Post-Quantum Isogeny-based Cryptography on ARM processors

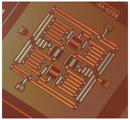
Reza Azarderakhsh

Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
PQSecure Technologies

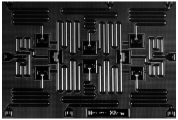
ARM Research Summit 2018

Why Quantum Computing? Why now?

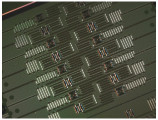
- The history of Integrated Circuits (IC)
  - 1958: First integrated circuit ($1cm^2$, 2 transistors)
  - 1971: Moore's Law is born (2,300 transistors)
  - 2014: IBM P8 Processor, 16 cores ($650mm^2$, $>$ 4.2 billion transistors)
- Quantum Computers[1]



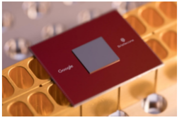
2015: 4-Qbit



2016: 8-Qbit

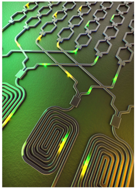


2017: 16-Qbit



2018: 72-Qbit

- Photon-based Quantum Computers are under construction!



[1] Pictures are taken from IBM Q Project

Quantum Computers

- Quantum Computers will be able to solve many problems in different areas:

[2] IBM's 50-bit Quantum Computer

Quantum Computers

- Quantum Computers will be able to solve many problems in different areas:
  - Weather prediction

[2] IBM's 50-bit Quantum Computer

Quantum Computers

- Quantum Computers will be able to solve many problems in different areas:
  - Weather prediction
  - Medical and healthcare

[2] IBM's 50-bit Quantum Computer

Quantum Computers

- Quantum Computers will be able to solve many problems in different areas:
  - Weather prediction
  - Medical and healthcare
  - Machine learning

[2] IBM's 50-bit Quantum Computer

Quantum Computers

- Quantum Computers will be able to solve many problems in different areas:
  - Weather prediction
  - Medical and healthcare
  - Machine learning
  - Many other NP problems can be solved in polynomial-time ☺

[2] IBM's 50-bit Quantum Computer

Quantum Computers

- Quantum Computers will be able to solve many problems in different areas:
  - Weather prediction
  - Medical and healthcare
  - Machine learning
  - Many other NP problems can be solved in polynomial-time ☺
- Current PKC is also constructed on NP problems!

[2] IBM's 50-bit Quantum Computer

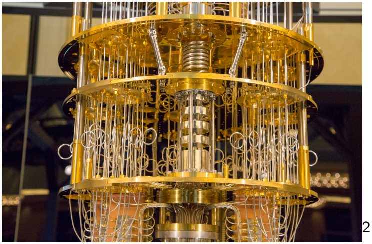
Quantum Computers

- Quantum Computers will be able to solve many problems in different areas:
  - Weather prediction
  - Medical and healthcare
  - Machine learning
  - Many other NP problems can be solved in polynomial-time ☺
- Current PKC is also constructed on NP problems!
  - RSA: Discrete Logarithm Problem (DLP)
  - ECC: Elliptic Curve Discrete Logarithm Problem (ECDLP)

[2] IBM's 50-bit Quantum Computer

Quantum Computers

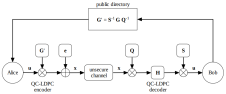
- Quantum Computers will be able to solve many problems in different areas:
  - Weather prediction
  - Medical and healthcare
  - Machine learning
  - Many other NP problems can be solved in polynomial-time ☺
- Current PKC is also constructed on NP problems!
  - RSA: Discrete Logarithm Problem (DLP)
  - ECC: Elliptic Curve Discrete Logarithm Problem (ECDLP)
  - Shor's quantum algorithm can solve these problems in polynomial-time ☹


[2]

[2] IBM's 50-bit Quantum Computer

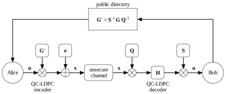
Primary PQC Candidates

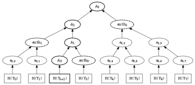
- Code-Based: McEliece



Primary PQC Candidates

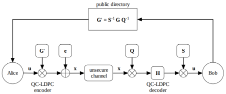
- Code-Based: McEliece
- Hash-Based: Lamport - Merkle Signatur

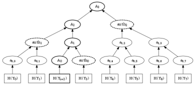


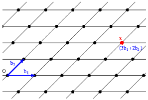


Primary PQC Candidates

- Code-Based: McEliece
- Hash-Based: Lamport - Merkle Signatur
- Lattice-Based: NTRU - LWE

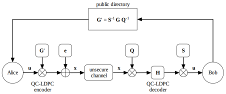




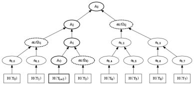


Primary PQC Candidates

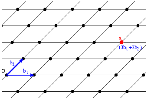
- Code-Based: McEliece



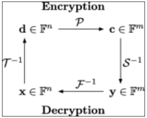
- Hash-Based: Lamport - Merkle Signatur



- Lattice-Based: NTRU - LWE

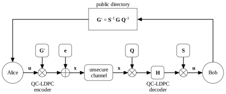


- Multivariate: Rainbow Signatures

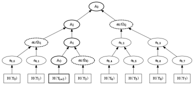


Primary PQC Candidates

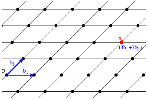
- Code-Based: McEliece



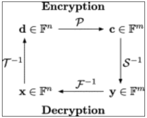
- Hash-Based: Lamport - Merkle Signatur



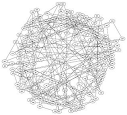
- Lattice-Based: NTRU - LWE



- Multivariate: Rainbow Signatures

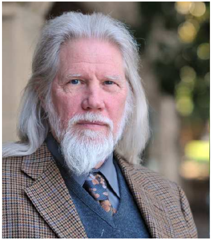


- Isogeny-Based: SIKE

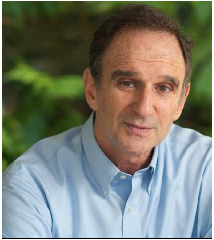


Diffie-Hellman Key-Exchange Protocol

Secret Key
Public Key



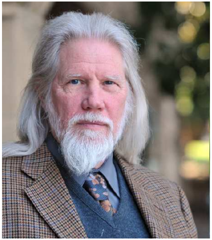
Secret Key
Public Key

[3]

