Wild McEliece Incognito

Christiane Peters

Technische Universiteit Eindhoven

joint work with Daniel J. Bernstein and Tanja Lange

Seminaire de Cryptographie
Rennes

April 1, 2011

Bad news

Quantum computers will break the most popular public-key cryptosystems:

- RSA,
- DSA,
- ECDSA,
- ECC,
- HECC
- ...

can be attacked in polynomial time using Shor's algorithm.

Good news

Post-quantum cryptography deals with cryptosystems that

- run on conventional computers and
- are secure against attacks by quantum computers.

Examples:

- Hash-based cryptography.
- Code-based cryptography.
- Lattice-based cryptography.
- Multivariate-quadratic-equations cryptography.
- Secret-key cryptography.

Overview:
Bernstein, Buchmann, and Dahmen, eds., Post-Quantum Cryptography. Springer, 2009.

Today's talk

Code-based cryptography.

1. Background
2. The McEliece cryptosystem
3. Wild McEliece
4. Decoding Wild Goppa codes
5. Notes on list decoding
6. Attacks
7. A new defense

1. Background

2. The McEliece cryptosystem

3. Wild McEliece

4. Decoding Wild Goppa codes

5. Notes on list decoding

6. Attacks

7. A new defense

Linear codes

A binary linear code C of length n and dimension k is a k-dimensional subspace of $\mathbf{F}_2^n$.

A generator matrix for C is a $k \times n$ matrix G such that $C = \left\{m\,G : m \in \mathbf{F}_2^k\right\}$.

The matrix G corresponds to a map $\mathbf{F}_2^k \to \mathbf{F}_2^n$ sending a message m of length k to an n-bit string.

Example: The matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

generates a code of length $n = 8$ and dimension $k = 4$.
Example of a codeword: $c = (0110)G = (11111011)$.

Hamming distance

- The Hamming distance between two words in $\mathbf{F}_2^n$ is the number of coordinates where they differ.
- The Hamming weight of a word is the number of non-zero coordinates.
- The minimum distance of a linear code C is the smallest Hamming weight of a non-zero codeword in C.

Decoding problem

Classical decoding problem: find the closest codeword $c \in C$ to a given $y \in \mathbf{F}_2^n$, assuming that there is a unique closest codeword.

There are lots of code families with fast decoding algorithms

- E.g., Goppa codes/alternant codes, Reed-Solomon codes, Gabidulin codes, Reed-Muller codes, Algebraic-geometric codes, BCH codes etc.

However, given a binary linear code with no obvious structure.

Berlekamp, McEliece, van Tilborg (1978) showed that the general decoding problem for linear codes is NP-complete.

- About $2^{(0.5+o(1))n/\log_2(n)}$ binary operations required for a code of length n and dimension $\approx 0.5n$.

Goppa codes

- Fix a prime power q; a positive integer m, a positive integer $n \leq q^m$; an integer $t < \frac{n}{m}$; distinct $a_1, \ldots, a_n \in \mathbf{F}_{q^m}$;
- and a polynomial $g(x)$ in $\mathbf{F}_{q^m}[x]$ of degree t such that $g(a_i) \neq 0$ for all i.

The Goppa code $\Gamma_q(a_1, \ldots, a_n, g)$ consists of all words $c = (c_1, \ldots, c_n)$ in $\mathbf{F}_q^n$ with

$$\sum_{i=1}^{n} \frac{c_i}{x - a_i} \equiv 0 \pmod{g(x)}$$

- $\Gamma_q(a_1, \ldots, a_n, g)$ has length n and dimension $k \geq n - mt$.
- The minimum distance is at least $\deg g + 1 = t + 1$ (in the binary case $2t + 1$).
- Patterson decoding efficiently decodes t errors in the binary case; otherwise only $t/2$ errors can be corrected.

