Cryptanalysis of Code-based Cryptography

Christiane Peters

IMACC 2013

Oxford – December 17, 2013

Outline

1. Motivation
2. Tutorial: How to hide a linear code
3. Tutorial: Decoding attacks
4. Tutorial: Further targets

1. Motivation

2. Tutorial: How to hide a linear code

3. Tutorial: Decoding attacks

4. Tutorial: Further targets

Code-based encryption scheme (Niederreiter version)

Public key: a random-looking $r \times n$ matrix H_{pub} with entries in $\mathbb{F}_q$.

Secret key: H_{pub} has a hidden (algebraic) structure allowing fast decoding.

Encryption of a weight-w word $\mathbf{e} \in \mathbb{F}_q^n$.

- Send ciphertext $\mathbf{s} = H_{pub} \cdot \mathbf{e}$.

Decryption:

- Use linear algebra to undo the conversion from the public code C_{pub} to the secret code C_{sec} and
- make use of the fast decoding algorithm for C_{sec} to find low-weight message $\mathbf{e}$.

Attacks

There are basically two types of attacks:

1. Structural attacks
   - Find the secret code given H_{pub}.
2. Decrypt a single ciphertext
   - Use a generic decoding algorithm.

Design goals

- Choose secret code and conversions so that retrieving C_{sec} is infeasible.
- Choose parameters so that generic attacks need $> 2^b$ bit ops to find a weight-w word (b-bit security).

1. Motivation

2. Tutorial: How to hide a linear code

3. Tutorial: Decoding attacks

4. Tutorial: Further targets

Todo: Hide a linear code

Start with a "linear code allowing fast decoding".

- Niederreiter (1986): use (Generalized) Reed–Solomon codes.

For a prime power q; an integer $0 \le t < q$; a primitive element $\alpha \in \mathbb{F}_q$ the Reed–Solomon code

$$\{(f(0), f(1), f(\alpha), \ldots, f(\alpha^{q-2})) : f \in \mathbb{F}_q[x], \deg f < q - t\}$$

- has length q, dimension $q - t$, and
- minimum distance $t + 1$ (MDS code).
- Berlekamp's algorithm decodes $t/2$ errors in $O(q^2)$.

Aim:

- add defenses against structural attacks while maintaining good error-correction.

Defenses

Scaling

- Pick q elements $\gamma_1, \ldots, \gamma_q \in \mathbb{F}_q^*$ to produce codewords $(\gamma_1 c_1, \ldots, \gamma_q c_q)$.

Permuting

- Pick a permutation $\pi \in S_q$ and permute the coordinates of the codewords to get $(c_{\pi(1)}, \ldots, c_{\pi(q)})$.

Puncturing

- Consider the shortened code containing codewords of the form $(c_{i_1}, \ldots, c_{i_n})$ where $1 \leq i_1 < \cdots < i_n \leq q$.

Generalized Reed–Solomon code

- Fix integers n, t with $0 \le t < n \le q$;
- an ordered set of distinct elements $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$;
- $\gamma_1, \ldots, \gamma_n \in \mathbb{F}_q^*$ (not necessarily distinct).

The Generalized Reed–Solomon code

$$\{(\gamma_1 f(\alpha_1), \ldots, \gamma_n f(\alpha_n)) : f \in \mathbb{F}_q[x], \deg f < n - t\}$$

- has length n, dimension $n - t$, and
- minimum distance $t + 1$ (MDS code).
- Can apply RS decoders to the punctured code after undoing the scaling and permuting.

A GRS parity-check matrix

A parity–check matrix of the Generalized Reed–Solomon code with parameters q, n, t and support $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$ and scalars $\{\gamma_1, \ldots, \gamma_n\} \subseteq \mathbb{F}_q^*$ is given by

$$H = \begin{pmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_n \\ \gamma_1\alpha_1 & \gamma_2\alpha_2 & \cdots & \gamma_n\alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_1\alpha_1^{t-1} & \gamma_2\alpha_2^{t-1} & \cdots & \gamma_n\alpha_n^{t-1} \end{pmatrix}$$

- This is the parity-check matrix of a permuted, scaled, punctured Reed–Solomon code.
- If we keep the α_i and γ_i private: can we use H as H_{pub} for the encryption scheme?

