The McEliece cryptosystem

Daniel J. Bernstein

Some reasons to study McEliece

Among all public-key encryption systems, the McEliece system has the strongest security track record. **Minimizes security risks.**

McEliece is already deployed in end-to-end secure-messaging systems, Adva's high-speed optical networks, Crypto4A's hardware security modules, and the Mullvad and Rosenpass VPNs.

Easy-to-use software library libmceliece has already been integrated into Debian and Ubuntu.

More environments: Bouncy Castle (Java and C#), Python, Rust, M4, FPGAs, McTiny, McOutsourcing.
Integrations: PQClean, liboqs, Node.js, OpenSSH.

One way to invent the McEliece system

0. Start with modern lattice-based cryptography.

One way to invent the McEliece system

0. Start with modern lattice-based cryptography.

1. Choose **modulus 2**. Bad: slower in software.
Good: simpler; easier analysis; much more stability against cryptanalysis; nicer for hardware.

One way to invent the McEliece system

0. Start with modern lattice-based cryptography.

1. Choose **modulus 2**. Bad: slower in software.
Good: simpler; easier analysis; much more stability against cryptanalysis; nicer for hardware.

2. Then switch to a **more powerful decoder**.
Bad: more complicated decoding algorithm.
Good: much better security vs. ciphertext size.

One way to invent the McEliece system

0. Start with modern lattice-based cryptography.

1. Choose **modulus 2**. Bad: slower in software.
Good: simpler; easier analysis; much more stability against cryptanalysis; nicer for hardware.

2. Then switch to a **more powerful decoder**.
Bad: more complicated decoding algorithm.
Good: much better security vs. ciphertext size.

3. Then **travel back in time** to publish in 1978.
Good: allows half century of security analysis and half century of implementation improvements.

The general framework

System parameters: $n, q, r \in \{1, 2, 3, \ldots\}$.

Public key determines $K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.
Notation: $\mathbb{Z}/q$ is the ring of integers mod q;
$(\mathbb{Z}/q)^r = \{(u_0, \ldots, u_{r-1}) : \text{each } u_i \in \mathbb{Z}/q\}$;
$a, b \in X$ means $a \in X$ and $b \in X$.

The general framework

System parameters: $n, q, r \in \{1, 2, 3, \ldots\}$.

Public key determines $K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.
Notation: $\mathbb{Z}/q$ is the ring of integers mod q;
$(\mathbb{Z}/q)^r = \{(u_0, \ldots, u_{r-1}) : \text{each } u_i \in \mathbb{Z}/q\}$;
$a, b \in X$ means $a \in X$ and $b \in X$.

Ciphertext: $C = s_0K_0 + \cdots + s_{n-1}K_{n-1} \in (\mathbb{Z}/q)^r$
where $s_0, \ldots, s_{n-1} \in \mathbb{Z}$ are small secrets.
Ciphertext has $r \log_2 q$ bits.

The general framework

System parameters: $n, q, r \in \{1, 2, 3, \ldots\}$.

Public key determines $K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.
Notation: $\mathbb{Z}/q$ is the ring of integers mod q;
$(\mathbb{Z}/q)^r = \{(u_0, \ldots, u_{r-1}) : \text{each } u_i \in \mathbb{Z}/q\}$;
$a, b \in X$ means $a \in X$ and $b \in X$.

Ciphertext: $C = s_0 K_0 + \cdots + s_{n-1} K_{n-1} \in (\mathbb{Z}/q)^r$
where $s_0, \ldots, s_{n-1} \in \mathbb{Z}$ are small secrets.
Ciphertext has $r \log_2 q$ bits.

This covers "code-based" and "lattice-based" encryption. Let's call this **cola encryption**.

A cola example: `ntruhps2048509`

System parameters: $(n, q, r) = (1018, 2048, 508)$.

Public key determines $K_0, \ldots, K_{1017} \in (\mathbb{Z}/2048)^{508}$.

Ciphertext: $C = s_0K_0 + \cdots + s_{1017}K_{1017}$
for secrets $s_0, \ldots, s_{1017} \in \{-1, 0, 1\}$.

Ciphertext has $508 \log_2 2048 = 5588$ bits,
i.e., $5588/8 = 698.5$ bytes, sent in 699 bytes.

(Exercise: What are n, q, r for `kyber512`?)

Lattice attacks

Attacker sees $C, K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.
Easy linear-algebra computation finds big
$t_0, \ldots, t_{n-1} \in \mathbb{Z}$ with $C = t_0K_0 + \cdots + t_{n-1}K_{n-1}$.

Lattice attacks

Attacker sees $C, K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.
Easy linear-algebra computation finds big
$t_0, \ldots, t_{n-1} \in \mathbb{Z}$ with $C = t_0 K_0 + \cdots + t_{n-1} K_{n-1}$.

