Review of Classic McEliece

Friday, November 12th

NIST Postquantum Crypto Seminar

Carl Miller & Ray Perlner

(Not for public distribution.)

The basics

Classic McEliece is a code-based KEM. It is based on the assumed hardness of decoding a certain family of linear codes.

CM makes strong security claims, although its public keys are huge.

Some options for us

1. Standardize Classic McEliece.

2. Standardize BIKE, HQC, or SIKE instead.

3. Standardize only the KEMs that are lattice-based.

Re-introduction to Classic McEliece

Goppa codes

Let $\mathbf{F}_q$ be a finite field (q = a power of 2), and choose distinct $\alpha_i \epsilon \mathbf{F}_q$.

The code generated by the rows of this matrix has Hamming distance $\geq n - l$.

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & \cdots & \alpha_n^2 \\ \vdots & & & \ddots & \vdots \\ \alpha_1^\ell & \alpha_2^\ell & \alpha_3^\ell & \cdots & \alpha_n^\ell \end{bmatrix}$$

Let g be a random irreducible polynomial, and let H be the same matrix with α_i^j replaced by $\alpha_i^j / g(\alpha_i)$.
This is an efficiently decodable code.

Goppa codes

Rewrite H as a binary matrix, and then row-reduce it.

If we're lucky, we get a matrix in **systematic form.**

$$L = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & \cdots & 0 & \\ 0 & 1 & 0 & \cdots & 0 & \\ 0 & 0 & 1 & \cdots & 0 & T \\ \vdots & & & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 & \end{array}\right]$$

The structure of the code is now hidden.

Goppa codes

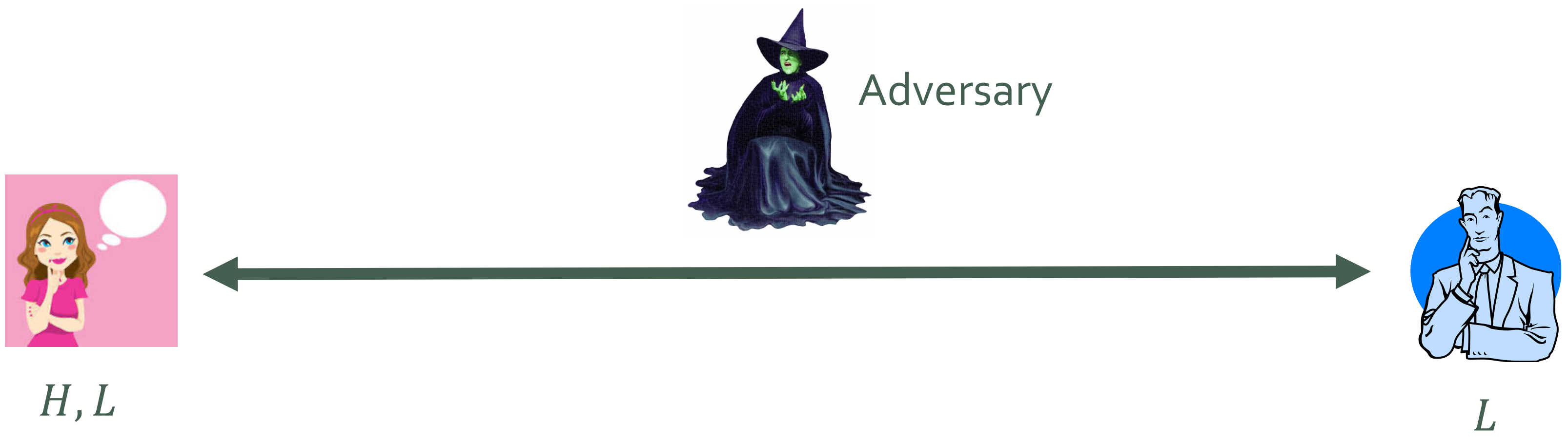
$$L = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & \cdots & 0 & \\ 0 & 1 & 0 & \cdots & 0 & \\ 0 & 0 & 1 & \cdots & 0 & T \\ \vdots & & & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 & \end{array}\right]$$

Let e be random weight-t vector (t small) and let $c = Le$.