1. Background

2. The McEliece cryptosystem

3. Wild McEliece

4. Decoding Wild Goppa codes

5. Notes on list decoding

6. Attacks

7. A new defense

Encryption

- Given public system parameters n, k, w.
- The public key is a random-looking $k \times n$ matrix $\hat{G}$ with entries in $\mathbf{F}_q$.
- Encrypt a message $m \in \mathbf{F}_q^k$ as

$$m\hat{G} + e$$

  where $e \in \mathbf{F}_q^n$ is a random error vector of weight w.
- Need to correct w errors to find m.
- Decoding is not easy without knowing the structure of the code generated by $\hat{G}$.

Secret key

The public key $\hat{G}$ has a hidden Goppa-code structure allowing fast decoding:

$$\hat{G} = SGP$$

where

- G is the generator matrix of a Goppa code Γ of length n and dimension k and error-correcting capability w;
- S is a random $k \times k$ invertible matrix; and
- P is a random $n \times n$ permutation matrix.

The triple (G, S, P) forms the secret key.

Note: Detecting this structure, i.e., finding G given $\hat{G}$, seems even more difficult than attacking a random $\hat{G}$.

Decryption

The legitimate receiver knows S, G and P with $\hat{G} = SGP$ and a decoding algorithm for Γ.

How to decrypt $y = m\hat{G} + e$.

1. Compute $yP^{-1} = mSG + eP^{-1}$.
2. Apply the decoding algorithm of Γ to find mSG which is a codeword in Γ from which one obtains m.

In practice (1)

Biswas and Sendrier. McEliece Cryptosystem Implementation: Theory and Practice. PQCrypto 2008.

- 3.0GHz Intel Core 2 Duo E6850 CPU (single-core implementation)

n	k	w	encryption (cycles/byte)	decryption (cycles/byte)	key size	sec level
1024	524	50	243	7938	32 kB	60
2048	1696	32	178	1848	74 kB	87
8192	7958	18	119	312	232 kB	91

Comparison (EBATS preliminary report 2007):

	encryption (cycles/byte)	decryption (cycles/byte)
RSA 1024	800	23100
RSA 2048	834	55922
NTRU	4753	8445

In practice (2)

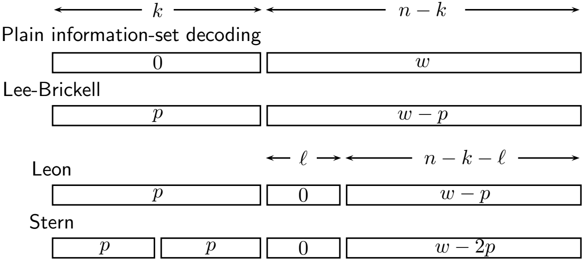
Eisenbarth, Güneysu, Heyse, and Paar. MicroEliece: McEliece for Embedded Devices. CHES 2009.

Linear binary code with $(n, k, w) = (2048, 1751, 27)$ providing 80-bit security.

1. ATxMega192A1 μC (16 kB of SRAM, 192 kB internal Flash memory) (clocked at 32 MHz)
   - generator matrix 448 kB does not fit into the 192 kB internal Flash memory
   - about $14 \cdot 10^6$ cycles for encryption of one message
   - about $20 \cdot 10^6$ cycles for decryption of one message
2. Xilinx Spartan-3AN XC3S1400AN-5 FPGA

Best known attacks

- Information-set decoding algorithms take as input the public generator matrix G, the ciphertext y, and the public error weight w.
- Attacker knows that $y = mG + e$. Try to find the weight-w error vector e by looking for certain error patterns.
- Repeat algorithm with another distribution of errors until e is found.



Parameters for the classical case

Bernstein, Lange, P., PQCrypto 2008:

- Break of McEliece's original parameters $[1024, 524, 50]$.
- Suggestion: for 128-bit security of the McEliece cryptosystem take a length-2960, dimension-2288 classical binary Goppa code ($t = 56$), with 57 errors added by the sender.
- The public-key size here is 1537536 bits.
- Smaller-key variants use other codes such as Reed-Solomon codes, generalized Reed-Solomon codes, quasi-cyclic codes, quasi-dyadic codes or geometric Goppa codes.