Note: $(t_0 - s_0)K_0 + \cdots + (t_{n-1} - s_{n-1})K_{n-1} = 0$;
i.e., $(t_0 - s_0, \ldots, t_{n-1} - s_{n-1}) \in L$ where
$L = \{(v_0, \ldots, v_{n-1}) : v_0 K_0 + \cdots + v_{n-1} K_{n-1} = 0\}$.

Lattice attacks

Attacker sees $C, K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.
Easy linear-algebra computation finds big
$t_0, \ldots, t_{n-1} \in \mathbb{Z}$ with $C = t_0K_0 + \cdots + t_{n-1}K_{n-1}$.

Note: $(t_0 - s_0)K_0 + \cdots + (t_{n-1} - s_{n-1})K_{n-1} = 0$;
i.e., $(t_0 - s_0, \ldots, t_{n-1} - s_{n-1}) \in L$ where
$L = \{(v_0, \ldots, v_{n-1}) : v_0K_0 + \cdots + v_{n-1}K_{n-1} = 0\}$.

Attack problem is now a "close-vector problem":
find v in lattice L with $v \approx (t_0, \ldots, t_{n-1})$.

NP-hardness myths for lattice encryption

Standard conjectures: "the polynomial hierarchy does not collapse"; in particular, P $\neq$ NP; so, for every NP-hard problem, every poly-time algorithm fails to solve *some* example of the problem.

NP-hardness myths for lattice encryption

Standard conjectures: "the polynomial hierarchy does not collapse"; in particular, P $\neq$ NP; so, for every NP-hard problem, every poly-time algorithm fails to solve *some* example of the problem.

Fact: The general problem of finding $v \in L$ with $v \approx t$ is NP-hard. (1981 van Emde Boas)

NP-hardness myths for lattice encryption

Standard conjectures: "the polynomial hierarchy does not collapse"; in particular, $\mathrm{P} \neq \mathrm{NP}$; so, for every NP-hard problem, every poly-time algorithm fails to solve *some* example of the problem.

Fact: The general problem of finding $v \in L$ with $v \approx t$ is NP-hard. (1981 van Emde Boas)

Common mistake: "Attacking lattice encryption is an example of this problem, so it's NP-hard."

No, there's no reason to think attacking lattice encryption is NP-hard. Fact: Every problem broken in poly time is an example of an NP-hard problem.

Lattice gaps

Picture from 2005 Aharanov–Regev:

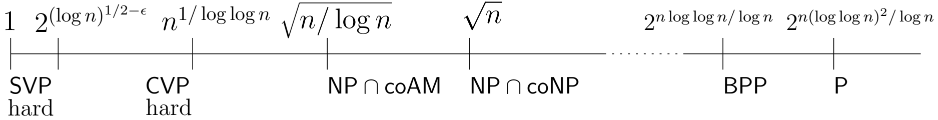


Figure 1: The complexity of lattice problems (some constants omitted)

Lattice gaps

Picture from 2005 Aharanov–Regev:

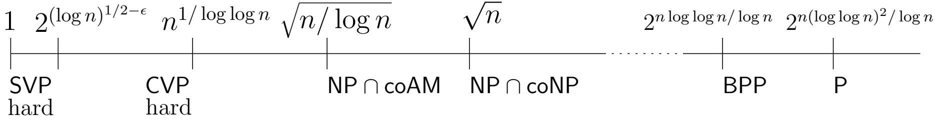


Figure 1: The complexity of lattice problems (some constants omitted)

Right side, large "gap": t is particularly close to L; fast algorithms find closest vector.

Lattice gaps

Picture from 2005 Aharanov–Regev:

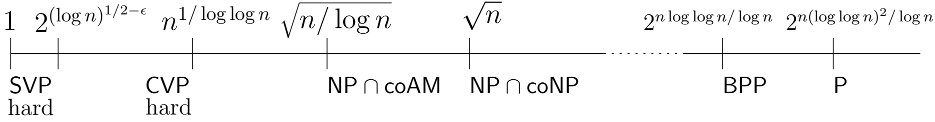


Figure 1: The complexity of lattice problems (some constants omitted)

Right side, large “gap”: t is particularly close to L; fast algorithms find closest vector.

Left side, small “gap”: t is far from L; NP-hard.

Lattice gaps

Picture from 2005 Aharanov–Regev:

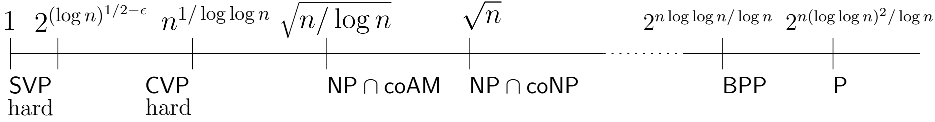


Figure 1: The complexity of lattice problems (some constants omitted)

Right side, large "gap": t is particularly close to L; fast algorithms find closest vector.