[3] Pictures are taken from https://news.bitcoin.com

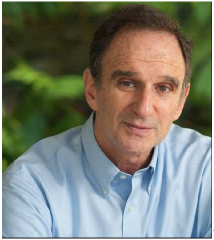
Diffie-Hellman Key-Exchange Protocol

Secret Key
Public Key



Public Key →

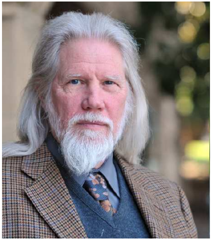
Secret Key
Public Key


[3]

[3] Pictures are taken from https://news.bitcoin.com

Diffie-Hellman Key-Exchange Protocol

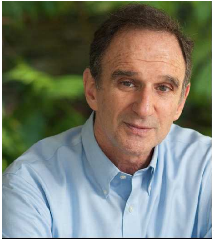
Secret Key
Public Key



Public Key →

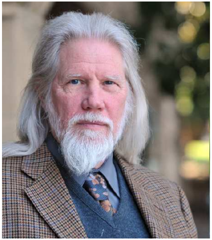
← Public Key

Secret Key
Public Key


[3]

[3] Pictures are taken from https://news.bitcoin.com

Secret Key
Public Key

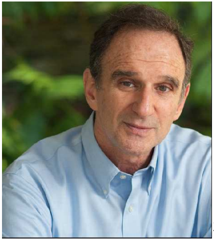


Public Key →

← Public Key

↔ Shared Key

Secret Key
Public Key


[3]

[3] Pictures are taken from https://news.bitcoin.com

Classical ECC vs. Post-Quantum Isogeny Cryptography

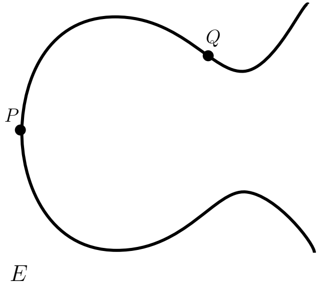


Figure: Classical Elliptic Curve Cryptography

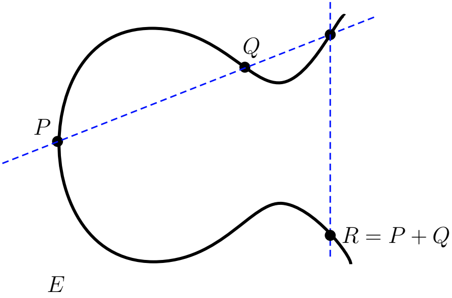


Figure: Classical Elliptic Curve Cryptography

Classical ECC vs. Post-Quantum Isogeny Cryptography

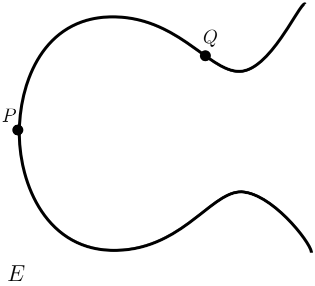


Figure: Post-Quantum Isogeny-based Cryptography

Classical ECC vs. Post-Quantum Isogeny Cryptography

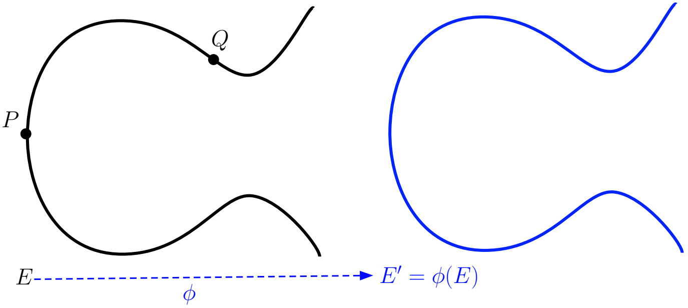


Figure: Post-Quantum Isogeny-based Cryptography

Classical ECC vs. Post-Quantum Isogeny Cryptography

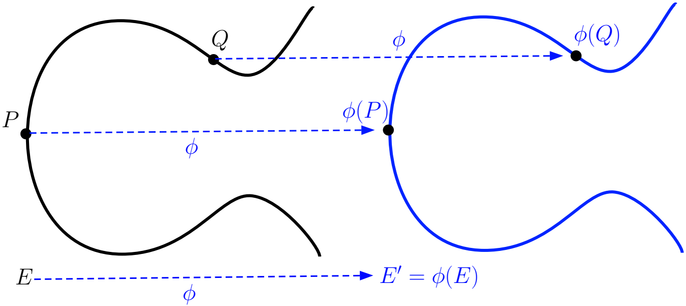


Figure: Post-Quantum Isogeny-based Cryptography

- The first suggestions to use isogenies in crypto by Couveignes in 1997.

Supersingular Isogeny-Based Cryptography History

- The first suggestions to use isogenies in crypto by Couveignes in 1997.

- Supersingular isogeny hash function by Charles, Lauter and Goren in 2005.

- The first suggestions to use isogenies in crypto by Couveignes in 1997.

- Supersingular isogeny hash function by Charles, Lauter and Goren in 2005.

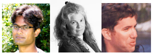
  

- Isogeny-based public-key cryptosystems by Rostovtsev and Stolbunov in 2006.

Supersingular Isogeny-Based Cryptography History

- The first suggestions to use isogenies in crypto by Couveignes in 1997.

- Supersingular isogeny hash function by Charles, Lauter and Goren in 2005.



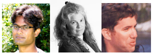
- Isogeny-based public-key cryptosystems by Rostovtsev and Stolbunov in 2006.

- The biggest impetus by Jao, De Feo (SIDH) in 2011.

