Code-based cryptography V

Information-set decoding

Tanja Lange
with some slides by Tung Chou and Christiane Peters

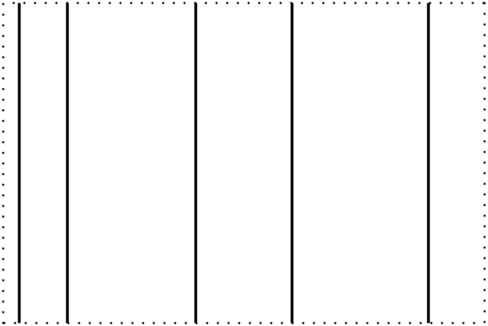
Eindhoven University of Technology

SAC – Post-quantum cryptography

Generic attack: Brute force

Given K and $\mathbf{s} = K\mathbf{e}$, find $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$.

$K =$

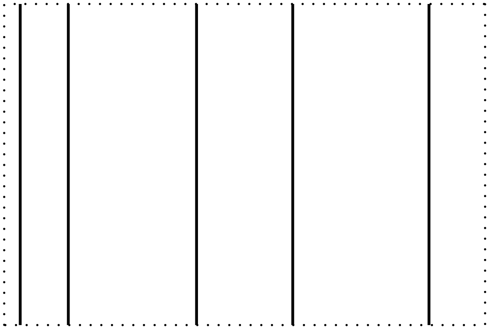


Pick any group of t columns of K, add them and compare with $\mathbf{s}$.

Cost:

Generic attack: Brute force

Given K and $\mathbf{s} = K\mathbf{e}$, find $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$.

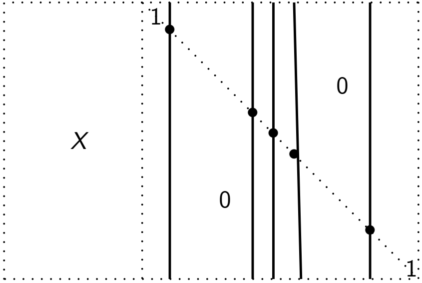
$K =$ 

Pick any group of t columns of K, add them and compare with $\mathbf{s}$.

Cost: $\binom{n}{t}$ sums of t columns.
Can do better so that each try costs only 1 column addition
(after some initial additions).
Cost: $O\binom{n}{t}$ additions of 1 column.

Generic attack: Information-set decoding, 1962 Prange

$K' =$



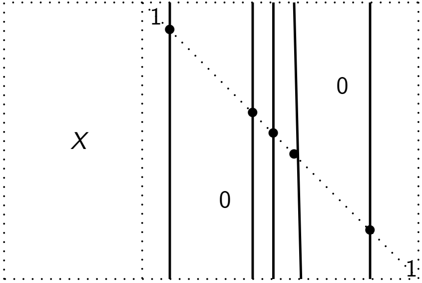
$\mathbf{s}' = K'\mathbf{e}'$

1. Permute K and bring to systematic form $K' = (X|I_{n-k})$.
   (If this fails, repeat with other permutation).
2. Then $K' = UKP$ for some permutation matrix P and U the matrix that produces systematic form.
3. This updates $\mathbf{s}$ to $U\mathbf{s}$.
4. If $\mathrm{wt}(U\mathbf{s}) = t$ then $\mathbf{e}' = (00\ldots0)||U\mathbf{s}$.
   Output unpermuted version of $\mathbf{e}'$.
5. Else return to 1 to rerandomize.

Cost:

Generic attack: Information-set decoding, 1962 Prange

$K' =$

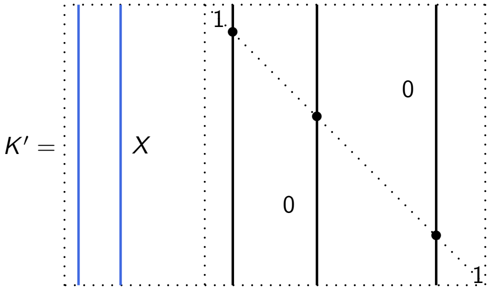


$\mathbf{s}' = K'\mathbf{e}'$

1. Permute K and bring to systematic form $K' = (X|I_{n-k})$.
   (If this fails, repeat with other permutation).
2. Then $K' = UKP$ for some permutation matrix P and U the matrix that produces systematic form.
3. This updates $\mathbf{s}$ to $U\mathbf{s}$.
4. If $\mathrm{wt}(U\mathbf{s}) = t$ then $\mathbf{e}' = (00\ldots0)||U\mathbf{s}$.
   Output unpermuted version of $\mathbf{e}'$.
5. Else return to 1 to rerandomize.

Cost: $O(\binom{n}{t}/\binom{n-k}{t})$ matrix operations.

