Approaches to the problem of making PAKEs quantum -safe

Oleg Taraskin

Vladimir Soukharev

David Jao

Jason T. LeGrow
The problem

All existing “industry” PAKE protocols are quantum-insecure:

Underlying hard problems (DLP, ECDLP and factoring) can be solved on quantum computer in polynomial time by Shor’s algorithm.
Example

Shor’s algorithm for ECDLP: space $\sim 6n$ qubits, time $\sim 360n^3$ (John Proos and Christof Zalka, 2003)

Most popular curves:

- ed25519 (Edwards curve)
- secp256k1 (Bitcoin, Ethereum)
- P-256 (NIST standard)

space $\sim 6*256 = 1536$ qubits, time $\sim 360*256^3$ operations
Isogeny basics

$E_1, E_2$ - elliptic curves over $F_q$

Isogeny $E_1 \rightarrow E_2$:

$\varphi(x, y) = \left( \frac{f_1(x,y)}{f_2(x,y)}, \frac{g_1(x,y)}{g_2(x,y)} \right)$

$\varphi(\infty) = \infty$

(equivalently, $\varphi(P + Q) = \varphi(P) + \varphi(Q)$)

$f_1, f_2, g_1, g_2$ are polynomials

Degree of isogeny $\varphi$ is max degree of $f_1(x, y)$ and $f_2(x, y)$
Example

\( E_1 : y^2 = x^3 + x + 1 \) and \( E_2 : y^2 = x^3 + 4x + 13 \) over \( F_{19} \)

\[
(x, y) = \left( \frac{x^3 - 4x^2 - 8x - 8}{(x-2)^2}, y \frac{x^3 - 6x^2 + 5x - 6}{(x-2)^3} \right)
\]

\( \deg \varphi = 3 \)

\( A = (9, 6), B = (14, 2) \) and \( C = A + B = (5, 6) \)

\( \varphi (9, 6) = (14, 1) \)

\( \varphi (14, 2) = (17, 4) \)

\( \varphi (5, 6) = (8, 5) \)

Group homomorphism: \( \varphi (9, 6) + \varphi (14, 2) = \varphi (5, 6) \)
Construction of isogenies

Isogeny is a group homomorphism:
\[ \ker \varphi = \{ P \in E : \varphi(P) = \infty \} \]

Let's \( K \) is some subgroup of \( E \),
exists \( \varphi_K : E \to E/K \) such that \( \ker \varphi_K \) is \( K \) and \( \deg \varphi_K = | K | \)

Isogeny can be calculated by Velu's algorithm (1971):
- Input : curve \( E_1, K \)
- Output: curve \( E_2 \), map \( \varphi \)
Construction of isogenies

Another way to express isogeny $E \to E/K:$

$E \to E/<G_K>$

where $G_K$ is generator of kernel group $K$

Tate’s theorem:

Two curves $E_1, E_2$ are isogenous over $F_q$ if and only if $\#E_1 = \#E_2$
Example

\[ E_1 : y^2 = x^3 + x + 1 \]  and  \[ E_2 : y^2 = x^3 + 4x + 13 \]

over field \( F_{19} \), \[ \#E_1 = \#E_2 = 21 \]

\[ \varphi(x, y) = \left( \frac{x^3 - 4x^2 - 8x - 8}{x^2 - 4x + 4}, \frac{x^3y - 6x^2y + 5xy - 6y}{x^3 - 6x^2 - 7x - 8} \right) \]

\[ \deg \varphi = 3 \]

Kernel of isogeny is a subgroup \( K = \{ \infty, (2, 7), (2, 12) \} \)

Kernel’s generators are \((2, 7), (2, 12),\)

so denote \( G_K = (2, 7) \) or \( G_K = (2, 12) \) \( E_2 = E_1/<G_K> \)
Hard problem

Given $E_1, E_2$ - elliptic curves over $F_q$, \quad \#E_1 = \#E_2

Find isogeny $\varphi$ between $E_1$ and $E_2$
$n$ -torsion subgroup

$$E[n] = \{ R \in E(F_q) : n \ast R = \infty \}$$

$E[n]$ is isomorphic to $\mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ (i.e. has order $= n^2$) if $\gcd(n, q) = 1$

Base points $P$ and $Q \in E[n]$: each $C \in E[n]$ can be expressed as $C = x \ast P + y \ast Q$ where $x, y \in [0, n)$
Supersingular curve

\[ \#E(GF(p^n)) = p^n + 1 - t \quad , \text{where} \quad t \text{ – trace of Frobenius} \]

if \( t \equiv 0 \mod p \):
   E is supersingular
else
   E is ordinary
SIDH (Supersingular Isogeny Diffie-Hellman)
D. Jao and L. De Feo, 2011

supersingular curve over $F_{p^2}$ that contains subgroups $E[2^{e_2}]$ and $E[3^{e_3}]$, 
where $2^{e_2} \approx 3^{e_3}$