Supersingular Isogeny-Based Cryptography History

- The first suggestions to use isogenies in crypto by Couveignes in 1997.

- Supersingular isogeny hash function by Charles, Lauter and Goren in 2005.



- Isogeny-based public-key cryptosystems by Rostovtsev and Stolbunov in 2006.

- The biggest impetus by Jao, De Feo (SIDH) in 2011.

- Supersingular Isogeny Key Encapsulation (SIKE) by Jao et al. submitted to NIST PQC Standardization 2017.

Supersingular Isogeny-Based Cryptography Underlying Problem

- Consider two supersingular elliptic curves defined over a large prime extension field: $E1/\mathbb{F}_{p^2}$ and $E2/\mathbb{F}_{p^2}$, where p is a large prime.

Supersingular Isogeny-Based Cryptography Underlying Problem

- Consider two supersingular elliptic curves defined over a large prime extension field: $E1/\mathbb{F}_{p^2}$ and $E2/\mathbb{F}_{p^2}$, where p is a large prime.
- There exists some isogeny $\phi : E_1 \rightarrow E_2$ with a fixed, smooth degree ℓ that is public which maps E_1 to E_2

- Consider two supersingular elliptic curves defined over a large prime extension field: $E1/\mathbb{F}_{p^2}$ and $E2/\mathbb{F}_{p^2}$, where p is a large prime.
- There exists some isogeny $\phi : E_1 \rightarrow E_2$ with a fixed, smooth degree ℓ that is public which maps E_1 to E_2

Supersingular Isogeny Problem

Given $P, Q \in E_1$ and $\phi(P), \phi(Q) \in E_2$, retrieve the secret isogeny map ϕ

- Consider two supersingular elliptic curves defined over a large prime extension field: $E1/\mathbb{F}_{p^2}$ and $E2/\mathbb{F}_{p^2}$, where p is a large prime.
- There exists some isogeny $\phi : E_1 \rightarrow E_2$ with a fixed, smooth degree ℓ that is public which maps E_1 to E_2

Supersingular Isogeny Problem

Given $P, Q \in E_1$ and $\phi(P), \phi(Q) \in E_2$, retrieve the secret isogeny map ϕ

- The best known quantum attack is based on Claw's finding algorithm

Supersingular Isogeny-Based Cryptography Underlying Problem

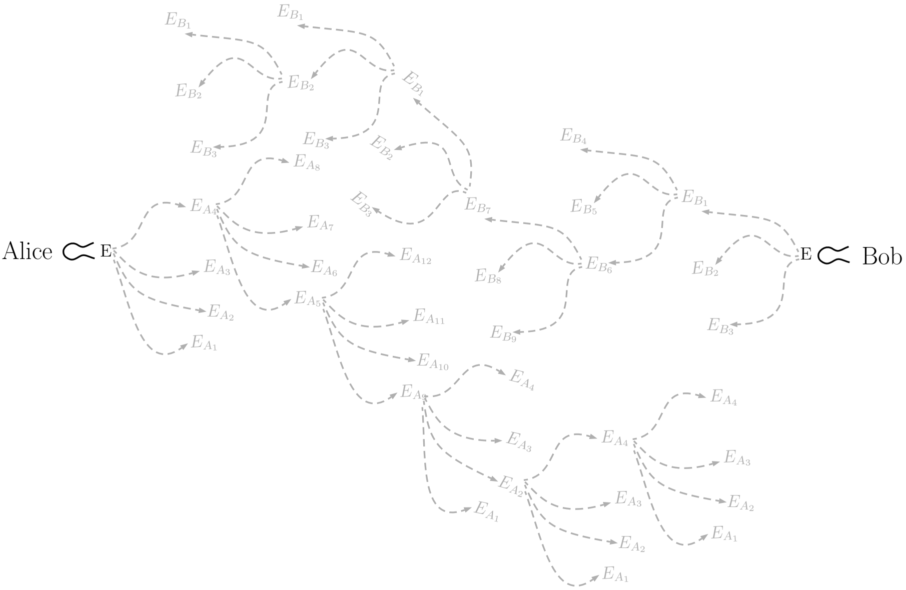
- Consider two supersingular elliptic curves defined over a large prime extension field: $E1/\mathbb{F}_{p^2}$ and $E2/\mathbb{F}_{p^2}$, where p is a large prime.
- There exists some isogeny $\phi : E_1 \rightarrow E_2$ with a fixed, smooth degree ℓ that is public which maps E_1 to E_2

Supersingular Isogeny Problem

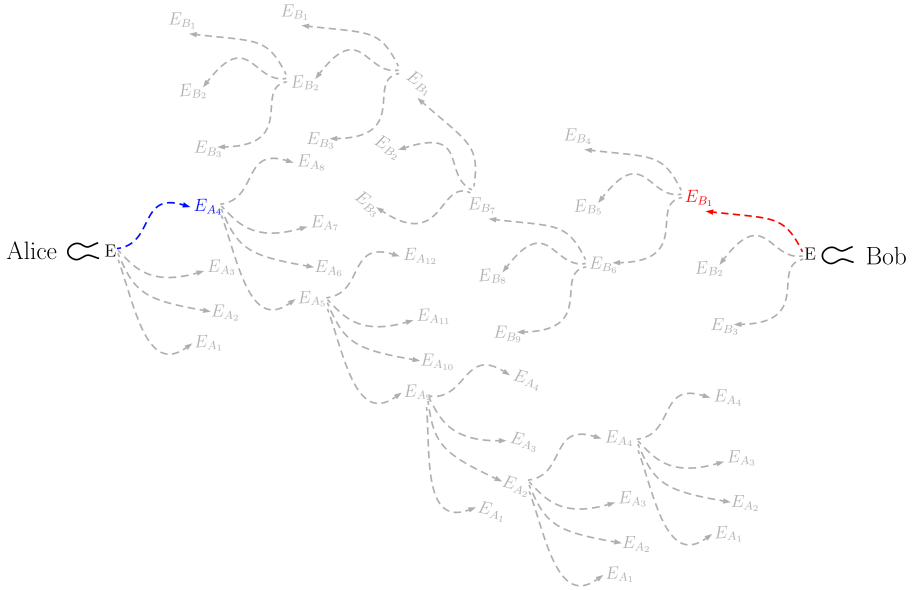
Given $P, Q \in E_1$ and $\phi(P), \phi(Q) \in E_2$, retrieve the secret isogeny map ϕ

