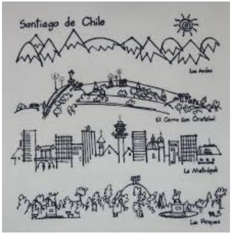
Modern Alice's Adventures in Cryptoland

Francisco Rodríguez-Henríquez
Cinvestav, México



Latincrypt 2019
Santiago de Chile
October first, 2019

Main primitives and building blocks in modern cryptography

Main primitives and building blocks in modern cryptography

- Primitives:
  - Encryption/decryption of digital documents [this task is typically solved using symmetric cryptography]

Main primitives and building blocks in modern cryptography

- Primitives:
  - Encryption/decryption of digital documents [this task is typically solved using symmetric cryptography]
  - Signature/verification of digital documents [This task is usually solved using public key cryptography]

Main primitives and building blocks in modern cryptography

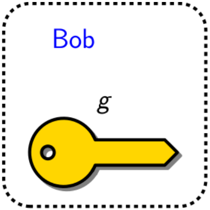
- Primitives:
  - Encryption/decryption of digital documents [this task is typically solved using symmetric cryptography]
  - Signature/verification of digital documents [This task is usually solved using public key cryptography]
  - Sharing a secret among two or more parties [this task is usually solved using the Diffie-Hellman protocol or its variants]

Main primitives and building blocks in modern cryptography

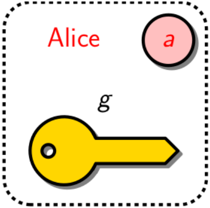
- Primitives:
  - Encryption/decryption of digital documents [this task is typically solved using symmetric cryptography]
  - Signature/verification of digital documents [This task is usually solved using public key cryptography]
  - Sharing a secret among two or more parties [this task is usually solved using the Diffie-Hellman protocol or its variants]
- Building blocks:
  - Block ciphers and stream ciphers
  - Hash functions
  - Public key crypto-schemes
  - ...

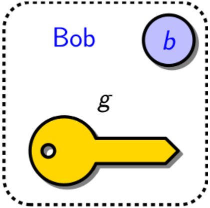
Design problem: How to share a secret?

Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



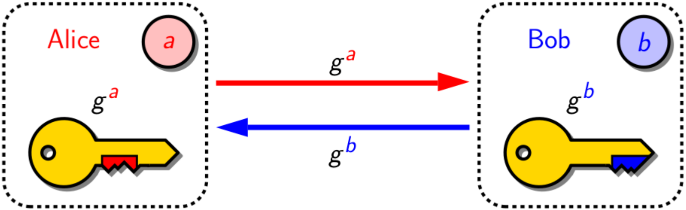
Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



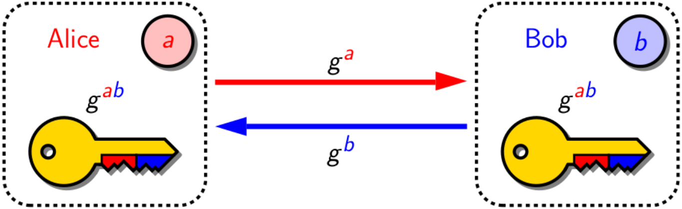


Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976

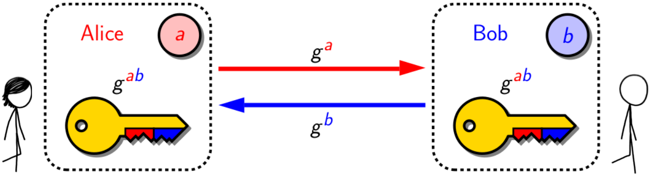
Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976

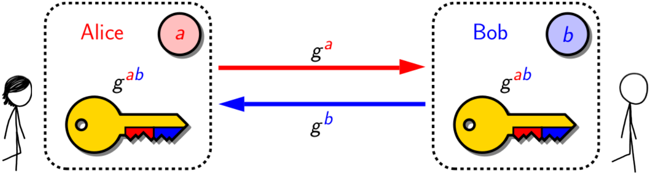


- Alice and Bob decide to work in the $\mathbb{Z}_p$ group, with p a large odd prime. They also choose a generator $g \in \mathbb{Z}_p$ (i.e., $Ord(g) = p - 1$).
- Alice and Bob select $a, b \in \mathbb{Z}_p$, respectively
- Alice and Bob compute a shared secret as,

$$K = (g^a)^b = (g^b)^a$$

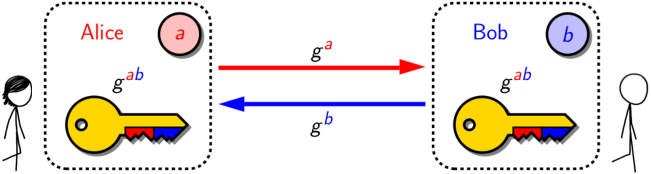
Note: This protocol can only be secure against passive attackers

Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



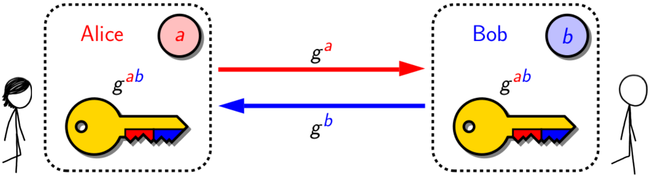
Protocol's security lies in the computational intractability of solving the Discrete Logarithm Problem (DLP), namely,
Given a prime p and a generator $g, h \in [1, p-1]$, find an integer k such that, $g^k \equiv h \bmod p$.

Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



Protocol's security lies in the computational intractability of solving the Discrete Logarithm Problem (DLP), namely,
Given a prime p and a generator $g, h \in [1, p-1]$, find an integer k such that, $g^k \equiv h \bmod p$.

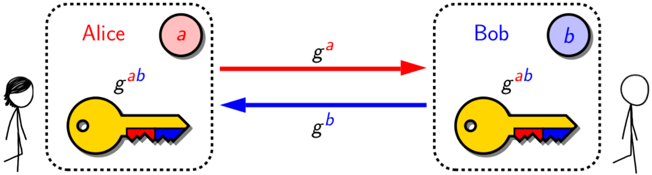
Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



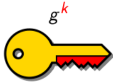
Protocol's security lies in the computational intractability of solving the Discrete Logarithm Problem (DLP), namely,

Given a prime p and a generator $g, h \in [1, p-1]$, find an integer k such that, $g^k \equiv h \bmod p$.

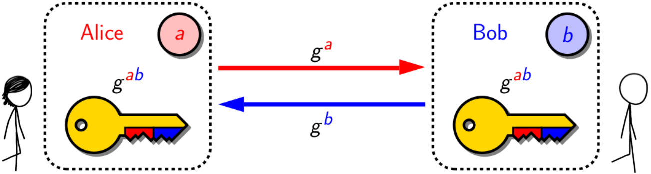
Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



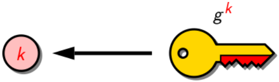
Protocol's security lies in the computational intractability of solving the Discrete Logarithm Problem (DLP), namely,
Given a prime p and a generator g, $h \in [1, p-1]$, find an integer k such that, $g^k \equiv h \bmod p$.



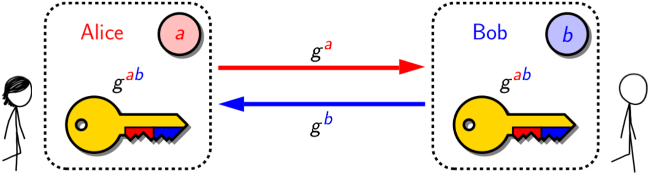
Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



Protocol's security lies in the computational intractability of solving the Discrete Logarithm Problem (DLP), namely,
Given a prime p and a generator $g, h \in [1, p-1]$, find an integer k such that,
$g^k \equiv h \bmod p$.

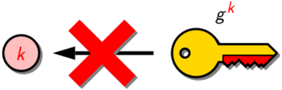


Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976

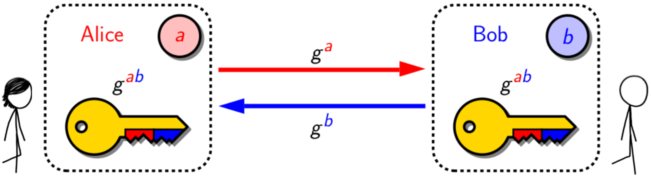


Protocol's security lies in the computational intractability of solving the Discrete Logarithm Problem (DLP), namely,

Given a prime p and a generator g, $h \in [1, p-1]$, find an integer k such that, $g^k \equiv h \bmod p$.



Design problem: How to share a secret?. Solution: Diffie-Hellman Protocol 1976



- Diffie and Hellman published their protocol in their breakthrough paper, Diffie, W.; Hellman, M. (1976). "New directions in cryptography". IEEE Transactions on Information Theory. 22 (6): 644–654."
- Diffie and Hellman won the 2015 Turing award
- Since its publication in 1976, "New directions in cryptography" has inspired many new ideas in the discipline. In this talk we will revisit four different versions of this protocol [!!]

