Migrating to the McEliece cryptosystem

Daniel J. Bernstein
Goal: Long-term security

Want to ensure that application data is solidly protected for the foreseeable future.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Goal: Long-term security

Want to ensure that application data is solidly protected for the foreseeable future.

Typical application today is not achieving this:

- Application relies on ECC (ECDH) for public-key encryption.
- Attackers today are recording the application’s ECC-encrypted data.
- Attackers will use future quantum computers to break the encryption.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Goal: Long-term security

Want to ensure that application data is solidly protected for the foreseeable future.

Typical application today is not achieving this:

- Application relies on ECC (ECDH) for public-key encryption.
- Attackers today are recording the application’s ECC-encrypted data.
- Attackers will use future quantum computers to break the encryption.

Typical response: Upgrade to post-quantum encryption. But is the new system secure?

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Example of a failure: SIKE

2011: SIKE is published, says it is better than previous isogeny-based cryptosystems.

2017: SIKE is submitted to NIST competition.

2019: Google and Cloudflare upgrade many users’ HTTPS connections to use SIKE.
Example of a failure: SIKE

2011: SIKE is published, says it is better than previous isogeny-based cryptosystems.

2017: SIKE is submitted to NIST competition.

2019: Google and Cloudflare upgrade many users’ HTTPS connections to use SIKE.

2022: NIST selects SIKE as just one of four candidates for further consideration. “SIKE remains an attractive candidate for standardization because of its small key and ciphertext sizes.”
Example of a failure: SIKE

2011: SIKE is published, says it is better than previous isogeny-based cryptosystems.

2017: SIKE is submitted to NIST competition.

2019: Google and Cloudflare upgrade many users’ HTTPS connections to use SIKE.

2022: NIST selects SIKE as just one of four candidates for further consideration. “SIKE remains an attractive candidate for standardization because of its small key and ciphertext sizes.”

2022: Attacks are published that break SIKE.
### Options for SIKE upgrades

<table>
<thead>
<tr>
<th></th>
<th>short-term security</th>
<th>long-term security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Pre-upgrade: ECC</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Option 1: Encrypt with SIKE and remove previous ECC encryption.
Option 2, what Google and Cloudflare did: Encrypt with ECC and encrypt with SIKE. "Double encryption"; "hybrid encryption".

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
## Options for SIKE upgrades

<table>
<thead>
<tr>
<th></th>
<th>short-term security</th>
<th>long-term security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Pre-upgrade: ECC</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Option 1: SIKE</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Option 2: ECC+SIKE</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Option 1: Encrypt with SIKE and *remove* previous ECC encryption.
Options for SIKE upgrades

<table>
<thead>
<tr>
<th></th>
<th>short-term security</th>
<th>long-term security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Pre-upgrade: ECC</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Option 1: SIKE</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Option 2: ECC+SIKE</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Option 1: Encrypt with SIKE and remove previous ECC encryption.

Option 2, what Google and Cloudflare did: Encrypt with ECC and encrypt with SIKE. “Double encryption”; “hybrid encryption”.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Many more failures

Out of 69 submissions in 2017 to the NIST competition from 260 people, 28% are now known to be breakable:

- CFPKM
- Giophantus
- MQDSS
- Rainbow-1
- SIKE
- Compact LWE
- Guess Again
- pqsigRM
- RankSign
- SRTPI
- DME
- HK17
- qTESLA-s
- Round2
- WalnutDSA
- Edon-K
- LUOV-7
- RaCoSS
- RVB

Attack algorithms have improved against almost all of the remaining submissions.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Many more failures

Out of 69 submissions in 2017 to the NIST competition from 260 people, 28% are now known to be breakable:

- CFPKM
- Giophantus
- MQDSS
- Rainbow-1
- SIKE
- Compact LWE
- Guess Again
- pqsigRM
- RankSign
- SRTP
- DME
- HK17
- qTESLA-s
- Round2
- Edon-K
- LUOV-7
- RaCoSS
- RVB
- WalnutDSA

Attack algorithms have improved against almost all of the remaining submissions.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Example of the dangers

FrodoKEM says it is the most conservative lattice-based system: an “instantiation and implementation” of 2010 Lindner–Peikert.

2010 Lindner–Peikert proposed dimension 256 to “currently offer security levels at least matching those of AES-128” (emphasis added).

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Example of the dangers

FrodoKEM says it is the most conservative lattice-based system: an “instantiation and implementation” of 2010 Lindner–Peikert.