- The best known quantum attack is based on Claw's finding algorithm
- Claw finding algorithm complexity for SIKE and SIDH:
  - $\mathcal{O}(p^{1/6}) \rightarrow$Quantum attacks
- The best known classical attack is based on meet in the middle
  - $\mathcal{O}(p^{1/4}) \rightarrow$Classical attacks

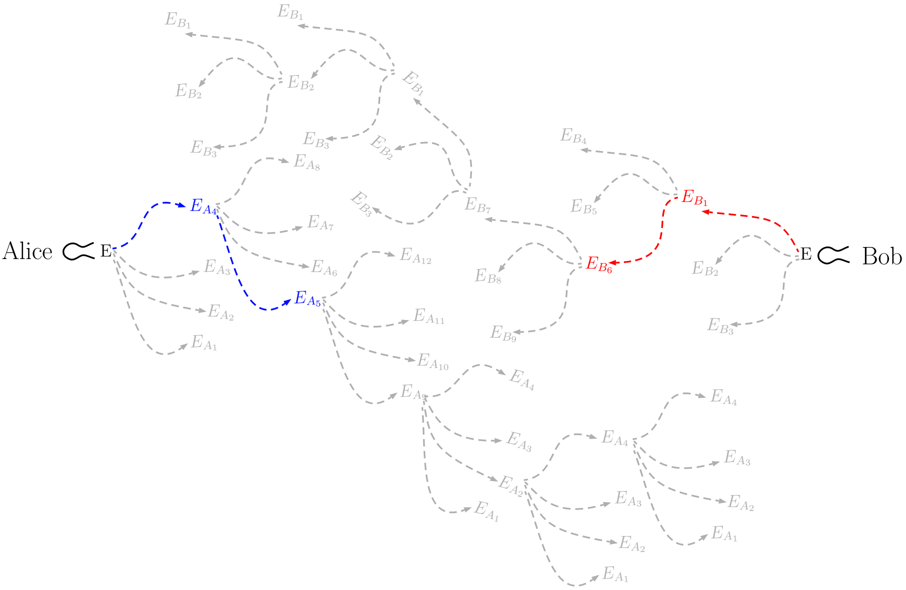
Alice and Bob Isogeny Walks from Different Degree Isogenies



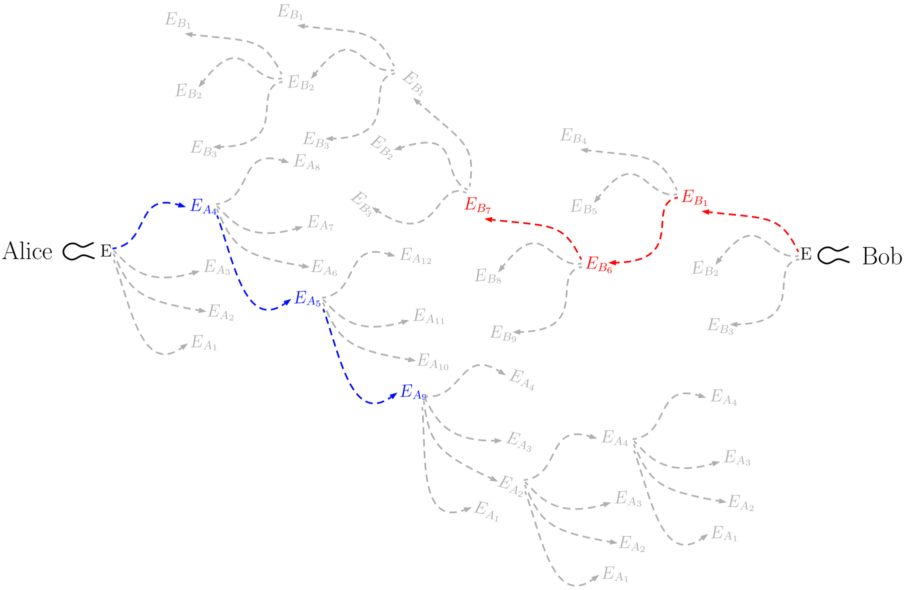
Alice and Bob Isogeny Walks from Different Degree Isogenies



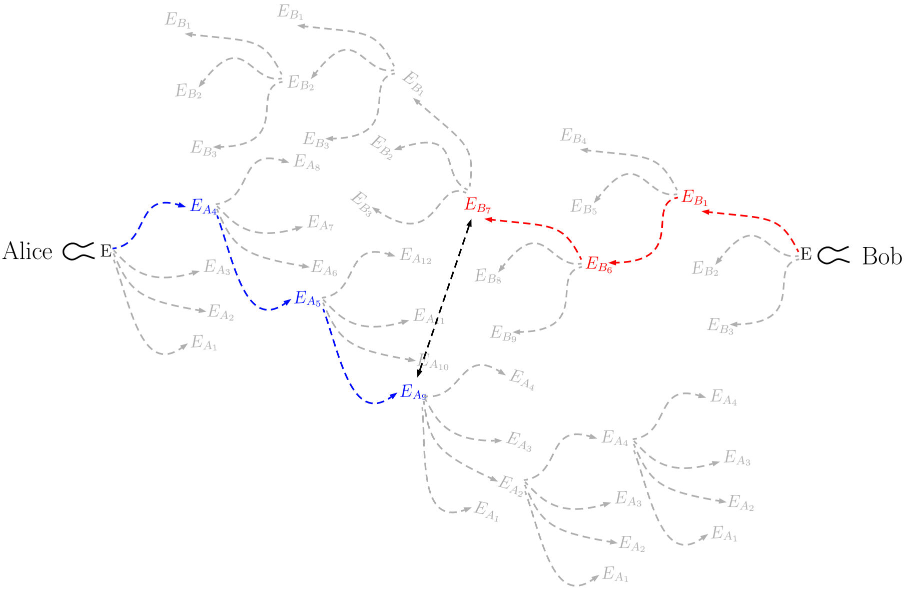
Alice and Bob Isogeny Walks from Different Degree Isogenies