Hard computational problems

1. Integer factorization problem: Given an integer $N = p \cdot q$ find its prime factors p and q. Find p, q such that $2019 = p \cdot q$

Hard computational problems

1. Integer factorization problem: Given an integer $N = p \cdot q$ find its prime factors p and q. Find p, q such that $2019 = p \cdot q$
   answer: $2019 = 3 \cdot 673$

Hard computational problems

1. Integer factorization problem: Given an integer $N = p \cdot q$ find its prime factors p and q. Find p, q such that $2019 = p \cdot q$
   answer: $2019 = 3 \cdot 673$
2. Discrete logarithm problem: Given a prime p and $g, h \in [1, p-1]$, find an integer x (if one exists) such that, $g^x \equiv h \bmod p$.
   find x such that $2^x \equiv 304 \bmod 419$

Hard computational problems

1. Integer factorization problem: Given an integer $N = p \cdot q$ find its prime factors p and q. Find p, q such that $2019 = p \cdot q$
   answer: $2019 = 3 \cdot 673$
2. Discrete logarithm problem: Given a prime p and $g, h \in [1, p-1]$, find an integer x (if one exists) such that, $g^x \equiv h \bmod p$.
   find x such that $2^x \equiv 304 \bmod 419$
   answer: $2^{343} \equiv 304 \bmod 419$.

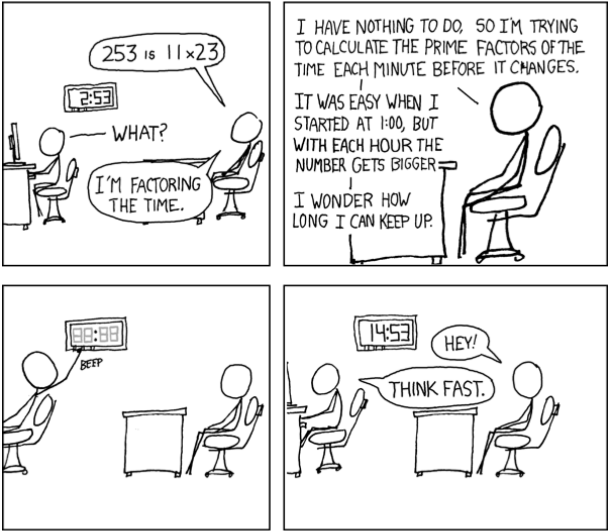
Hard computational problems

1. Integer factorization problem: Given an integer $N = p \cdot q$ find its prime factors p and q. Find p, q such that $2019 = p \cdot q$
   answer: $2019 = 3 \cdot 673$
2. Discrete logarithm problem: Given a prime p and $g, h \in [1, p-1]$, find an integer x (if one exists) such that, $g^x \equiv h \bmod p$.
   find x such that $2^x \equiv 304 \bmod 419$
   answer: $2^{343} \equiv 304 \bmod 419$.
   More generally: Given $g, h \in \mathbb{F}_q^*$, find an integer x (if one exists) such that, $g^x \equiv h$, where $q = p^k$ is the power of a prime

Hard computational problems

1. Integer factorization problem: Given an integer $N = p \cdot q$ find its prime factors p and q. Find p, q such that $2019 = p \cdot q$
   answer: $2019 = 3 \cdot 673$
2. Discrete logarithm problem: Given a prime p and $g, h \in [1, p-1]$, find an integer x (if one exists) such that, $g^x \equiv h \bmod p$.
   find x such that $2^x \equiv 304 \bmod 419$
   answer: $2^{343} \equiv 304 \bmod 419$.
   More generally: Given $g, h \in \mathbb{F}_q^*$, find an integer x (if one exists) such that, $g^x \equiv h$, where $q = p^k$ is the power of a prime
3. Elliptic curve discrete logarithm problem: Given an elliptic curve $E/\mathbb{F}_q$ and $P, Q \in E(\mathbb{F}_q)$, find an integer x (if one exists) such that, $xP = Q$ [More ECDLP material will be discussed later]

Time complexity



borrowed from the xkcd site.

Running time complexity

- The efficiency of an algorithm is measured in terms of its input size.

Running time complexity

- The efficiency of an algorithm is measured in terms of its input size.
  - For the discrete logarithm problem in $\mathbb{F}_q$, the input size is $O(\log q)$ bits.

Running time complexity

- The efficiency of an algorithm is measured in terms of its input size.
  - For the discrete logarithm problem in $\mathbb{F}_q$, the input size is $O(\log q)$ bits.
- A polynomial-time algorithm is one whose running time is bounded by a polynomial in the input size: $(\log q)^c$, where c is a constant.

Running time complexity

- The efficiency of an algorithm is measured in terms of its input size.
  - For the discrete logarithm problem in $\mathbb{F}_q$, the input size is $O(\log q)$ bits.
- A polynomial-time algorithm is one whose running time is bounded by a polynomial in the input size: $(\log q)^c$, where c is a constant.
- A fully exponential-time algorithm is one whose running time is of the form q^c, where c is a constant.

Running time complexity

- The efficiency of an algorithm is measured in terms of its input size.
  - For the discrete logarithm problem in $\mathbb{F}_q$, the input size is $O(\log q)$ bits.
- A polynomial-time algorithm is one whose running time is bounded by a polynomial in the input size: $(\log q)^c$, where c is a constant.
- A fully exponential-time algorithm is one whose running time is of the form q^c, where c is a constant.
- A subexponential-time algorithm as one whose running time is of the form,

$$L_q[\alpha, c] = e^{c(\log q)^{\alpha}(\log \log q)^{1-\alpha}},$$

where $0 < \alpha < 1$, and c is a constant.
$\alpha = 0$: polynomial $\alpha = 1$: fully exponential

Attacks on discrete log computation over small char $\mathbb{F}_{q^n}$: Main developments in the last 30+ years

Let Q be defined as $Q = q^n$.

- Hellman-Reyneri 1982: Index-calculus $L_Q[\frac{1}{2}, 1.414]$
- Coppersmith 1984: $L_Q[\frac{1}{3}, 1.526]$
- Joux-Lercier 2006: $L_Q[\frac{1}{3}, 1.442]$ when q and n are "balanced"
- Hayashi et al. 2012: Used an improved version of the Joux-Lercier method to compute discrete logs over the field $\mathbb{F}_{3^{6\cdot 97}}$
- Joux 2012: $L_Q[\frac{1}{3}, 0.961]$ when q and n are "balanced"
- Joux 2013: $L_Q[\frac{1}{4} + o(1), c]$ when $Q = q^{d\cdot m}$, d a small integer (e.g. $d = 2, 3$) and $q \approx m$
- Göloğlu et al. 2013: similar to Joux 2013, BPA @ Crypto'2013

Attacks on discrete log computation over small char $\mathbb{F}_{q^{3n}}$: security level consequences

Let us assume that one wants to compute discrete logarithms in the field $\mathbb{F}_{q^{3n}}$, with $q = 3^6$, $n = 509$, Notice that the group size of that field is,

$$\#\mathbb{F}_{3^{6 \cdot 509}} = \lceil \log_2(3) \cdot 6 \cdot 509 \rceil = 4841 \text{ bits.}$$

Algorithm	Time complexity	Equiv. bit security level
Hellman-Reyneri 1982	$L_{q^{6n}}[\frac{1}{2}, 1.414]$	337
Coppersmith 1984	$L_{q^{6n}}[\frac{1}{3}, 1.526]$	134
Joux-Lercier 2006	$L_{q^{6n}}[\frac{1}{3}, 1.442]$	126
Joux-Lercier 2006 (as revised by Shinohara et al. 2012)	$L_{q^{6n}}[\frac{1}{3}, 1.270]$	111
Joux 2012 (personal estimation)	$L_{q^{6n}}[\frac{1}{3}, 1.175]$	103
Joux 2013 (as analyzed by Adj et al. Pairing 2013)	$L_{q^{6n}}[\frac{1}{4}, 1.530]$	81
Joux-Pierrot 2014 (as analyzed by Adj et al. Waifi 2014)	$L_{q^{6n}}[\frac{1}{4}, 1.530]$	58

Recommended key sizes (circa 2013)

Security in bits	RSA $\|\|N\|\|_2$	DL: $\mathbb{F}_p$ $\|\|p\|\|_2$	DL: $\mathbb{F}_{2^m}$ m	ECC $\|\|q\|\|_2$
80	1024	1024	1500	160
112	2048	2048	3500	224
128	3072	3072	4800	256
192	7680	7680	12500	384
256	15360	15360	25000	512

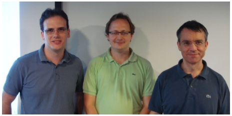
Recommended key sizes (2019)

Security in bits	RSA $\|\|N\|\|_2$	DLP: $\mathbb{F}_p$ $\|\|p\|\|_2$	~~DL: $\mathbb{F}_{2^m}$ m~~	ECC $\|\|q\|\|_2$
≈ 74	1024	1024	~~1500~~	160
≈ 106	2048	2048	~~3500~~	224
128	3072	3072	~~4800*~~	256
192	7680	7680	~~12500~~	384
256	15360	15360	~~25000~~	512

Recommended key sizes (2019)

Security in bits	RSA $\|\|N\|\|_2$	DLP: $\mathbb{F}_p$ $\|\|p\|\|_2$	~~DL: $\mathbb{F}_{2^m}$ m~~	ECC $\|\|q\|\|_2$
≈ 74	1024	1024	~~1500~~	160
≈ 106	2048	2048	~~3500~~	224
128	3072	3072	~~4800*~~	256
192	7680	7680	~~12500~~	384
256	15360	15360	~~25000~~	512

- * Nowadays, the extension $\mathbb{F}_{2^{4800}}$ is estimated to provide a security level of around 60 bits (see [Granger-Kleinjung-Zumbrägel'18], [AMOR'16]).



