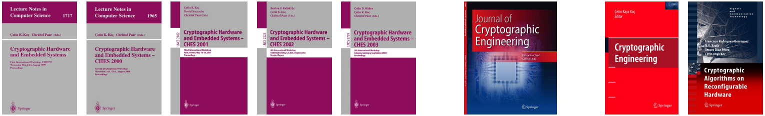
Supersingular Isogeny Diffie-Hellman

Çetin Kaya Koç

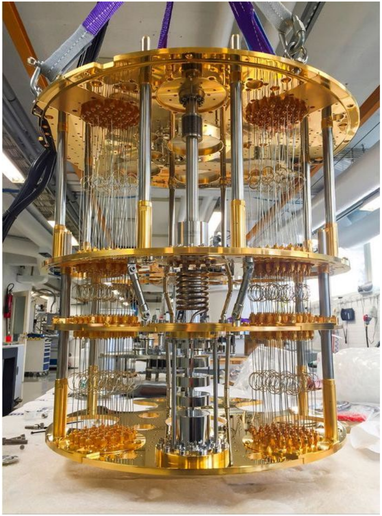
The Speaker: Çetin Kaya Koç *

- Professor at UC Santa Barbara and İstinye University, Turkey
- Director of **Koç Lab** `http://koclab.org` — supervising grad students and posdtocs from US, Turkey, and China
- Co-founder of the **CHES Conference**
- Founding editor-in-chief of **Journal of Cryptographic Engineering**
- Author of 2 successful books in cryptography — the book *Cryptographic Engineering* is translated to Chinese
- Crypto consulting and development: Cryptocode Inc (Santa Barbara) and Koç Lab Technology Ltd (İstanbul)



* Koç is pronounced as "Coach"

Quantum Computer



A quantum computer is composed of:

- A register containing of n **qubits**

- Multiqubit logic gates applied to the register according to an algorithm

- A measurement system determining the states of selected qubits at the end of computation

Due to superposition principle, a single quantum register is capable of simultaneously processing all inputs at once

A quantum computer is useful **only if** we have a quantum algorithm to solve a particular problem

The End of Traditional Cryptography

- The public-key cryptosystems currently in use are based on integer factorization, discrete logarithm, and elliptic curve discrete logarithm problems
- These problems are believed to be **intractable** with current computing technology, which makes them useful as cryptographic one-way functions
- However, they can be solved in polynomial time by using Shor's algorithm (or one of its variants) on a quantum computer
- An analysis of Shor's algorithm indicates that the k-bit composite integer with 2 prime factors $n = pq$ can be factored in $O(k^3 \log k)$ operations using a $2k + 3$ bit (qubit) quantum computer

Security Levels Pre- and Post-Quantum

Public-key cryptography		Pre	Post
RSA-3072 [7]	encryption	128	broken (Shor)
RSA-3072 [7]	signature	128	broken (Shor)
DH-3072 [8]	key exchange	128	broken (Shor)
DSA-3072 [9, 10]	signature	128	broken (Shor)
256-bit ECDH [11, 12, 13]	key exchange	128	broken (Shor)
256-bit ECDSA [14, 15]	signature	128	broken (Shor)

Bernstein and Lange: https://eprint.iacr.org/2017/314

Post-Quantum Cryptography (PQC)

- The progress in quantum computer development indicates that the RSA, Diffie-Hellman and elliptic curve Diffie-Hellman cryptosystems might be broken within 10 to 20 years
- Thus, investigations on cryptographic algorithms that cannot be broken on quantum computers in polynomial time were initiated
- We are in a race against time to develop and deploy **post-quantum cryptography** before quantum computers arrive
- These terms are synonymous:
  Post-quantum cryptography
  Quantum-safe cryptography
  Quantum-computer-safe cryptography
  Quantum-resistant cryptography
  Quantum-computer-resistant cryptography

Changes in Cryptography

- There are other reasons for anticipating changes in crypto standards
- The existing standards (developed by NIST and adapted by various standard organizations) seem to have big security holes in them
- For example, it was shown in 2007 that an RNG which is used as default in several software packages may have a "back door"
- According to Snowden (published in the media in 2013):