Left side, small "gap": t is far from L; NP-hard.

Middle: the standard conjectures imply that the problem is *not* NP-hard for, e.g., "gap" $\sqrt{n}$.

Warning: multiple gap concepts

A typical "decisional" gap problem:
you're guaranteed that *either* $\operatorname{dist}(t, L) \le d/G$ *or* $\operatorname{dist}(t, L) > d$; problem is to figure out which.

Warning: multiple gap concepts

A typical "decisional" gap problem:
you're guaranteed that *either* $\operatorname{dist}(t, L) \le d/G$ *or* $\operatorname{dist}(t, L) > d$; problem is to figure out which.

"Search" problem: find $v \in L$ with $\operatorname{dist}(t, v) \le d/G$ given that all other $w \in L$ have $\operatorname{dist}(t, w) > d$.

Warning: multiple gap concepts

A typical "decisional" gap problem:
you're guaranteed that *either* $\text{dist}(t, L) \le d/G$ *or* $\text{dist}(t, L) > d$; problem is to figure out which.

"Search" problem: find $v \in L$ with $\text{dist}(t, v) \le d/G$ given that all other $w \in L$ have $\text{dist}(t, w) > d$.

Suffices to solve an "approximation" problem:
find $v \in L$ with $\text{dist}(t, v) \le G \, \text{dist}(t, L)$.

Warning: multiple gap concepts

A typical "decisional" gap problem:
you're guaranteed that *either* $\operatorname{dist}(t, L) \le d/G$ *or* $\operatorname{dist}(t, L) > d$; problem is to figure out which.

"Search" problem: find $v \in L$ with $\operatorname{dist}(t, v) \le d/G$ given that all other $w \in L$ have $\operatorname{dist}(t, w) > d$.

Suffices to solve an "approximation" problem:
find $v \in L$ with $\operatorname{dist}(t, v) \le G \operatorname{dist}(t, L)$.

If $d = \max\{\operatorname{dist}(u, L)\}$ then the guarantee forces $\operatorname{dist}(t, L) \le d/G$ so $G \le \max\{\operatorname{dist}(u, L)\}/\operatorname{dist}(t, L)$.
For simplicity, this talk focuses on computing this cutoff gap: $\max\{\operatorname{dist}(u, L)\}/\operatorname{dist}(t, L)$.

What's the NTRU cutoff gap?

NTRU has $s_0, \ldots, s_{n-1} \in \{-1, 0, 1\}$,
so $\mathrm{dist}(t, L) \le |(s_0, \ldots, s_{n-1})| \le n^{1/2}$.

What's the NTRU cutoff gap?

NTRU has $s_0, \ldots, s_{n-1} \in \{-1, 0, 1\}$,
so $\mathrm{dist}(t, L) \le |(s_0, \ldots, s_{n-1})| \le n^{1/2}$.

Typically q is chosen as $\Theta(n)$. Can then show that most vectors have distance $\Omega(n)$ from L, so cutoff gap is $\Omega(n)/n^{1/2}$, i.e., $\Omega(n^{1/2})$.

(Exercise: Prove this gap.)

What's the NTRU cutoff gap?

NTRU has $s_0, \ldots, s_{n-1} \in \{-1, 0, 1\}$,
so $\operatorname{dist}(t, L) \le |(s_0, \ldots, s_{n-1})| \le n^{1/2}$.

Typically q is chosen as $\Theta(n)$. Can then show that most vectors have distance $\Omega(n)$ from L, so cutoff gap is $\Omega(n)/n^{1/2}$, i.e., $\Omega(n^{1/2})$.

(Exercise: Prove this gap.)

This doesn't mean NTRU is broken! Maybe attacking NTRU is hard without being NP-hard.

The NTRU decoder

Alice generates an NTRU secret key and a public key determining $K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.

The secret key determines a linear transformation φ such that $\varphi(K_0), \ldots, \varphi(K_{n-1})$ are small.

The NTRU decoder

Alice generates an NTRU secret key and a public key determining $K_0, \ldots, K_{n-1} \in (\mathbb{Z}/q)^r$.

The secret key determines a linear transformation φ such that $\varphi(K_0), \ldots, \varphi(K_{n-1})$ are small.

Bob computes $C = s_0 K_0 + \cdots + s_{n-1} K_{n-1}$. Alice computes $\varphi(C) = s_0 \varphi(K_0) + \cdots + s_{n-1} \varphi(K_{n-1})$, which is small, so the reduction mod q disappears. A fast algorithm solves for $s_0, \ldots, s_{n-1}$.