2010 Lindner–Peikert proposed dimension 256 to “currently offer security levels at least matching those of AES-128” (emphasis added).

Many newer advances in attacks have been published. 2010 Lindner–Peikert proposal has much lower security level than AES-128.
Example of the dangers

FrodoKEM says it is the most conservative lattice-based system: an “instantiation and implementation” of 2010 Lindner–Peikert. 2010 Lindner–Peikert proposed dimension 256 to “currently offer security levels at least matching those of AES-128” (emphasis added).

Many newer advances in attacks have been published. 2010 Lindner–Peikert proposal has much lower security level than AES-128. FrodoKEM claims dimension 640 matches AES-128 with a “comfortable security margin”.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
McEliece’s cryptosystem is different

(Robert J. McEliece, 1942–2019)

The McEliece cryptosystem was published in 1978 and has a remarkably stable security level despite many papers trying to break it.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Stability metric #1: asymptotics

\[
\lim_{K \to \infty} \frac{\log_2 \text{AttackCost}_{\text{year}}(K)}{\log_2 \text{AttackCost}_{2023}(K)}
\]

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org) 8
Stability metric #1: asymptotics

Red: Lattices keep losing asymptotic security.

Green: McEliece is asymptotically stable.

\[
\lim_{K \to \infty} \frac{\log_2 \text{AttackCost}_{\text{year}}(K)}{\log_2 \text{AttackCost}_{2023}(K)}
\]

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Stability metric #2: challenges

Important McEliece parameter: “length”. There are *scaled-down challenges* to see what lengths academics can break.

The two most recent records:

- Length-1284 challenge broken as title of a Eurocrypt 2022 *paper*.
Stability metric #2: challenges

Important McEliece parameter: “length”. There are scaled-down challenges to see what lengths academics can break.

The two most recent records:

- Length-1284 challenge broken as title of a Eurocrypt 2022 paper.
Stability metric #2: challenges

Important McEliece parameter: “length”.
There are scaled-down challenges to see what lengths academics can break.

The two most recent records:

- Length-1284 challenge broken as title of a Eurocrypt 2022 paper.

The 2008 software is as fast as the 2022 software. The records come from running attacks on larger computer clusters.
Stability metric #3: bit operations

2023 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 (length 3488):
  • $2^{156.96}$: isd1, attack ideas from the 1980s.
  • $2^{150.59}$: isd2, latest attacks.
What about quantum computers?

McEliece attacks, like AES attacks, are bottlenecked by big searches.

Replacing searches with quantum searches (and “random walks” with “quantum walks”) at worst chops exponents in half. Probably actual impact is much smaller.

Classic McEliece parameter selections use lengths 3488, 4608, 6688, 6960, 8192. 6688, 6960 are recommended for long-term “will never have to change this” security.
Another security metric: tightness

1978 McEliece system was designed to be one-way. This is the natural mathematical concept of security for public-key encryption, but does not stop chosen-ciphertext attacks.

For comparison, typical lattice proof says:

\[ \text{QROMCCASecLevel(lattice-based system)} \geq \text{OneWaySecLevel(new lattice problem)} - 100. \]

Actually, most proofs are worse than this.
Another security metric: tightness

1978 McEliece system was designed to be one-way. This is the natural mathematical concept of security for public-key encryption, but does not stop chosen-ciphertext attacks.

2017 “Classic McEliece” has CCA protection. $\text{QROMCCASecLevel(Classic McEliece)} \geq \text{OneWaySecLevel(1978 McEliece)} - 5$.
Another security metric: tightness

1978 McElieic system was designed to be one-way. This is the natural mathematical concept of security for public-key encryption, but does not stop chosen-ciphertext attacks.

2017 “Classic McElieic” has CCA protection.

\[
\text{QROMCCASecLevel(Classic McElieic)} \geq \text{OneWaySecLevel(1978 McElieic)} - 5.
\]

For comparison, typical lattice proof says:

\[
\text{QROMCCASecLevel(lattice-based system)} \geq \text{OneWaySecLevel(new lattice problem)} - 100.
\]

Actually, most proofs are worse than this.
Lattices strike back

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Lattices strike back

“The mceliece6960119 public key is 1MB. That’s unusable.”

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
OK, let’s talk about performance

1MB is very fast on a modern network. Are Netflix and YouTube unusable?
OK, let’s talk about performance

1MB is very fast on a modern network. Are Netflix and YouTube unusable?