  > *... the agency planted vulnerabilities in a standard adopted in 2006 by the NIST and later by the ISO, which has 163 countries as members. Classified NSA memos appear to confirm that the fatal weakness, discovered by cryptographers in 2007, was engineered by the agency.*

- Furthermore, evaluation of the security of the elliptic curves shows possibility manipulation in order to weaken security
  `https://bada55.cr.yp.to/pubs.html`

Quantum Cryptography

- Quantum Cryptography $\neq$ Post-Quantum Cryptography
- **Quantum cryptography** refers to quantum mechanical techniques to achieve communication secrecy or quantum key distribution
- With the help of quantum cryptography, various cryptographic operations can be performed, which would be impossible using only classical communication
- For example, it is impossible to copy data encoded in a quantum state
- This property could be used to detect eavesdropping in a channel
- The quantum key distribution offers an information-theoretic security to the key exchange problem

Standardization of Post-Quantum Cryptography

- Post-Quantum Cryptography Standardization is a project by National Institute of Standards and Technology (NIST) to standardize post-quantum cryptography
- It is a multiyear effort aimed at selecting and standardizing the next-generation of quantum-resistant cryptographic algorithms
- Efforts by NIST and NSA started as early as 2009
- https://www.nist.gov/pqcrypto

Timeline of Standardization of PQC

- 2009: NIST publishes a PQC survey (Perlner & Cooper)
- 2012: NIST begins PQC project (Team & Work with other orgs)
- Apr 2015: The 1st NIST PQC Workshop
- Feb 2016: NIST Report on PQC (NISTIR 8105)
- Dec 2016: Requirements and evaluation criteria published
- Nov 2017: Deadline for submissions
- Apr 2018: The 1st NIST PQC Standardization Workshop
- Aug 2019: The 2nd NIST PQC Standardization Workshop
- (Possibly) 2020: The 3rd NIST PQC Standardization Workshop
- 2022: Draft standards to be completed
- 2024: The PQC project to be concluded

Timeline of Standardization of PQC

- Apr 2018: The 1st Round Candidates: 64 complete

	Signatures	KEM/Encryption	Overall
Lattice-based	5	21	26
Code-based	2	17	19
Multi-variate	7	2	9
Symmetric-based	3		3
Other	2	5	7
Total	**19**	**45**	**64**

- Jan 2019: The 2nd Round Candidates: 26 complete

	Signatures		KEM/Encryption		Overall	
Lattice-based	~~5~~	3	~~21~~	9	~~26~~	12
Code-based	~~2~~	0	~~17~~	7	~~19~~	7
Multi-variate	~~7~~	4	~~2~~	0	~~9~~	4
Symmetric-based	~~3~~	2			~~3~~	2
Other	~~2~~	0	~~5~~	1	~~7~~	1
Total	~~19~~	**9**	~~45~~	**17**	~~64~~	**26**

Encryption/KEM Algorithms

Big Quake	Codes	Goppa
Classic McEliece	Codes	Goppa
NTS-KEM	Codes	Goppa
BIKE	Codes	short Hamming
HQC	Codes	short Hamming
LEDAkem	Codes	short Hamming
LEDApkc	Codes	short Hamming
QC-MDPC KEM	Codes	short Hamming
LAKE	Codes	low rank
LOCKER	Codes	low rank
Ouroboros-R	Codes	low rank
RQC	Codes	low rank
SIKE	Isogeny	Isogeny

→

Classic McEliece	Codes	Goppa
NTS-KEM	Codes	Goppa
BIKE	Codes	short Hamming
HQC	Codes	short Hamming
LEDAcrypt	Codes	short Hamming
Rollo	Codes	low rank
RQC	Codes	low rank
SIKE	Isogeny	Isogeny