Select “starting” curve: $y^2 = x^3 + ax + b$ over $F_{p^2}$

with characteristic $p = 2^{e_2}3^{e_3} \pm 1$
such that $\#E = (2^{e_2}3^{e_3})^2$ i.e. has $E[2^{e_2}]$ and $E[3^{e_3}]$

Fix base points:
$P_a$ and $Q_a$ of $E[2^{e_2}]$ - basis of Alice
$P_b$ and $Q_b$ of $E[3^{e_3}]$ - basis of Bob
SIDH (Supersingular Isogeny Diffie-Hellman )
D. Jao and L. De Feo, 2011

Fixed public parameters:

\[ y^2 = x^3 + ax + b \text{ over } F_{p^2} \]

\{P_a, Q_a\} - basis of \ E[2^{e_2}]
\{P_b, Q_b\} - basis of \ E[3^{e_3}]
Alice generates key pair:

- picks up random private key $a : 0 < a < 2^{e^2}$
- kernel group generator $G_a = P_a + a \cdot Q_a$
- calculates isogeny $\varphi_a$ with kernel group generated by $G_a$:
  $$E_a = E / \langle G_a \rangle$$
- maps Bob’s basis $\{P_b, Q_b\}$ to curve $E_a : \{\varphi_a(P_b), \varphi_a(Q_b)\}$
- sends to Bob her public key:
  $$E_a, \varphi_a(P_b), \varphi_a(Q_b)$$
SIDH (Supersingular Isogeny Diffie-Hellman)  
D. Jao and L. De Feo, 2011

Upon receiving public key of Alice, Bob generates key pair:

- picks up random private key $b : 0 < b < 3^{e^3}$
- $G_b = P_b + b \times Q_b$: point of order $3^{e^3}$

calculates isogeny $\varphi_b$ with kernel group generated by $G_b$:
- $E_b = E/< G_b >$
- maps Alice’s basis $\{P_a, Q_a\}$ to curve $E_b : \{\varphi_b(P_a), \varphi_b(Q_a)\}$
- sends to Alice his public key:

$$E_b, \varphi_b(P_a), \varphi_b(Q_a)$$
SIDH (Supersingular Isogeny Diffie-Hellman )  
D. Jao and L. De Feo, 2011

Bob:

\[ G_{ba} = \varphi_a(P_b) + b\varphi_a(Q_b) \]
\[ E_{ba} = E_a/\langle G_{ba} \rangle \]

Alice:

\[ G_{ab} = \varphi_b(P_a) + a\varphi_b(Q_a) \]
\[ E_{ab} = E_b/\langle G_{ab} \rangle \]

Shared secret : \[ j(E_{ba}) = j(E_{ab}) \]
Commutative diagram

\[
\begin{array}{c}
E \\
\downarrow_{\varphi_B} \\
E/<G_b> \\
\downarrow_{\varphi_B} \\
E_{ab}
\end{array}
\quad \xrightarrow{\varphi_A} \quad
\begin{array}{c}
E/<G_a> \\
\downarrow_{\varphi_A} \\
E/<G_a>/\langle \varphi_B(G_a) \rangle \\
\downarrow_{\varphi_A} \\
E_{ab}
\end{array}
\]

\[
E_{ab} = E/<G_b>/\langle \varphi_B(G_a) \rangle = E/<G_a>/\langle \varphi_A(G_b) \rangle
\]
Towards Isogeny-Based Password-Authenticated Key Establishment

Oleg Taraskin\(^1\), Vladimir Soukharev, David Jao, and Jason T. LeGrow

\(^1\) Waves Platform. Moscow, Russian Federation. tog.postquant@gmail.com
\(^2\) InfoSec Global. Toronto, Ontario, Canada. vladimir.soukharev@infosecglobal.com
\(^3\) Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada. \{djao, jlegrow\}@uwaterloo.ca

Abstract. Password authenticated key establishment (PAKE) is a cryptographic primitive that allows two parties who share a low-entropy secret (a password) to securely establish cryptographic keys in the absence of public key infrastructure. We propose the first quantum-resistant password-authenticated key exchange scheme based on supersingular elliptic curve isogenies. The scheme is built upon supersingular isogeny Diffie-Hellman [15], and uses the password to generate permutations which obscure the auxiliary points. We include elements of a security proof, and discuss roadblocks to obtaining a proof in the BPR model [1]. We also include some performance results.
From SIDH to SIDH PAKE

Ephemeral public key of Alice:

\[ E_a, \varphi_a(P_b), \varphi_a(Q_b) \]

Alice calculates masked public key:

\[ \text{Masked}_P = \text{MaskP} + \varphi_a(P_b), \text{Masked}_Q = \text{MaskQ} + \varphi_a(Q_b) \]

(where \( \text{MaskP} = F(1 || \text{password}) \), \( \text{MaskQ} = F(2 || \text{password}) \))

Alice sends to Bob \( E_a, \text{Masked}_P, \text{Masked}_Q \)
From SIDH to SIDH PAKE

