Post-quantum cryptography

David Jao

August 14, 2023

UNIVERSITY OF
WATERLOO

Motivation: quantum computers

Quantum computers represent a huge potential threat to existing public-key cryptosystems: RSA, ECC, etc.

- Transitioning cryptographic algorithms takes time. If you wait until the threat arrives, **it's too late**.
- NIST is currently finalizing its first suite of post-quantum cryptosystems (2024) and evaluating additional candidates for signatures (2024-2027).
- Only public-key cryptography is threatened.

Post-quantum public-key cryptosystems:

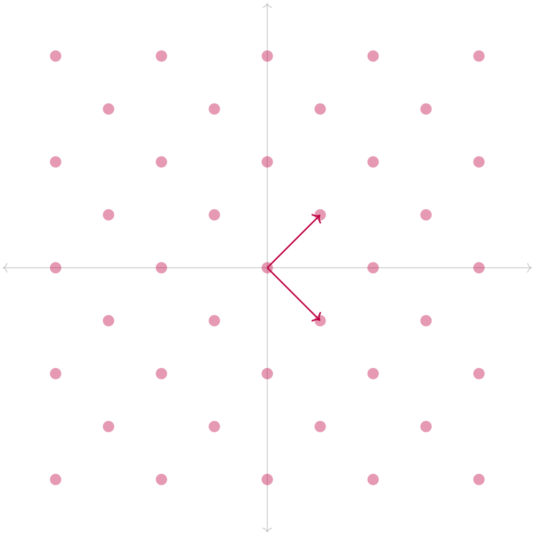
- Lattice-based cryptography
- Code-based cryptography
- Multivariate polynomials
- Hash-based cryptography
- Isogeny-based cryptography

Lattices

Definition

A *lattice* is a discrete subgroup of $\mathbb{R}^n$.

- *subgroup* — closed under addition and subtraction.
- *discrete* — there exists a minimum distance ε between distinct points
- Typically in cryptography we have $n > 500$



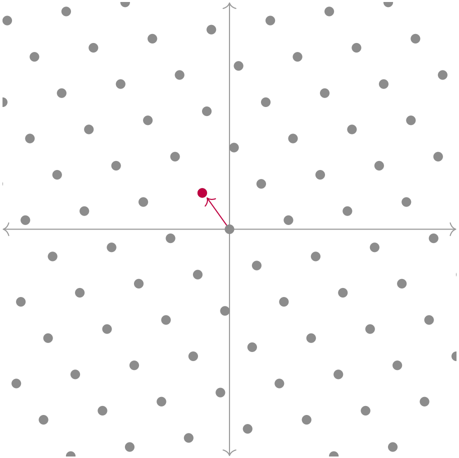
Shortest vector problem

Definition

Given $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$, find a nonzero $\mathbf{v} \in \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ of smallest norm.

Variants include:

- Approximate SVP: Find $\mathbf{v} \neq \mathbf{0}$ within a factor of γ of smallest norm.
- Decision SVP: Given $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$, determine whether $\mathbf{v}$ has smallest possible norm.
- Approximate Decision SVP, etc.



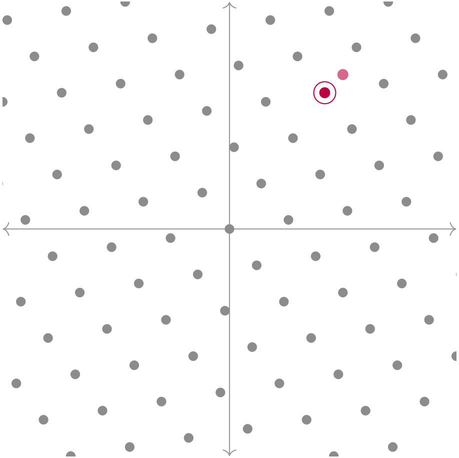
Closest vector problem

Definition

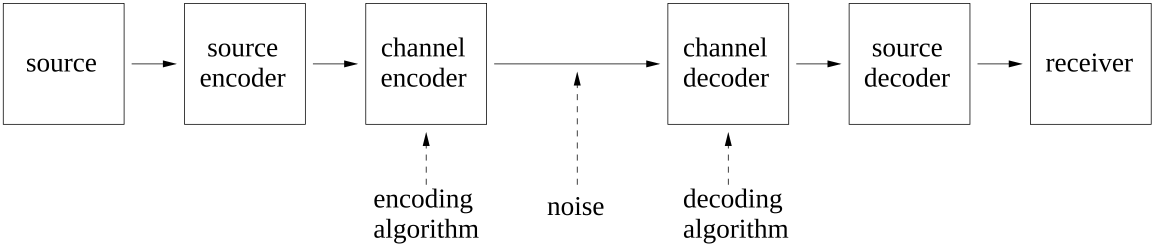
Given $\mathbf{v}$ and $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$, find $\mathbf{w} \in \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ minimizing $|\mathbf{w} - \mathbf{v}|$.