Sidelnikov–Shestakov attack

Recover private key (the α_i's and the γ_i's) from public key in polynomial time.

- Reconstruct codewords of weight $t+1$ from the rows of the systematic generator matrix of the public code (MDS code).

$$\begin{matrix} 1 & 0 & 0 & \cdots & 0 & b_{1,k+1} & \cdots & b_{1,n} \\ 0 & 1 & 0 & \cdots & 0 & b_{2,k+1} & \cdots & b_{2,n} \\ 0 & 0 & 1 & \cdots & 0 & b_{3,k+1} & \cdots & b_{3,n} \\ & & & \ddots & & & & \\ 0 & 0 & 0 & \cdots & 1 & b_{k,k+1} & \cdots & b_{k,n} \end{matrix}$$

$$\underbrace{\qquad\qquad}_{k=n-t} \qquad \underbrace{\qquad\qquad}_{t}$$

Each row corresponds to a codeword polynomial

$$f_{b_i}(x) = c_{b_i} \cdot \prod_{j=1, j\neq i}^{k} (x - \alpha_j) \text{ of degree } k-1 = n-t-1$$

whose coeffs c_{b_i} can be reconstructed from the b_i in $O(k^2 n)$.

Rescuing GRS codes?

Fix: Berger–Loidreau (2005): add ℓ parity checks to the matrix to hide the GRS code.

- Fake parity checks decrease the dimension of the public code (no longer MDS) and thus remove codewords needed for Sidelnikov–Shestakov attack.

Rescuing GRS codes?

~~Fix~~: Berger–Loidreau (2005): add ℓ parity checks to the matrix to hide the GRS code.

- Fake parity checks decrease the dimension of the public code (no longer MDS) and thus remove codewords needed for Sidelnikov–Shestakov attack.

Wieschebrink (2006, 2010): apply Sidelnikov–Shestakov to the square of the public code (likely to be a GRS code containing minimum-weight word of the desired form).

Subfield subcodes

- Let $q = 2^m$;
- fix n, k with $0 \leq k < n \leq q$;
- consider a linear code C over $\mathbb{F}_q$.

The subfield subcode $C|_{\mathbb{F}_2}$ of C is the restriction of C to $\mathbb{F}_2$.

$$C|_{\mathbb{F}_2} = \{(c_1, \ldots, c_n) \in C \mid c_i \in \mathbb{F}_2 \text{ for } i = 1, \ldots, n\}.$$

- Dimension: $\dim(C|_{\mathbb{F}_2}) \geq n - m(n - \dim C)$.
- Minimum distance: $d(C|_{\mathbb{F}_2}) \geq d(C)$.

A family of GRS codes

Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{2^m}$, $h = \prod_{i=1}^{n}(x - \alpha_i)$, and g a degree-t polynomial in $\mathbb{F}_{2^m}[x]$ with $g(\alpha_i) \neq 0$.

- The words $c = (c_1, \ldots, c_n)$ in $\mathbb{F}_{2^m}^n$ with

$$\left\{ \left(\frac{fg}{h'}(\alpha_1), \ldots, \frac{fg}{h'}(\alpha_n) \right) : f \in \mathbb{F}_{2^m}[x],\ \deg(f) < n - t \right\}$$

  form a linear $[n, n-t]$ code in $\mathbb{F}_{2^m}^n$, denoted as $\Gamma_{2^m}(g) = \Gamma_{2^m}(\alpha_1, \ldots, \alpha_n, g)$.

Properties of $\Gamma_{2^m}(g)$

- Minimum distance $d(\Gamma_{2^m}(g)) \geq t + 1$.
- Use Berlekamp's algorithm for decoding up to half the minimum distance.

Goppa codes

The restriction $\Gamma_2(g)$ of $\Gamma_{2^m}(g)$ to the field $\mathbb{F}_2$ is called a Goppa code.

Properties of $\Gamma_2(g)$

- Dimension $k \geq n - mt$.
- Minimum distance $\geq t + 1$.

q-ary Goppa codes

Let q be an arbitrary prime power.

The restriction $\Gamma_q(g)$ of $\Gamma_{q^m}(g)$ to the field $\mathbb{F}_q$ is called a Goppa code.