Bob:

receives \( E_a, \text{MaskedP}, \text{MaskedQ} \)

calculates \( \varphi_a(P_b) = \text{MaskedP} - \text{MaskP} \),
\[ \varphi_a(Q_b) = \text{MaskedQ} - \text{MaskQ} \]

and get Alice’s public key: \( E_a, \varphi_a(P_b), \varphi_a(Q_b) \)
Offline dictionary attack

Tate pairing
\[ e(P_b, Q_b)^{\deg(\varphi_a)} = e(\varphi_a(P_b), \varphi_a(Q_b)) \]
\[ (\deg(\varphi_a) = 2^{e^2}) \]

Attacker has \( E_a, MaskedP, MaskedQ \)
Calculates \( MaskP_i \) and \( MaskQ_i \) for candidates on password
If \[ e(P_b, Q_b)^{\deg(\varphi_a)} = e(MaskedP - MaskP_i, MaskedQ - MaskQ_i) \]
Then password is found (with high probability)
Möbius Action

\[ SL_2(l, e) = \{ \Psi \in (Z/l^e Z)^{2 \times 2} : \det (A) = 1 \mod l^e \} \]
\[ \Upsilon_2(l, e) = \{ \Psi \in SL_2(l, e) : A \text{ is upper triangular mod } l \} \]

\( \Upsilon_2(l, e) \) acts on \( E[l^e] \times E[l^e] \) like matrix-vector multiplication:

i.e. if \( \Psi = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \) then \( \Psi \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \alpha X + \beta Y \\ \gamma X + \delta Y \end{pmatrix} \)
SIDH PAKE

Alice masks her ephemeral public key \( E_a, \varphi_a(P_b), \varphi_a(Q_b) \):

\[
\begin{pmatrix}
X_a \\
Y_a
\end{pmatrix} = \Psi_A \left( \begin{pmatrix}
\varphi_a(P_b) \\
\varphi_a(Q_b)
\end{pmatrix} \right)
\]

\( \Psi_A \) is a function of password and \( j \)-invariant of \( E_a \)
Sends \( E_a, X_a, Y_a \) to Bob

Bob, upon receiving \( E_a, X_a, Y_a \):

checks that \( e(P_b, Q_b)^{\deg(\varphi_a)} = e(X_a, Y_a) \) - if not, abort
SIDH PAKE

If pairing check is ok:

Bob unmask masked ephemeral public key \( E_a, X_a, Y_a \):

calculates inverse \( \Psi_A^{-1} \) from matrix \( \Psi_A = H_A (\text{password}, j(E_a)) \)
restores ephemeral public key:

\[
\begin{pmatrix}
\varphi_a(P_b) \\
\varphi_a(Q_b)
\end{pmatrix} = \Psi_A^{-1} \begin{pmatrix}
X_a \\
Y_a
\end{pmatrix}
\]

And obtains “clear” SIDH ephemeral public key \( E_a, \varphi_a(P_b), \varphi_a(Q_b) \)
SIDH PAKE

Bob generates his key pair:

- picks up random private key $b$: $0 < b < 3^{e^3}$
- calculates public key:
  \[ E_b = E / < P_b + b \cdot Q_b >, \quad \varphi_b(P_a), \quad \varphi_b(Q_a) \]
- masks his public key:
  \[ \Psi_B = H_B(\text{password}, j(E_b)) \]
  \[ \begin{pmatrix} X_b \\ Y_b \end{pmatrix} = \Psi_B \begin{pmatrix} \varphi_b(P_a) \\ \varphi_b(Q_a) \end{pmatrix} \]
- sends $E_b$, $X_b$, $Y_b$ to Alice
SIDH PAKE

Calculates shared secret:

$$E_{ba} = E_a / \langle \varphi_a (P_b) + b \varphi_a (Q_b) \rangle$$

Shared secret:

$$KDF\left( (E_a, X_a, Y_a) || (E_b, X_b, Y_b) || j(E_{ba}) || \psi_A || \psi_B \right)$$
SIDH PAKE

Upon receiving $E_b, X_b, Y_b$ from Bob, Alice:

checks that $e(P_a, Q_a)^{\deg(\phi_b)} = e(X_b, Y_b)$ - if not, abort

demasks:  
\[
\begin{pmatrix}
\phi_b(P_a) \\
\phi_b(Q_a)
\end{pmatrix} = \psi_B^{-1} \begin{pmatrix} X_b \\ Y_b \end{pmatrix}
\]

$E_{ab} = E_b < \phi_b(P_a) + a*\phi_b(Q_a) >$

Shared secret:

$KDF((E_a, X_a, Y_a) || (E_b, X_b, Y_b) || j(E_{ab}) || \psi_A || \psi_B)$
Practical aspects

Curves:
- from SIKE algorithm (now in a second round of NIST Post-Quantum Cryptography Standardization Process)

Ephemeral key sizes:
- just the same as in SIDH
  - (for SIKE’s curves p434 and p503: 330 and 378 bytes resp.)

Time:
- from 1.7 to 2 of “pure” SIDH:
  - (for SIKE’s curves p434 and p503: 142 and 228 of $10^6$ clock cycles resp.
    - Ubuntu 18.04, 1.6 GHz Intel Core i5-8250U)
Questions ?