Goal: reduce the key size!

Quasi-dyadic codes

Misoczki-Barreto. Compact McEliece Keys from Goppa Codes. SAC 2009.

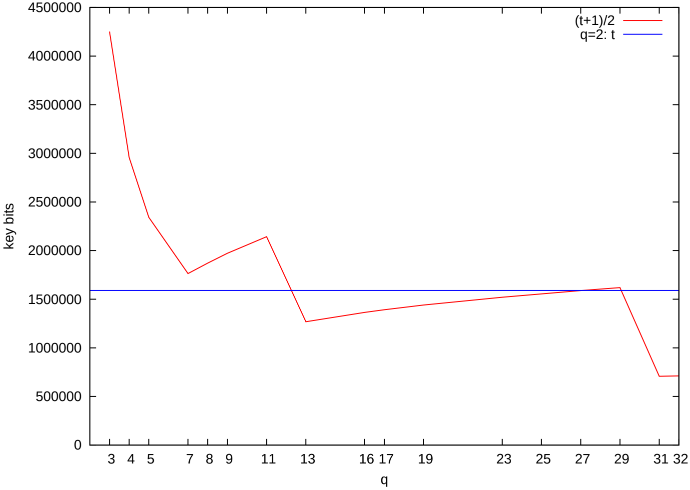
- Hide quasi-dyadic Goppa code as quasi-dyadic public key.
- Certain instances broken (Faugere et al, Eurocrypt 2010; Gauthier Umana and Leander, 2010).
- Binary quasi-dyadic Goppa codes still hold up. `http://eprint.iacr.org/2009/187`
- For 128-bit security the dyadic public key has only 32768 key bits.

Reducing the key size (2)

- Classical Goppa codes are the most confidence-inspiring choice.
- Using Goppa codes over larger fields decreases the key size at the same security level against information-set decoding (P., PQCrypto 2010).
- A Goppa code over $\mathbf{F}_{31}$ leads to a 725741-bit key for 128-bit security.
- Drawback: can correct only $t/2$ errors if $q > 2$ (vs. t in the binary case).
- However, Goppa codes over smaller fields such as $\mathbf{F}_3$ are not competitive in key size with codes over $\mathbf{F}_2$.

Key sizes for various q at a 128-bit security level

McEliece with $\Gamma_q(a_1, \ldots, a_n, g)$ with an alternant decoder.



1. Background

2. The McEliece cryptosystem

3. Wild McEliece

4. Decoding Wild Goppa codes

5. Notes on list decoding

6. Attacks

7. A new defense

Proposal

Use the McEliece cryptosystem with Goppa codes of the form

$$\Gamma_q(a_1, \ldots, a_n, g^{q-1})$$

where g is an irreducible monic polynomial in $\mathbf{F}_{q^m}[x]$ of degree t.

- Note the exponent $q-1$ in g^{q-1}.
- We refer to these codes as wild Goppa codes.

Minimum distance of wild Goppa codes

Theorem (Sugiyama-Kasahara-Hirasawa-Namekawa, 1976)

$$\Gamma_q(a_1, \ldots, a_n, g^{q-1}) = \Gamma_q(a_1, \ldots, a_n, g^q)$$

for a monic squarefree polynomial $g(x)$ in $\mathbf{F}_{q^m}[x]$ of degree t.

- The case $q = 2$ of this theorem is due to Goppa, using a different proof that can be found in many textbooks.

Proof

1. $\Gamma_q(a_1, \ldots, a_n, g^{q-1}) \supseteq \Gamma_q(a_1, \ldots, a_n, g^q)$:
   - If
   $$\sum_i \frac{c_i}{x - a_i} = 0 \text{ in } \mathbf{F}_{q^m}[x]/g^q$$
   then certainly
   $$\sum_i \frac{c_i}{x - a_i} = 0 \text{ in } \mathbf{F}_{q^m}[x]/g^{q-1}.$$

Proof (cont.)