Encryption/KEM Algorithms

Crystals-Kyber	Lattice	MLWE
KINDI	Lattice	MLWE
Saber	Lattice	MLWR
FrodoKEM	Lattice	LWE
Lotus	Lattice	LWE
Lizard	Lattice	LWE/RLWE
Emblem/R.emblem	Lattice	LWE/RLWE
KCL	Lattice	LWE/RLWE/LWR
Round 2	Lattice	LWR/RLWR
Hila5	Lattice	RLWE
Ding's key exchange	Lattice	RLWE
LAC	Lattice	RLWE
Lima	Lattice	RLWE
NewHope	Lattice	RLWE
Three Bears	Lattice	IMLWE
Mersenne-756839	Lattice	ILWE
Titanium	Lattice	MP-LWE
Ramstake	Lattice	LWE like
Odd Manhattan	Lattice	Generic
NTRU Encrypt	Lattice	NTRU
NTRU-HRSS-KEM	Lattice	NTRU
NTRUprime	Lattice	NTRU

Crystals-Kyber	Lattice	MLWE
Saber	Lattice	MLWR
FrodoKEM	Lattice	LWE
Round 5	Lattice	LWR/RLWR
LAC	Lattice	RLWE
NewHope	Lattice	RLWE
Three Bears	Lattice	IMLWE
NTRU	Lattice	NTRU
NTRUprime	Lattice	NTRU

Digital Signature Algorithms

Signatures		
CRYSTALS-Dilithium	Lattice	Fiat-Shamir
qTesla	Lattice	Fiat-Shamir
Falcon	Lattice	Hash then sign
pqNTRUSign	Lattice	Hash then sign
Gravity-SPHINCS	Symm	Hash
SPHINCS+	Symm	Hash
Picnic	Symm	ZKP
GeMMS	MultVar	HFE
Gui	MultVar	HFE
HiMQ-3	MultVar	UOV
LUOV	MultVar	UOV
Rainbow	MultVar	UOV
MQDSS	MultVar	Fiat-Shamir

Signatures		
CRYSTALS-Dilithium	Lattice	Fiat-Shamir
qTesla	Lattice	Fiat-Shamir
Falcon	Lattice	Hash then sign
SPHINCS+	Symm	Hash
Picnic	Symm	ZKP
GeMMS	MultVar	HFE
LUOV	MultVar	UOV
Rainbow	MultVar	UOV
MQDSS	MultVar	Fiat-Shamir

Competing PQC Families

- There are 5 competing families of PQC algorithms:
  - code-based encryption
  - isogeny encryption
  - lattice-based encryption and signatures
  - multivariate signatures
  - hash-based signatures
- The PQC includes a diverse set of algorithms based on different mathematical structures, properties, and one-way functions
- The PQC borrows some of the structures and tools from the traditional PKC and also brings in new methods and techniques
- The PQC algorithms are mathematically interesting and there are challenges in their high-speed implementations

Supersingular Isogeny Key Encapsulation

- An interesting and important algorithm in the NIST PQC list is Supersingular Isogeny Key Encapsulation (SIKE)
- The SIKE protocol specifies
  - Parameter sets
  - Key and ciphertext formats
  - Encapsulation and decapsulation mechanisms
  - Choice of symmetric primitives (hash functions)
- SIKE is based on Supersingular Isogeny Diffie-Hellman (SIDH)
- The work of De Fao, Jao, Plut, Childs, and Soukharev led to the discovery of the SIDH algorithm

Supersingular Isogeny Diffie-Hellman (SIDH)

- The traditional Diffie-Hellman (DH) establishes a secret key between two parties over on insecure channel
- The one-way function that makes DH work comes either from the prime field discrete logarithm problem or elliptic curve discrete logarithm problem over a prime or binary field
- The SIDH works with the set of (isomorphism classes of) supersingular elliptic curves and their isogenies
- The SIDH is based on the hardness of finding an isogeny map between two elliptic curves

Variations of Diffie-Hellman

	DH	ECDH	SIDH
elements	integers g modulo prime	points P in curve group	curves $\mathcal{E}$ in isogeny class
secrets	exponents x	scalars k	isogenies ϕ
computations	$g, x \mapsto g^x$	$k, P \mapsto [k]P$	$\phi, \mathcal{E} \mapsto \phi(\mathcal{E})$
hard problem	given g, g^x find x	given $P, [k]P$ find k	given $\mathcal{E}, \phi(\mathcal{E})$ find ϕ