A cola example mod 2: `bikel1`

System parameters: $(n, q, r) = (24646, 2, 12323)$.

Public key determines $K_0, \ldots, K_{24645} \in (\mathbb{Z}/2)^{12323}$.

Ciphertext: $C = s_0 K_0 + \cdots + s_{24645} K_{24645}$
for "weight-134" vector $(s_0, \ldots, s_{24645}) \in \{0, 1\}$;
i.e., $\#\{i : s_i \neq 0\} = 134$.
Ciphertext has 12323 bits $\approx$ 1541 bytes.

Alice generated weight-71 $\varphi(K_0), \ldots, \varphi(K_{24645})$.
Then $\varphi(C) = s_0 \varphi(K_0) + \cdots + s_{24645} \varphi(K_{24645})$
involves some reductions mod 2, but
fast statistics usually solve for $s_0, \ldots, s_{24645}$.

What's the BIKE cutoff gap?

BIKE takes $(s_0, \ldots, s_{n-1})$ of weight $\Theta(n^{1/2})$, so t has distance $\Theta(n^{1/4})$ from lattice L.

Can show that most vectors have distance $\Theta(n^{1/2})$ from L, so cutoff gap is $\Theta(n^{1/4})$.

What's the BIKE cutoff gap?

BIKE takes $(s_0, \ldots, s_{n-1})$ of weight $\Theta(n^{1/2})$, so t has distance $\Theta(n^{1/4})$ from lattice L.

Can show that most vectors have distance $\Theta(n^{1/2})$ from L, so cutoff gap is $\Theta(n^{1/4})$.

Compared to NTRU:

- Gap sounds smaller. More secure?
- But t sounds closer to L. Fewer s possibilities. Less secure?

`ntruhps2048509` (699-byte ciphertexts) and `bikel1` (1541-byte ciphertexts) are both designed to have roughly 128 bits of security.

The basic ISD attack

There are $\binom{n}{w}$ weight-w vectors $s \in (\mathbb{Z}/2)^n$.

For $(n, w) = (24646, 134)$: $\binom{n}{w} \approx 2^{1196}$.

The basic ISD attack

There are $\binom{n}{w}$ weight-w vectors $s \in (\mathbb{Z}/2)^n$.
For $(n, w) = (24646, 134)$: $\binom{n}{w} \approx 2^{1196}$.

Faster than searching through all s:
1962 Prange "information-set decoding".
Basic idea: Maybe $s_r = s_{r+1} = \cdots = s_{n-1} = 0$;
probability $\binom{r}{w}/\binom{n}{w} \approx 2^{-134.52}$.
Then $C = s_0K_0 + \cdots + s_{r-1}K_{r-1}$.
Solve for $s_0, \ldots, s_{r-1}$ by linear algebra.
If this fails, permute $\{0, \ldots, n-1\}$ and try again.

The basic ISD attack

There are $\binom{n}{w}$ weight-w vectors $s \in (\mathbb{Z}/2)^n$.
For $(n, w) = (24646, 134)$: $\binom{n}{w} \approx 2^{1196}$.

Faster than searching through all s:
1962 Prange "information-set decoding".
Basic idea: Maybe $s_r = s_{r+1} = \cdots = s_{n-1} = 0$;
probability $\binom{r}{w}/\binom{n}{w} \approx 2^{-134.52}$.
Then $C = s_0 K_0 + \cdots + s_{r-1} K_{r-1}$.
Solve for $s_0, \ldots, s_{r-1}$ by linear algebra.
If this fails, permute $\{0, \ldots, n-1\}$ and try again.

See https://isd.mceliece.org for 50 papers studying ISD. Noticeable speedups, mostly in linear algebra. No change in asymptotic attack exponent.

NTRU security vs. BIKE security

NTRU has 3^n possible choices of s encrypted as $r \log_2 q \approx (n/2) \log_2 n$ ciphertext bits.

e.g. `ntruhps2048509`: $3^{1018} \approx 2^{1613}$ choices of s encrypted as 5588 ciphertext bits.

Compared to BIKE, less information about more choices of s. Why isn't this a higher security level?

NTRU security vs. BIKE security

NTRU has 3^n possible choices of s encrypted as $r \log_2 q \approx (n/2) \log_2 n$ ciphertext bits.

e.g. `ntruhps2048509`: $3^{1018} \approx 2^{1613}$ choices of s encrypted as 5588 ciphertext bits.

Compared to BIKE, less information about more choices of s. Why isn't this a higher security level?

Answer: NTRU attacks use combinatorial searches and linear algebra *and* size variations mod q.
Size variations have led to big attack speedups.

