BIG QUAKE

Carl Miller
NIST Computer Security Division
July 13, 2018

NIST PQC Seminar (not for public distribution)

The Basics

- It’s a key encapsulation scheme. (The submission also includes an encryption scheme.)

- It is based on binary quasi-cyclic Goppa codes.

The Basics

- It's a key encapsulation scheme. (The submission also includes an encryption scheme.)

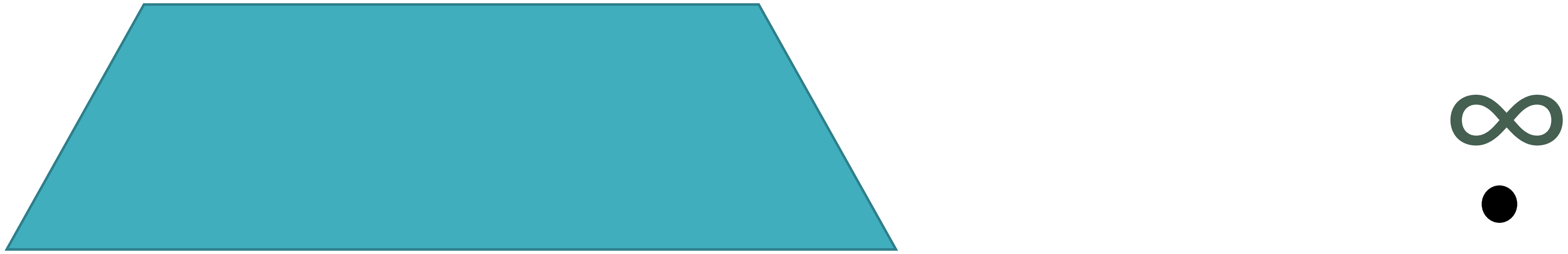
- It is based on binary quasi-cyclic Goppa codes.

(= "big quake")

Binary Goppa Codes

Rational Functions

Consider the complex plane, plus a point at infinity:



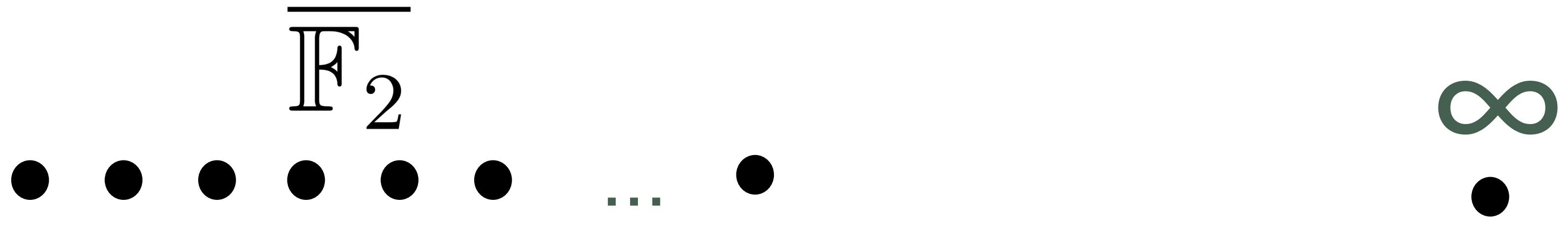
Then, for any rational function,

$$\frac{f(z)}{g(z)} = \frac{(z-y_1)(z-y_2)\cdots(z-y_m)}{(z-x_1)(z-x_2)\cdots(z-x_n)}$$

# of poles = # of zeroes (counting multiplicities).

Rational Functions

The same is true over the algebraic closure of a finite field.



Then, for any rational function,

$$\frac{f(z)}{g(z)} = \frac{(z-y_1)(z-y_2)\cdots(z-y_m)}{(z-x_1)(z-x_2)\cdots(z-x_n)}$$

\# of poles = # of zeroes (counting multiplicities).

Building a code

Fix $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$ in $\overline{\mathbb{F}_2}$. Let $G \subseteq \mathbb{F}_2^n$ be the set of all vectors $(g_1, g_2, \ldots, g_n)$ for which the rational function

$$R(z) = \sum_{i=1}^{n} \frac{g_i}{z - x_i}$$

has zeroes at $y_1, y_2, \ldots, y_m$.

Then, G is a linear code with minimum distance at least m! This is a **binary Goppa code.**

Building a code

Some binary Goppa codes have very good parameters (i.e., size vs. minimum distance).

Goppa codes are easy to decode, given $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$.