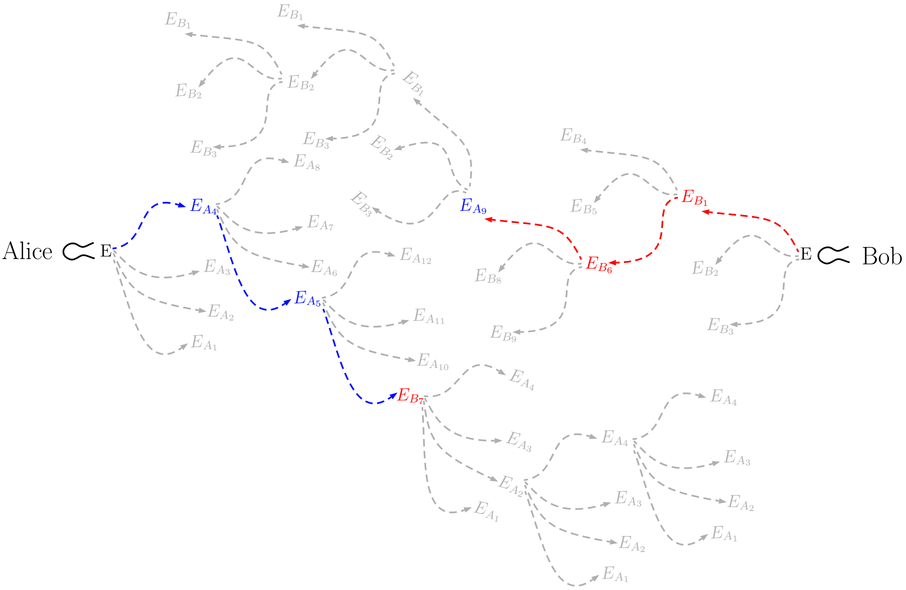
Alice and Bob Isogeny Walks from Different Degree Isogenies



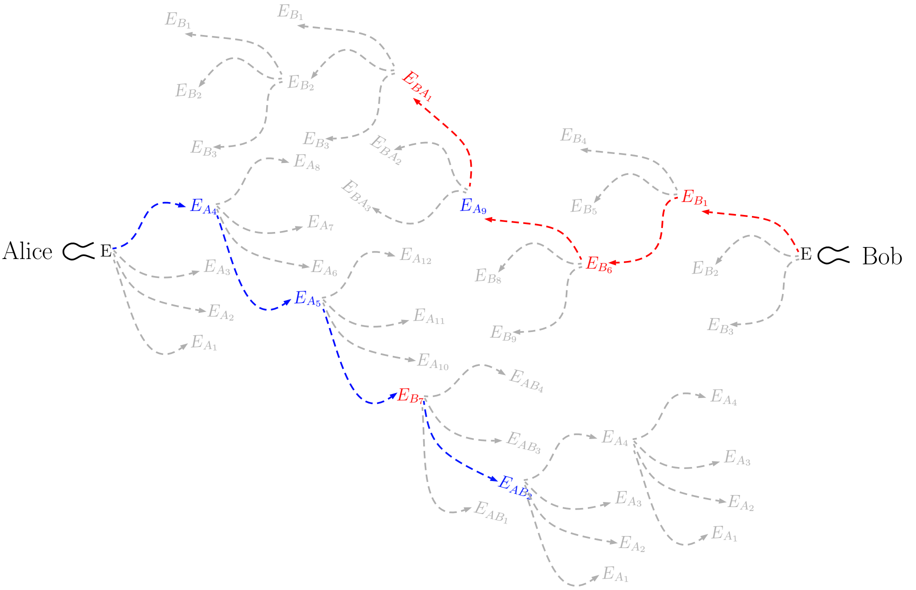
Alice and Bob Isogeny Walks from Different Degree Isogenies



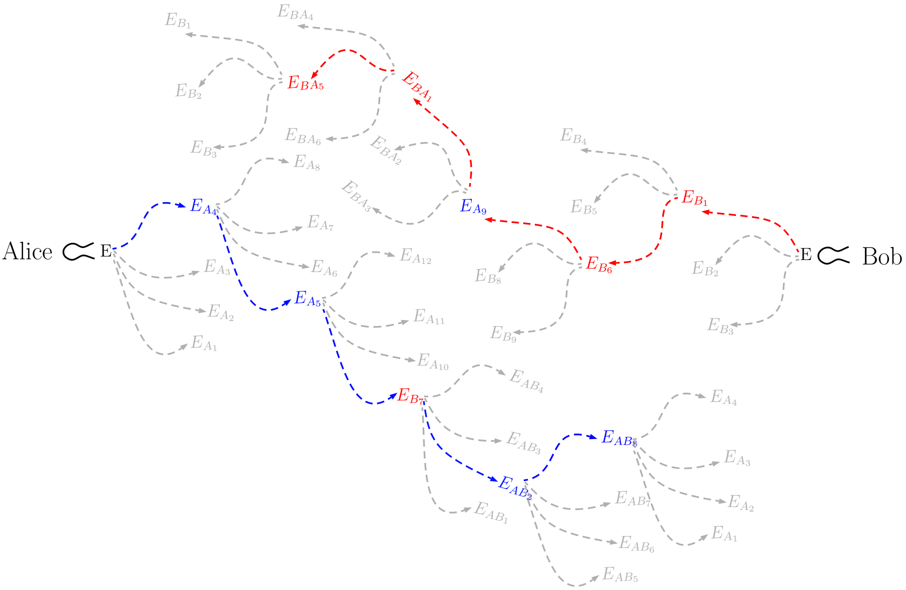
Alice and Bob Isogeny Walks from Different Degree Isogenies



