Classic McEliece: conservative code-based cryptography

Martin R. Albrecht, Daniel J. Bernstein, Tung Chou,
Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram,
Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen,
Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters,
Peter Schwabe, Nicolas Sendrier, Jakub Szefer,
Cen Jung Tjhai, Martin Tomlinson, Wen Wang

https://classic.mceliece.org/

29 Nov. – 1 Dec. 2022
Fourth NIST PQC workshop

Plaintext confirmation and patent 9912479

- previous-round versions of Classic McEliece use "**plaintext confirmation**" and "implicit rejection" to achieve CCA security.
- A ciphertext is $(He, \text{Hash}(e))$. $\text{Hash}(e)$ is the confirmation value.

Plaintext confirmation and patent 9912479

- previous-round versions of Classic McEliece use "**plaintext confirmation**" and "implicit rejection" to achieve CCA security.
- A ciphertext is $(He, \text{Hash}(e))$. $\text{Hash}(e)$ is the confirmation value.

- Patent 9912479 is about plaintext confirmation.
- The patent does not literally apply to any version of Classic McEliece.
- All of the overlap between the patent and Classic McEliece is already in the prior art.

Plaintext confirmation and patent 9912479

- previous-round versions of Classic McEliece use "**plaintext confirmation**" and "implicit rejection" to achieve CCA security.
- A ciphertext is $(He, \text{Hash}(e))$. $\text{Hash}(e)$ is the confirmation value.

- Patent 9912479 is about plaintext confirmation.
- The patent does not literally apply to any version of Classic McEliece.
- All of the overlap between the patent and Classic McEliece is already in the prior art.

- Changes for the 4th round: remove plaintext confirmation.
- To achieve CCA security, implicit rejection is sufficient.

Changes in encapsulation

3rd round specification:

- Generate a uniform random vector $e \in \mathbb{F}_2^n$ of weight t.
- Compute $C_0 = \text{Encode}(e, T)$.
- Compute $C_1 = \text{Hash}(2, e)$.
- Set $C = (C_0, C_1)$.
- Compute $K = \text{Hash}(1, e, C)$.
- Output ciphertext C and session key K.

Changes in encapsulation

4th round specification:

- Generate a uniform random vector $e \in \mathbb{F}_2^n$ of weight t.
- Compute **C** $= \text{Encode}(e, T)$.
- Compute $C_1 = \text{Hash}(2, e)$.
- Set $C = (C_0, C_1)$.
- Compute $K = \text{Hash}(1, e, C)$.
- Output ciphertext C and session key K.

Changes in decapsulation

3rd round specification:

- Split the ciphertext C as (C_0, C_1) with $C_0 \in \mathbb{F}_2^{n-k}$ and $C_1 \in \mathbb{F}_2^{\ell}$.
- Set $b \leftarrow 1$.
- Extract $s \in \mathbb{F}_2^n$ and $\Gamma' = (g, \alpha'_1, \alpha'_2, \ldots, \alpha'_n)$ from the private key.
- Compute $e \leftarrow \text{Decode}(C_0, \Gamma')$. If $e = \perp$, set $e \leftarrow s$ and $b \leftarrow 0$.
- Compute $C'_1 = \text{Hash}(2, e)$;
- If $C'_1 \neq C_1$, set $e \leftarrow s$ and $b \leftarrow 0$.
- Compute $K = \text{Hash}(b, e, C)$;
- Output session key K.

Changes in decapsulation

4th round specification:

- Split the ciphertext C as (C_0, C_1) with $C_0 \in \mathbb{F}_2^{n-k}$ and $C_1 \in \mathbb{F}_2^{\ell}$.
- Set $b \leftarrow 1$.
- Extract $s \in \mathbb{F}_2^n$ and $\Gamma' = (g, \alpha'_1, \alpha'_2, \ldots, \alpha'_n)$ from the private key.
- Compute $e \leftarrow \text{Decode}(\mathbf{C}, \Gamma')$. If $e = \perp$, set $e \leftarrow s$ and $b \leftarrow 0$.
- Compute $C'_1 = \text{Hash}(2, e)$;
- If $C'_1 \neq C_1$, set $e \leftarrow s$ and $b \leftarrow 0$.
- Compute $K = \text{Hash}(b, e, C)$;
- Output session key K.

Changes in ciphertext sizes

	3rd round	4th round
`mceliece348864` `mceliece348864f`	128	96
`mceliece460896` `mceliece460896f`	188	156
`mceliece6688128` `mceliece6688128f`	240	208
`mceliece6960119` `mceliece6960119f`	226	194
`mceliece8192128` `mceliece8192128f`	240	208

Table: Ciphertext sizes in bytes.

- `Kyber` has 768- to 1568-byte ciphertexts.