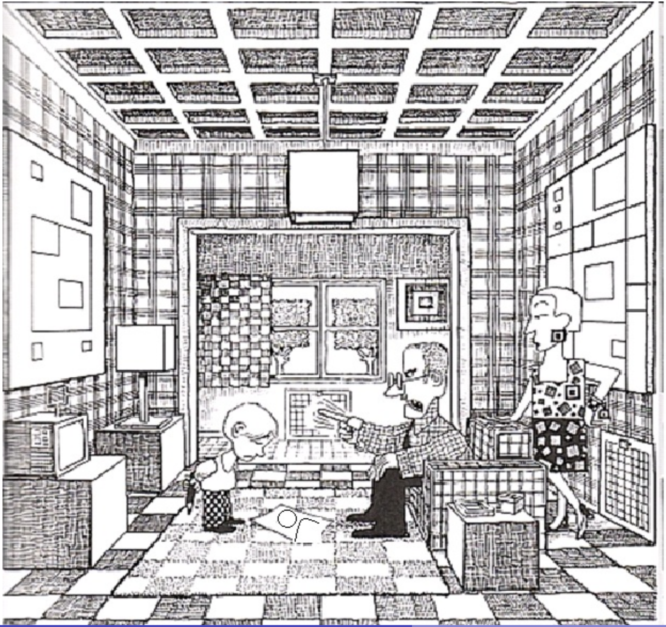
Barbulescu-Gaudry-Joux-Thomé: "A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic". EUROCRYPT 2014: 1-16

Recommended key sizes (2019)

Security in bits	RSA $\|\|N\|\|_2$	DLP: $\mathbb{F}_p$ $\|\|p\|\|_2$	~~DL: $\mathbb{F}_{2^m}$ m~~	ECC $\|\|q\|\|_2$
≈ 74	1024	1024	~~1500~~	160
≈ 106	2048	2048	~~3500~~	224
128	3072	3072	~~4800*~~	256
192	7680	7680	~~12500~~	384
256	15360	15360	~~25000~~	512

- Factorization (RSA): Using the Number Field Sieve (NFS) method leads to subexponential complexity, $\approx L_N\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$, Where N is the RSA modulus
- DLP over $\mathbb{F}_p$: Using index-calculus methods leads to subexponential complexity, $\approx L_p\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$,
- ECDLP: Using the Pollard's rho method leads to exponential complexity $\sqrt{\pi \cdot q}/2$, where $q = p^k$ is the prime field extension where the elliptic curve has been defined

Elliptic-curve-based cryptography

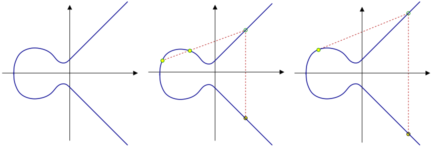


Elliptic-curve-based cryptography

Figure: Professors Neal Koblitz and Victor Miller and many Mexican graduate students at ECC 2012 in Querétaro, México

- Elliptic-curve-based cryptography (ECC) was independently proposed by Victor Miller and Neal Koblitz in 1985.
- It took more than two decades for ECC to be widely accepted and become the most popular public-key cryptographic scheme (above its archrival RSA)
- Nowadays ECC is massively used in everyday applications

Elliptic-curve-based cryptography



An elliptic curve is defined by the set of affine points $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$, with $p > 3$ an odd large prime, which satisfies the short Weierstrass equation given as,

$$E : y^2 = x^3 + ax + b,$$

along with a point at infinity denoted as $\mathcal{O}$.
Let $E(\mathbb{F}_p)$ be the set of points that satisfy the elliptic curve equation above. This set forms an Abelian group with order (size) given as, $\#E(\mathbb{F}_p) = h \cdot r$, where r is a large prime and the cofactor is a small integer.

Elliptic curves

- E defined by a Weierstraß equation of the form

$$y^2 = x^3 + Ax + B$$

Elliptic curves

- E defined by a Weierstraß equation of the form

$$y^2 = x^3 + Ax + B$$

- $E(K)$ set of rational points over a field K

Elliptic curves

- E defined by a Weierstraß equation of the form

$$y^2 = x^3 + Ax + B$$

- $E(K)$ set of rational points over a field K
- Additive group law over $E(K)$

Elliptic curves

- E defined by a Weierstraß equation of the form

$$y^2 = x^3 + Ax + B$$

- $E(K)$ set of rational points over a field K
- Additive group law over $E(K)$
- Many applications in cryptography since 1985
  - EC-based Diffie-Hellman key exchange
  - EC-based Digital Signature Algorithm
  -

Elliptic curves

- E defined by a Weierstraß equation of the form

$$y^2 = x^3 + Ax + B$$

- $E(K)$ set of rational points over a field K
- Additive group law over $E(K)$
- Many applications in cryptography since 1985
  - EC-based Diffie-Hellman key exchange
  - EC-based Digital Signature Algorithm
  -
- Interest: smaller keys than usual cryptosystems (RSA, ElGamal, ...)

Elliptic curves

- E defined by a Weierstraß equation of the form

$$y^2 = x^3 + Ax + B$$

- $E(K)$ set of rational points over a field K
- Additive group law over $E(K)$
- Many applications in cryptography since 1985
  - EC-based Diffie-Hellman key exchange
  - EC-based Digital Signature Algorithm
  -
- Interest: smaller keys than usual cryptosystems (RSA, ElGamal, ...)
- But there's more:
  - Bilinear pairings
  - Isogenous elliptic curves

Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$

Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$

Group cryptography

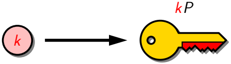
- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have
  $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$

Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have
  $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$

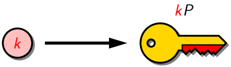
Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have
  $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$



Group cryptography

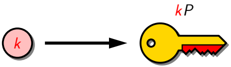
- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have
  $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$



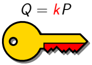
- Discrete logarithm: given $Q \in \mathbb{G}_1$, compute k such that $Q = kP$

Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$

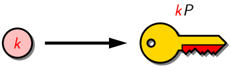


- Discrete logarithm: given $Q \in \mathbb{G}_1$, compute k such that $Q = kP$

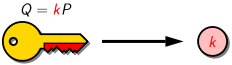


Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have
  $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$

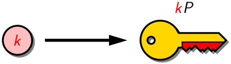


- Discrete logarithm: given $Q \in \mathbb{G}_1$, compute k such that $Q = kP$

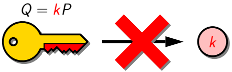


Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have
  $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$

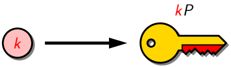


- Discrete logarithm: given $Q \in \mathbb{G}_1$, compute k such that $Q = kP$

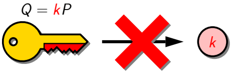


Group cryptography

- $(\mathbb{G}_1, +)$, an additively-written cyclic group of prime order $\#\mathbb{G}_1 = \ell$
- P, a generator of the group: $\mathbb{G}_1 = \langle P \rangle$
- Scalar multiplication: for any integer k, we have
  $kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$



- Discrete logarithm: given $Q \in \mathbb{G}_1$, compute k such that $Q = kP$



- We assume that the discrete logarithm problem (DLP) in $\mathbb{G}_1$ is hard

The Elliptic Curve Diffie-Hellman (ECDH) Protocol

Algorithm 1 The elliptic curve Diffie-Hellman protocol

Public parameters: Prime p, curve $E/\mathbb{F}_p$, point $P = (x, y) \in E(\mathbb{F}_p)$ of order r

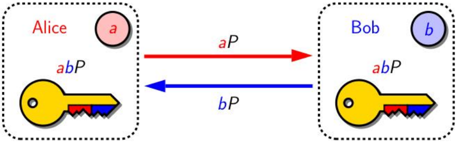
Phase 1: Key pair generation

	Alice		Bob
1:	Select the private key $d_A \xleftarrow{\$} [1, r-1]$	1:	Select the private key $d_B \xleftarrow{\$} [1, r-1]$
2:	Compute the public key $Q_A \leftarrow d_A P$	2:	Compute the public key $Q_B \leftarrow d_B P$

Phase 2: Shared secret computation

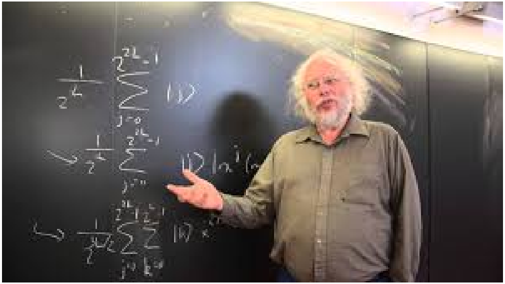
	Alice		Bob
3:	Send Q_A to Bob	3:	Send Q_B to Alice
4:	Compute $R \leftarrow d_A Q_B$	4:	Compute $R \leftarrow d_B Q_A$

Final phase: The shared secret is x-coordinate of the point R



[Apocalyptic] scenario for the next years: The arrival of large-scale quantum computers

[Apocalyptic] scenario for the next years: The arrival of large-scale quantum computers

-  

  - A quantum computer implementation of Peter Shor algorithm for factorization of integer numbers will imply that the computational effort for breaking elliptic-curve discrete logs will become polynomial.
  - In practice, this means that breaking commercial [EC]DLP would go from billions of years to hundred of hours.