Properties of $\Gamma_q(g)$

- Dimension $k \geq n - mt$.
- Minimum distance $\geq t + 1$.

Wild Goppa codes

Let q be an arbitrary prime power and g squarefree in $\mathbb{F}_q[x]$.

The restriction $\Gamma_q(g)$ of $\Gamma_{q^m}(g)$ to the field $\mathbb{F}_q$ is called a Goppa code.

Properties of $\Gamma_q(g^{q-1})$

- Dimension $k \geq n - mt$.
- Minimum distance $\geq qt + 1$ since $\Gamma_q(g^q) = \Gamma_q(g^{q-1})$ for squarefree g.

Goppa codes of the form $\Gamma_q(g^{q-1})$ are called wild Goppa codes.

Structural security

Many possible codes for a given parameter set m, n, k.

- Guessing the Goppa polynomial g or the support set $\{\alpha_1, \ldots, \alpha_n\}$ is made infeasible.

Wieschebrink's version of Sidelnikov–Shestakov attack for subcodes not applicable

- square code is not GRS.

Faugère et al. (2010): distinguish hidden Goppa-code matrix from random matrix for high-rate Goppa codes.

- No key recovery.

Alternative constructions

More compact keys:

- Misoczki–Barreto: quasi-dyadic Goppa codes
- Berger et al: quasi-cyclic GRS codes
- Misoczki et al: quasi-cyclic MDPC

So far no good attacks known.

Warning

- Don't overdo it! Compact keys are desirable BUT keep your key space large enough
- Biasi et al. (2012) claimed keys as small as AES; unfortunately, construction yields $\lll 2^{80}$ keys for claimed 80-bit security.

Generic decoding is hard

Syndrome-decoding problem:

- given an $r \times n$ binary matrix H,
- a vector $\mathbf{s} \in \mathbb{F}_2^r$,
- and $w \geq 0$,

find $\mathbf{e} \in \mathbb{F}_2^n$ of weight $\leq w$ such that $H\mathbf{e} = \mathbf{s}$.

Berlekamp, McEliece, van Tilborg (1978) showed that the syndrome-decoding problem is NP-hard.

- The best known generic decoding algorithms all take exponential time.
- About $2^{(0.5+o(1))n/\log n}$ binary operations required for a code of length n, dimension $\approx 0.5n$, and minimum distance $\approx n/\log n$.

Information-set decoding

Best known generic decoding methods rely on so-called information-set decoding or in short: ISD.

Quite a long history:

1962 Prange;
1981 Clark (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–P.;
2009 Bernstein–Lange–P.–van Tilborg;
2009 Finiasz–Sendrier;
2010 P.;
2011 Bernstein–Lange–P.;
2011 Sendrier;
2011 May–Meurer–Thomae;
2012 Becker–Joux–May–Meurer.

- Papers in the last 5 years were aiming at attacking actual cryptographic parameters,
- focusing on either implementations or asymptotic analyses.

Todo: build a generic decoder

Build a (w-bounded) decoder that gets as input

- a parity-check matrix H,
- a ciphertext $\mathbf{s} \in \mathbb{F}_2^r$, and
- an integer $w \geq 0$.

The algorithm tries to determine an error vector $\mathbf{e}$ of weight w such that

$$\mathbf{s} = H\mathbf{e}.$$

Note: from now on consider only linear codes over $\mathbb{F}_2$.

Problem

$$\begin{array}{ccccc|c}
 & \vdots & & & \vdots & \vdots \\
1 & 1 & 1 & & 0 & 0 \\
1 & 0 & 0 & & 1 & 1 \\
0 & 1 & 1 & \cdots\cdots & 0 & 0 \\
0 & 1 & 0 & & 1 & 1 \\
1 & 1 & 1 & & 1 & 0 \\
 & \vdots & & & \vdots & \vdots
\end{array}$$

$\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3 \quad \cdots\cdots \quad \mathbf{c}_n \quad \mathbf{s} = \mathbf{c}_2 + \mathbf{c}_3 + \mathbf{c}_{18} + \mathbf{c}_{20} + \mathbf{c}_{24}$

Given an $r \times n$ matrix, a syndrome $\mathbf{s}$.

Goal: find w columns of H with xor $\mathbf{s}$.