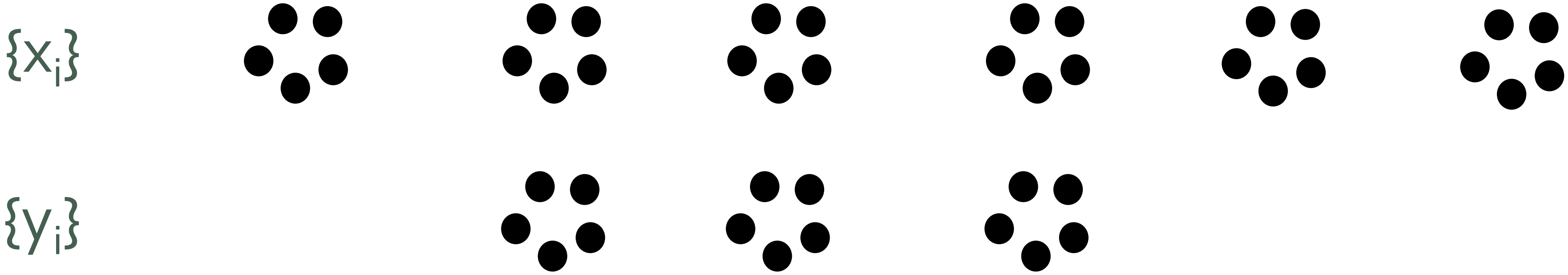
McEliece Cryptosystem

McEliece is an encryption scheme based on codes. The original 1978 paper used Goppa codes.

Idea: Alice prepares an easily-decodable encoding scheme, and then scrambles the generator matrix so it's hard to decode.

Quasi-cyclic binary Goppa codes

We assume that the sets $\{x_1, x_2, \ldots, x_n\}$ and $\{ y_1, y_2, \ldots, y_m\}$ are stabilized by multiplication by some root of unity ζ_l ($l < 20$).



This structure makes the code description more compact.

Protocols

A Public Key Encryption Scheme

1. Bob chooses a random quasi-cyclic binary Goppa code (represented here by $\{x_i\}$, $\{y_i\}$).
2. Bob chooses a parity-check matrix **H** for the code (one that does **<u>not</u>** allow easy decoding). He gives **H** to Alice.

This step is complicated. **H** also has some of the quasi-cyclic structure.

Private key: $\{x_i\}$, $\{y_i\}$.

Public key: **H**

A Public Key Encryption Scheme

3. Alice chooses a random low-weight vector e, and sends an encryption of her message m as

$$(m \oplus \text{hash}(e), \mathbf{H}e)$$

4. Bob decrypts e and recovers m.

$(m \oplus \text{hash}(e), \mathbf{H}e)$

Private key: $\{x_i\}, \{y_i\}$.

Public key: **H**

A Public Key Encryption Scheme

The key encapsulation protocol is a more complicated version of this.

m is chosen uniformly at random, and then e is derived deterministically from m (why?).

$$(m \oplus \mathrm{hash}(e), \mathbf{H}e)$$

Private key: $\{x_i\}$, $\{y_i\}$.

Public key: **H**

Security

Security is argued based on the following assumptions:

- A QCB Goppa code is indistinguishable from a random QC code.
- Syndrome decoding of random QC codes is hard.
- An assumption about the hash function?

$$(m \oplus \mathrm{hash}(e), \mathbf{H}e)$$

Private key: $\{x_i\}$, $\{y_i\}$.

Public key: **H**

Security

For attacks the protocol, one can try to take **H** and recover the original binary Goppa code. The authors describe various ways this could be attempted.
They discuss quantum attacks with Grover's algorithm.

$(m \oplus \text{hash}(e), \mathbf{H}e)$

Private key: $\{x_i\}$, $\{y_i\}$.

Public key: **H**

Parameters

$\{x_i\}$ is chosen from the finite field $\mathbb{F}_{2^m}$, where m = 12, 14, 16, or 18.

$\{y_i\}$ is specified as the set of roots of a polynomial $g(z^\ell)$ with coefficients from the same field.

$(m \oplus \text{hash}(e), \mathbf{H}e)$

Private key: $\{x_i\}$, $\{y_i\}$.

Public key: **H**

Parameters

$\{x_i\}$ is chosen from the finite field $\mathbb{F}_{2^m}$, where m = 12, 14, 16, or 18.

$\{y_i\}$ is specified as the set of roots of a polynomial $g(z^\ell)$ with coefficients from the same field.

5.3.2 Parameters for reaching NIST security level 3 (AES192)

m	n	k	ℓ	Size (bytes)	r	$t = r\ell$ ($\deg g(z^\ell)$)	w_{msg}	Keys	Max Dreg
14	6000	4236	3	311346	42	126	193	5751	11
16	7000	5080	5	243840	24	120	195	6798	12
18	7410	4674	19	84132	8	152	195	2696	16

Performance

7.1 Running time in Milliseconds

	BIG_QUAKE_1	BIG_QUAKE_3	BIG_QUAKE_5
Key Generation	268	2 469	4 717
Encapsulation	1.23	3.00	4.46
Decapsulation	1.41	9.11	13.7

7.2 Space Requirements in Bytes

	BIG_QUAKE_1	BIG_QUAKE_3	BIG_QUAKE_5
Public Key	25 482	84 132	149 800
Secret Key	14 772	30 860	41 804
Ciphertext	201	406	492

No standalone “Advantages & Limitations” section, but the intro talks about savings on computation and key size.

BIG QUAKE

Carl Miller
NIST Computer Security Division
July 13, 2018

NIST PQC Seminar (not for public distribution)