[Apocalyptic] scenario for the next years: The arrival of large-scale quantum computers

-  

  Along with ECC, RSA and DSA public key crypto-schemes will also go to extinction

Design problem: How to construct a post-quantum Diffie-Hellman protocol?

Answers against the [Apocalyptic] scenario: Post-Quantum Cryptography (PQC)

- About two years ago, NIST launched a Post-Quantum Cryptography (PQC) standardization contest. NIST stated that

  'regardless of whether we can estimate the exact time of the arrival of the quantum computing era, we must begin now to prepare our information security systems to be able to resist quantum computing."

- The main focus of the contest is to find new PQC signature/verification and shared key establishment protocols. The latter task should be done using a scheme known as Key Encapsulation Mechanism (KEM).

Answers against the [Apocalyptic] scenario: Post-Quantum Cryptography (PQC)

- Out of 82 initial candidates only 23 made it to the second round. The surviving candidates have been classified in five main categories.
- Here at Latincrypt2019 and ASCrypto 2019, we will be hearing a lot about,
  - Lattice-based cryptography
  - Code-based crypto
  - Multivariate-based crypto
  - hash-based crypto
  - isogeny-based crypto

Design problem: How to construct a post-quantum Diffie-Hellman protocol using isogeny-based crypto?

[More] Mathematical definitions: recap

An *Elliptic Curve* in Weierstrass short model over a finite field $\mathbb{F}_q$ where $q = p^m$ for some prime $p > 3$, is given by the equation

$$E/\mathbb{F}_q : Y^2 = X^3 + AX + B$$

where $A, B \in \mathbb{F}_q$.

The *j-invariant* $j(E)$ of a curve acts like a fingerprint of a curve and it is given by

$$j(E) = \frac{1728 \cdot 4A^2}{4A^2 + 27B^2}.$$

A point P in $E(\mathbb{F}_q)$ is a pair (x, y) such that $x^3 + Ax + B - y^2 = 0$.

[More] Mathematical definitions: recap

- We can **Add** points

$$R := P + Q,$$

- **Double** a point

$$[2]P := P + P$$

- and multiply by a scalar as,

$$[m]P := P + P + \cdots + P, (m-1)(\text{times}).$$

- The minimum integer m such that $[m]P = \mathcal{O}$ is called the **order** of P.
- The **subgroup generated** by P is the set $\{P, [2]P, [3]P, \ldots, [m-1]P, \mathcal{O}\}$ and is denoted by $\langle P \rangle$.
- The m-**torsion subgroup** is defined as $E[m] = \{P \in E \mid [m]P = \mathcal{O}\}$.

[More] Mathematical definitions: recap

- (Hasse's Theorem)The number of rational points in an elliptic curve is bounded by

$$\#E(\mathbb{F}_q) = q + 1 - t, \qquad | t | \leq 2\sqrt{q}.$$

- E is supersingular if $p|t$, i.e., if

$$\#E(\mathbb{F}_q) = q + 1 \bmod p.$$

Otherwise E is said to be ordinary.

Basic definitions of isogenies

- An *Isogeny* $\phi : E_0 \rightarrow E_1$ is an homomorphism between elliptic curves given by rational functions. Given P and Q in E_0 is fulfilled that
  - $\phi(P + Q) = \phi(P) + \phi(Q)$,
  - $\phi(\mathcal{O}) = \mathcal{O}$.
- The *Kernel* of an Isogeny ϕ is the set

$$K = \{P \in E \mid \phi(P) = \mathcal{O}\}.$$

  Note: In this talk the degree of an isogeny is $s := \#K$.

- Let E and E' be two elliptic curves defined over $\mathbb{F}_q$. If there exists an isogeny $\phi : E \rightarrow E'$, then we say that E and E' are isogenous.

Basic definitions of isogenies

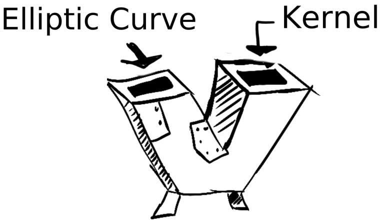
- Tate's theorem states that two elliptic curves E and E' are isogenous over $\mathbb{F}_q$, iff $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$.
- If two elliptic curves E and E' are isogenous over $\mathbb{F}_q$, either both of them are supersingular or both of them are ordinary.

Basic definitions of isogenies

- Let E be an elliptic curve and $P \in E$ be an order m point.
- Then there exists an elliptic curve E_P and an isogeny $\phi_P : E \to E_P$ such that the *Kernel* of ϕ_P is $\langle P \rangle$, *i.e.* $\phi_P(p) = \mathcal{O}$ for each $p \in \langle P \rangle$. We write

$$E_P = E/\langle P \rangle$$

- Moreover, given E defined over $\mathbb{F}_q$, and $K = \langle P \rangle$, Vélu's formulas outputs E_P and ϕ_P. The running time of Vélu's formulas is polynomial in $s = \#K$ and $\log_2(q)$.

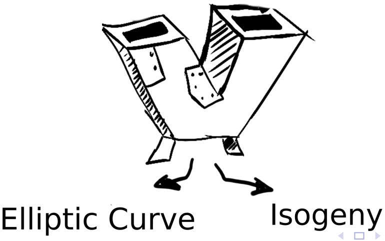


Basic definitions of isogenies

- Let E be an elliptic curve and $P \in E$ be an order m point.
- Then there exists an elliptic curve E_P and an isogeny $\phi_P : E \to E_P$ such that the *Kernel* of ϕ_P is $\langle P \rangle$, *i.e.* $\phi_P(p) = \mathcal{O}$ for each $p \in \langle P \rangle$. We write

$$E_P = E/\langle P \rangle$$

- Moreover, given E defined over $\mathbb{F}_q$, and $K = \langle P \rangle$, Vélu's formulas outputs E_P and ϕ_P. The running time of Vélu's formulas is polynomial in $s = \#K$ and $\log_2(q)$.



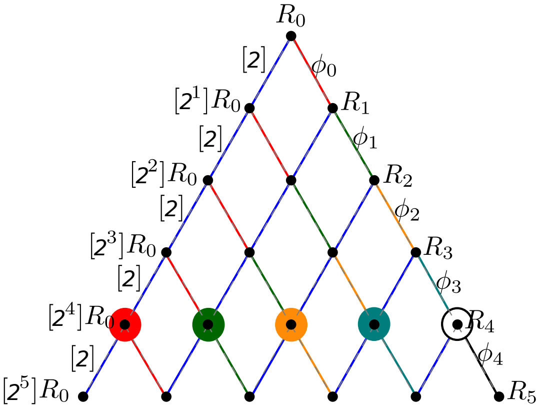
Basic definitions of isogenies

- Let E and E' be two elliptic curves defined over $\mathbb{F}_q$. If there exists a degree-1 isogeny between E and E' then $j(E) = j(E')$. We say that E and E' are isomorphic. We denote that by $E \cong E'$.
- Given an isogeny $\phi : E_0 \to E_1$ of degree d^e then
  - Then we can decompose ϕ as the composition

$$\phi_{e-1} \circ \phi_{e-2} \circ \cdots \phi_1 \circ \phi_0$$

    where ϕ_i has degree d.
  - There exists an isogeny $\hat{\phi} : E_1 \to E_0$ (called the dual isogeny of ϕ) such that,
    $\hat{\phi} \circ \phi = [d^e]$ and $\phi \circ \hat{\phi} = [d^e]$.

Computing composition of isogenies

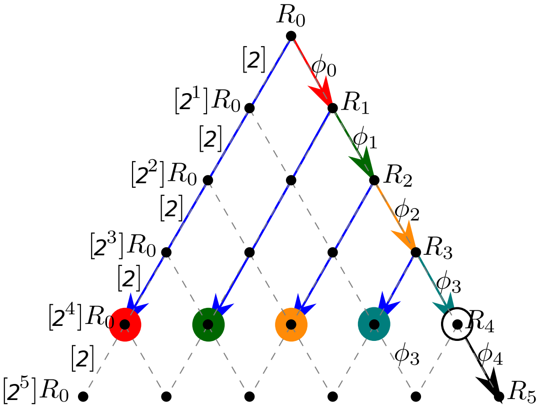


Example for a 2^5-isogeny.