Another cola example: `mceliece348864`

System parameters: $(n, q, r) = (3488, 2, 768)$.

Public key determines $K_0, \ldots, K_{3487} \in (\mathbb{Z}/2)^{768}$.

Ciphertext: $C = s_0 K_0 + \cdots + s_{3487} K_{3487}$
for weight-64 vector $(s_0, \ldots, s_{3487}) \in \{0, 1\}$.
Ciphertext has 768 bits, i.e., 96 bytes.

This encrypts $\binom{3488}{64} \approx 2^{456}$ choices of s into just 768 bits. Alice decrypts using a more powerful decoder than the NTRU or BIKE decoders.

This is another system designed for 128-bit security.
Prange uses $\binom{n}{64}/\binom{r}{64} \approx 2^{142.78}$ iterations.

What's the McEliece cutoff gap?

Normally take $n \approx 5r$, weight $w \approx 0.2n/\log_2 n$.

Now $|s| = w^{1/2} \in \Theta(n^{1/2}/(\log n)^{1/2})$.

Can show that most vectors have distance $\Theta(n^{1/2})$ from L. Gap is just $\Theta((\log n)^{1/2})$. "Polylog-gap poly-distance cola encryption".

i.e.: t is *almost* as far from L as most vectors are. This relies critically on the power of Alice's decoder!

Summary of numerical features

Comparing PKEs (public-key encryption systems) by orders of magnitude of $|s|$ etc.:

PKE	q	ct size	$\lvert s\rvert$	cutoff gap
NTRU	n	$n \log n$	$n^{1/2}$	$n^{1/2}$
BIKE	2	n	$n^{1/4}$	$n^{1/4}$
McEliece	2	n	$(n/\log n)^{1/2}$	$(\log n)^{1/2}$

Summary of numerical features

Comparing PKEs (public-key encryption systems) by orders of magnitude of $|s|$ etc.:

PKE	q	ct size	$\lvert s\rvert$	cutoff gap
NTRU	n	$n \log n$	$n^{1/2}$	$n^{1/2}$
BIKE	2	n	$n^{1/4}$	$n^{1/4}$
McEliece	2	n	$(n/\log n)^{1/2}$	$(\log n)^{1/2}$

Can reduce NTRU gaps by having q grow somewhat more slowly than n; but getting down to a polylog gap requires more powerful decoder, as in McEliece.

(Exercise: GAM/LPR is also $n, n \log n, n^{1/2}, n^{1/2}$.)

So McEliece is NP-hard?

So McEliece is NP-hard?

Fact: **No PKE has ever been proven NP-hard.**

So McEliece is NP-hard?

Fact: **No PKE has ever been proven NP-hard.**

The polylog-gap poly-distance close-vector problem is NP-hard, but this doesn't guarantee security or NP-hardness for the McEliece PKE:

- Maybe it's breakable for *almost all* public keys.
- Maybe it's breakable for public keys that *correspond to McEliece secret keys*.

So McEliece is NP-hard?

Fact: **No PKE has ever been proven NP-hard.**

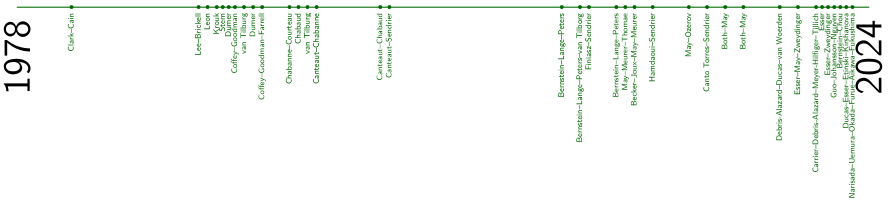
The polylog-gap poly-distance close-vector problem is NP-hard, but this doesn't guarantee security or NP-hardness for the McEliece PKE:

- Maybe it's breakable for *almost all* public keys.
- Maybe it's breakable for public keys that *correspond to McEliece secret keys*.

So the McEliece attack literature studies performance of attacks against uniform random matrices, and studies ways to distinguish Alice's public key from a uniform random matrix.

