Attacking and defending the McEliece cryptosystem

(Joint work with Daniel J. Bernstein and Tanja Lange)

Christiane Peters

Technische Universiteit Eindhoven

PQCrypto — 2nd Workshop on Postquantum Cryptography

October 18, 2008

1. Review of the McEliece cryptosystem
2. Stern's attack
3. Attack optimization and comparison
4. A successfull attack on the original McEliece parameters
5. Defending the McEliece cryptosystem

Linear codes

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

A generator matrix of an $[n, k]$ code C is a $k \times n$ matrix G such that $C = \{\mathbf{x}\, G : \mathbf{x} \in \mathbf{F}_2^k\}$.

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

A parity-check matrix of an $[n, k]$ code C is an $(n-k) \times n$ matrix H such that $C = \{\mathbf{c} \in \mathbf{F}_2^n : H\, \mathbf{c}^T = 0\}$.

A systematic generator matrix is a generator matrix of the form $(I_k|Q)$ where I_k is the $k \times k$ identity matrix and Q is a $k \times (n-k)$ matrix (redundant part).

The matrix $H = (Q^T|I_{n-k})$ is then a parity-check matrix for C.

Decoding problem

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 nonzero codeword in C.

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

In particular: Decoding a generic binary code of length n and without knowing anything about its structure requires about $2^{(0.5+o(1))n/\log_2(n)}$ binary operations (assuming a rate $\approx 1/2$)

The McEliece cryptosystem

Given a length-n binary Goppa code Γ of dimension k with minimum distance $2t+1$ where $t \approx (n-k)/\log_2(n)$.
(original parameters: $n = 1024$, $k = 524$, $t = 50$)

The McEliece secret key consists of a generator matrix G for Γ, an efficient t-error correcting decoding algorithm for Γ; an $n \times n$ permutation matrix P and a nonsingular $k \times k$ matrix S.

n, k, t are public; but Γ, P, S are randomly generated secrets.

The McEliece public key is the $k \times n$ matrix SGP.

Encryption of a message $\mathbf{m}$ of length k: Compute $\mathbf{m}SGP$ and add a random error vector $\mathbf{e}$ of weight t and length n.
Send $\mathbf{y} = \mathbf{m}SGP + \mathbf{e}$.

McEliece decryption: Compute $\mathbf{y}P^{-1} = \mathbf{m}SG + \mathbf{e}P^{-1}$.
Use decoding algorithm to find $\mathbf{m}S$ and thereby $\mathbf{m}$.

Attacks on the McEliece PKC

Most effective attack against the McEliece cryptosystem is information-set decoding.

Many variants: McEliece (1978), Leon (1988), Lee and Brickell (1988), Stern (1989), van Tilburg (1990), Canteaut and Chabanne (1994), Canteaut and Chabaud (1998), and Canteaut and Sendrier (1998).

Note: Our complexity analysis showed that Stern's original attack beats Canteaut et al. when aiming for 128-bit security

Our attack is most easily understood as a variant of Stern's attack.

Reduce decoding to search for minimum weight words

McEliece ciphertext $\mathbf{y} \in \mathbf{F}_2^n$ has distance t from a unique closest codeword $\mathbf{c} = \mathbf{m}G$ in a code C which has minimum distance at least $2t + 1$.

Find $\mathbf{e}$ of weight t such that $\mathbf{c} = \mathbf{y} - \mathbf{e}$:

- append $\mathbf{y}$ to the list of generators
- and form a generator matrix for $C + \{0, \mathbf{y}\}$.

Then

$$\mathbf{e} = (\mathbf{m}, 1) \left(\frac{G}{\mathbf{m}G + \mathbf{e}} \right)$$

is a codeword in $C + \{0, \mathbf{y}\}$; and it is the only weight-t word.

Bottleneck in all of these attacks is finding the weight-t codeword in $C + \{0, \mathbf{y}\}$ which has slightly larger dimension, namely $k + 1$.