Rules:

- Once you go down, you can't go back.
- The only way to go down along a non-blue line is reaching first the dot rounded by the same color of the line.
  Example: if you want to go down on a red line, first you need to reach the red rounded circle node.

Computing composition of isogenies

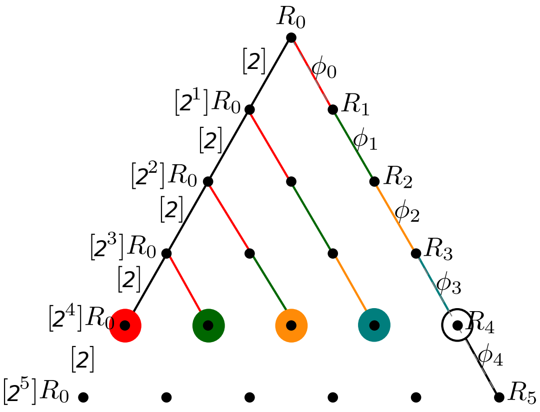


Example for a 2^5-isogeny.

Rules:

- Once you go down, you can't go back.
- The only way to go down along a non-blue line is reaching first the dot rounded by the same color of the line. Example: if you want to go down on a red line, first you need to reach the red rounded circle node.

Computing composition of isogenies



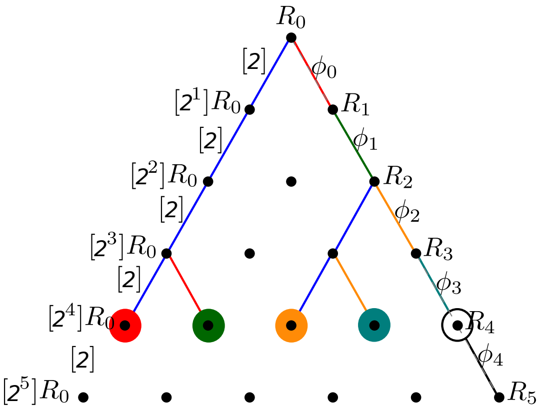
Unbalanced path: Isogeny evaluation oriented

Costs:

- [2] : 4
- Evaluations : 10

Fully parallelizable. (Needs more than 250 cores for real world implementations)

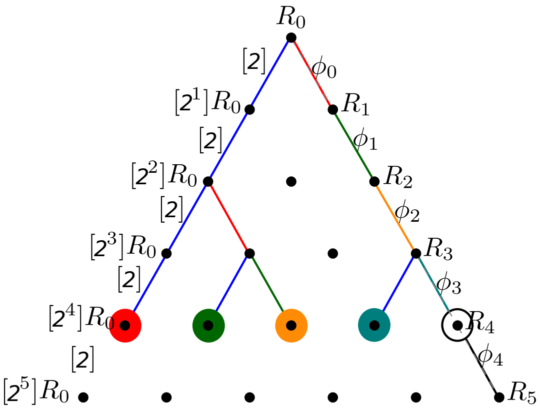
Computing composition of isogenies



Balanced path
Costs:

- [2] : 6
- Evaluations : 6

Computing composition of isogenies

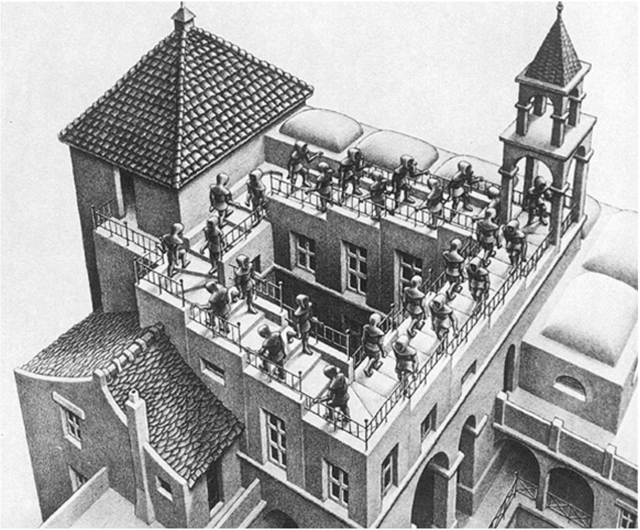


Balanced path

Costs:

- [2] : 6
- Evaluations : 6

Design problem: How to construct a post-quantum Diffie-Hellman protocol using isogeny-based crypto?



Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$,
- Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$.
- $E[2^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[3^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$,
- Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$.
- $E[2^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[3^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description of the SIDH protocol

E

$E/\langle R_A, R_B \rangle$

Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$,
- Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$.
- $E[2^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[3^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description of the SIDH protocol

$$E \xrightarrow{\phi_A} E/\langle R_A \rangle$$

$$R_A \leftarrow [n_A]P_A + [m_A]Q_A$$
$$R_B \leftarrow [n_B]P_B + [m_B]Q_B$$

$$E \xrightarrow{\phi_B} E/\langle R_B \rangle$$

$$E/\langle R_A, R_B \rangle$$

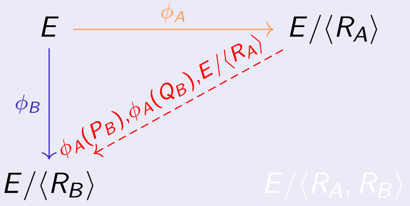
Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$,
- Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$.
- $E[2^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[3^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description of the SIDH protocol

$$R_A \leftarrow [n_A]P_A + [m_A]Q_A$$
$$R_B \leftarrow [n_B]P_B + [m_B]Q_B$$



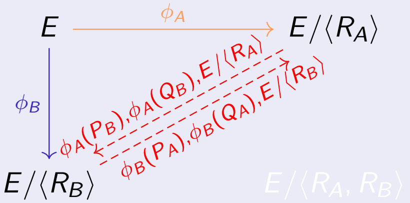
Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$,
- Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$.
- $E[2^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[3^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description of the SIDH protocol

$$R_A \leftarrow [n_A]P_A + [m_A]Q_A$$
$$R_B \leftarrow [n_B]P_B + [m_B]Q_B$$



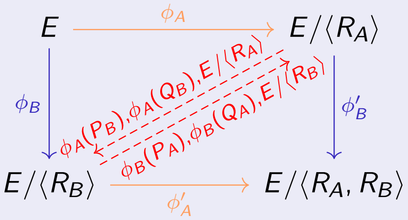
Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$,
- Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$.
- $E[2^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[3^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description of the SIDH protocol

$$\phi_B(R_A) \leftarrow [n_A]\phi_B(P_A) + [m_A]\phi_B(Q_A)$$
$$\phi_A(R_B) \leftarrow [n_B]\phi_A(P_B) + [m_B]\phi_A(Q_B)$$



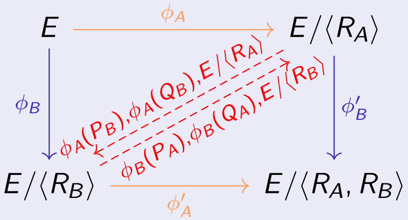
Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$,
- Let E be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E(\mathbb{F}_{p^2}) = (p+1)^2$.
- $E[2^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[3^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description of the SIDH protocol

$$\phi_B(R_A) \leftarrow [n_A]\phi_B(P_A) + [m_A]\phi_B(Q_A)$$
$$\phi_A(R_B) \leftarrow [n_B]\phi_A(P_B) + [m_B]\phi_A(Q_B)$$



where the shared secret key is the j-invariant $j(E/\langle R_A, R_B \rangle)$.

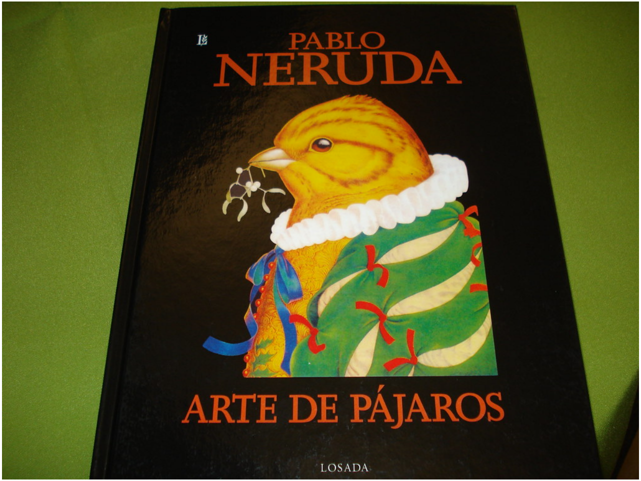
The CSSI problem [Charles-Goren-Lauter 2005]

The SIDH protocol bases its security guarantees in the hardness of the following hard problem,

Problem (CSSI)

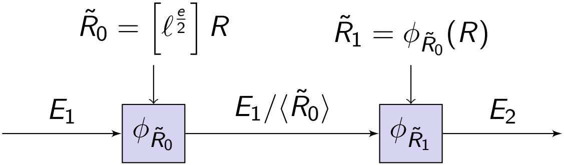
Given the public parameters e_A, e_B, p, E, P_A, Q_A, and the elliptic curve $E/\langle R_A \rangle$, compute a degree-2^{e_A} isogeny $\phi_A : E \to E/\langle R_A \rangle$.