Assumption: Given L and c, it is hard to recover e.

Classic McEliece

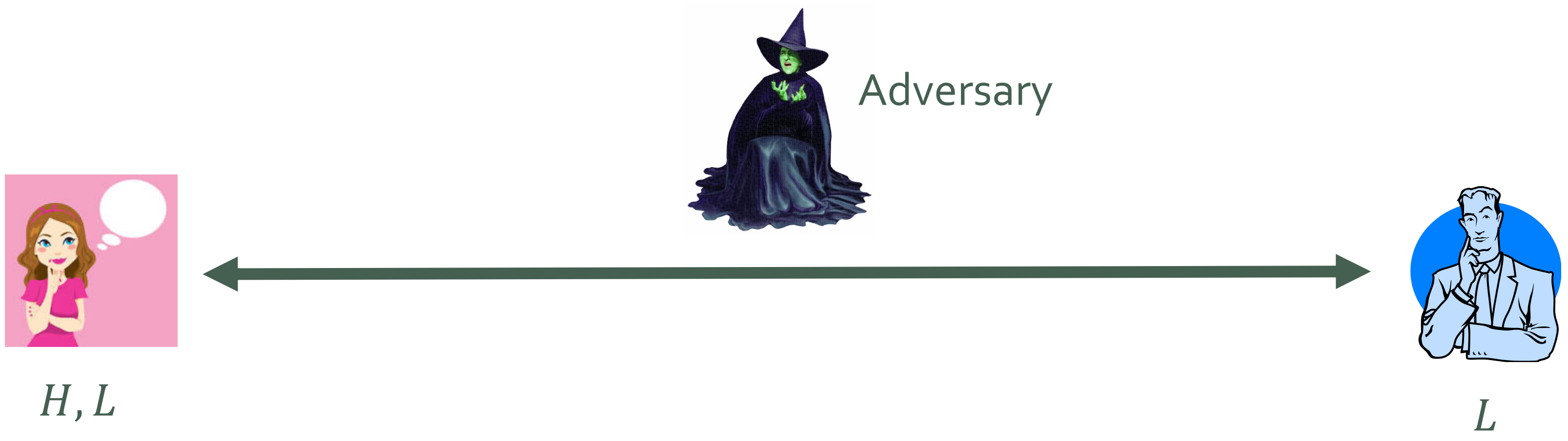
1. Alice broadcasts the (systematic form) matrix L.
2. Bob generates random e, computes $c = Le$, and obtains the key K by hashing e.
3. Bob broadcasts c (+ additional hash info). Alice determines K.



Classic McEliece

Security argument:

1. Assume that the Goppa code L is hard to decode. (The syndrome map is OW-CPA.)
2. Prove that the scheme is IND-CCA2 secure.



Classic McEliece

Security argument:

1. Assume that the Goppa code L is hard to decode. (The syndrome map is OW-CPA.)
2. Prove that the scheme is IND-CCA2 secure.

A well-studied, though not terribly natural (?) assumption.

The authors point to the 40+ year history of work on this protocol.

Classic McEliece

Security argument:

1. Assume that the Goppa code L is hard to decode. (The syndrome map is OW-CPA.)
2. Prove that the scheme is IND-CCA2 secure.

The following paper finishes off the proof: N. Bindel et al., "Tight proofs of CCA security in the quantum random oracle model." (2019)

The authors imply that step 2 is made easier by the fact that their OW-CPA scheme is deterministic and has no decryption failures.

Classic McEliece

Security argument:

1. Assume that the Goppa code L is hard to decode. (The syndrome map is OW-CPA.)
2. Prove that the scheme is IND-CCA2 secure.

The authors have now introduced "f variant" protocols, which allow more general semi-systematic Goppa matrices.

(Small change in performance, no real effect on security.)