**Quantify the costs in context.**
See if they’re affordable. Skip the hype.
(Should decisions be based on hype wars?)

OK, let’s talk about performance

1MB is very fast on a modern network. Are Netflix and YouTube unusable?

Quantify the costs in context. See if they’re affordable. Skip the hype. (Should decisions be based on hype wars?)

McEliece is already used in some end-to-end secure-messaging systems and the Mullvad and Rosenpass VPNs. Recommended by BSI (Germany) and NCSC (Netherlands). Under consideration by NIST and by ISO.
Revenge of the lattices
Revenge of the lattices

“Even if McEliece is usable, it’s much bigger than lattices. Sending extra network traffic damages the environment.”

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Want to minimize cost? Reuse keys!

Google’s public key can be used to protect any number of ciphertexts to/from Google. Ciphertexts have to be sent end-to-end, and usually have to be sent immediately, even if you’re on an expensive network.

Public keys can be shared locally through existing caching mechanisms (e.g., DNS), and can be distributed in advance.
Want to minimize cost? Reuse keys!

Google’s public key can be used to protect *any number of ciphertexts* to/from Google. Ciphertexts have to be sent end-to-end, and usually have to be sent immediately, even if you’re on an expensive network.

Public keys can be *shared locally* through existing caching mechanisms (e.g., DNS), and can be distributed in advance.

Next slide: 1 site; 10 ISPs; 3 users per ISP. Real world: can easily be 20000 users per ISP.
Public keys out, ciphertexts in

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
Public keys out, ciphertexts in

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
The McEliece size advantage

<table>
<thead>
<tr>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>The site’s public key, M</td>
<td>1047319 bytes</td>
</tr>
<tr>
<td>The site’s public key, K</td>
<td>800 bytes</td>
</tr>
<tr>
<td>Each user’s ciphertext, K</td>
<td>768 bytes</td>
</tr>
<tr>
<td>Each user’s ciphertext, M</td>
<td>194 bytes</td>
</tr>
<tr>
<td>20000 ct + 20000 pk copies, M</td>
<td>20950260000 bytes</td>
</tr>
<tr>
<td>20000 ct + 20000 pk copies, K</td>
<td>31360000 bytes</td>
</tr>
<tr>
<td>20000 ct + 1 pk copy, M</td>
<td>15360800 bytes</td>
</tr>
<tr>
<td>20000 ct + 1 pk copy, M</td>
<td>4927319 bytes</td>
</tr>
</tbody>
</table>

If we’re trying to minimize environmental impact, we should aim for the last line.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem (https://mceliece.org)
## The McEliece size advantage

<table>
<thead>
<tr>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>The site’s public key, $M$</td>
<td>1047319 bytes</td>
</tr>
<tr>
<td>The site’s public key, $K$</td>
<td>800 bytes</td>
</tr>
<tr>
<td>Each user’s ciphertext, $K$</td>
<td>768 bytes</td>
</tr>
<tr>
<td>Each user’s ciphertext, $M$</td>
<td>194 bytes</td>
</tr>
<tr>
<td>20000 ct + 20000 pk copies, $M$</td>
<td>20950260000 bytes</td>
</tr>
<tr>
<td>20000 ct + 20000 pk copies, $K$</td>
<td>31360000 bytes</td>
</tr>
<tr>
<td>20000 ct + 1 pk copy, $K$</td>
<td>15360800 bytes</td>
</tr>
<tr>
<td>20000 ct + 1 pk copy, $M$</td>
<td>4927319 bytes</td>
</tr>
</tbody>
</table>

If we’re trying to minimize environmental impact, we should aim for the last line.

K: kyber512.

M: mceliece6960119, much higher security.

Daniel J. Bernstein, Migrating to the McEliece cryptosystem ([https://mceliece.org](https://mceliece.org))
Classic McEliece implementations

Official software for Classic McEliece is distributed via SUPERCOP benchmarking framework. Four implementations for each parameter set, all passing TIMECOP:

- **ref**: portable, prioritizing simplicity.
- **vec**: portable, 64-bit vectorization.
- **sse**: Intel/AMD, 128-bit vectorization.
- **avx**: Intel/AMD, 256-bit vectorization.

Unofficial: Bouncy Castle (Java and C#), Rust, M4, FPGAs, McTiny, McOutsourcing. Integrations: PQClean, liboqs, Node.js. New: Easy-to-use libmceliece.