How to [classically] attack the SIDH protocol



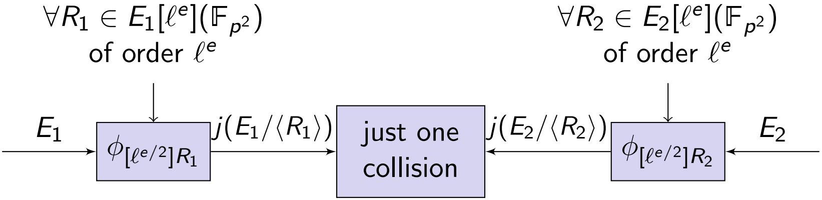
How to attack SIDH: The CSSI problem modeled as a collision finding problem [Adj-Cervantes-Chi-Menezes-RH'2018]

Let's write (R, ℓ, e) to mean either $(R_A, 2, e_A)$ or $(R_B, 3, e_B)$, $E_1 = E$, and $E_2 = E/\langle R\rangle$. Notice that the degree-(ℓ^e) isogeny $\phi\colon E \to E/\langle R\rangle$ can be written as the composition of two degree-$\ell^{e/2}$ isogenies.



How to attack SIDH: The CSSI problem modeled as a collision finding problem [Adj-Cervantes-Chi-Menezes-RH'2018]

Let's write (R, ℓ, e) to mean either $(R_A, 2, e_A)$ or $(R_B, 3, e_B)$, $E_1 = E$, and $E_2 = E/\langle R \rangle$. Therefore, E_1 and E_2 satisfies:



Meet-in-the-middle attack

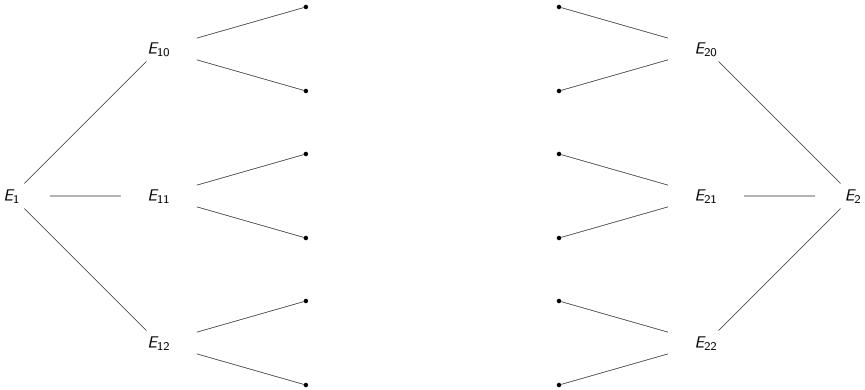
Let us illustrate how MITM works by an example. Let $e_A = 4$, $e_B = 2$, $p = 2^4 \cdot 3^2 \cdot 5 - 1$,

$$\begin{aligned} E_1 &\colon y^2 = x^3 + \big(\text{0x040} \cdot i + \text{0x1F0}\big)x + \big(\text{0x1E6} \cdot i + \text{0x0C7}\big), \\ P_1 &= (\text{0x16E} \cdot i + \text{0x1B4}, \text{0x10B} \cdot i + \text{0x05F}), \\ Q_1 &= (\text{0x203} \cdot i + \text{0x0CC}, \text{0x047} \cdot i + \text{0x0C5}), \text{ and} \\ E_2 &\colon y^2 = x^3 + \big(\text{0x1CF} \cdot i + \text{0x047}\big)x + \big(\text{0x1EA} \cdot i + \text{0x00D}\big). \end{aligned}$$

Then, the goal is to find a degree-2^4 isogeny from E_1 to E_2 using the following strategy:

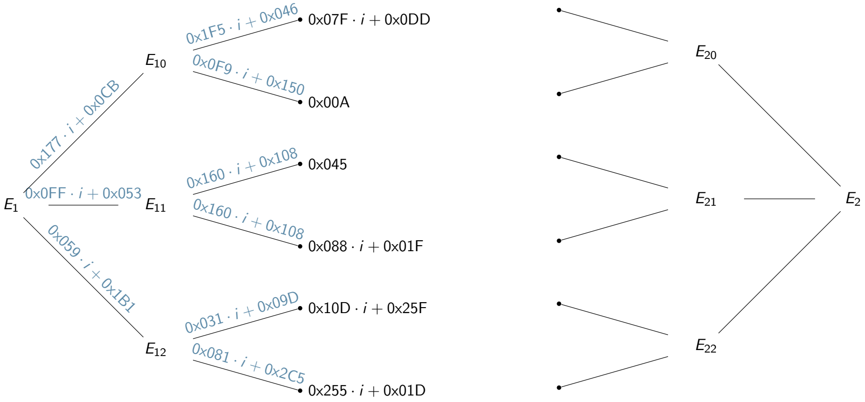
Meet-in-the-middle attack

First, compute the degree-2^2 isogeny tree rooted at E_1, and store its leaves.



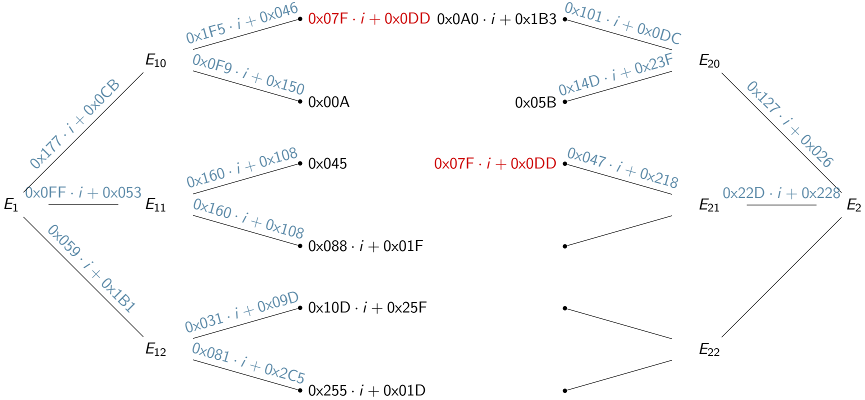
Meet-in-the-middle attack

First, compute the degree-2^2 isogeny tree rooted at E_1, and store its leaves.



Meet-in-the-middle attack

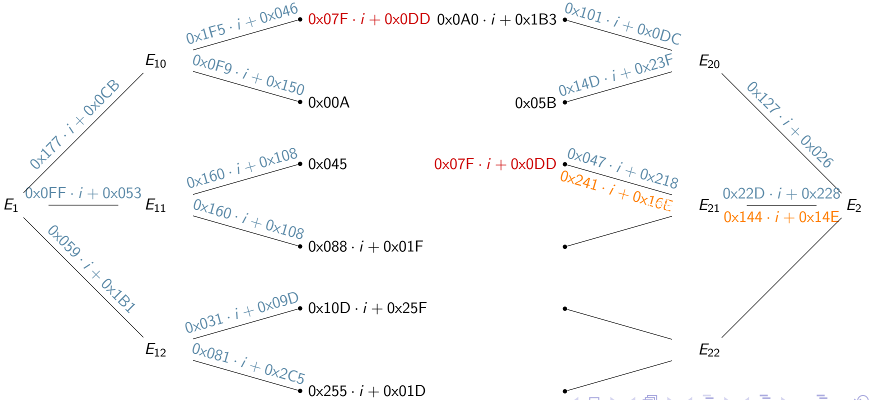
Second, compute degree-2^2 isogenies at E_2 until the match is found.



Meet-in-the-middle attack

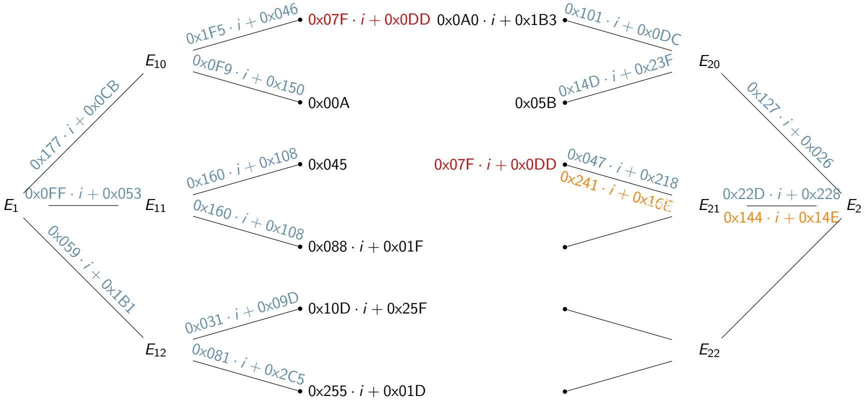
Then, we can reconstruct $\phi_A\colon E_1 \to E_2$ by composing the following isogenies:

$$E_1 \xrightarrow{\phi_0} E_{10} \xrightarrow{\phi_1} E_{100} \xrightarrow[\psi]{\mathbb{F}_{p^2}\text{-isomorphism}} E_{210} \xrightarrow{\hat{\phi}_2} E_{21} \xrightarrow{\hat{\phi}_3} E_2$$



Meet-in-the-middle attack

Now, let λ be the discrete log of $\phi_A(Q_A)$ in base $\phi_A(P_A)$ (or vice versa). Then, the secret kernel of Alice is $\langle Q_A - [\lambda]P_A \rangle$ (or $P_A - [\lambda]Q_A$). In our toy example, $\lambda = 3$.