Known cryptanalysis

- Key recovery
  - Try to find α_i, g (best info on these attacks actually comes from the BIGQUAKE submission)
    - Brute force guess g and solve linearly for α_i or vice versa
    - Solve a bilinear system for both (see e.g. https://hal.inria.fr/hal-00964265/document)
  - These attacks do not appear competitive with message recovery attacks for CM
  - Public key is generated from a 256-bit seed, so attacker can brute force search for the seed. (May be best attack at category 5, esp. in multi-keypair setting.)
- Message recovery (Information Set Decoding (ISD))
  - Guess a random subset of the error bits (almost all 0s)
  - Linearly solve for the rest of the error bits and check the total weight
  - Use meet in the middle techniques to try a lot of guesses at once
    - Many variants: Stern, Dumer, MMT, BJMM , MO, May Both …

Issues with Concrete Security ISD

- Concrete security estimates for MMT, BJMM etc.
  - Getting Accurate Numbers
  - How much does memory count?
- Multi-ciphertext security
  - Not part of standard IND-CCA definition
  - DOOM
    - Also applies to BIKE, HQC
- Multi-keypair security
  - Not part of standard IND-CCA definition
  - Small Seed (256-bits)
    - Applies to lots of schemes (we've basically said we don't care as long as the seed isn't less than 256 bits)
    - Not clear if this is also an issue for multi-ciphertext security, but it doesn't matter much
- Misuse
  - Kirk Fleming brought up a misuse scenario applying also to several other schemes
  - Kirk Fleming also brought up a misleading (at best) implementation note in the CM spec

Getting Accurate Numbers

- One widely cited source had surprisingly low concrete security estimates for the MMT algorithm (Baldi et al: https://www.researchgate.net/publication/336203573_A_Finite_Regime_Analysis_of_Information_Set_Decoding_Algorithms)
  - If accurate, this would be a problem not just for CM, but BIKE and HQC
  - We made some noise on the forum and crypto stack exchange concerning this
- Seemingly in response to our pleas, a new analysis paper came out: (Esser, Bellini https://eprint.iacr.org/2021/1243.pdf)
  - This paper finds a flaw in Baldi et al's estimate for MMT
  - I will assume Esser, Bellini gives accurate numbers

ISD complexity estimates (Esser, Bellini)

- Magic numbers, Category 1: 143, Category 3: 207, Category 5: 272

	Category 1 ($n = 3488$)		Category 3 ($n = 4608$)		Category 5 (n=6688)		Category 5 ($n = 6960$)		Category 5 ($n = 8192$)	
	T	M	T	M	T	M	T	M	T	M
PRANGE	173	22	217	23	296	24	297	24	334	24
STERN	151	50	193	60	268	80	268	90	303	109
BOTH-MAY	143	88	182	101	250	136	249	137	281	141
MAY-OZEROV	141	89	180	113	246	165	246	160	276	194
BJMM	142	97	183	121	248	160	248	163	278	189
BJMM-P-DW	143	86	183	100	249	160	248	161	279	166
BJMM-DW	144	97	183	100	250	130	250	160	282	164
$M \leq 60$	145	60	187	60	262	58	263	60	298	59
$M \leq 80$	143	74	183	77	258	76	258	74	293	77
$\log M$ access	147	89	187	113	253	165	253	160	283	194
$\sqrt[3]{M}$ access	156	25	199	26	275	36	276	36	312	47

Table 2: Bit security estimates for the suggested parameter sets of the Classic McEliece scheme.

ISD Quantum Security Estimate (Esser Bellini)

- Good news: Even if "Cat 3" parameters are below target, they're still likely to meet category 2.

Scheme	Category	n	quantum security margin
McEliece	1	3488	21
	3	4608	3
	5	6688	18
	5	6960	18
	5	8192	56
BIKE (message)	1	24646	41
	3	49318	47
	5	81946	53
BIKE (key)	1	24646	32
	3	49318	40
	5	81946	43
HQC	1	35338	33
	3	71702	43
	5	115274	44

Table 5: Quantum bit security margin of the corresponding schemes in comparison to breaking AES quantumly.

Decoding One Out of Many (DOOM)

- An attacker can decaps 1 out of N ciphertexts using ISD for about $\frac{1}{\sqrt{N}}$ times the cost of attacking 1 ciphertext out of 1
- ISD works by finding a low weight codeword in some code
  - 1 out of 1 attack: Code is generated by
    - k words $(0, x)$, st. $Lx = 0$,
    - 1 word $(1, s)$ s.t. $Ls = Le$.
  - 1 out of N attack: Code is generated by
    - k words $(0 \dots 0, x)$, st. $Lx = 0$,
    - N words $(0 \dots 010 \dots 0, s_i)$ s.t. $Ls_i = Le_i$.
    - Increasing N makes guessing enough bits of each target about $\sqrt{N}$ times as hard, but there are N times as many targets.