SIDH Key Size

- The SIDH algorithm has the smallest public and private keys among known quantum-resistant algorithms
- The SIDH over 128-bit quantum security has a key size of 576 Bytes
- With key compression, the public key size is reduced to only 336 Bytes

Algorithm	NTRU [19]	New Hope [20]	McBits [21]	SIDH [7]	SIDH (with Compression) [11]
Type	Lattice	Ring-LWE	Code	Isogeny	Isogeny
Public Key	6,130	2,048	1,046,739	576	336
Private Key	6,743	2,048	10,992	48	48
Perfect Forward Secrecy	×	✓	×	✓	✓
Performance	Slow	Very Fast	Slow	Very Slow	Very Slow

Key sizes are in Bytes.

Supersingular Isogeny Diffie-Hellman

- The best-known algorithms against the SIDH protocol have an exponential time complexity for both classical and quantum attackers
- The SIDH algorithm uses conventional elliptic curve operations
- The SIDH algorithm uses only supersingular elliptic curves
- The SIDH algorithm also provides perfect forward secrecy which improves the long-term security of encrypted communications
- Compromise of a key does not affect the security of past communication

j-invariance and Isogenies

- The SIDH algorithm establishes the secret key by computing the j-invariant of two isomorphic supersingular elliptic curves generated by the two communicating parties that happens to be isogenous to an initial supersingular curve $\mathcal{E}_0$
- The j-invariant values are equal for two isomorphic curves
- Isomorphisms are a special case of isogenies where the kernel is trivial $\phi : \mathcal{E}_1 \rightarrow \mathcal{E}_2$ (the kernel is the point at infinity of $\mathcal{E}_1$)
- Endomorphisms are a special case of isogenies where the domain and co-domain are the same curve $\phi : \mathcal{E}_1 \rightarrow \mathcal{E}_1$
- Isogenies are *almost* isomorphisms

Supersingular Isogeny Diffie-Hellman

- Given two elliptic curves $\mathcal{E}_0$ and $\mathcal{E}_1$ defined over a finite field GF(q), an isogeny $\phi : \mathcal{E}_0 \to \mathcal{E}_1$ is a rational map defined over GF(q) such that ϕ is a group homomorphism from $\mathcal{E}_0$ to $\mathcal{E}_1$
- Two elliptic curves are isogenous over GF(q) if and only if they have the same cardinality
- Isogeny preserves the identity $\phi(\mathcal{O}_0) = \phi(\mathcal{O}_1)$
- Given finite subgroup $G \in \mathcal{E}_0$, there is a unique curve $\mathcal{E}_1$ and isogeny $\phi : \mathcal{E}_0 \to \mathcal{E}_1$ having kernel G

SIDH Domain (Public) Parameters

- A prime number of the form

$$p = (l_A)^{e_A}(l_B)^{e_B} f \pm 1$$

- Here l_A and l_B are small prime numbers, e_A and e_B are positive integers, and f is a small cofactor
- Choose a supersingular elliptic curve $\mathcal{E}$ over $\mathrm{GF}(p^2)$
- Fixed elliptic curve points P_A, Q_A, P_B, Q_B on $\mathcal{E}$
- The cardinality of $\mathcal{E}$ is $((l_A)^{e_A}(l_B)^{e_B} f)^2$
- The order of P_A and Q_A is $(l_A)^{e_A}$
- The order of P_B and Q_B is $(l_B)^{e_B}$

SIDH Key Exchange Steps

- Before the key exchange protocol starts the parties A and B will each create an isogeny from the shared elliptic curve $\mathcal{E}$
- The parties A and B each will do this by creating random points which will be the kernels of their isogeny
- The kernel of each isogeny will be spanned by the pairs of points P_A, Q_A and P_B, Q_B, respectively
- The different pairs of points used ensure that parties A and B create different, non-commuting, isogenies