Stability metric #1: asymptotics

$$\lim_{K \to \infty} \frac{\log_2 \mathrm{AttackCost}_{\mathrm{year}}(K)}{\log_2 \mathrm{AttackCost}_{2024}(K)}$$

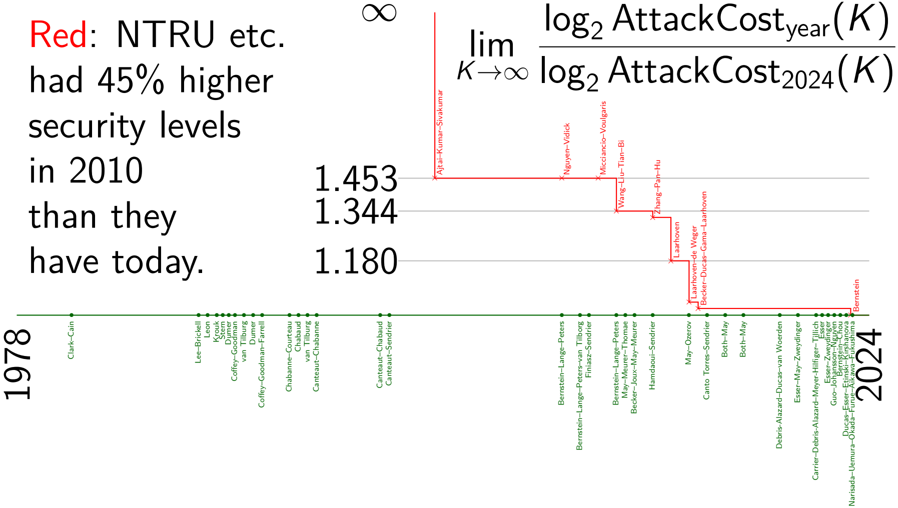


Stability metric #1: asymptotics

Green: McEliece.

Red: NTRU etc. had 45% higher security levels in 2010 than they have today.

$$\lim_{K\to\infty} \frac{\log_2 \mathrm{AttackCost}_{\mathrm{year}}(K)}{\log_2 \mathrm{AttackCost}_{2024}(K)}$$



Stability metric #2: challenges

There are scaled-down challenges to see which values of n academics can break. Latest records:

- $n = 1284$ challenge broken as title of a Eurocrypt 2022 paper.
- $n = 1347$ challenge broken using the 2008 Bernstein–Lange–Peters software, **which is as fast as the 2022 software**.
- $n = 1409$ challenge broken on a GPU cluster.

(Exercise: Find lattice-attack software from 2008. See how slow it is compared to current software.)

Stability metric #3: bit operations

Crypto 2024 Bernstein–Chou "CryptAttackTester: high-assurance attack analysis": software to

- build complete attack circuits,
- predict circuit cost and probability,
- run small attacks to check accuracy.

Bit operations predicted by CryptAttackTester to attack `mceliece348864` ($n = 3488$):

- $2^{156.96}$: `isd1`, attack ideas from the 1980s.
- $2^{150.59}$: `isd2`, latest attacks.

Attacking keys vs. attacking ciphertexts

Asiacrypt 2023 Couvreur–Mora–Tillich distinguish `mceliece348864` key from random using estimated 2^{2231} operations.

Attacking keys vs. attacking ciphertexts

Asiacrypt 2023 Couvreur–Mora–Tillich distinguish `mceliece348864` key from random using estimated 2^{2231} operations. State-of-the-art attacks instead attack ciphertexts, treating public keys as random.

Attacking keys vs. attacking ciphertexts

Asiacrypt 2023 Couvreur–Mora–Tillich distinguish `mceliece348864` key from random using estimated 2^{2231} operations. State-of-the-art attacks instead attack ciphertexts, treating public keys as random.

For each ciphertext size, speed of known attacks:

1. Fastest: Attacking NTRU/LPR/... ciphertexts.
2. Also fastest: Attacking NTRU/LPR/... keys.
3. Much slower: Attacking McEliece ciphertexts.
4. Slowest: Attacking McEliece keys.

Attacking keys vs. attacking ciphertexts

Asiacrypt 2023 Couvreur–Mora–Tillich distinguish `mceliece348864` key from random using estimated 2^{2231} operations. State-of-the-art attacks instead attack ciphertexts, treating public keys as random.

For each ciphertext size, speed of known attacks:

1. Fastest: Attacking NTRU/LPR/… ciphertexts.
2. Also fastest: Attacking NTRU/LPR/… keys.
3. Much slower: Attacking McEliece ciphertexts.
4. Slowest: Attacking McEliece keys.

1+2 exploit weaknesses shared by keys and ciphertexts. (Some people *praise* this sharing.)

Another McEliece security advantage

BIKE has a "quasi-cyclic" structure:
$K_0, \ldots, K_{n-1}$ are actually
$K, xK, x^2K, \ldots, x^{r-1}K, 1, x, x^2, \ldots, x^{r-1}$
for some public $K \in (\mathbb{Z}/2)[x]/(x^r - 1)$.

Another McEliece security advantage

BIKE has a "quasi-cyclic" structure:
$K_0, \ldots, K_{n-1}$ are actually
$K, xK, x^2K, \ldots, x^{r-1}K, 1, x, x^2, \ldots, x^{r-1}$
for some public $K \in (\mathbb{Z}/2)[x]/(x^r - 1)$.