Lee–Brickell attack

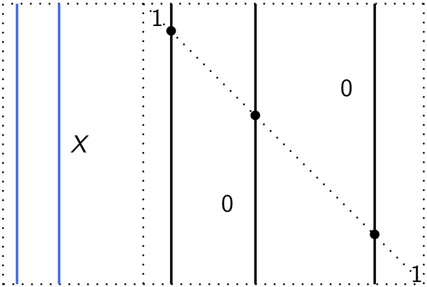


$\mathbf{s}' = K'\mathbf{e}'$

1. Permute K and bring to systematic form $K' = (X|I_{n-k})$.
   (If this fails, repeat with other permutation). $\mathbf{s}$ is updated to $\mathbf{s}'$.
2. For small p, pick p of the k columns on the left, compute their sum $X\mathbf{p}$. ($\mathbf{p}$ is the vector of weight p).
3. If $\mathrm{wt}(\mathbf{s}' + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s}' + X\mathbf{p})$.
   Output unpermuted version of $\mathbf{e}'$.
4. Else return to 2 or return to 1 to rerandomize.

Cost:

Lee–Brickell attack

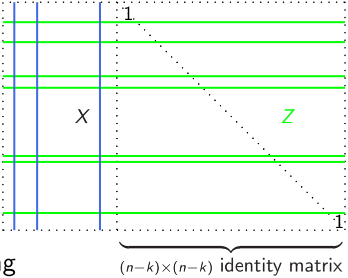
$K' =$ 

$\mathbf{s}' = K'\mathbf{e}'$

1. Permute K and bring to systematic form $K' = (X|I_{n-k})$. (If this fails, repeat with other permutation). $\mathbf{s}$ is updated to $\mathbf{s}'$.
2. For small p, pick p of the k columns on the left, compute their sum $X\mathbf{p}$. ($\mathbf{p}$ is the vector of weight p).
3. If $\mathrm{wt}(\mathbf{s}' + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s}' + X\mathbf{p})$. Output unpermuted version of $\mathbf{e}'$.
4. Else return to 2 or return to 1 to rerandomize.

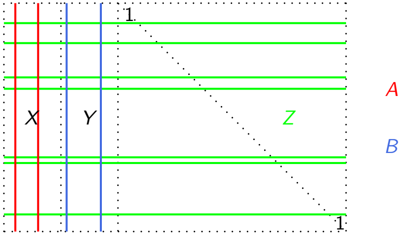
Cost: $O(\binom{n}{t}/(\binom{k}{p}\binom{n-k}{t-p}))$ [matrix operations$+\binom{k}{p}$ column additions].

Leon's attack



- Setup similar to Lee-Brickell's attack.
- Random combinations of p vectors will be dense, so have $\mathrm{wt}(\mathbf{s}' + X\mathbf{p}) \sim (n-k)/2$.
- Idea: Introduce early abort by checking only ℓ positions (selected by set Z, green lines in the picture). This forms $\ell \times k$ matrix X_Z, length-ℓ vector $\mathbf{s}'_Z$.
- Inner loop becomes:
  1. Pick $\mathbf{p}$ with $\mathrm{wt}(\mathbf{p}) = p$.
  2. Compute $X_Z\mathbf{p}$.
  3. If $\mathbf{s}'_Z + X_Z\mathbf{p} \neq 0$ goto 1.
  4. Else compute $X\mathbf{p}$.
     1. If $\mathrm{wt}(\mathbf{s}' + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s}' + X\mathbf{p})$. Output unpermuted version of $\mathbf{e}'$.
     2. Else return to 1 or rerandomize K.
- Note that $\mathbf{s}'_Z + X_Z\mathbf{p} = 0$ means that there are no ones in the positions specified by Z. Small loss in success, big speedup.

Stern's attack



- Setup similar to Leon's and Lee-Brickell's attacks.
- Use the early abort trick, so specify set Z.
- Improve chances of finding $\mathbf{p}$ with $\mathbf{s}' + X_Z\mathbf{p} = 0$:
  - Split left part of K' into two disjoint subsets X and Y.
  - Let $A = \{\mathbf{a} \in \mathbb{F}_2^{k/2} | \mathrm{wt}(\mathbf{a}) = p\}$, $B = \{\mathbf{b} \in \mathbb{F}_2^{k/2} | \mathrm{wt}(\mathbf{b}) = p\}$.
  - Search for words having exactly p ones in X and p ones in Y and exactly $t - 2p$ ones in the remaining columns.
  - Do the latter part as a collision search:
    Compute $\mathbf{s}'_Z + X_Z\mathbf{a}$ for all (many) $\mathbf{a} \in A$, sort.
    Then compute $Y_Z\mathbf{b}$ for $\mathbf{b} \in B$ and look for collisions; expand.
  - Iterate until word with $\mathrm{wt}(\mathbf{s}' + X\mathbf{a} + Y\mathbf{b}) = t - 2p$ is found for some X, Y, Z.
- Select p, ℓ, and the subset of A to minimize overall work.