Stern's attack

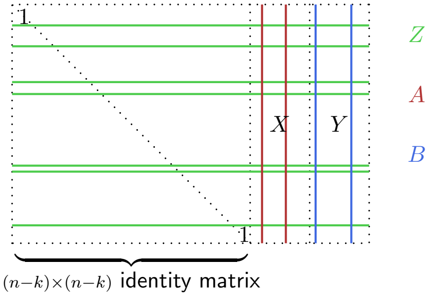
Given $w \geq 0$ and an $(n-k) \times n$ parity check matrix H for a binary $[n, k]$ code C. Find $\mathbf{c} \in C$ of weight w.

Construct $\mathbf{c}$ by looking for exactly w columns of H which add up to 0.

Stern: Choose three disjoint subsets X, Y, Z among the columns of H.
Search for words having exactly $p, p, 0$ ones in those column sets and exactly $w - 2p$ nonzero in the remaining columns.

One iteration of Stern's algorithm

- Select $n-k$ linearly independent columns; apply elementary row operations to get the identity matrix
- Form a set Z of ℓ rows
- Divide remaining k columns into two subsets X and Y.



- For every size-p subset A of X compute the ℓ-bit vector $\pi(A)$ by adding up the columns of the matrix $H' = (H_{i,j})_{i \in Z, j \in A}$. Similarly, compute $\pi(B)$.
- For each collision $\pi(A) = \pi(B)$ compute the sum of the $2p$ columns in $A \cup B$. This sum is an $(n-k)$-bit vector.
- **If** the sum has weight $w - 2p$, we obtain 0 by adding the corresponding $w - 2p$ columns in the $(n-k) \times (n-k)$ submatrix.

  Else select $n-k$ new columns.

1. Review of the McEliece cryptosystem

2. Stern's attack

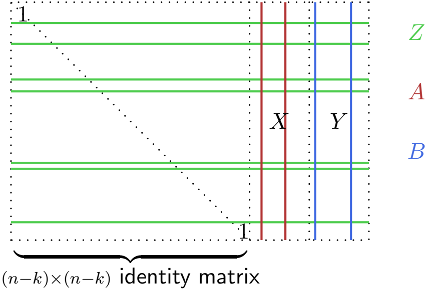
3. Attack optimization and comparison

4. A successfull attack on the original McEliece parameters

5. Defending the McEliece cryptosystem

Our improvements

Step 1



- Starting linear algebra part by using column selection from previous iteration.
- Forcing more existing pivots: *reuse exactly* $n-k-c$ *column selections (Canteaut et al.:* $c=1$*)*
- Faster pivoting
- Multiple choices of Z: *allow* m *disjoint sets* $Z_1, \ldots, Z_m$ *s.t. the word we're looking for has weight* $p, p, 0 \ldots, 0$ *on the sets* $X, Y, Z_1, \ldots, Z_m$

Step 2

- Reusing additions of the ℓ-bit vectors for p-element subsets A of X
- Faster additions after collisions: *consider at most* w *instead of* $n-k$ *cols*

Iterations

Stern: iterations are independent (in each step $n-k$ linearly independent columns are randomly chosen);

Our attack reuses existing pivots: Number of errors in the selected $n-k$ columns is correlated with the number of errors in the columns selected in the next iteration.

Extreme case $c = 1$ considered by Canteaut et al.:
swapping one selected column for one deselected column is quite likely to preserve the number of errors in the selected columns.

We analyzed the impact of selecting c new columns on the number of iterations with a Markov chain computation (generalizing from Canteaut et al.)

`www.win.tue.nl/~cpeters/mceliece.html`

Complexity

Canteaut, Chabaud, and Sendrier: an attacker can decode 50 errors in a $[1024, 524]$ code over $\mathbf{F}_2$ in $2^{64.1}$ bit operations.