2. $\Gamma_q(a_1, \ldots, a_n, g^{q-1}) \subseteq \Gamma_q(a_1, \ldots, a_n, g^q)$:
   - Consider any $(c_1, c_2, \ldots, c_n) \in \mathbf{F}_q^n$ such that $\sum_i c_i/(x - a_i) = 0$ in $\mathbf{F}_{q^m}[x]/g^{q-1}$.
   - Find an extension k of $\mathbf{F}_{q^m}$ so that g splits into linear factors in $k[x]$.
   - Then
     $$\sum_i \frac{c_i}{x - a_i} = 0 \text{ in } k[x]/g^{q-1},$$
     so
     $$\sum_i \frac{c_i}{x - a_i} = 0 \text{ in } k[x]/(x - r)^{q-1}$$
     for each factor $x - r$ of g.

Proof (cont.)

- The elementary series expansion

$$\frac{1}{x-a_i} = -\frac{1}{a_i-r} - \frac{x-r}{(a_i-r)^2} - \frac{(x-r)^2}{(a_i-r)^3} - \cdots$$

  then implies

$$\sum_i \frac{c_i}{a_i-r} + (x-r)\sum_i \frac{c_i}{(a_i-r)^2} + (x-r)^2\sum_i \frac{c_i}{(a_i-r)^3} + \cdots = 0$$

  in $k[x]/(x-r)^{q-1}$.

- I.e., $\sum_i c_i/(a_i-r) = 0$,
  $\sum_i c_i/(a_i-r)^2 = 0$,
  $\ldots$,
  $\sum_i c_i/(a_i-r)^{q-1} = 0$.

Proof (cont.)

- Take the qth power of the equation $\sum_i c_i/(a_i - r) = 0$, to obtain $\sum_i c_i/(a_i - r)^q = 0$.
- Work backwards to see that $\sum_i c_i/(x - a_i) = 0$ in $k[x]/(x - r)^q$.
- By hypothesis g is the product of its distinct linear factors $x - r$.
- Therefore g^q is the product of the coprime polynomials $(x - r)^q$, and $\sum_i c_i/(x - a_i) = 0$ in $k[x]/g^q$.
- I.e.,
$$\sum_i \frac{c_i}{x - a_i} = 0 \text{ in } \mathbf{F}_{q^m}[x]/g^q.$$
- And thus $(c_1, \ldots, c_n) \in \Gamma_q(a_1, \ldots, a_n, g^q)$.

Error-correcting capability

- Since $\Gamma_q(\dots, g^{q-1}) = \Gamma_q(\dots, g^q)$ the minimum distance of $\Gamma_q(\dots, g^{q-1})$ equals the one of $\Gamma_q(\dots, g^q)$ and is thus $\geq \deg g^q + 1 = qt + 1$.

- We present an alternant decoder that allows efficient correction of $\lfloor qt/2 \rfloor$ errors for $\Gamma_q(\dots, g^{q-1})$.

- Note that the number of efficiently decodable errors increases by a factor of $q/(q-1)$ while the dimension $n - m(q-1)t$ of $\Gamma_q(\dots, g^{q-1})$ stays the same.

Sidestep: Number fields

- Consider the ring of integers $\mathcal{O}_L$ of a number field L and $Q_1, Q_2, \ldots$, the distinct maximal ideals of $\mathcal{O}_L$.

- A prime p ramifies in a number field L if the unique factorization $p\mathcal{O}_L = Q_1^{e_1} Q_2^{e_2} \cdots$ has an exponent e_i larger than 1.

- Each Q_i with $e_i > 1$ is ramified over p; this ramification is wild if e_i is divisible by p.

