Decoding McEliece with a Hint

Elena Kirshanova, joint work Alex May

ACCESS Seminar

https://eprint.iacr.org/2022/525

Agenda

Part I. Classic McEliece KEM & Goppa Code

Part II. Decoding McEliece with a hint

Part I

Classic McEliece KEM & Goppa Code

Classic McEliece

- is a code-based Key Encapsulation Mechanism
- is NIST's 4th Round Candidate
- has large public key size and comparatively slow key generation.
- has the smallest ciphertext sizes of any of the NIST PQC candidates

"NIST is confident in the security of Classic McEliece and would be comfortable standardizing the submitted parameter sets (under a different claimed security strength in some cases). However, it is unclear whether Classic McEliece represents the best option for enough applications to justify standardizing it at this time."[1]

[1] From *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process* `https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf`

Overview of the construction[2]

I. KeyGen :

- Let C be a code over $\mathbb{F}$ of length n correcting $\mathcal{T}$ errors with efficient encoding/decoding algorithms
- $\mathsf{sk} = H_{\mathsf{sk}} \in \mathbb{F}^{n-k \times n}$ - a 'good' parity-check matrix of C
- $\mathsf{pk} = H_{\mathsf{pk}} = [I_{n-k}|T]$ - the systematic form of H_{sk}.
  In general, $\mathsf{pk} = M \cdot H_{\mathsf{sk}} \cdot P$, M – random non-signular, P – permutation matrix.

[2] See `https://classic.mceliece.org/nist/mceliece-20201010.pdf` for the details

Overview of the construction[2]

I. KeyGen :

- Let C be a code over $\mathbb{F}$ of length n correcting $\mathcal{T}$ errors with efficient encoding/decoding algorithms
- $\mathsf{sk} = H_{\mathsf{sk}} \in \mathbb{F}^{n-k \times n}$ - a 'good' parity-check matrix of C
- $\mathsf{pk} = H_{\mathsf{pk}} = [I_{n-k}|T]$ - the systematic form of H_{sk}.
  In general, $\mathsf{pk} = M \cdot H_{\mathsf{sk}} \cdot P$, M – random non-signular, P – permutation matrix.

II. Encaps :

- Encapsulate the key (message) into an error vector $\mathbf{e}$ of weight $\mathcal{T}$
- Return $\mathbf{s} = H_{\mathsf{pk}} \cdot \mathbf{e}$

III. Decaps :

- Decode $\mathbf{s}$ using H_{sk}
- In general, compute $\mathbf{s}' = M^{-1}\mathbf{s}$, decode $\mathbf{s}'$ into $\mathbf{e}'$, compute $\mathbf{e} = \mathbf{e}'P^{-1}$.

[2] See `https://classic.mceliece.org/nist/mceliece-20201010.pdf` for the details

Which code should we use?

A good code should have a large error-correcting capacity and efficient encoding/decoding routines.

- In 1978 McEliece proposes to use Goppa code
- In 1986 Niederraiter proposes to use Reed-Solomon code
- Since then several proposals to use AG-codes

Which code should we use?

A good code should have a large error-correcting capacity and efficient encoding/decoding routines.

- In 1978 McEliece proposed to use Goppa code
- ~~In 1986 Niederrraiter proposes to use Reed-Solomon code~~
- ~~Since then, several proposals to use AG-codes~~

Goppa code

- Fix $L = \{\alpha_1, \ldots, \alpha_n\} \subset \mathbb{F}_{2^m}$ – a set of Goppa points
- Fix $g(x) \in \mathbb{F}_{2^m}[x]$ – a polynomial of degree t s.t. $g(\alpha_i) \neq 0, \forall i$ – a Goppa polynomial

Goppa Code C of length n is

$$C = \Gamma(L, g) = \left\{ \mathbf{c} \in \mathbb{F}_2^n \ : \ \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \bmod g(x) \right\}$$

Goppa code

- Fix $L = \{\alpha_1, \ldots, \alpha_n\} \subset \mathbb{F}_{2^m}$ – a set of Goppa points
- Fix $g(x) \in \mathbb{F}_{2^m}[x]$ – a polynomial of degree t s.t. $g(\alpha_i) \neq 0, \forall i$ – a Goppa polynomial

Goppa Code C of length n is

$$C = \Gamma(L, g) = \left\{ \mathbf{c} \in \mathbb{F}_2^n \ : \ \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \bmod g(x) \right\}$$

Every codeword $\mathbf{c} = (c_1, \ldots, c_n)$ satisfies

$$\sum_i c_i \prod_{j \neq i} (x - \alpha_j) = 0 \bmod g(x).$$

Parity-check matrix

$$\mathbf{c} \in \Gamma(L, g) \iff H\mathbf{c} = 0 \text{ for a parity-check matrix } H.$$

$$\overline{H}_{\text{Goppa}} = \begin{pmatrix} g^{-1}(\alpha_1) & \dots & g^{-1}(\alpha_n) \\ \alpha_1 g^{-1}(\alpha_1) & \dots & \alpha_n g^{-1}(\alpha_n) \\ \vdots & \ddots & \vdots \\ \alpha_1^{t-1} g^{-1}(\alpha_1) & \dots & \alpha_n^{t-1} g^{-1}(\alpha_n) \end{pmatrix} \in \mathbb{F}_{2^m}^{t \times n}$$

- Obtain a parity-check matrix over $\mathbb{F}_q^{tm \times n}$ by considering a natural bijection $\mathbb{F}_{2^m}^{t \times n} \to \mathbb{F}_2^{tm \times n}$ using a fixed basis.
- Gaussian elimination of this matrix gives McEliece public key pk.

