찾는다면 추가적인 최적화가 가능할 것으로 보인다.

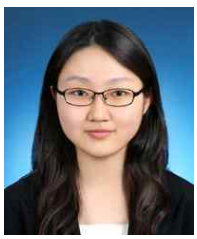
References

〔1〕 J.J. Chi-Domiguez et al. “On new Velu’s formulae and their applications to CSIDH and BSIDH constant-time implementations,” IACR Cryptology ePrint

Archive, 2020:1109, 2020
[2] D. Bernstein et al. "Faster computation of isogenies of large prime degree," IACR Cryptology ePrint Archive, 2020:341, 2020
[3] W. Beullens et al. "CSI-FiSh: efficient isogeny based signatures through class group computations," ASIACRYPT, LNCS 11921, pp. 227-247, Dec. 2019
[4] W. Castryck and T. Decru "CSIDH on the surface," PQCrypto, LNCS 12100, pp.111-129, April, 2020
[5] W. Castryck et al. "CSIDH: An efficient post-quantum commutaitve group action," ASIACRYPT, LNCS 11274, Dec. 2018
[6] D. Cervantes-Vazquez et al. "Stronger and faster side-channel protections for CSIDH," LATINCRYPT, LNCS 11774, Sept. 2019
[7] A. Childs et al. "Constructing elliptic curve isogenies in quantum subexponential time," Journal of Mathematical Cryptology, vol. 8, no. 1, pp. 1-29, 2014
[8] C. Costello, "B-SIDH supersingular isogeny Diffie-Hellman using twisted torsion," ASIACRYPT, LNCS 12492, pp. 440-463, Dec. 2020
[9] C. Costello and H. Hisil, "A simple and compact algorithm for SIDH with arbitrary degree isogenies," ASIACRYPT, LNCS 10625, pp. 303-329, Dec. 2017
[10] J.M. Couveignes, "Hard homogenous spaces," IACR Cryptology ePrint Archive, 2006:291, 2006
[11] De Feo. et al. "Towards practical key exchange from ordinary isogeny graphs," ASIACRYPT, LNCS 11274, Dec. 2018
[12] D. Heo et al. "On the performance analysis for CSIDH-based cryptosystems," Applied Sciences, vol. 10, no. 19, 2020
[13] D. Heo et al. "Optimized CSIDH implementation using a 2-torsion point," Cryptography, vol. 4, no. 3, 2020
[14] A. Jalali, "Towards optimized and constant-time CSIDH on embedded devices," International Workshop on Constructive Side-Channel Analysis and Secure Design, pp. 215-231, 2019
[15] D. Jao, L. De Feo "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," PQCrypto, LNCS 7071, pp. 19-34, Aug. 2011
[16] T. Kawashima, "An efficient authenticated key exchange from random self-reducibility on CSIDH," IACR Cryptology ePrint Archive, 2020:1178, 2020
[17] 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
[18] M. Meyer et al. "On lions and elligators: An efficient constant-time implementations of CSIDH", PQCrypto, LNCS 11505, pp. 307-325, 2019
[19] M. Meyer and S. Reith "A faster way to the CSIDH," INDOCRYPT, LNCS 11356, pp. 137-152, 2018
[20] M. Meyer et al. "On hybrid SIDH schemes using Edwards and Montgomery curve arithmetic," IACR Cryptology ePrint Archive, 2017:1213, 2017
[21] 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
[22] H. Onuki et al. "A constant-time algorithm of CSIDH keeping two points," IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, vol. E103.A, no. 10, pp. 1174-1182, 2020

[23] A. Stolbunov, "Constructing public-key cryptographic schemes based on class group action on a set of isogenous elitpic curves," Advances in Mathematics of Communication, vol. 4, no. 2, pp. 215-235, 2010

〈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 박사후연구원
2020년 7월~2021년 2월: KU Leuven ESAT/COSIC 박사후연구원
2021년 3월~현재: 성신여자대학교 수리통계데이터사이언스학부 조교수
〈관심분야〉 공개키 암호시스템, 후양자암호