Row randomization

$$
\left[\begin{array}{ccccc|c}
 & \vdots & & & \vdots & \vdots \\
1 & 1 & 1 & & 0 & 0 \\
1 & 0 & 0 & & 1 & 1 \\
0 & 1 & 1 & \cdots\cdots & 0 & 0 \\
0 & 1 & 0 & & 1 & 1 \\
1 & 1 & 1 & & 1 & 0 \\
 & \vdots & & & \vdots & \vdots
\end{array}\right]
$$

$\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3 \quad \cdots\cdots \quad \mathbf{c}_n \quad \mathbf{s} = \mathbf{c}_2 + \mathbf{c}_3 + \mathbf{c}_{18} + \mathbf{c}_{20} + \mathbf{c}_{24}$

Can arbitrarily permute rows without changing the problem.

Goal: find w columns of H with xor $\mathbf{s}$.

Row randomization

$$
\left[\begin{array}{ccccccc|c}
 & \vdots & & & & & \vdots & \vdots \\
1 & 0 & 0 & & & & 1 & 1 \\
1 & 1 & 1 & & & & 0 & 0 \\
0 & 1 & 1 & & \cdots\cdots & & 0 & 0 \\
0 & 1 & 0 & & & & 1 & 1 \\
1 & 1 & 1 & & & & 1 & 0 \\
 & \vdots & & & & & \vdots & \vdots
\end{array}\right]
$$

$\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3 \quad \ldots\ldots \quad \mathbf{c}_n \quad \mathbf{s} = \mathbf{c}_2 + \mathbf{c}_3 + \mathbf{c}_{18} + \mathbf{c}_{20} + \mathbf{c}_{24} +$

Can arbitrarily permute rows without changing the problem.

Goal: find w columns of H with xor $\mathbf{s}$.

Column normalization

$$
\begin{array}{ccc ccc c|c}
 & \vdots & & & & \vdots & \vdots \\
1 & 0 & 0 & & & 1 & 1 \\
1 & 1 & 1 & & & 0 & 0 \\
0 & 1 & 1 & & \cdots\cdots & 0 & 0 \\
0 & 1 & 0 & & & 1 & 1 \\
1 & 1 & 1 & & & 1 & 0 \\
 & \vdots & & & & \vdots & \vdots
\end{array}
$$

$\mathbf{c}_1\mathbf{c}_2\mathbf{c}_3 \quad \cdots\cdots \quad \mathbf{c}_n \quad \mathbf{s} = \mathbf{c}_2 + \mathbf{c}_3 + \mathbf{c}_{18} + \mathbf{c}_{20} + \mathbf{c}_{24}$ ·

Can arbitrarily permute columns without changing the problem.

Goal: find w columns of H with xor $\mathbf{s}$.

Column normalization

$$
\begin{array}{ccccccc|c}
 & \vdots & & & & \vdots & & \vdots \\
0 & 1 & 0 & & & 1 & & 1 \\
1 & 1 & 1 & & & 0 & & 0 \\
1 & 0 & 1 & & \cdots\cdots & 0 & & 0 \\
1 & 0 & 0 & & & 1 & & 1 \\
1 & 1 & 1 & & & 1 & & 0 \\
 & \vdots & & & & \vdots & & \vdots \\
\mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & & \cdots\cdots & \mathbf{c}_n & & \mathbf{s} = \mathbf{c}_1 + \mathbf{c}_3 + \mathbf{c}_{18} + \mathbf{c}_{20} + \mathbf{c}_{24}
\end{array}
$$

Can arbitrarily permute columns without changing the problem.

Goal: find w columns of H with xor $\mathbf{s}$.

Information-set decoding

$$
\left[\begin{array}{ccccccccccc|c}
1 & 0 & 1 & \cdots & 1 & 1 & 0 & 0 & \cdots & & 0 & 0 \\
0 & 1 & 1 & \cdots & 1 & 0 & 1 & 0 & \cdots & & 0 & 1 \\
1 & 1 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & & 0 & 1 \\
 & & & & & & & & & & & 0 \\
 & & & & & & & & \ddots & & & \vdots \\
1 & 0 & 1 & \cdots & 1 & 0 & 0 & 0 & \cdots & & 1 & 0
\end{array}\right]
$$

$$\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3 \qquad \mathbf{c}_k \mathbf{c}_{k+1} \qquad \mathbf{c}_n \quad \mathbf{s} = \mathbf{c}_3 + \mathbf{c}_7 + \mathbf{c}_{28} + \mathbf{c}_{30} -$$

Can add one row to another ⇒ build identity matrix.