Software speed without plaintext confirmation

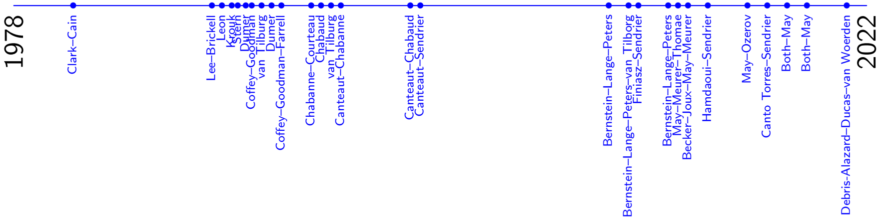
	encapsulation	decapsulation
mceliece348864 mceliece348864f	35495	123480
mceliece460896 mceliece460896f	74870	264845
mceliece6688128 mceliece6688128f	146364	306230
mceliece6960119 mceliece6960119f	158274	287934
mceliece8192128 mceliece8192128f	160866	312147

Table: Haswell cycles from https://bench.cr.yp.to/results-kem.html

- For Cortex-M4, see https://ia.cr/2021/492.
- Ongoing project: libmceliece.

McEliece security stability

$$\lim_{K\to\infty} \frac{\log_2 \text{AttackCost}_{\text{year}}(K)}{\log_2 \text{AttackCost}_{2022}(K)}$$



McEliece security stability

Blue: McEliece.

Red: Lattices have lost much more security. Lattices had 42% higher security levels a decade ago than they have today.

McEliece security stability

Blue: McEliece.

Red: Lattices have lost much more security. Lattices had 42% higher security levels a decade ago than they have today.

Length of the history does matter:

SIKE's security was stable during 2011–2022.

The Esser-Bellini paper "Syndrome Decoding Estimator"

- In one of their "models", to find collisions between two lists of vectors L_1 and L_2, such that n collisions are expected, the cost is considered as

$$|L_1| + |L_2| + n$$

  "operations".

- What is the definition of an "operation"? Hashing operations? Vector additions? Hamming-weight computation?
- Many existing papers do not really count bit operations.

Big ongoing project: Isdbitops

- Representing each ISD variant as a circuit.
- Counts **every bit operation** in the circuit.
- Predicts success probability of the circuit.
- Example:

```
N=3488,K=2720,W=64 stern I=1048576,RE=1024,X=64,YX=22,
P=2,L=30,QU=11,QF=1536,WI=1 [157.110225,157.110226]

N=3488,K=2720,W=64 bjmm I=1048576,RE=1024,X=8,YX=22,
PIJ=1,PI=2,L0=10,L1=21,CP=0,CS=1,D=11,Z=0,QU0=6,QF0=6,
WI0=5,QU1=3,QF1=2048,WI1=1 [156.3598,156.359926]
```

- We hope to announce detailed results next year.

Applications – MULLVAD VPN

https://mullvad.net/en/blog/2022/7/11/experimental-post-quantum-safe-vpn-tunnels/

Experimental post-quantum safe VPN tunnels

11 July 2022 FEATURES APP

Our latest beta (app version 2022.3-beta1) and some WireGuard servers now support VPN tunnels that protect against attackers with access to powerful quantum computers.

The encryption used by WireGuard has no known vulnerabilities. However, the current establishment of a shared secret to use for the encryption is known to be crackable with a strong enough quantum computer.

Although strong enough quantum computers have yet to be demonstrated, having post-quantum secure tunnels today protect against attackers that record encrypted traffic with the hope of decrypting it with a future quantum computer.

Our solution

A WireGuard tunnel is established, and is used to share a secret in such a way that a quantum computer can't figure out the secret even if it had access to the network traffic. We then disconnect and start a new WireGuard tunnel specifying the new shared secret with WireGuard's pre-shared key option. The Post-Quantum secure algorithm used here is Classic McEliece.

Is slow key generation a problem?

- Encapsulation and decapsulation are fast, but key generation is ≈ 1000 times slower.
- Having CCA security means that key pairs can be reused.
- To achieve decent **forward secrecy**, it suffices to generate a key pair, say, every 5 minutes.

Are large public keys a problem?

- Classic McEliece public keys are large: we have been recommending parameter sets with 1MB keys.

- There will be more and more users that can afford 1MB keys.
- Average webpage size is over 2MB now according to httparchive.org ($\approx$ 55% growth rate since 2017).

- End users or local ISPs can cache frequently-used static keys.
- In many applications, much more data needs to be transferred after key agreement, such as video streaming.

Submission documents

Things are divided into several documents:

- Submission overview
  `classic.mceliece.org/nist/mceliece-submission-20221023.pdf`
- Cryptosystem specification
  `classic.mceliece.org/mceliece-spec-20221023.pdf`
- Design rationale
  `classic.mceliece.org/mceliece-rationale-20221023.pdf`
- Guide for security reviewers
  `classic.mceliece.org/mceliece-security-20221023.pdf`
- Guide for implementors
  `classic.mceliece.org/mceliece-impl-20221023.pdf`