Variants include:

- Approximate CVP: Find $\mathbf{w} \in \mathcal{L}$ such that $|\mathbf{w} - \mathbf{v}|$ is within a factor of γ of smallest possible.
- Decision CVP: Given $\mathbf{w} \in \mathcal{L}$, determine if $|\mathbf{w} - \mathbf{v}|$ is as small as possible.
- Approximate Decision CVP, etc.



Codes


source
source encoder
channel encoder
channel decoder
source decoder
receiver
encoding algorithm
noise
decoding algorithm

Repetition code

Source message	Codeword	# errors/codeword that can be detected	#errors/codeword that can be corrected	Information rate
0 1	0 1	0	0	1
0 1	00 11	1	0	$\frac{1}{2}$
0 1	000 111	2	1	$\frac{1}{3}$
0 1	0000 1111	3	1	$\frac{1}{4}$
0 1	00000 11111	4	2	$\frac{1}{5}$
		$\vdots$		
0 1	0^n 1^n	$n-1$	$\lfloor \frac{n-1}{2} \rfloor$	$\frac{1}{n}$

Terminology

- An *alphabet* is a finite set of $q \geq 2$ symbols. (e.g. $A = \{0, 1\}$)
- A *word* is a finite sequence of symbols from A. (also: *vector*, *tuple*)
- The *length* of a word is the number of symbols in it.
- A *code* over A is a set of words (of size ≥ 2).
- A *codeword* is a word in the code.
- A *block code* is a code in which all codewords have the same length.
- A block code of length n containing M codewords over A is called an $[n, M]$-code over A. (Hence such a code is a subset $C \subset A^n$, with $|C| = M$.)

Hamming distance

Definition

The *Hamming distance* between two words of length n is

$$d(x, y) = \#\{i \in \{1, \dots, n\} : x_i \neq y_i\}.$$

The *Hamming distance* of a block code is

$$d(C) = \min\{d(x, y) : x, y \in C,\ x \neq y\}.$$

$d(x, y)$ is actually a metric: For all x, y, z,

- $d(x, y) \geq 0$
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

Example

$C = \{00000, 11100, 00111, 10101\}$ is a $[5, 4]$ code over $A = \{0, 1\}$. We might encode messages as follows:

Message		Codeword
00	$\longrightarrow$	00000
01	$\longrightarrow$	00111
10	$\longrightarrow$	11100
11	$\longrightarrow$	10101

Linear codes

Definition

Let F be a finite field of size q. A *linear code* is a block code $C \subset F^n$ of length n over F such that C is a vector subspace of F^n.

- If $C \subset F^n$ is a linear code of dimension k (as a vector space over F), we say C is an (n, k)-code.
- An (n, k)-code has q^k codewords, so it is an $[n, q^k]$-code over F.
- The information rate of an (n, k)-code is $\frac{k}{n}$.

Hamming weight

Definition

The *Hamming weight* of a vector $v \in F^n$ is

$$w(v) = d(\mathbf{0}, v).$$

The *Hamming weight* of a linear code C is

$$w(C) = \min\{w(c) : c \in C \setminus \{\mathbf{0}\}\}.$$

Theorem

For a linear code C, $w(C) = d(C)$.

Proof.

$d(C) = \min\{d(x,y) : x \neq y\} = \min\{w(x-y) : x \neq y\} = \min\{w(c) : c \neq \mathbf{0}\} = w(C)$. □

Encoding

Let C be an (n, k)-code. A natural way to encode messages is

$$(m_1, m_2, \ldots, m_k) \mapsto m_1 v_1 + m_2 v_2 + \cdots + m_k v_k$$

where $\{v_1, v_2, \ldots, v_k\}$ is a basis for C.

Definition

A generator matrix G for an (n, k)-code C is a $k \times n$ matrix whose rows form a basis for C over F.