The “wild” terminology

- If $\mathcal{O}_L/p$ is $\mathbf{F}_p[x]/f$ for f a monic polynomial in $\mathbf{F}_p[x]$. Then $Q_1, Q_2, \ldots$ correspond to the irreducible factors of f, and $e_1, e_2, \ldots$ to the exponents in the factorization of f.
- The ramification corresponding to an irreducible factor ϕ of f is wild if and only if the exponent is divisible by p.
- We also refer to φ^p as being wild, and refer to the corresponding Goppa codes as wild Goppa codes.
- The traditional concept of wild ramification is defined by the characteristic of the base field.
- We take the freedom to generalize the definition of wildness to use the size of $\mathbf{F}_q$ rather than just the characteristic of $\mathbf{F}_q$.

1. Background

2. The McEliece cryptosystem

3. Wild McEliece

4. Decoding Wild Goppa codes

5. Notes on list decoding

6. Attacks

7. A new defense

Polynomial description of Goppa codes

Recall that

$$\begin{aligned}\Gamma &= \Gamma_q(a_1, \ldots, a_n, g^q) \\ &\subseteq \Gamma_{q^m}(a_1, \ldots, a_n, g^q) \\ &= \left\{ \left(\frac{f(a_1)}{h'(a_1)}, \ldots, \frac{f(a_n)}{h'(a_n)} \right) : f \in g^q \mathbf{F}_{q^m}[x], \deg f < n \right\}\end{aligned}$$

where $h = (x - a_1) \cdots (x - a_n)$.

- View target codeword $c = (c_1, \ldots, c_n) \in \Gamma$ as a sequence

$$\left(\frac{f(a_1)}{h'(a_1)}, \ldots, \frac{f(a_n)}{h'(a_n)} \right)$$

  of function values, where f is a multiple of g^q of degree below n.

Classical decoding

Given y, a word of distance $\lfloor qt/2 \rfloor$ from our target codeword.

Reconstruct c from $y = (y_1, \ldots, y_n)$ as follows:

- Interpolate

$$\frac{y_1 h'(a_1)}{g(a_1)^q}, \frac{y_2 h'(a_2)}{g(a_2)^q}, \ldots, \frac{y_n h'(a_n)}{g(a_n)^q}$$

  into a degree-n polynomial $\varphi \in \mathbf{F}_{q^m}[x]$.
- Compute the continued fraction of φ/h to degree $\lfloor qt/2 \rfloor$.: i.e., apply the Euclidean algorithm to h and φ, stopping with the first remainder $v_0 h - v_1 \varphi$ of degree $< n - \lfloor qt/2 \rfloor$.
- Compute $f = (\varphi - v_0 h/v_1) g^q$.
- Compute $c = (f(a_1)/h'(a_1), \ldots, f(a_n)/h'(a_n))$.

Efficiency

This algorithm uses $n^{1+o(1)}$ operations in $\mathbf{F}_{q^m}$ using standard FFT-based subroutines.

- A Python script can be found on my website:
  `http://pqcrypto.org/users/christiane/wild.html`

More decoders

- Can use any Reed-Solomon decoder to reconstruct f/g^q from the values $f(a_1)/g(a_1)^q, \ldots, f(a_n)/g(a_n)^q$ with $\lfloor qt/2 \rfloor$ errors.

- This is an illustration of the following sequence of standard transformations:

  $$\text{Reed–Solomon decoder} \Rightarrow \text{generalized Reed–Solomon decoder}$$
  $$\Rightarrow \text{alternant decoder} \Rightarrow \text{Goppa decoder.}$$

- The resulting decoder corrects $\lfloor (\deg g)/2 \rfloor$ errors for general Goppa codes $\Gamma_q(a_1, \ldots, a_n, g)$.

- In particular, $\lfloor q(\deg g)/2 \rfloor$ errors for $\Gamma_q(a_1, \ldots, a_n, g^q)$; and so $\lfloor q(\deg g)/2 \rfloor$ errors for $\Gamma_q(a_1, \ldots, a_n, g^{q-1})$.