SIDH Key Exchange Steps

- A pair of random points R_A and R_B are created as a random linear combination of the points P_A, Q_A and P_B, Q_B
- Parties A and B then use R_A and R_B and Velu's formulas to create isogenies ϕ_A and ϕ_B
- A computes the images of P_B and Q_B under ϕ_A
- B computes the images of P_A and Q_A under ϕ_B
- After the exchange step, both parties (simultaneously) compute an elliptic curve whose j-invariant becomes the shared key

Velu Formulas

- Given any finite subgroup of G of $\mathcal{E}$, we may form an isogeny $\phi : \mathcal{E} \to \mathcal{E}'$ with kernel G using Velu's formulas
- Example: $\mathcal{E} : y^2 = (x^2 + b_1 x + b_0)(x - a)$
- The point $(a, 0)$ has order 2
- The quotient of $\mathcal{E}$ with kernel $(a, 0)$ gives an isogeny $\phi : \mathcal{E} \to \mathcal{E}'$

$$\mathcal{E}' : y^2 = x^3 + (-(4a + 2b_1))x^2 + (b_1^2 - 4b_0)x$$

- The map ϕ takes (x, y) to

$$\left(\frac{x^3 - (a - b_1)x^2 - (b_1 a - b_0)x - b_0 a}{x - a} , \frac{(x^2 - (2a)x - (b_1 a + b_0))y}{(x - a)^2} \right)$$

SIDH Typical Parameters

Curve: $E_0/\mathbb{F}_{p^2}:\ y^2 = x^3 + x$

Prime	Classical/Quantum Security (bits)	Public Key Size (Bytes)	P_A	P_B
$p_{503} = 2^{250}3^{159} - 1$	125/83	378	$[3^{159}](14, \sqrt{14^3 + 14})$	$[2^{250}](6, \sqrt{6^3 + 6})$
$p_{751} = 2^{372}3^{239} - 1$	186/124	564	$[3^{239}](11, \sqrt{11^3 + 11})$	$[2^{372}](6, \sqrt{6^3 + 6})$
$p_{1019} = 2^{508}3^{319}35 - 1$	253/168	765	$[3^{319}35](13, \sqrt{13^3 + 13})$	$[2^{508}35](7, \sqrt{7^3 + 7})$
$p_{1533} = 2^{776}3^{477} - 1$	378/252	1,150	$[3^{477}](5, \sqrt{5^3 + 5})$	$[2^{776}](6, \sqrt{6^3 + 6})$

SIDH Key Exchange Steps $A \rightarrow B$

- A generates two random integers $m_A, n_A < (l_A)^{e_A}$
- A computes $R_A = [m_A]P_A + [n_A]Q_A$
- A uses R_A to create an isogeny mapping $\phi_A : \mathcal{E} \rightarrow \mathcal{E}_A$ and the elliptic curve $\mathcal{E}_A$ isogenous to $\mathcal{E}$
- A applies ϕ_A to P_B and Q_B to create $\phi_A(P_B)$ and $\phi_A(Q_B)$ on $\mathcal{E}_A$
- A sends $\mathcal{E}_A, \phi_A(P_B), \phi_A(Q_B)$ to B

SIDH Key Exchange Steps $B \rightarrow A$

- B generates two random integers $m_B, n_B < (l_B)^{e_B}$
- B computes $R_B = [m_B]P_B + [n_B]Q_B$
- B uses R_B to create an isogeny mapping $\phi_B : \mathcal{E} \rightarrow \mathcal{E}_B$ and the elliptic curve $\mathcal{E}_B$ isogenous to $\mathcal{E}$
- B applies ϕ_B to P_A and Q_A to create $\phi_B(P_A)$ and $\phi_B(Q_A)$ on $\mathcal{E}_B$
- B sends $\mathcal{E}_B, \phi_B(P_A), \phi_B(Q_A)$ to A