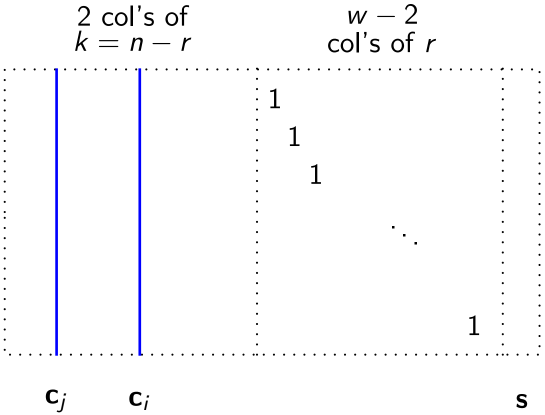
Goal: find w columns which xor $\mathbf{s}$.

Basic information-set decoding

Prange (1962):

- Perhaps xor involves none of the first $n - r$ columns.
- If so, immediately see that **s** is constructed from w columns from the identity submatrix.
- If not, re-randomize and restart – this is a probabilistic algorithm.
- Expect about $\frac{\binom{n}{w}}{\binom{r}{w}}$ iterations.

Lee–Brickell



Check for each pair (i, j) with $1 < i < j \leq k$ if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$ has weight $w - 2$.

Decreasing the number of iterations

Lee–Brickell (1988):

- More likely that xor involves exactly 2 of the first $n - r$ columns.
- Check for each pair (i, j) with $1 < i < j \leq n - r$ if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$ has weight $w - 2$.
- Expect about $\frac{\binom{n}{w}}{\binom{n-r}{2}\binom{r}{w-2}}$ iterations, each checking $\binom{n-r}{2}$ sums $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$.

Decreasing the number of iterations

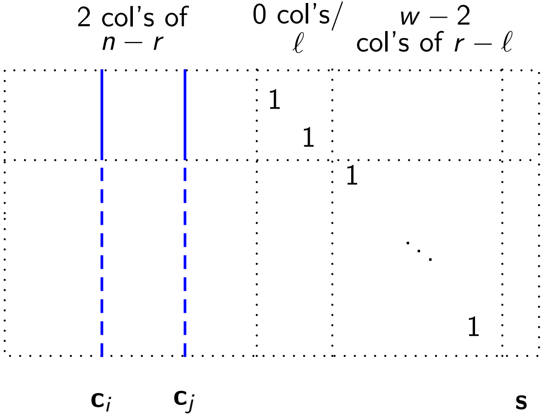
Lee–Brickell (1988):

- More likely that xor involves exactly p of the first $n - r$ columns.
- Check for each pair (i, j) with $1 < i < j \leq n - r$ if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$ has weight $w - p$.
- Expect about $\frac{\binom{n}{w}}{\binom{n-r}{p}\binom{r}{w-p}}$ iterations, each checking $\binom{n-r}{p}$ sums $\mathbf{s} + \mathbf{c}_{i_1} + \cdots + \mathbf{c}_{i_p}$.

Note

- Cost for computing these sums grows with p.
- Choosing $p = \frac{w}{2}$ would minimize # iterations but increase cost of each iterations enormously; $p = 2$ is optimal.

Leon, Krouk, Stern



Check for each pair (i, j) with $1 < i < j \leq n - r$ if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$ has weight $w - 2$ and the first ℓ bits all zero.

- Early abort if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j \neq \mathbf{0}$ on first ℓ bits.

Improvements

Leon (1989), Krouk (1989):

- Check for each (i, j) if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$ has weight $w - 2$ and the first ℓ bits all zero.

- Fast to test, iteration cost decreases.

- Expect about $\frac{\binom{n}{w}}{\binom{n-r}{2}\binom{r-\ell}{w-2}}$ iterations – only a few more than for Lee–Brickell.