1. Background

2. The McEliece cryptosystem

3. Wild McEliece

4. Decoding Wild Goppa codes

5. Notes on list decoding

6. Attacks

7. A new defense

List decoding (1)

- Using the Guruswami–Sudan list-decoding algorithm we can efficiently correct $n - \sqrt{n(n-qt)} > \lfloor qt/2 \rfloor$ errors in the function values $f(a_1)/g(a_1)^q, \ldots, f(a_n)/g(a_n)^q$.
- Not as fast as a classical decoder but still takes polynomial time.
- Consequently we can handle $n - \sqrt{n(n-qt)}$ errors in the wild Goppa code $\Gamma_q(a_1, \ldots, a_n, g^{q-1})$.

List decoding (2)

- This algorithm can produce several possible codewords c. Unique decoding is ensured by CCA2-secure variants.
- Use conversions of the McEliece cryptosystem by Kobara and Imai (PKC 2001).
- We do not claim that this algorithm is the fastest possible decoder.
- See Bernstein (2008), Augot et al. (2010), and Bernstein (2011) for more efficient (but also more complicated) versions.

1. Background

2. The McEliece cryptosystem

3. Wild McEliece

4. Decoding Wild Goppa codes

5. Notes on list decoding

6. Attacks

7. A new defense

Attacks on Wild McEliece

- The wild McEliece cryptosystem includes, as a special case, the original McEliece cryptosystem.

- A complete break of the wild McEliece cryptosystem would therefore imply a complete break of the original McEliece cryptosystem.

Generic attacks

- The top threat against the original McEliece cryptosystem is information-set decoding.
- The same attack also appears to be the top threat against the wild McEliece cryptosystem for $\mathbf{F}_3$, $\mathbf{F}_4$, etc.
- Use complexity analysis of state-of-the-art information-set decoding for linear codes over $\mathbf{F}_q$ from [P. 2010] to find parameters (q, n, k, t) for Wild McEliece.

Structural attacks

Polynomial-searching attacks:

- There are approximately q^{mt}/t monic irreducible polynomials g of degree t in $\mathbf{F}_{q^m}[x]$, and therefore approximately q^{mt}/t choices of g^{q-1}.

- An attacker can try to guess the Goppa polynomial g^{q-1} and then apply Sendrier's "support-splitting algorithm" to compute a permutation-equivalent code using the set $\{a_1, \ldots, a_n\}$.

- The support-splitting algorithm takes $\{a_1, \ldots, a_n\}$ as an input along with g.

Defense against structural attacks

The first defense is well known and appears to be strong:

- Keep q^{mt}/t extremely large, so that guessing g^{q-1} has negligible chance of success. Our recommended parameters have q^{mt}/t dropping as q grows.
- In fact: our experiments showed that the number of irreducible polynomials g becomes smaller than 2^{128} if $q \geq 11$ when aiming for 128-bit security against information-set decoding.
- So enumerating all possible g's is more efficient than performing information-set decoding.

Defense (2)

The second defense is unusual (strength is unclear):

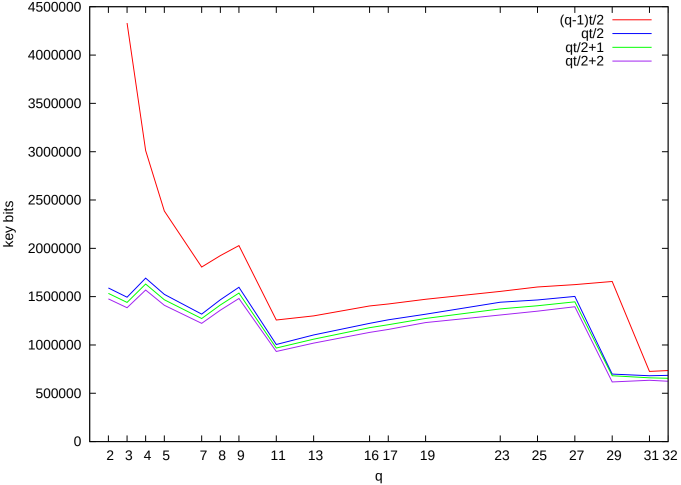
- It is traditional, although not universal, to take $n = 2^m$ and $q = 2$, so that the only possible set $\{a_1, \ldots, a_n\}$ is $\mathbf{F}_{2^m}$.