Meet-in-the-middle attack

Clearly, The average-case time complexity is $1.5N$ and it has space complexity N, where $N \approx (\ell_A + 1)\ell_A^{e_A/2-1} \approx p^{1/4}$ (Infeasible for $N \geq 2^{80}$).

Meet-in-the-middle attack

Clearly, The average-case time complexity is $1.5N$ and it has space complexity N, where $N \approx (\ell_A + 1)\ell_A^{e_A/2-1} \approx p^{1/4}$ (Infeasible for $N \geq 2^{80}$). Consequently, using m processors and w cells of memory, the running time of MITM is approximately

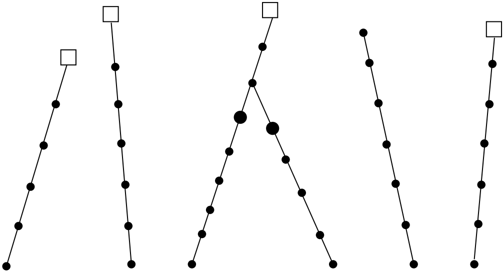
$$(w/m + N/m)\frac{N}{w} \approx N^2/(w \cdot m) \approx p^{1/2}/(w \cdot m).$$

Collision search problem: Modeling

Let S be a finite set of size N. The goal is to find a collision for a random function $f : S \to S$. Note: Recall that in the case of SIDH, $N \approx= p^{\frac{1}{4}}$.

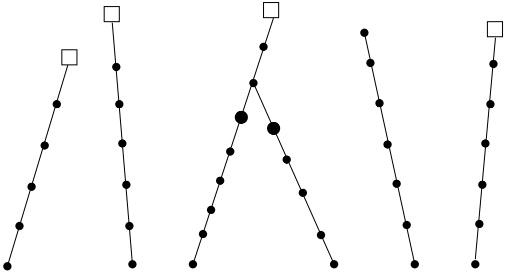
van Oorschot-Wiener (VW) collision search

First, let us define an element x of S to be *distinguished* if it has some easily-testable distinguishing property, and let θ be the proportion of elements of S that are distinguished.



van Oorschot-Wiener (VW) collision search

First, let us define an element x of S to be *distinguished* if it has some easily-testable distinguishing property, and let θ be the proportion of elements of S that are distinguished.



Then, using m processors, the expected time complexity of the VW method is approximately $\frac{1}{m}\sqrt{\pi N/2} + 2.5/\theta$.

van Oorschot-Wiener (VW) golden collision search

A random function $f : S \to S$ is expected to have $(N-1)/2$ unordered collisions.

van Oorschot-Wiener (VW) golden collision search

A random function $f : S \to S$ is expected to have $(N-1)/2$ unordered collisions. Suppose that we seek a particular one of these collisions, called a golden collision, which can be efficiently recognized.

van Oorschot-Wiener (VW) golden collision search

A random function $f : S \to S$ is expected to have $(N-1)/2$ unordered collisions. Suppose that we seek a particular one of these collisions, called a golden collision, which can be efficiently recognized.

Consequently, one continues generating distinguished points and collisions until the golden collision is encountered.

van Oorschot-Wiener (VW) golden collision search

The golden collision might occur with very small probability compared to other collision.

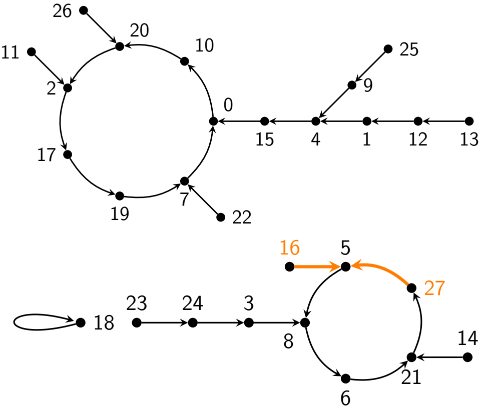


Figure: Functional graph of a random function $f: \{0, \ldots, 27\} \to \{0, \ldots, 27\}$. The desire golden collision is marked with Orange.

van Oorschot-Wiener (VW) golden collision search

The golden collision might occur with very small probability compared to other collision. Thus, it is necessary to change the version of f periodically.

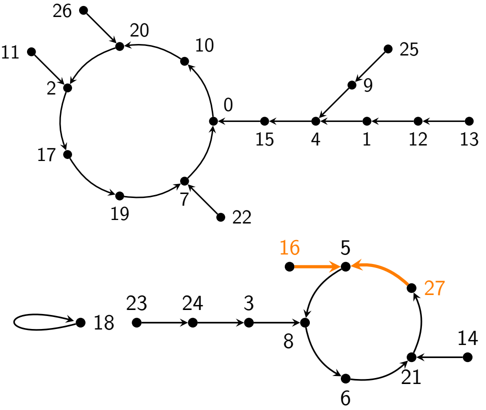


Figure: Functional graph of a random function $f : \{0, \ldots, 27\} \rightarrow \{0, \ldots, 27\}$. The desire golden collision is marked with Orange.

van Oorschot-Wiener (VW) golden collision search

Let

- w be the number of elements we can store in memory,
- $\theta = 2.25\sqrt{w/N}$,
- $10w$ be the number of distinguished elements that each version of f produces,
- $2^{10} \leq w \leq N/2^{10}$.

van Oorschot-Wiener (VW) golden collision search

Let

- w be the number of elements we can store in memory,
- $\theta = 2.25\sqrt{w/N}$,
- $10w$ be the number of distinguished elements that each version of f produces,
- $2^{10} \leq w \leq N/2^{10}$.

Heuristically, van Oorschot and Wiener observed that each version of f generates approximately $1.3w$ collisions, of which approximately $1.1w$ are distinct.

van Oorschot-Wiener (VW) golden collision search

Let

- w be the number of elements we can store in memory,
- $\theta = 2.25\sqrt{w/N}$,
- $10w$ be the number of distinguished elements that each version of f produces,
- $2^{10} \leq w \leq N/2^{10}$.

Heuristically, van Oorschot and Wiener observed that each version of f generates approximately $1.3w$ collisions, of which approximately $1.1w$ are distinct. In summary, the expected running time to find the golden collisions when m processors are employed is

$$\frac{1}{m}\left(2.5\sqrt{N^3/w}\right). \tag{1}$$

Solving CSSI with VW golden collision search

Therefore, using m processors and w cells of memory, the VW method can be used to find this golden collision in expected time

$$\frac{1}{m}\left(2.5\sqrt{8N^3/w}\right) \approx 7.1p^{3/8}/(w^{1/2}m).$$

Solving CSSI with VW golden collision search: 128-, 160-, 192-bit security

# processors m	space w	$p \approx 2^{448}$ calendar time	$p \approx 2^{448}$ total time	$p \approx 2^{512}$ calendar time	$p \approx 2^{512}$ total time	$p \approx 2^{536}$ calendar time	$p \approx 2^{536}$ total time	$p \approx 2^{614}$ calendar time	$p \approx 2^{614}$ total time
Meet-in-the-middle using Depth-first search									
48	64	106	154	138	186	150	198	188	236
48	80	90	138	122	170	134	182	172	220
64	80	74	138	106	170	118	182	156	220
van Oorschot and Wiener golden collision search									
48	64	88	136	112	160	121	169	149	197
48	80	80	128	104	152	113	161	141	189
64	80	64	128	88	152	97	161	125	189

Table: Time complexity estimates of CSSI attacks for $p \approx 2^{448}$, $p \approx 2^{512}$, $p \approx 2^{536}$ and $p \approx 2^{614}$. All numbers are expressed in their base-2 logarithms. The unit of time is a $2^{e/2}$-isogeny computation [2], and we are ignoring communication costs.

[2] Calendar time is the elapsed time taken for a computation, whereas total time is the sum of the time expended by all m processors.

Solving CSSI with VW golden collision search: 128-, 160-, 192-bit security

# processors m	space w	$p \approx 2^{448}$ calendar time	total time	$p \approx 2^{512}$ calendar time	total time	$p \approx 2^{536}$ calendar time	total time	$p \approx 2^{614}$ calendar time	total time
Meet-in-the-middle using Depth-first search									
48	64	106	154	138	186	150	198	188	236
48	80	90	138	122	170	134	182	172	220
64	80	74	138	106	170	118	182	156	220
van Oorschot and Wiener golden collision search									
48	64	88	136	112	160	121	169	149	197
48	80	80	128	104	152	113	161	141	189
64	80	64	128	88	152	97	161	125	189

Table: Time complexity estimates of CSSI attacks for $p \approx 2^{448}$, $p \approx 2^{512}$, $p \approx 2^{536}$ and $p \approx 2^{614}$. All numbers are expressed in their base-2 logarithms. The unit of time is a $2^{e/2}$-isogeny computation [2], and we are ignoring communication costs.