Choosing parameters $p = 2$, $m = 2$, $\ell = 20$, $c = 7$, and $r = 7$ in our new attack shows that the same computation can be done in only $2^{60.55}$ bit operations, almost a $12\times$ improvement over Canteaut et al.

The number of iterations drops from $9.85 \cdot 10^{11}$ to $4.21 \cdot 10^{11}$, and the number of bit operations per iteration drops from $20 \cdot 10^6$ to $4 \cdot 10^6$.

1. Review of the McEliece cryptosystem
2. Stern's attack
3. Attack optimization and comparison
4. A successfull attack on the original McEliece parameters
5. Defending the McEliece cryptosystem

Running time in practice

Our attack software extracts a plaintext from a ciphertext by decoding 50 errors in a $[1024, 524]$ binary code.

Attack on a single computer with a 2.4GHz Intel Core 2 Quad Q6600 CPU would need, on average, approximately 1400 days (2^{58} CPU cycles) to complete the attack.

Running the software on 200 such computers would reduce the average time to one week.

Canteaut, Chabaud, and Sendrier: implementation on a 433MHz DEC Alpha CPU; one such computer would need approximately 7400000 days (2^{68} CPU cycles).

Note: Hardware improvements only reduce 7400000 days to 220000 days.

The remaining speedup factor of 150 comes from our improvements of the attack itself.

First successful attack

We were able to extract a plaintext from a ciphertext by decoding 50 errors in a $[1024, 524]$ binary code.

- there were about 200 computers involved, with about 300 cores
- computation finished in under 90 days
  (most of the cores put in far fewer than 90 days of work; some of which were considerably slower than a Core 2)
- used about 8000 core-days
- error vector found by Walton cluster at SFI/HEA Irish Centre of High-End Computing (ICHEC)
- the new parameters $m = 2$, $c = 12$ take only 5000 core-days on average

We gratefully acknowledge contributions of CPU time from

- the Coding and Cryptography Computer Cluster (C4) at TU/e;
- the FACS cluster at CWI;
- the Walton cluster at SFI/HEA Irish Centre for High-End Computing (ICHEC);
- the Department of Electrical Engineering at National Taiwan University;
- the CACAO cluster at LORIA;
- the sandpit Cluster at TU/e;
- the Argo cluster at the University of Illinois at Chicago (UIC);
- the Center for Research and Instruction in Technologies for Electronic Security (RITES) at UIC;
- and D. J. Bernstein and Tanja Lange.

1. Review of the McEliece cryptosystem
2. Stern's attack
3. Attack optimization and comparison
4. A successfull attack on the original McEliece parameters
5. Defending the McEliece cryptosystem

Improvements

Increasing n The most obvious way to defend McEliece's cryptosystem is to increase the code length n.

Allowing values of n *between powers of* 2 allows considerably better optimization of (e.g.) the McEliece public-key size.

Using list decoding to increase w
2008: Bernstein introduced a *list-decoding algorithm for classical irreducible binary Goppa codes*.

The receiver can efficiently decode approximately $n - \sqrt{n(n-2t-2)} \geq t+1$ errors instead of t errors.
The sender can introduce correspondingly more errors.

Unique decoding is ensured by CCA2-secure variants.

Proposed parameters $[n, k]$ for various security levels

For 80-bit security against our attack we propose $[1632, 1269]$ Goppa codes (degree $t = 33$), with 34 errors added by the sender. Public-key size: $k \cdot (n - k) = 460647$ bits.

Without list decoding, and restriction $n = 2^d$: $[2048, 1751]$ Goppa codes ($t = 27$). Public key size: 520047 bits.

For 128-bit security: we propose $[2960, 2288]$ Goppa codes ($t = 56$), with 57 errors added by the sender. Public-key size: 1537536 bits.

For 256-bit security: $[6624, 5129]$ Goppa codes ($t = 115$), with 117 errors added by the sender. Public-key size: 7667855 bits.

Thank you for your attention!