Example

A generator matrix G is in *standard form* if $G = [I_k \mid A]$ for some $k \times (n-k)$ matrix A.

Example

$$G = \left[\begin{array}{ccc|cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right]$$

$$\begin{aligned}
(000) &\mapsto (00000)\\
(001) &\mapsto (00110)\\
(010) &\mapsto (01001)\\
(011) &\mapsto (01111)\\
(100) &\mapsto (10011)\\
(101) &\mapsto (10101)\\
(110) &\mapsto (11010)\\
\underbrace{(111)}_{\text{source messages}} &\mapsto \underbrace{(11100)}_{\text{codewords}}
\end{aligned}$$

Systematic codes

- A linear code C is *systematic* if there exists a generator matrix for C in standard form.
- Two linear codes are *equivalent* if there exists a permutation of coordinates which maps one code into the other.
- Theorem: Every linear code is equivalent to a systematic code.

Dual code

Definition

Let C be an (n, k)-code over F. The *dual code* $C^\perp$ of C is

$$C^\perp = \{x \in F^n : x \cdot y = 0 \text{ for all } y \in C\}.$$

A *parity-check matrix* for C is a generator matrix H for $C^\perp$.

Properties:

- If C is an (n, k)-code over F, then $C^\perp$ is an $(n, n-k)$-code over F.
- $(C^\perp)^\perp = C$.
- If C is systematic with generator matrix $G = [I_k \mid A]$, then $H = [-A^T \mid I_{n-k}]$ is a generator matrix for $C^\perp$ (and a parity-check matrix for C).
- For all $x \in F^n$, $x \in C$ if and only if $Hx^T = \mathbf{0}$.

Hamming code

We often define codes by their parity-check matrix. For example

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

defines a $(7, 4)$-code over $\mathbb{F}_2$, with generator matrix

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This particular code is a *Hamming code* of distance 3.

Decoding example

For the $(7,4)$ Hamming code with parity-check matrix

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

- Suppose we receive $r = (0111110)$.
- We compute $Hr^T = (011)^T$.
- This is not zero, so $r \notin C$.
- However, if we set $e = (0000100)$, then $He^T = (011)^T$.
- Hence $H(r-e)^T = (000)^T$, so $c = r - e = (0111010)$ is a codeword.
- Since $d(c, r) = 1$, it is likely that c was the intended codeword.

Syndrome decoding

Definition

For an (n, k)-code C with parity-check matrix H, the *syndrome* of a vector $x \in F^n$ is the (column) vector

$$s = Hx^T \in (F^{n-k})^T.$$

Properties:

- The syndrome of a codeword is $\mathbf{0}^T$.
- Two vectors in F^n are in the same coset of C if and only if they have the same syndrome.
- Syndrome decoding: Make a giant table of every possible syndrome and the corresponding intended codeword. This table has q^{n-k} entries.
- Decoding an arbitrary linear code optimally is known to be NP-hard.

McEliece cryptosystem

- Public parameters: $\mathbb{F}, n, k, t$ with $k < n$
- Key generation:
  - Choose an (n, k)-code C such that C can correct t errors and C admits an efficient decoding algorithm A (e.g. a binary Goppa code).
  - Let G be the generator matrix for C.
  - Choose a random invertible $k \times k$ matrix S and a random $n \times n$ permutation matrix P.
  - The public key is the $k \times n$ matrix $\hat{G} = SGP$. The private key is A.
- Encryption: To encrypt $\mathbf{m} \in \mathbb{F}^k$:
  - Choose a random vector $\mathbf{z} \in \mathbb{F}^n$ of weight t.
  - The ciphertext is $\mathbf{c} = \mathbf{m}\hat{G} + \mathbf{z}$.
- Decryption: To decrypt $\mathbf{c}$:
  - Compute $\hat{\mathbf{c}} = \mathbf{c}P^{-1}$.
  - Use the decoding algorithm A to decode $\hat{\mathbf{c}}$ to $\hat{\mathbf{m}}$.
  - Output $\mathbf{m} = \hat{\mathbf{m}}S^{-1}$.