Possible Misuse Scenario
Same Error vector/ Different Keypair

- The attack:
  - Attacker has $L_1 e$, $L_2 e$
  - Attacker can use ISD on a much smaller rank code by taking the intersection of the codes generated by:
    - First code:
      - k words $(0, y)$, st. $L_1 y = 0$,
      - 1 word $(1, s_1)$ s.t. $L_1 s_1 = L_1 e$.
    - Second code:
      - k words $(0, y)$, st. $L_2 y = 0$,
      - 1 word $(1, s_2)$ s.t. $L_2 s_2 = L_2 e$.
  - New code has rank no more than $2k - n + 2$
    - Attack complexity drops approximately from $\left(\frac{n}{n-k}\right)^t$ to $\left(\frac{n}{2(n-k)}\right)^t$
    - E.g. Category 1 parameters lose about 64 bits of security.
- Countermeasure: Hash randomness with public key to generate error vector
- Good enough?: Just use fresh randomness for each ciphertext (should anyway)

Bad Implementation Note

- Assume s is replaced by a constant e_0 at step 4
- Consider a ciphertext consisting of a mauled C_0 and $C_1 = H(2, e_0)$
- Seems like if C_0 is t bits from a codeword, step 6 will fail resulting in an unpredictable K
- But if C_0 is not t bits from a codeword, step 4 will fail and step 6 will succeed, resulting in $K = H(0, e_0, C)$

2.3.3 Decapsulation

The following algorithm DECAP takes as input a ciphertext C and a private key, and outputs a session key K. Here is the algorithm:

1. Split the ciphertext C as (C_0, C_1) with $C_0 \in \mathbb{F}_2^{n-k}$ and $C_1 \in \mathbb{F}_2^{\ell}$.
2. Set $b \leftarrow 1$.
3. Extract $s \in \mathbb{F}_2^n$ and $\Gamma' = (g, \alpha'_1, \alpha'_2, \ldots, \alpha'_n)$ from the private key.
4. Compute $e \leftarrow \text{DECODE}(C_0, \Gamma')$. If $e = \perp$, set $e \leftarrow s$ and $b \leftarrow 0$.
5. Compute $C'_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings.
6. If $C'_1 \neq C_1$, set $e \leftarrow s$ and $b \leftarrow 0$.
7. Compute $K = \mathsf{H}(b, e, C)$; see Section 2.5.2 for H input encodings.
8. Output session key K.

If C is a legitimate ciphertext then $C = (C_0, C_1)$ with $C_0 = He$ for some $e \in \mathbb{F}_2^n$ of weight t and $C_1 = \mathsf{H}(2, e)$. The decoding algorithm will return e as the unique weight-t vector and the $C'_1 = C_1$ check will pass, thus $b = 1$ and K matches the session key computed in encapsulation.

As an implementation note, the output of decapsulation is unchanged if "$e \leftarrow s$" in Step 4 is changed to assign something else to e. Implementors may prefer, e.g., to set e to a fixed n-bit string, or a random n-bit string other than s. However, the definition of decapsulation does depend on e being set to s in Step 6.

Bad Implementation Note History

- Kirk Fleming brought this up on the forum
- Some other people agreed with him
- DJB said everyone was willfully misinterpreting the note
- I think the interpretation which results in an insecure implementation is the obvious interpretation
- We don't have to (and shouldn't) include the note if we publish a Classic McEliece standard
- Are we worried that implementers may implement from the CM submission rather than our standard, though?

Summary

- There's been a lot of discussion on the forum about the concrete security of CM
- Most of the issues are not dealbreakers. If we standardize CM:
  - We should downgrade the claimed category 3 parameters to category 2
  - We should remove the implementation note
  - We may consider minor tweaks for better misuse resistance
  - There is some security loss in the multi-target setting, but probably not enough to be worth doing anything about