Similar comment applies to NTRU and to many other cola systems, but not McEliece.

Another McEliece security advantage

BIKE has a “quasi-cyclic” structure:
$K_0, \ldots, K_{n-1}$ are actually
$K, xK, x^2K, \ldots, x^{r-1}K, 1, x, x^2, \ldots, x^{r-1}$
for some public $K \in (\mathbb{Z}/2)[x]/(x^r - 1)$.

Similar comment applies to NTRU and to many other cola systems, but not McEliece.

Why this matters: Some cryptosystems (e.g., the original STOC 2009 Gentry FHE system for cyclotomics) have been broken by attacks exploiting this structure. Crypto 2023 $2^{98.77}$ attack against `bikel1` also exploited this structure.

Does quasi-cyclic reduce network traffic?

Quasi-cyclic structure allows smaller public keys: e.g., 699 bytes for `ntruhps2048509` or 1541 bytes for `bike11` vs. 261120 bytes for `mceliece348864`.

Does quasi-cyclic reduce network traffic?

Quasi-cyclic structure allows smaller public keys: e.g., 699 bytes for `ntruhps2048509` or 1541 bytes for `bikel1` vs. 261120 bytes for `mceliece348864`.

However, minimum network traffic comes from (1) reusing keys for many ciphertexts and (2) choosing McEliece. People who say network traffic is important should support the smallest option!

Does quasi-cyclic reduce network traffic?

Quasi-cyclic structure allows smaller public keys: e.g., 699 bytes for `ntruhps2048509` or 1541 bytes for `bikel1` vs. 261120 bytes for `mceliece348864`.

However, minimum network traffic comes from (1) reusing keys for many ciphertexts and (2) choosing McEliece. People who say network traffic is important should support the smallest option!

Quantifying total costs shows that all of these systems are affordable anyway, even scaled up. What really matters is security.

History: knapsack cryptosystems

1978 Hellman–Merkle: a cryptosystem that uses "trapdoor knapsacks" to hide information.

History: knapsack cryptosystems

1978 Hellman–Merkle: a cryptosystem that uses "trapdoor knapsacks" to hide information.

1982 Shamir, 1983 Adleman, 1983 Brickell–Lagarias–Odlyzko, etc.: breaks of practically all "knapsack" proposals.

History: knapsack cryptosystems

1978 Hellman–Merkle: a cryptosystem that uses "trapdoor knapsacks" to hide information.

1982 Shamir, 1983 Adleman, 1983 Brickell–Lagarias–Odlyzko, etc.: breaks of practically all "knapsack" proposals.

This gave "knapsacks" a very bad reputation. "Lattice-based cryptosystems" are knapsack-based cryptosystems trying to avoid this reputation.

In the meantime: McEliece

1978 McEliece: "A public key cryptosystem based on algebraic coding theory".

Uses a powerful decoder from 1970 Goppa. I'll look at this decoder later.

In the meantime: McEliece

1978 McEliece: "A public key cryptosystem based on algebraic coding theory".

Uses a powerful decoder from 1970 Goppa. I'll look at this decoder later.

1986 Niederreiter: space improvement, producing the short ciphertexts that I've been talking about.

More history: NTRU

1996 Hoffstein–Pipher–Silverman preprint “NTRU: a new high speed public key cryptosystem”:

- “In conclusion, for appropriate choice of parameters, NTRU appears to be secure against lattice reduction methods, including any future progress in solving the lattice proximity problem.”
- “NTRU bears a superficial resemblance to the McEliece public key cryptosystem.”

More history: NTRU

1996 Hoffstein–Pipher–Silverman preprint "NTRU: a new high speed public key cryptosystem":

- "In conclusion, for appropriate choice of parameters, NTRU appears to be secure against lattice reduction methods, including any future progress in solving the lattice proximity problem."
- "NTRU bears a superficial resemblance to the McEliece public key cryptosystem."

1997 Coppersmith–Shamir: better lattice attacks.
1998 Hoffstein–Pipher–Silverman: bigger NTRU.

Perspectives on cola cryptography

2003 Bernstein posting that coined the phrase "post-quantum cryptography" mentioned "lattice-type public-key systems, such as McEliece and NTRU".

2017 Barak similarly summarizes "the 'geometric' or 'coding/lattice'-based systems of the type first proposed by McEliece"—but claims without justification that "known lattice-based public-key encryption schemes can be broken using oracle access to an $O(\sqrt{n})$ approximation algorithm for the lattice closest vector problem". Does "lattice-based" exclude McEliece? Why?

Classic McEliece

McEliece's original security goal was one-wayness: stopping attacker from finding random s given C.