- Keep n noticeably lower than q^m, so that there are many possible subsets $\{a_1, \ldots, a_n\}$ of $\mathbf{F}_{q^m}$.

- Can the support-splitting idea be generalized to handle many sets $\{a_1, \ldots, a_n\}$ simultaneously?

Key sizes for various q at a 128-bit security level

McEliece with $\Gamma_q(a_1, \ldots, a_n, g^{q-1})$ and $\lfloor (q-1)t/2 \rfloor$, $\lfloor qt/2 \rfloor$, $\lfloor qt/2 \rfloor + 1$, or $\lfloor qt/2 \rfloor + 2$ added errors.



1. Background
2. The McEliece cryptosystem
3. Wild McEliece
4. Decoding Wild Goppa codes
5. Notes on list decoding
6. Attacks
7. A new defense

Hiding wildness

Beelen: proof of Sugiyama et al.'s theorem based on Chinese Remainder Theorem. Hide Goppa codes by using an extra factor.

Wild McEliece Incognito (joint work with Bernstein and Lange):

- Can completely avoid the potential problem of polynomial-searching attacks by using codes with Goppa polynomial $f \cdot g^{q-1}$.
- In particular: Goppa codes of the form $\Gamma_q(a_1, \ldots, a_n, fg^{q-1})$ where f is a squarefree monic polynomial in $\mathbf{F}_{q^m}[x]$ of degree s and g a squarefree monic polynomial in $\mathbf{F}_{q^m}[x]$ of degree t.
- Choose f so that the number of polynomials fg^{q-1} becomes too large to search.

Getting wilder

- For $\deg(f) = s$ and $\deg(g) = t$ the codes can correct up to $\lfloor (s + qt)/2 \rfloor$ errors.
- Efficient decoding of $\lfloor (s + qt)/2 \rfloor$ errors can be done using the same alternant decoders as described before.
- Still “wild.”

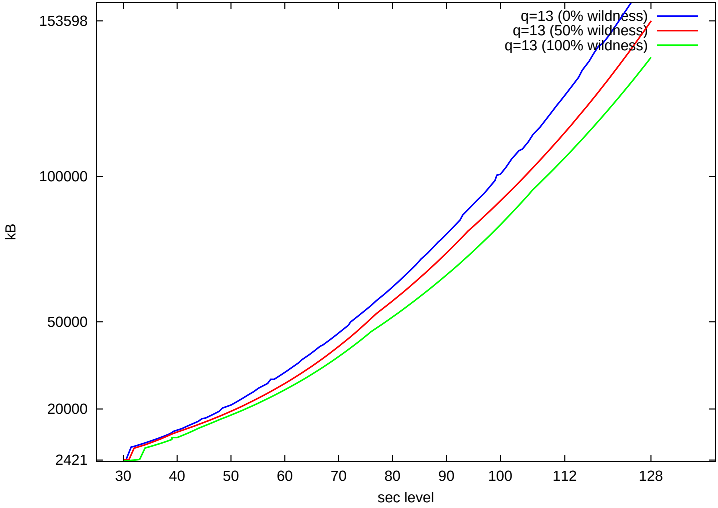
Wildness comparison

Given a wild Goppa code $\Gamma_q(a_1, \ldots, a_n, fg^{q-1})$ with f and g both squarefree and f a degree-s polynomial and g a degree t-polynomial.

- Restrict to "50% wildness", i.e., where the degrees of f and g^{q-1} are balanced by setting $s = (q-1)t$.
- Experiment: consider wild McEliece keys with 0%, 50%, and 100% wildness percentage for $q = 13$.

Key sizes for $q = 13$ for various security levels

McEliece with $\Gamma_q(a_1, \ldots, a_n, fg^{q-1})$ and $\lfloor (s + qt)/2 \rfloor$ added errors.



PQCrypto 2011

Nov 29 – Dec 2, Taipei

http://pq.crypto.tw/pqc11/

Thank you for your attention!