Elliptic curves

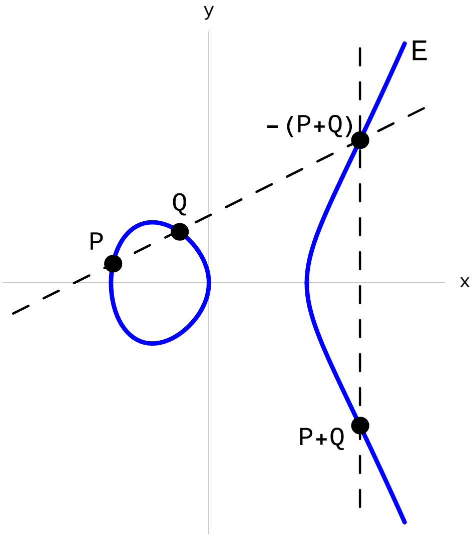
Definition

An elliptic curve over a field F is a nonsingular curve E of the form

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

for fixed constants $a, b \in F$.

The set of projective points on an elliptic curve forms a group.



Isogenies

Definition

An isogeny is a morphism ϕ of algebraic varieties between two elliptic curves, such that ϕ is a group homomorphism.

Concretely:

$$\phi\colon E \to E'$$
$$\phi(x, y) = (\phi_x(x, y), \phi_y(x, y))$$
$$\phi_x(x, y) = \frac{f_1(x, y)}{f_2(x, y)}$$
$$\phi_y(x, y) = \frac{g_1(x, y)}{g_2(x, y)}$$

(f_1, f_2, g_1, and g_2 are all polynomials)

Degree 2 example

- Let $E : y^2 = x^3 + ax + b$.
- Suppose $\ker\phi = \{\infty, P\}$. Then $P + P = \infty$, so $P = (x_P, 0)$ with $x_P^3 + ax_P + b = 0$.
- We have

$$E' : y^2 = x^3 + (a - 5(3x_P^2 + a))x + (b - 7x_P(3x_P^2 + a))$$

$$\phi(x, y) = \left(x + \frac{3x_P^2 + a}{x - x_P},\; y - \frac{y(3x_P^2 + a)}{(x - x_P)^2} \right)$$

Degree 3 example

- Let $E : y^2 = x^3 + ax + b$.
- Suppose $\ker\phi = \{\infty, P, -P\}$. Then $P = (x_P, y_P)$ with $3x_P^4 + 6ax_P^2 - a^2 + 12bx_P = 0$ and $y_P^2 = x_P^3 + ax_P + b$.
- We have

$$E' : y^2 = x^3 + (a - 10(3x_P^2 + a))x + (b - 28y_P^2 - 14x_P(3x_P^2 + a))$$

$$\phi(x,y) = \left(x + \frac{2(3x_P^2 + a)}{x - x_P} + \frac{4y_P^2}{(x - x_P)^2}, y - \frac{8yy_P^2}{(x - x_P)^3} - \frac{2y(3x_P + a)}{(x - x_P)^2}\right)$$

Supersingular Isogeny Key Encapsulation (NIST Round 4 Candidate)

Based on Supersingular Isogeny Diffie-Hellman (Jao & De Feo, 2011)

1. Public parameters: Supersingular elliptic curve E over $\mathbb{F}_{p^2}$.
2. Alice chooses a kernel $A \subset E[2^e] \subset E(\mathbb{F}_{p^2})$ of size 2^e and sends E/A and $\phi_A|_{E[3^f]}$.
3. Bob chooses a kernel $B \subset E[3^f] \subset E(\mathbb{F}_{p^2})$ of size 3^f and sends E/B and $\phi_B|_{E[2^e]}$.
4. The shared secret is

$$E/\langle A, B\rangle = (E/A)/\phi_A(B) = (E/B)/\phi_B(A).$$

Diffie-Hellman (DH)

$$\begin{tikzcd} g \arrow[r] \arrow[d] & g^x \arrow[d] \\ g^y \arrow[r] & g^{xy} \end{tikzcd}$$

SIDH

$$\begin{tikzcd} E \arrow[r, "\phi_A"] \arrow[d, "\phi_B"] & E/A \arrow[d] \\ E/B \arrow[r] & E/\langle A, B\rangle \end{tikzcd}$$

CSIDH (2018) — Castryck, Lange, Martindale, Panny, Renes

Based on Couveignes (1996), Rostovstev & Stolbunov (2006), using supersingular curves to obtain smooth order kernels.