Part II

Decoding McEliece with a hint

Advanced Goppa

Theorem 1: On input of $H_{\mathsf{pk}} \in \mathbb{F}_2^{tm \times n}$, an index set $\mathcal{I} \subset \{1, \ldots, n\}$ with $\ell := |I| > tm$, and Goppa points $\alpha_i \in \mathbb{F}_{2^m}, i \in I$, algorithm ADVANCED-GOPPA computes a list $\mathcal{L}$ with the Goppa polynomial $g(x) \in \mathcal{L}$ in $O(n^3)$ operations in $\mathbb{F}_{2^m}$.

Idea: Observe that

$$\sum_{i \in \mathrm{supp}(\mathbf{c})} c_i \prod_{j \in \mathrm{supp}(\mathbf{c}) \setminus \{i\}} (x - \alpha_j) \equiv 0 \bmod g(x). \tag{1}$$

- Assume we know Goppa points α_i for some index set $\mathcal{I} \subset \{1, \ldots, n\}, |\mathcal{I}| \geq tm + 1$.
- Advanced-Goppa constructs $\mathbf{c}$ with $\mathrm{supp}(\mathbf{c}) \subseteq \mathcal{I}$ from H_{pk}.
- Easy: H_{pk} projected to the columns in $\mathcal{I}$ has non-trivial kernel, from which we take $\mathbf{c}$.
- Given $\mathbf{c}$ we compute the lhs of (1), and factor into irreducible polynomials over $\mathbb{F}_{2^m}[x]$.
- Every irreducible deg-t factor is stored in some list $\mathcal{L}$.

Experimental Results

(n, t, m)	$\ell = tm + 1$	$\lvert\mathcal{L}\rvert = 1$	$\ell = tm + 2$	$\lvert\mathcal{L}\rvert = 1$	Av. time
$(3488, 64, 12)$	769	97%	770	100%	18 sec
$(4608, 96, 13)$	1249	99%	1250	100%	54 sec
$(6960, 119, 13)$	1548	99%	1549	100%	91 sec
$(8192, 128, 13)$	1665	99%	1666	100%	105 sec

Table: Experimental results for Advanced-Goppa on a laptop

Recovering the remaining points

Theorem 2: On input $H_{\mathsf{pk}} \in \mathbb{F}_2^{tm \times n}$, the Goppa polynomial $g(x)$, an index set $\mathcal{I} \subset \{1, \ldots, n\}$ with $\ell := |\mathcal{I}| > tm$ such that $\mathrm{rank}(H_{\mathsf{pk}}[\mathcal{I}]) = tm$, and Goppa points $\alpha_i \in \mathbb{F}_{2^m}, i \in \mathcal{I}$, algorithm Goppa-Points outputs in time $\mathcal{O}\left(n^4\right)$ all Goppa points $\alpha_1, \ldots, \alpha_n$.

Idea:

- Construct from H_{pk} a codeword $\mathbf{c}$ with $\mathrm{supp}(\mathbf{c}) \subseteq \mathcal{I} \cup \{r\}$ for some $r \notin \mathcal{I}$.
- Then

$$\sum_{i \in \mathcal{I}} \frac{c_i}{x - \alpha_i} \equiv \frac{1}{x - \alpha_r} \bmod g(x).$$

- Compute lhs, and then solve for α_r.

Recovering the remaining points

(n, t, m)	$\ell = tm + 1$	time
$(3488, 64, 12)$	769	42 sec
$(4608, 96, 13)$	1249	130 sec
$(6960, 119, 13)$	1548	167 sec
$(8192, 128, 13)$	1665	183 sec

Table: Experimental results for Goppa-Points

Faulty Goppa Points

Error model: an attacker obtains $\tilde{\alpha}_1, \ldots \tilde{\alpha}_n$, each $\tilde{\alpha}_i = \alpha_i$ with probability $1 - p$, $p = \Theta(1)$.

Theorem 3: On input $H_{\mathsf{pk}} \in \mathbb{F}_2^{tm \times n}$, erroneous Goppa points $\tilde{\alpha}_1, \ldots \tilde{\alpha}_n$, where pn α_i's are faulty and $n(1-p) \geq tm + 1$, algorithm Faulty-Goppa outputs $g(x)$ and $\alpha_1, \ldots, \alpha_n$ in expected time

$$T = \mathcal{O}\left(n^5 \cdot \frac{\binom{n}{tm+1}}{\binom{n(1-p)}{tm+1}} \right).$$

- Guess a size-$(tm+1)$ subset of $\tilde{\alpha}_1, \ldots \tilde{\alpha}_n$ that contains only correct Goppa points (essentially, Prange)
- Use Advanced-Goppa (Theorem 1) and Goppa-Points (Theorem 2) as tests for the guess.
- In practice either of these two algorithms fails on incorrect Goppa points.

Conclusions and potential directions

Given $tm + 1$ Goppa points

- one can efficiently recover $g(x)$
- one can efficiently recover the remaining points

Open questions:

- What can we do only with $g(x)$?
- Real Side-channel cryptanalysis? See Q.Guo, A.Johansson, T.Johansson "A Key-Recovery Side-Channel Attack on Classic McEliece" [3]

[3] https://eprint.iacr.org/2022/514.pdf

Conclusions and potential directions

Given $tm + 1$ Goppa points

- one can efficiently recover $g(x)$
- one can efficiently recover the remaining points

Open questions:

- What can we do only with $g(x)$?
- Real Side-channel cryptanalysis? See Q.Guo, A.Johansson, T.Johansson "A Key-Recovery Side-Channel Attack on Classic McEliece" [3]

Thank you! Q?

[3] https://eprint.iacr.org/2022/514.pdf