2017 "Classic McEliece" converts this into a KEM, adding protection against chosen-ciphertext attacks. QROMCCASecLevel(Classic McEliece) $\geq$ OneWaySecLevel(1978 McEliece) $-$ 5.

Classic McEliece is the main focus of current McEliece deployment.

How does the decoder work?

For the rest of this talk: I'll look at how Alice decodes $(s_0, \ldots, s_{n-1})$ with high weight (small gap).

System parameters:	Typical
• Integer $m \geq 1$.	$m \in \{12, 13\}$
• Integer $n \geq 1$ with $n \leq 2^m$.	$2^{m-1} < n \leq 2^m$
• Integer $w \geq 2$ with $mw < n$.	$w \approx 0.2n/\log_2 n$
• Integer $r = mw$.	$r \approx 0.2n$
• Finite field F with $\#F = 2^m$.	

For `mceliece348864`: $m = 12$; $n = 3488$; $w = 64$; $r = 768$; $F = (\mathbb{Z}/2)[z]/(z^{12} + z^3 + 1)$.

The McEliece secret key

Alice chooses the following secrets:

- Distinct elements $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ of F.
- Monic irreducible deg-w polynomial $g \in F[x]$: i.e., $g = x^w + g_{w-1}x^{w-1} + \cdots + g_1 x + g_0$, each $g_j \in F$, and g is irreducible in $F[x]$.

Note that $g(\alpha_i) \neq 0$ since $w \geq 2$.

Obvious secret-key format has $(n+w)m$ bits.
There are $(2^m)(2^m-1)\cdots(2^m-n+1)$ choices of α, and about $2^{wm}/w$ choices of g.

The McEliece public key

Think of the public key as a linear transformation $H : (\mathbb{Z}/2)^n \to (\mathbb{Z}/2)^{mw}$. Note that everyone can compute the lattice $\{c \in \mathbb{Z}^n : H(c) = 0\}$.

Alice chooses a transformation H satisfying the **Goppa property**: $H(c) = 0$ if and only if $\sum_i c_i A/(x - \alpha_i) \in gF[x]$, where $A = \prod_i (x - \alpha_i)$.

To avoid revealing any information other than the lattice, Alice chooses H in **systematic form**. This means $H(\text{zeropad}(v)) = v$ for all $v \in (\mathbb{Z}/2)^{mw}$, where $\text{zeropad}(v) = (v, 0, 0, \ldots, 0) \in (\mathbb{Z}/2)^n$.

The decoding algorithm

How Alice decodes a ciphertext:

- Input $C \in (\mathbb{Z}/2)^{mw}$.

The decoding algorithm

How Alice decodes a ciphertext:

- Input $C \in (\mathbb{Z}/2)^{mw}$.
- Interpolate $B \in F[x]$ with $\deg B < n$ and $B(\alpha_i) = \text{zeropad}(C)_i A'(\alpha_i)/g^2(\alpha_i)$ for each i, where A' is the derivative of A.

The decoding algorithm

How Alice decodes a ciphertext:

- Input $C \in (\mathbb{Z}/2)^{mw}$.
- Interpolate $B \in F[x]$ with $\deg B < n$ and $B(\alpha_i) = \text{zeropad}(C)_i A'(\alpha_i)/g^2(\alpha_i)$ for each i, where A' is the derivative of A.
- Compute $a, b \in F[x]$ with $\deg a \le w$, $\deg(aB - bA) < n - w$, and $\gcd\{a, b\} = 1$. (This is a "half-gcd" computation.)

The decoding algorithm

How Alice decodes a ciphertext:

- Input $C \in (\mathbb{Z}/2)^{mw}$.
- Interpolate $B \in F[x]$ with $\deg B < n$ and $B(\alpha_i) = \text{zeropad}(C)_i A'(\alpha_i)/g^2(\alpha_i)$ for each i, where A' is the derivative of A.
- Compute $a, b \in F[x]$ with $\deg a \le w$, $\deg(aB - bA) < n - w$, and $\gcd\{a, b\} = 1$. (This is a “half-gcd” computation.)
- Compute $s \in (\mathbb{Z}/2)^n$ with $s_i = [a(\alpha_i) = 0]$, i.e., $s_i = 1$ if and only if $a(\alpha_i) = 0$.
- Output s.

Magic fact: The algorithm works

Fact: If $s \in (\mathbb{Z}/2)^n$ has weight w and $C = H(s)$ then the algorithm outputs s.

Converse: If the algorithm outputs $s \in (\mathbb{Z}/2)^n$ and s has weight w then $C = H(s)$.

To understand *why* this works,
take a course on coding theory,
or read my minicourse on this algorithm:
`cr.yp.to/papers.html#goppadecoding`.