Collision decoding

Stern (1989): enforce 0's on first ℓ bits using a meet-in-the-middle approach $\Rightarrow$ square-root improvement.

Strategy

- Split first $n - r$ columns in two disjoint sets of equal size; draw $\mathbf{c}_i$'s from the left, $\mathbf{c}_j$'s from the right set.
- Find collisions between first ℓ bits of $\mathbf{s} + \mathbf{c}_i$ and the first ℓ bits of $\mathbf{c}_j$.
- For each collision, check if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$ has weight $w - 2$.

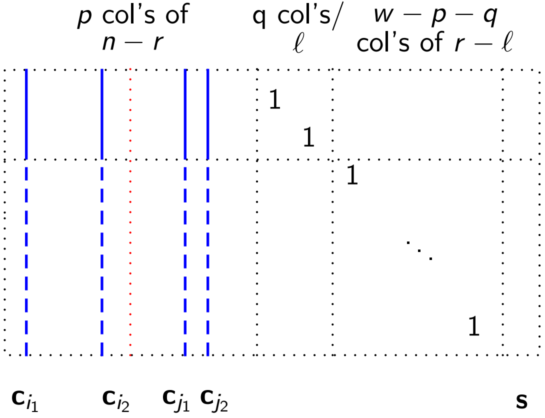
Collision decoding

Stern (1989): enforce 0's on first ℓ bits using a meet-in-the-middle approach $\Rightarrow$ square-root improvement.

Strategy

- Split first $n - r$ columns in two disjoint sets of equal size; draw $\mathbf{c}_i$'s from the left, $\mathbf{c}_j$'s from the right set.
- Find collisions between first ℓ bits of $\mathbf{s} + \mathbf{c}_{i_1} + \cdots + \mathbf{c}_{i_{p/2}}$ and the first ℓ bits of $\mathbf{c}_{j_1} + \cdots + \mathbf{c}_{j_{p/2}}$.
- For each collision, check if $\mathbf{s} + \mathbf{c}_{i_1} + \cdots + \mathbf{c}_{i_{p/2}} + \mathbf{c}_{j_1} \cdots + \mathbf{c}_{j_{p/2}}$ has weight $w - p$.
- Expect about $\frac{\binom{n}{w}}{\binom{(n-r)/2}{p/2}^2 \binom{r-\ell}{w-p}}$ iterations.

Ball-collision decoding



- Disjoint split of columns on the left.
- Allow a few zeros in the previously “forbidden zone”.

Ball-collision decoding

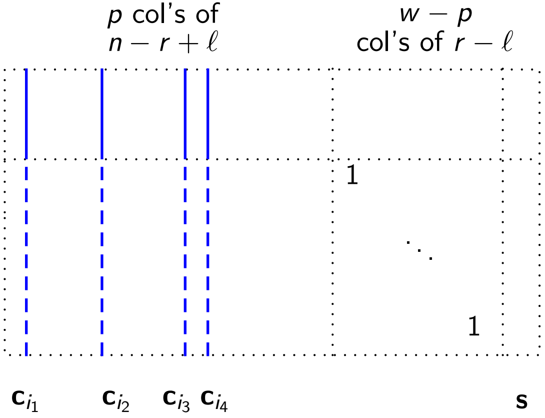
Bernstein, Lange, P. (2011):

- Find collisions between the Hamming ball of radius q around $\mathbf{s} + \mathbf{c}_{i_1} + \cdots + \mathbf{c}_{i_p}$ and the Hamming ball of radius q around $\mathbf{c}_{j_1} + \cdots + \mathbf{c}_{j_p}$.

- Main theorem: (asymptotically) exponential speedup of ball-collision decoding over Stern's collision decoding.

- Reference implementation of ball-collision decoding: `http://cr.yp.to/ballcoll.html`

Using representations



- Only partial Gauss elimination.
- Consider selected sums of p columns out of $n - r + \ell$.