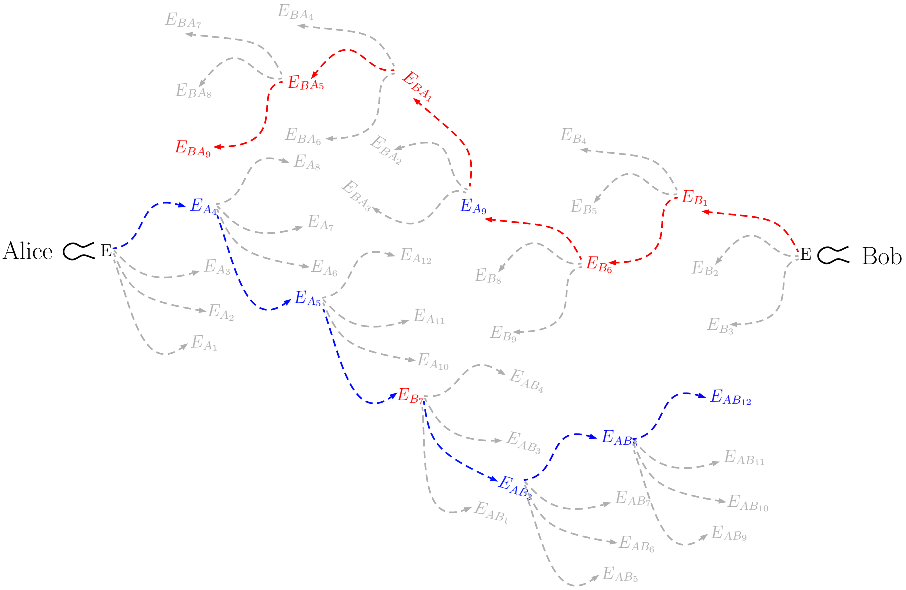
Alice and Bob Isogeny Walks from Different Degree Isogenies



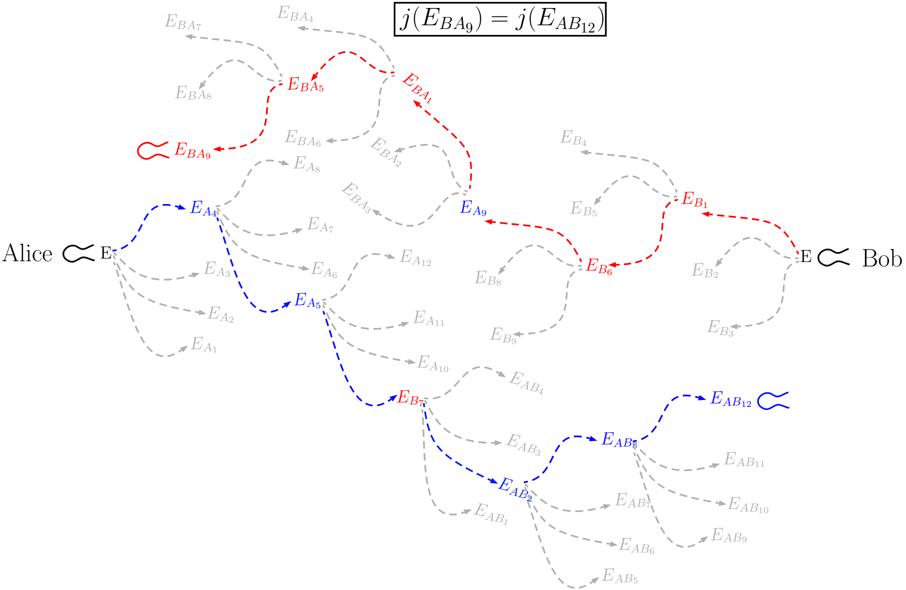
Alice and Bob Isogeny Walks from Different Degree Isogenies



Alice and Bob Isogeny Walks from Different Degree Isogenies



Alice and Bob Isogeny Walks from Different Degree Isogenies

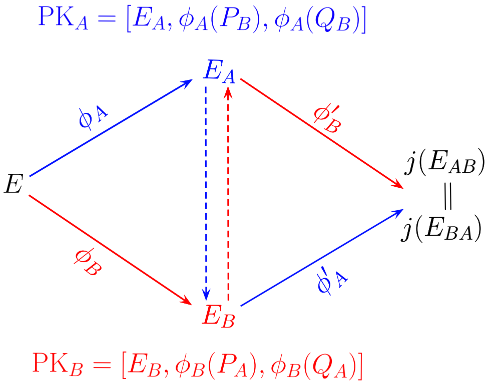


Public Parameters

E, p

$P_A, Q_A \in E$

$P_B, Q_B \in E$



Supersingular Isogeny-Based Cryptography Pros and Cons

- Pros
  - Very small public/private key size.

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.

Supersingular Isogeny-Based Cryptography Pros and Cons

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.
  - No complicated error distribution, rejection sampling, etc.

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.
  - No complicated error distribution, rejection sampling, etc.
  - Conservative security analysis on generic attacks.

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.
  - No complicated error distribution, rejection sampling, etc.
  - Conservative security analysis on generic attacks.
- Cons
  - Youngest PQC candidate.

Supersingular Isogeny-Based Cryptography Pros and Cons

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.
  - No complicated error distribution, rejection sampling, etc.
  - Conservative security analysis on generic attacks.

- Cons
  - Youngest PQC candidate.
  - Slower compared to some other candidates.

Supersingular Isogeny-Based Cryptography Pros and Cons

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.
  - No complicated error distribution, rejection sampling, etc.
  - Conservative security analysis on generic attacks.

- Cons
  - Youngest PQC candidate.
  - Slower compared to some other candidates.
  - Security concerns when reuse keys.

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.
  - No complicated error distribution, rejection sampling, etc.
  - Conservative security analysis on generic attacks.
- Cons
  - Youngest PQC candidate.
  - Slower compared to some other candidates.
  - Security concerns when reuse keys.
  - New schemes based on isogeny-based cryptography needs to be analyzed on practical settings.