Conclusion: MITM is more costly than VW golden collision search.

[2] Calendar time is the elapsed time taken for a computation, whereas total time is the sum of the time expended by all m processors.

Comments about quantum attacks

Tani's algorithm

The fastest known quantum attack on CSSI is Tani's algorithm [Tani'09], which has an running time equal to $O(p^{1/6})$ and requires $O(p^{1/6})$ space.

Comments about quantum attacks

Tani's algorithm

The fastest known quantum attack on CSSI is Tani's algorithm [Tani'09], which has an running time equal to $O(p^{1/6})$ and requires $O(p^{1/6})$ space.

Grover's algorithm

Clearly, CSSI can also be solved by an application of Grover's quantum search [Grover'96], which has a running time equal to $O(p^{1/4})$. However, using m quantum circuits only yields a speedup by a factor of $\sqrt{m}$ [Zalka'99].

Comments about quantum attacks

Tani's algorithm

The fastest known quantum attack on CSSI is Tani's algorithm [Tani'09], which has an running time equal to $O(p^{1/6})$ and requires $O(p^{1/6})$ space.

Grover's algorithm

Clearly, CSSI can also be solved by an application of Grover's quantum search [Grover'96], which has a running time equal to $O(p^{1/4})$. However, using m quantum circuits only yields a speedup by a factor of $\sqrt{m}$ [Zalka'99].

Tani vs Grover: the recent work of Jaques and Schanck in their Crypto'2019 paper (which won the BPA) argue that Tani's algorithm is more costly than Grover's algorithm using all reasonable cost measures

Comments about quantum attacks

NIST suggests that 2^{40} is the maximum depth of a quantum circuit that can be executed in one year using presently envisioned quantum computing architectures [NIST'16].

Comments about quantum attacks

NIST suggests that 2^{40} is the maximum depth of a quantum circuit that can be executed in one year using presently envisioned quantum computing architectures [NIST'16].

Thus, assuming that the maximum circuit depth is 2^k, the number of quantum circuits needed to perform Grover's search in one year for $p \approx 2^r$ is approximately $\left(\frac{2^{\frac{r}{4}}}{2^k}\right)^2$.

Maximum depth of a quantum circuit	$p \approx 2^{448}$	$p \approx 2^{512}$	$p \approx 2^{536}$	$p \approx 2^{614}$
	m	m	m	m
40	144	176	188	227
64	96	128	140	179

Table: Number of quantum circuits needed to perform Grover's search in one year for $p \approx 2^{448}$, $p \approx 2^{512}$, $p \approx 2^{536}$, and $p \approx 2^{614}$. All numbers are expressed in their base-2 logarithms.

Recommendations

Assuming $m \leq 2^{64}$ and $w \leq 2^{80}$, we suggest

- $p_{434} = 2^{216}3^{137} - 1$ (instead of $p_{751} = 2^{372}3^{239} - 1$ [Costello *et al.*'16]) in order to achieve 128-bit security,
- $p_{546} = 2^{273}3^{172} - 1$ (instead of $p_{964} = 2^{486}3^{301} - 1$ [Jao *et al.*'17]) in order to achieve 160-bit security, and
- $p_{610} = 2^{305}3^{192} - 1$ in order to achieve 192-bit security.

Recommendations

SIDH operations are about 4.8 times faster when p_{434} is used instead of p_{751}.

Protocol phase		CLN library [Costello *et al.*'16]			CLN + enhancements		
		p_{751}	p_{434}	p_{546}	p_{751}	p_{434}	p_{546}
Key Gen.	Alice	35.7	7.51	13.20	26.9	5.3	10.5
	Bob	39.9	8.32	14.84	30.5	6.0	11.7
Shared Secret	Alice	33.6	7.01	12.56	24.9	5.0	10.0
	Bob	38.4	7.94	14.35	28.6	5.8	11.5

Table: Performance of the SIDH protocol. All timings are reported in 10^6 clock cycles, measured on an Intel Core i7-6700 supporting a Skylake micro-architecture. The "CLN + enhancements" columns incorporates improved formulas for degree-4 and degree-3 isogenies from [Costello & Hisil'17] and Montgomery ladders from [Faz-Hernández *et al.*'17] into the CLN library.

Summary

- Golden collision search is more cost effective that the meet-in-the-middle attack.
- SIDH operations are about 4.8 times faster when p_{434} is used instead of p_{751}.

Summary

SIDH parameters with p_{434} could be deemed to meet the security requirements in NIST's Category 2 [NIST'16] (classical and quantum security comparable or greater than that of SHA-256 with respect to collision resistance).

SIDH parameters with p_{610} could be deemed to meet the security requirements in NIST's Category 4 [NIST'16] (classical and quantum security comparable to that of SHA-384).

Note: The above suggestions have been endorsed by the SIKE team for the NIST round-2 version of their protocol

Design problem: How to construct a post-quantum Diffie-Hellman protocol?

Design problem: How to construct a post-quantum Diffie-Hellman protocol?

Secret Key

$[\mathfrak{a}] = \mathfrak{l}_1^{e_1^a} \cdots \mathfrak{l}_n^{e_n^a}$

$\{e_1^a \cdots e_n^a\}$

Public key

$E_A = E_0 \Big/ [\mathfrak{a}]$

Shared secret

$E_{AB} = E_b \Big/ [\mathfrak{a}]$

$\rightleftharpoons$

Secret Key

$[\mathfrak{b}] = \mathfrak{l}_1^{e_1^b} \cdots \mathfrak{l}_n^{e_n^b}$

$\{e_1^b \cdots e_n^b\}$

Public key

$E_B = E_0 \Big/ [\mathfrak{b}]$

Shared secret

$E_{AB} = E_a \Big/ [\mathfrak{b}]$

Castryck-Lange-Martindale-Panny-Renes: "CSIDH: An Efficient Post-Quantum Commutative Group Action". ASIACRYPT (3) 2018: 395-427

Gracias

- The concrete analysis and experiments of the CSSI problem shown in this presentation are joint work with Gora Adj, Daniel Cervantes-Vázquez, Jesús Javier Chi-Domínguez and Alfred Menezes.
- Thanks are due to Jean-Luc Beuchat, Daniel Cervantes-Vázquez and Jesús Chi-Domínguez for designing several of the animations of this presentation
- All pictures shown in this presentation were taken by the author in the Botero Museum at Bogotá.

Reference I

D. Jao and L. De Feo, “Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies”, Post-Quantum Cryptography — PQCrypto 2011, LNCS 7071 (2011), 19–34.

D. Charles, E. Goren and K. Lauter, “Cryptographic hash functions from expander graphs”, Journal of Cryptology, 22 (2009), 93–113.

J.M. Pollard, “Monte Carlo Methods for Index Computation (mod p)”. Mathematics of Computation, 32 (1978).

P. van Oorschot and M. Wiener, “Improving implementable meet-in-the-middle attacks by orders of magnitude”, Advances in Cryptology — CRYPTO ’96, LNCS 1109 (1996), 229–236.

L. De Feo, D. Jao and J. Plût, “Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies”, Journal of Mathematical Cryptology, 8 (2014), 209–247.

D. Jao et al., “Supersingular isogeny key encapsulation”, Round 1 submission, NIST Post-Quantum Cryptography Standardization, November 30, 2017.

Wikipedia, “Sunway TaihuLight”, https://en.wikipedia.org/wiki/Sunway_TaihuLight.

Wikipedia, “Exabyte”, https://en.wikipedia.org/wiki/Exabyte#Google.

Reference II

National Institute of Standards and Technology, “Submission requirements and evaluation criteria for the post-quantum cryptography standardization process”, December 2016.

L. Grover, “A fast quantum mechanical algorithm for database search”, Proceedings of the Twenty-Eighth Annual Symposium on Theory of Computing — STOC '96, ACM Press (1996), 212–219.

S. Tani, “Claw finding algorithms using quantum walk”, Theoretical Computer Science, 410 (2009), 5285–5297.

C. Zalka, “Grover's quantum searching algorithm is optimal”, Physical Review A, 60 (1999), 2746–2751.

C. Costello and H. Hisil, “A simple and compact algorithm for SIDH with arbitrary degree isogenies”, Advances in Cryptology — ASIACRYPT 2017, LNCS 10624 (2017), 303–329.

A. Faz-Hernández, J. López, E. Ochoa-Jiménez and F. Rodríguez-Henríquez, “A faster software implementation of the supersingular isogeny Diffie-Hellman key exchange protocol”, IEEE Transactions on Computers, to appear; also available from http://eprint.iacr.org/2017/1015.

C. Costello, P. Longa and M. Naehrig, “Efficient algorithms for supersingular isogeny Diffie-Hellman”, Advances in Cryptology — CRYPTO 2016, LNCS 9814 (2016), 572–601.

S. Jaques and J. Schanck, “Cost analyses of Tani's algorithm”, in preparation.