SIDH Key Exchange Steps by A

- A has $m_A, n_A, \phi_B(P_A), \phi_B(Q_A)$
- A computes $S_{BA} = [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A)$
- A creates the isogeny mapping ψ_{BA}
- A uses ψ_{BA} to create the elliptic curve $\mathcal{E}_{BA}$ which is isogenous to $\mathcal{E}$
- A computes the j-invariant of the curve $\mathcal{E}_{BA}$
- This number (or a part thereof) is the shared secret key

SIDH Key Exchange Steps by B

- B has $m_B, n_B, \phi_A(P_B), \phi_A(Q_B)$
- B computes $S_{AB} = [m_B]\phi_A(P_B) + [n_B]\phi_A(Q_B)$
- B creates the isogeny mapping ψ_{AB}
- B uses ψ_{AB} to create the elliptic curve $\mathcal{E}_{AB}$ which is isogenous to $\mathcal{E}$
- B computes the j-invariant of the curve $\mathcal{E}_{AB}$
- This number (or a part thereof) is the shared secret key

j-invariance and Isogenies

- Two isogenous elliptic curves have the same j-invariant values
- The j-invariant of an elliptic curve given by the Weierstrass equation $y^2 = x^3 + ax + b$ is calculated as

$$j = 1728 \, \frac{4a^3}{4a^3 + 27b^2}$$

- Similarly, the j-invariant of an elliptic curve given by the Montgomery equation $by^2 = x^3 + ax^2 + x$ is calculated as

$$j = 256 \, \frac{(a^2 - 3)^3}{a^2 - 4}$$

- This number (or a part thereof) is the shared secret key

j-invariance and Isogenies

- For example, consider the Weierstrass curves over GF(13)

$$\begin{aligned} \mathcal{E}_1 &: \quad y^2 = x^3 + 9x + 8 \\ \mathcal{E}_2 &: \quad y^2 = x^3 + 3x + 5 \end{aligned}$$

- They have the same j-invariant

$$j(\mathcal{E}_1) = \frac{1728 \cdot 4 \cdot 9^3}{4 \cdot 9^3 + 27 \cdot 8^2} = 3$$

$$j(\mathcal{E}_2) = \frac{1728\dot{4} \cdot 3^3}{4 \cdot 9^3 + 27 \cdot 5^2} = 3$$

Performance Issues for SIDH

- Initial analysis suggested primes of size 546 bytes for the SIDH protocol, however it was later reduced to 330 bytes
- Such key sizes produce slow runtime performance, on the order of milliseconds on high-end Intel processors
- This timing is significantly higher than the one achieved by several other quantum-resistant cryptosystem proposals
- Consequently recent focus is on devising strategies to reduce the runtime cost of the SIDH protocol
- For example, parallel evaluation of isogenies can be implemented on FPGA architectures

Arithmetic of SIDH

- The arithmetic of the SIDH elliptic curve is over GF(p^2)
- The arithmetic over the field GF(p^2) is implemented using the field arithmetic of GF(p)
- Thus, highly-optimized field arithmetic mod p is necessary for developing SIDH fast implementation
- The needed operations are finite field addition, squaring, multiplication, and inversion

Arithmetic of SIDH

- There have been several proposals to create efficient software implementations of the SIDH algorithm
- For the fast arithmetic computation inside the SIDH protocol (regardless of affine or projective formulas), it is more convenient to use the primes of the form $p = 2^{e_A} l^{e_B} \pm 1$
- The second prime l can also be selected as $l = 3$, however, larger primes ($l = 19$) were also suggested and used

Selecting Primes for SIDH

- The 751-bit prime $p = 2^{372} \cdot 3^{239} - 1$ provides 125-bit post-quantum security level, and it can be implemented on 64-bit platforms using 12 64-bit words
- The selection of the primes, the selection of the curve equation and the elliptic curve point representation (affine vs various projective) together yield efficient implementations of the SIDH algorithm
- On the other hand, the 964-bit prime $p = 2^{486} \cdot 3^{301} - 1$ is also implementation friendly and provides theoretical 160-bit post-quantum security