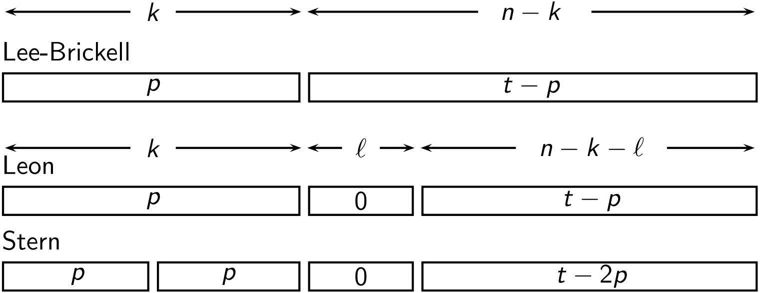
Running time in practice

2008 Bernstein, Lange, Peters.

- Wrote attack software against original McEliece parameters, decoding 50 errors in a $[1024, 524]$ code.
- Lots of optimizations, e.g. cheap updates between $\mathbf{s}'_Z + X_Z\mathbf{a}$ and next value for $\mathbf{a}$; optimized frequency of K randomization.
- Attack on a single computer with a 2.4GHz Intel Core 2 Quad Q6600 CPU would need, on average, 1400 days (2^{58} CPU cycles) to complete the attack.
- About 200 computers involved, with about 300 cores.
- Most of the cores put in far fewer than 90 days of work; some of which were considerably slower than a Core 2.
- Computation used about 8000 core-days.
- Error vector found by Walton cluster at SFI/HEA Irish Centre of High-End Computing (ICHEC).

Information-set decoding

Methods differ in where the “errors” are allowed to be.



Running time is exponential for Goppa parameters n, k, d.

Information-set decoding

Methods differ in where the errors are allowed to be.

$\longleftarrow k \longrightarrow$	$\longleftarrow n-k \longrightarrow$

Lee-Brickell

p	$t-p$

$\longleftarrow k \longrightarrow$	$\longleftarrow \ell \longrightarrow$	$\longleftarrow n-k-\ell \longrightarrow$

Leon

p	0	$t-p$

Stern

p	p	0	$t-2p$

Ball-collision decoding/Dumer/Finiasz-Sendrier

p	p	q	q	$t-2p-2q$
$\longleftarrow k_1 \longrightarrow$	$\longleftarrow k_2 \longrightarrow$	$\leftarrow \ell_1 \rightarrow$	$\leftarrow \ell_2 \rightarrow$	$\longleftarrow n-k-\ell \longrightarrow$

2011 May-Meurer-Thomae and 2012 Becker-Joux-May-Meurer refine multi-level collision search.

Security analysis

Some papers studying algorithms for attackers:
1962 Prange; 1981 Clark–Cain, crediting Omura; 1988 Lee–Brickell; 1988 Leon; 1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman; 1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell; 1993 Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg; 1994 Canteaut–Chabanne; 1998 Canteaut–Chabaud; 1998 Canteaut–Sendrier; 2008 Bernstein–Lange–Peters; 2009 Bernstein–Lange–Peters–van Tilborg; 2009 Bernstein (**post-quantum**); 2009 Finiasz–Sendrier; 2010
Bernstein–Lange–Peters; 2009 Bernstein–Lange–Peters–van Tilborg; 2009 Bernstein (**post-quantum**); 2009 Finiasz–Sendrier; 2010
Bernstein–Lange–Peters; 2011 May–Meurer–Thomae; 2012 Becker–Joux–May–Meurer; 2013 Hamdaoui–Sendrier; 2015 May–Ozerov; 2016 Canto Torres–Sendrier; 2017 Kachigar–Tillich (**post-quantum**); 2017 Both–May; 2018 Both–May; 2018 Kirshanova (**post-quantum**).

Improvements

- Increase n: The most obvious way to defend McEliece's cryptosystem is to increase the code length n.
- Allow values of n between powers of 2: Get considerably better optimization of (e.g.) the McEliece public-key size.
- Use list decoding to increase t: Unique decoding is ensured by CCA2-secure variants.
- 1962 Prange: simple attack idea guiding sizes in 1978 McEliece. The McEliece system (with later key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against Prange's attack. Here $c_0 \approx 0.7418860694$.

Improvements

- Increase n: The most obvious way to defend McEliece's cryptosystem is to increase the code length n.
- Allow values of n between powers of 2: Get considerably better optimization of (e.g.) the McEliece public-key size.
- Use list decoding to increase t: Unique decoding is ensured by CCA2-secure variants.
- 1962 Prange: simple attack idea guiding sizes in 1978 McEliece.
  The McEliece system (with later key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg\lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against Prange's attack.
  Here $c_0 \approx 0.7418860694$.
- Today, the McEliece system (with later key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg\lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against all these attacks.

Improvements

- Increase n: The most obvious way to defend McEliece's cryptosystem is to increase the code length n.
- Allow values of n between powers of 2: Get considerably better optimization of (e.g.) the McEliece public-key size.
- Use list decoding to increase t: Unique decoding is ensured by CCA2-secure variants.
- 1962 Prange: simple attack idea guiding sizes in 1978 McEliece.
  The McEliece system (with later key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against Prange's attack.
  Here $c_0 \approx 0.7418860694$.
- Today, the McEliece system (with later key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against all these attacks.
  Here $c_0 \approx 0.7418860694$.