Supersingular Isogeny-Based Cryptography Pros and Cons

- Pros
  - Very small public/private key size.
  - Data-structure and implementation similar to ECC.
  - Different security assumption compared to other candidates.
  - No possibility of decryption error.
  - No complicated error distribution, rejection sampling, etc.
  - Conservative security analysis on generic attacks.
- Cons
  - Youngest PQC candidate.
  - Slower compared to some other candidates.
  - Security concerns when reuse keys.
  - New schemes based on isogeny-based cryptography needs to be analyzed on practical settings.

Small Key-Size Makes it Suitable for Embedded Devices

Table: Communication bandwidth of some NIST PQC candidate KEMs in terms of public-key, secret-key, and transmitted ciphertext during the key encapsulation process.

Candidate	Primitive	Size (Bytes)		
		Public key	Secret key	Ciphertext
NewHope1024	RLWE	1824	3680	2208
Saber	Mod-LWR	992	2304	1088
NTRU-HRSS17	LWE	1138	1418	1278
Kyber-768	LWE	1088	2400	1152
NTRU Prime	RLWE	1218	1600	1047
SIKEp751	SI	**564**	**644**	**596**

Isogeny-based Cryptography Implementation Perspective

- Cryptography protocols deal with big integers → field arithmetic

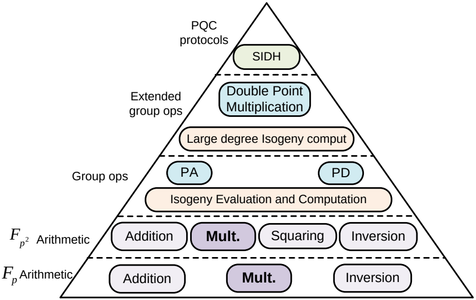
- Cryptography protocols deal with big integers $\rightarrow$ field arithmetic
- From top to bottom, the number of operations increases

Isogeny-based Cryptography Implementation Perspective

- Cryptography protocols deal with big integers $\rightarrow$ field arithmetic
- From top to bottom, the number of operations increases
- Optimization on the lowest level operations

Isogeny-based Cryptography Implementation Perspective

- Cryptography protocols deal with big integers $\rightarrow$ field arithmetic
- From top to bottom, the number of operations increases
- Optimization on the lowest level operations



Different Families of Processors:

- ARMv7-M → 32-bit Low-Power (Performance is challenging)
- ARMv7-A → 32-bit High-Performance with NEON Instruction set
- ARMv8-A → 64-bit High-Performance with Adv. SIMD instruction set

[4]

[5]

Figure: ARMv7-A Cortex-A15 (Jetson TK1 Board) and ARMv8-A Cortex-A57 (Nexus smartphone)

[4] Taken from https://developer.nvidia.com

[5] Taken from https://www.huawei.com

- Using divide-and-conquer strategy for big integer multiplication

- Using divide-and-conquer strategy for big integer multiplication
- Replace one $n-$bit multiplication with $3 \times \frac{n}{2}$-bit multiplications

- Using divide-and-conquer strategy for big integer multiplication
- Replace one $n-$bit multiplication with $3 \times \frac{n}{2}$-bit multiplications
- Replace $\frac{n}{2}$ multiplications with $3 \times \frac{n}{4}$-bit multiplications

- Using divide-and-conquer strategy for big integer multiplication
- Replace one $n-$bit multiplication with $3 \times \frac{n}{2}$-bit multiplications
- Replace $\frac{n}{2}$ multiplications with $3 \times \frac{n}{4}$-bit multiplications
    - Two-level Additive Karatsuba multiplication:

- Using divide-and-conquer strategy for big integer multiplication
- Replace one $n-$bit multiplication with $3 \times \frac{n}{2}$-bit multiplications
- Replace $\frac{n}{2}$ multiplications with $3 \times \frac{n}{4}$-bit multiplications
  - Two-level Additive Karatsuba multiplication:

$$A.B = A_h B_h 2^n + [(A_h + A_l)(B_h + B_l) - A_h B_h A_l B_l] 2^{\frac{n}{2}} + A_l B_l.$$

- Using divide-and-conquer strategy for big integer multiplication
- Replace one $n-$bit multiplication with $3 \times \frac{n}{2}$-bit multiplications
- Replace $\frac{n}{2}$ multiplications with $3 \times \frac{n}{4}$-bit multiplications
  - Two-level Additive Karatsuba multiplication:

$$A.B = A_h B_h 2^n + [(A_h + A_l)(B_h + B_l) - A_h B_h A_l B_l] 2^{\frac{n}{2}} + A_l B_l.$$

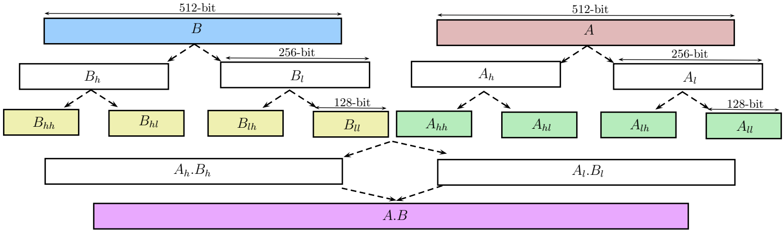
$$A_h.B_h = A_{hh} B_{hh} 2^{\frac{n}{2}} + [(A_{hh} + A_{hl})(B_{hh} + B_{hl}) - A_{hh} B_{hh} - A_{hl} B_{hl}] 2^{\frac{n}{4}} + A_{hl} B_{hl}$$

$$A_l.B_l = A_{lh} B_{lh} 2^{\frac{n}{2}} + [(A_{lh} + A_{ll})(B_{lh} + B_{ll}) - A_{lh} B_{lh} - A_{ll} B_{ll}] 2^{\frac{n}{4}} + A_{ll} B_{ll}.$$