1. Public parameters: Supersingular elliptic curve $E/\mathbb{F}_p$ with $G = \mathrm{Cl}(\mathrm{End}_p(E))$.
2. Alice chooses $\mathfrak{a} \in G$ and sends $\mathfrak{a} * E = E/\{P \in E : \forall\ \phi \in \mathfrak{a},\ \phi(P) = \infty\}$
3. Bob chooses $\mathfrak{b} \in G$ and sends $\mathfrak{b} * E$.
4. The shared secret is $(\mathfrak{a}\mathfrak{b}) * E = \mathfrak{a} * (\mathfrak{b} * E) = \mathfrak{b} * (\mathfrak{a} * E)$.

$$\begin{tikzcd} E \arrow[r, "\phi_{\mathfrak{a}}"] \arrow[d, "\phi_{\mathfrak{b}}"'] & \mathfrak{a} * E \arrow[d] \\ \mathfrak{b} * E \arrow[r] & (\mathfrak{a}\mathfrak{b}) * E \end{tikzcd}$$

Isogeny-based signature schemes

SIDH signatures (surprisingly, still viable)

1. Public key: $(E, E/A)$
2. Commitment: E/B
3. Challenge: $c \in \{1, 2, 3\}$
4. Response: ϕ_c

SeaSign / CSI-FiSh signatures

1. Public key: $E, \mathfrak{a} * E$
2. Commitment: $\mathfrak{b} * E$
3. Challenge: $c \in \{0, 1\}$
4. Response: $\phi_{\mathfrak{b}\mathfrak{a}^{-c}}$

SIDH

$$\begin{tikzcd} E \arrow[r, "\phi_A"] \arrow[d, "\phi_1"'] & E/A \\ E/B \arrow[r, "\phi_2"'] & E/\langle A, B\rangle \arrow[u, "\phi_3"'] \end{tikzcd}$$

SeaSign / CSI-FiSh

$$\begin{tikzcd} E \arrow[r, "\phi_{\mathfrak{a}}"] \arrow[d, "\phi_{\mathfrak{b}}"'] & \mathfrak{a} * E \arrow[dl, "\phi_{\mathfrak{b}\mathfrak{a}^{-1}}"] \\ \mathfrak{b} * E & \end{tikzcd}$$

Optimizations

- Hashing: Publish $H(\mathfrak{b} * E)$ instead of $\mathfrak{b} * E$
- Multiple challenges: Use n simultaneous commitments $\mathfrak{b}_1, \mathfrak{b}_2, \ldots, \mathfrak{b}_n$
- Twists: Commit to $\mathfrak{b} * E$ and $\mathfrak{b}^{-1} * E$ simultaneously

$$\begin{tikzcd} E \arrow[r, "\phi_{\mathfrak{a}}"] \arrow[d, "\phi_{\mathfrak{b}_1}"'] & \mathfrak{a} * E \arrow[dl, "\phi_{\mathfrak{b}_1 \mathfrak{a}^{-1}}"] \\ \mathfrak{b}_1 * E & \end{tikzcd}$$

$$\vdots$$

$$\vdots$$

$$\begin{tikzcd} E \arrow[r, "\phi_{\mathfrak{a}}"] \arrow[d, "\phi_{\mathfrak{b}_n}"'] & \mathfrak{a} * E \arrow[dl, "\phi_{\mathfrak{b}_n \mathfrak{a}^{-1}}"] \\ \mathfrak{b}_n * E & \end{tikzcd}$$

Optimizing for shortest $|\,\mathrm{pk} + \mathrm{sig}\,|$:

$\vert$ sk $\vert$	$\vert$ pk $\vert$	$\vert$ sig $\vert$	KeyGen	Sign	Verify
16 B	512 B	956 B	400 ms	1.48 s	1.48 s

Note: "CSI-FiSh really isn't polynomial-time" (`https://yx7.cc/blah/2023-04-14.html`)

SQISign

De Feo, Kohel, Leroux, Petit, Wesolowski

1. Public key: E, E_A, τ
2. Commitment: E_1
3. Challenge: ϕ
4. Response: σ

$$\begin{tikzcd} E \arrow[r, "\tau"] \arrow[d, "\psi"'] & E_A \arrow[d, "\sigma"] \\ E_1 \arrow[r, "\phi"'] & E_2 \end{tikzcd}$$

$\lvert\text{sk}\rvert$	$\lvert\text{pk}\rvert$	$\lvert\text{sig}\rvert$	KeyGen	Sign	Verify
16 B	64 B	204 B	0.6 s	2.5 s	50 ms