Selecting Primes for SIDH

- The $p = 2^{486} \cdot 3^{301} - 1$ has its 7 least significant 64-bit words all equal to 1s, which allows efficient Montgomery reduction

```
>>> bin(2**486*3**301-1)
'0b10000110101101011011111111110111011001000011110001100100111
10111101000010000000001010000001001001000101011010100111111000
10010110001010100001100101110101010011101011010001010100001111
11010010001001100101110101011001001000101000100110101010001101
00110111010101100001001001111110010101110001101101011010101001
01000000111100111111001010110111011110110101010100101011110101
00101000111010010011010101110110011100011101010101101000110011
11010110011011110010101000101000111001101010010111111111111111
11111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111'
```

Computing $[m]P + [n]Q$

- In order to reduce the running time of the SIDH protocol it is important to identify performance-critical operations
- Close inspection reveals that the SIDH algorithm a shared secret by performing a high number of elliptic curve and field arithmetic operations
- At each stage of the SIDH protocol, the parties A and B compute the kernel of an isogeny by calculating the point $[m]P + [n]Q$
- Here, P and Q are linearly independent points, and m and n are secret values

Computing $[m]P + [n]Q$

- The Montgomery ladder algorithm actually computes the operation $P + [k]Q$ given the points P, Q and the integer k
- The Montgomery ladder algorithm efficiently computes the x coordinate of $[k]Q = (x_2, y_2)$, after which the y coordinate is recovered y_2 using Okeya-Sakurai formula
- Then, we perform a projective addition of the points $P = (x_1 : y_1 : 1)$ and $[k]Q = (x_2 : y_2 : 1)$ to obtain the coordinates of $P + [k]Q$

Montgomery Ladder

- The Montgomery ladder algorithm computes $[k]P$

1: $(R_0, R_1) \leftarrow (\mathcal{O}, P)$
2: **for** $i = t - 1$ **downto** 0
2a: **if** $k_i = 1$ **then** $(R_0, R_1) \leftarrow (R_0 + R_1, [2]R_1)$
2b: **else** $(R_0, R_1) \leftarrow ([2]R_0, R_0 + R_1)$
3: **return** R_0

- Example: $k = 13 = (1101)_2$

		$k_3 = 1$	$k_2 = 1$	$k_1 = 0$	$k_0 = 1$
R_0:	$\mathcal{O}$	P	$[3]P$	$[6]P$	$\boxed{[13]P}$
R_1:	P	$[2]P$	$[4]P$	$[7]P$	$[14]P$

Montgomery and De Fao Three-Point Ladders

- The Montgomery ladder a left-to-right algorithm, since it computes $[k[P$ by scanning the bits of the scalar k from the most-significant to the least-significant bit
- A right-to-left evaluation of the Montgomery ladder also exists and was recently applied to the Montgomery curves
- **De Fao suggested** instead to compute $P + [m^{-1}n]Q$ for a scalar m that has multiplicative inverse mod $l_A^{e_A}$ or mod $l_B^{e_B}$
- The right-to-left Montgomery ladder algorithm can be combined with the three-point ladder procedure introduced by De Fao
- This algorithm computes the x coordinate of $P + [k]Q$ given the x coordinates of the points P, Q, and $Q - P$

De Fao Three-Point Ladder

- The De Fao three-point ladder algorithm computes $P + [k]Q$

  1: $(R_0, R_1, R_2) \leftarrow (Q, P, Q - P)$
  2: **for** $i = 0$ **to** $t - 1$
  2a: **if** $k_i = 1$ **then** $R_1 \leftarrow R_0 + R_1$
  2b: **else** $R_2 \leftarrow R_0 + R_2$
  3: $R_0 \leftarrow [2]R_0$
  4: **return** R_1

- Example: $k = 13 = (1101)_2$

		$k_0 = 1$	$k_1 = 0$	$k_2 = 1$	$k_3 = 1$
R_0:	Q	$[2]Q$	$[4]Q$	$[8]Q$	$[16]Q$
R_1:	P	$P + Q$		$P + [5]Q$	$\boxed{P + [13]Q}$
R_2:	$Q - P$		$[3]Q - P$		