- Using divide-and-conquer strategy for big integer multiplication
- Replace one $n-$bit multiplication with $3 \times \frac{n}{2}$-bit multiplications
- Replace $\frac{n}{2}$ multiplications with $3 \times \frac{n}{4}$-bit multiplications
  - Two-level Additive Karatsuba multiplication:

$$A.B = A_hB_h2^n + [(A_h + A_l)(B_h + B_l) - A_hB_hA_lB_l]2^{\frac{n}{2}} + A_lB_l.$$

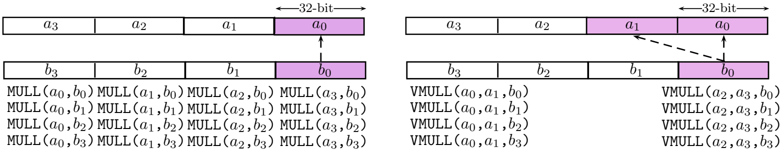
$$A_h.B_h = A_{hh}B_{hh}2^{\frac{n}{2}} + [(A_{hh} + A_{hl})(B_{hh} + B_{hl}) - A_{hh}B_{hh} - A_{hl}B_{hl}]2^{\frac{n}{4}} + A_{hl}B_{hl}$$
$$A_l.B_l = A_{lh}B_{lh}2^{\frac{n}{2}} + [(A_{lh} + A_{ll})(B_{lh} + B_{ll}) - A_{lh}B_{lh} - A_{ll}B_{ll}]2^{\frac{n}{4}} + A_{ll}B_{ll}.$$



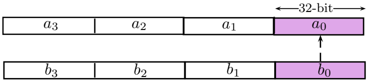
- Exploit SIMD capabilities of ARMv7-A cores for arithmetic implementation
    - NEON Assembly Implementation:

- Exploit SIMD capabilities of ARMv7-A cores for arithmetic implementation
  - NEON Assembly Implementation:
    - SIMD multiplication instructions reduce the total number of multiplications significantly

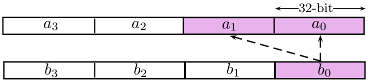
- Exploit SIMD capabilities of ARMv7-A cores for arithmetic implementation
  - NEON Assembly Implementation:
    - SIMD multiplication instructions reduce the total number of multiplications significantly
    - 128 $\times$ 128-bit multiplication using A32 and NEON:



- Exploit SIMD capabilities of ARMv7-A cores for arithmetic implementation
  - NEON Assembly Implementation:
    - SIMD multiplication instructions reduce the total number of multiplications significantly
    - 128 $\times$ 128-bit multiplication using A32 and NEON:



MULL(a_0,b_0)	MULL(a_1,b_0)	MULL(a_2,b_0)	MULL(a_3,b_0)
MULL(a_0,b_1)	MULL(a_1,b_1)	MULL(a_2,b_1)	MULL(a_3,b_1)
MULL(a_0,b_2)	MULL(a_1,b_2)	MULL(a_2,b_2)	MULL(a_3,b_2)
MULL(a_0,b_3)	MULL(a_1,b_3)	MULL(a_2,b_3)	MULL(a_3,b_3)



VMULL(a_0,a_1,b_0)	VMULL(a_2,a_3,b_0)
VMULL(a_0,a_1,b_1)	VMULL(a_2,a_3,b_1)
VMULL(a_0,a_1,b_2)	VMULL(a_2,a_3,b_2)
VMULL(a_0,a_1,b_3)	VMULL(a_2,a_3,b_3)

- 16 $\times$ `MULL` instructions in A32 vs. 8 $\times$ `VMULL` in NEON Vector Instructions.

- SIDH performance evaluation on different families of ARM processors
- Different security levels

Work	Lang.	Device	Field size	PQ Security	Total Time (ms)
AFJ14	C	Cortex-A15	771	128	1,308
			1035	170	2,816
KJAJM16	ASM	Cortex-A15	1008	167	**982**
JAMJ17	ASM	Cortex-A57	751	125	**331**
			964	160	**652**
	C		751	125	1,846
			964	160	4,212
	ASM	Cortex-A72	751	125	**271**
			964	160	**528**
	C		751	125	1,495
			964	160	3,495

Performance Reports on Various Platforms

- SIDH performance evaluation on different families of ARM processors
- Different security levels

Work	Lang.	Device	Field size	PQ Security	Total Time (ms)
AFJ14	C	Cortex-A15	771	128	1,308
			1035	170	2,816
KJAJM16	ASM	Cortex-A15	1008	167	**982**
JAMJ17	ASM	Cortex-A57	751	125	**331**
			964	160	**652**
	C		751	125	1,846
			964	160	4,212
	ASM	Cortex-A72	751	125	**271**
			964	160	**528**
	C		751	125	1,495
			964	160	3,495

- Reasonable performance, but we still need to improve these results.

Performance Reports on Various Platforms

- SIDH performance evaluation on different families of ARM processors
- Different security levels

Work	Lang.	Device	Field size	PQ Security	Total Time (ms)
AFJ14	C	Cortex-A15	771	128	1,308
			1035	170	2,816
KJAJM16	ASM	Cortex-A15	1008	167	**982**
JAMJ17	ASM	Cortex-A57	751	125	**331**
			964	160	**652**
	C		751	125	1,846
			964	160	4,212
	ASM	Cortex-A72	751	125	**271**
			964	160	**528**
	C		751	125	1,495
			964	160	3,495

- Reasonable performance, but we still need to improve these results.
- ARMv7-M platforms require further investigations due to the low working frequency (latest reported timings are in seconds).

- Quantum computers and their exceptional computational power will solve all the underlying problems that current PKC is constructed upon.
- We need to be prepared for this threat.
- NIST has already started the PQC standardization procedure.
- Different proposals have been submitted.
- SIKE is the only primitive which is constructed on the popular elliptic curves.
- SIKE offers the smallest key and ciphertext size among other candidates and it is suitable for embedded devices.

SIKE team

Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, David Jao, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik

Thank You!