De Fao Three-Point Ladder

- In the case the point Q is available in advance, and a table lookup approach can be utilized the computation of $[k]Q$
- We compute $T_i = [2^i]Q$ in advance and save them in a table
- They are then used in the De Fao three-point ladder algorithm

1: $(R_1, R_2) \leftarrow (P, Q - P)$
2: **for** $i = 0$ **to** $t - 1$
2a: **if** $k_i = 1$ **then** $R_1 \leftarrow T_i + R_1$
2b: **else** $R_2 \leftarrow T_i + R_2$
3: **return** R_1

De Fao Three-Point Ladder

- Example for $k = 12 = (01100)_2$

$$
\begin{tikzcd}
 & R_1 = P \arrow[d, equal] & R_0 = Q \arrow[d, Rightarrow] \arrow[dr] & R_2 = Q - P \arrow[d] \\
k_0 = 0 & P \arrow[d, equal] & 2Q \arrow[d, Rightarrow] \arrow[dr] & 2Q - P \arrow[d] \\
k_1 = 0 & P \arrow[d] & 4Q \arrow[d, Rightarrow] \arrow[dl] & 4Q - P \arrow[d, equal] \\
k_2 = 1 & P + 4Q \arrow[d] & 8Q \arrow[d, Rightarrow] \arrow[dl] & 4Q - P \arrow[d, equal] \\
k_3 = 1 & P + 12Q \arrow[d, equal] & 16Q \arrow[d, Rightarrow] \arrow[dr] & 4Q - P \arrow[d] \\
k_4 = 0 & P + 12Q & 32Q & 20Q - P
\end{tikzcd}
$$

Various SIDH Implementations

Work	Quantum Security (bits)	Platform	Smooth Isogeny Prime	Time (ms)				
				Alice Rnd. 1	Bob Rnd. 1	Alice Rnd. 2	Bob Rnd. 2	Total Time
~85-bit Quantum Security Level								
Jao and De Feo [5]	84	2.4 GHz Opt.	$2^{253}3^{161}7-1$	365	318	363	314	1360
Jao et al. [6]	85	2.4 GHz Opt.	$2^{258}3^{161}186-1$	28.1	28.0	23.3	22.7	102.1
Azarderakhsh et al. [10]	85	4.0 GHz i7	$2^{258}3^{161}186-1$	-	-	-	-	54.0
Koziel et al. [16]	84	Virtex-7	$2^{253}3^{161}7-1$	9.35	8.41	8.53	7.41	33.70
Koziel et al. [17]	83	Virtex-7	$2^{250}3^{159}-1$	4.83	5.25	4.41	4.93	19.42
This Work (12 Mults.)	83	Virtex-7	$2^{250}3^{159}-1$	3.59	3.87	3.22	3.53	14.22
~128-bit Quantum Security Level								
Jao et al. [6]	128	2.4 GHz Opt.	$2^{387}3^{242}-1$	65.7	54.3	65.6	53.7	239.3
Azarderakhsh et al. [10]	128	4.0 GHz i7	$2^{387}3^{242}-1$	-	-	-	-	133.7
Costello et al. [7]	124	3.4 GHz i7	$2^{372}3^{239}-1$	15.0	17.3	13.8	16.8	62.9
Koziel et al. [17]	124	Virtex-7	$2^{372}3^{239}-1$	10.6	11.6	9.5	10.8	42.5
This Work (12 Mults.)	124	Virtex-7	$2^{372}3^{239}-1$	7.99	8.63	7.14	7.86	31.61
~170-bit Quantum Security Level								
Jao et al. [6]	170	2.4 GHz Opt.	$2^{514}3^{323}353-1$	122	101	125	102	450
Azarderakhsh et al. [10]	170	4.0 GHz i7	$2^{514}3^{323}353-1$	-	-	-	-	266.9
This Work (12 Mults.)	168	Virtex-7	$2^{508}3^{319}35-1$	14.97	15.72	13.43	14.28	58.4

IEEE Transactions on Computers, November 2018