Increase number of p-sums

May–Meurer–Thomae (2011), Becker–Joux–May–Meurer (2012):

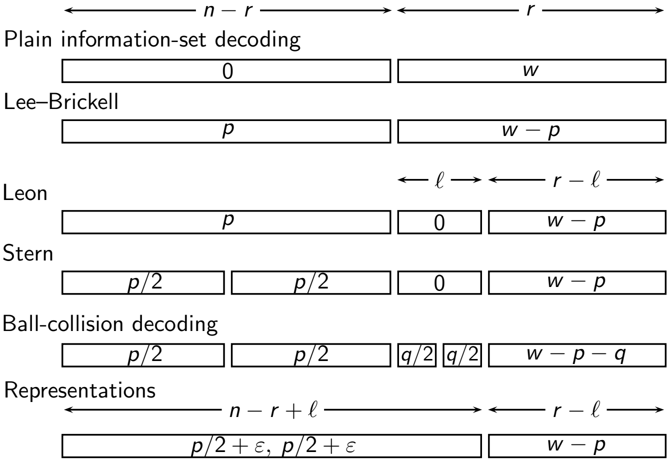
- Increase number of words with 0's on first ℓ positions by removing the split of $n - r$ columns into in two disjoint sets.

- Do not check all $\binom{k}{p}$ sums $\mathbf{s} + \mathbf{c}_{i_1} + \cdots + \mathbf{c}_{i_p}$.

- Examine a fraction of those sums using representation technique by Howgrave-Graham–Joux (2010).

- Main theorem: (asymptotically) exponential speedup of representation technique over ball-collision decoding.

Error distributions



Asymptotics

Recent papers are mostly asymptotic speedups.

- Gains are significant for coding-theoretic values for the minimum distance (Gilbert–Varshamov radius).

- For cryptographic applications, only small differences in cost between Stern's algorithm, ball-collision decoding, representation decoding.

- Bernstein, Lange, P., van Tilborg (2009): asymptotic analysis of ISD for McEliece minimum distances $d \approx n/\log n$.

Practical ISD

Bernstein, Lange, P. (2008):

- use variant of Stern's algorithm

  to extract a plaintext from a ciphertext by decoding $w = 50$ errors in a binary code with $n = 2^{10}$ and $r = 500$.

- Faster by a factor of more than 150 than previous attacks; within reach of a moderate cluster of computers.

Break of original McEliece parameters:

- About 200 (academic) computers involved, with about 300 cores; computation finished in under 90 days; used about 8000 core-days.

Challenges

`http://pqcrypto.org/wild-challenges.html`

- Inspired by `latticechallenge.org` project at TU Darmstadt.
- Want: cryptanalytic benchmarks.
- Build confidence in new setups (e.g., wild McEliece).

How it works:

- Different setups, challenges are indexed by field size and key size.
- Each challenge consists of a public key and a ciphertext.
- Find matching plaintext (or even to find the secret keys).

Gregory Landais at INRIA decrypted ciphertexts for binary Goppa codes with keys up to 28 kB (almost 60-bit security)

Key sizes

Typical key sizes for binary Goppa codes:

- 187*kB* for 128-bit security against ISD

Typical key sizes for *q*-ary Goppa codes:

- 88kB for $\Gamma_{31}(g)$ (small subfield $m = 2$, secure?). (P., PQCrypto 2010).

Typical key sizes for wild Goppa codes:

- 88kB for $\Gamma_{31}(g^{30})$ (extra structural security "incognito") (Bernstein, Lange, P., SAC 2010).

1. Motivation

2. Tutorial: How to hide a linear code

3. Tutorial: Decoding attacks

4. Tutorial: Further targets

Code-based signatures

Bleichenbacher attack against CFS digital signatures

- Replaces classical birthday attacks with cost $2^{r/2}$ by general birthday attacks with four lists and cost $2^{r/3}$.

Sendrier (2011): DOOM (Decoding one out of many)

- improves on Johansson and Jönsson when solving the decoding problem for N instances at once (gaines $\sqrt{N}$).

Beyond Coding Theory

Learning Parity with Noise (LPN) problem: an LPN oracle $\Pi(\mathbf{x}, \epsilon)$ with secret $\mathbf{x}$ in $\mathbb{F}_2^k$ returns a sample

$$(\mathbf{a}, \langle \mathbf{x}, \mathbf{a} \rangle + e)$$

- where $\mathbf{a}$ is chosen uniformly at random from $\mathbb{F}_2^k$ and
- the bit e is chosen with $\text{Prob}(e = 1) = \epsilon$ and $\text{Prob}(e = 0) = 1 - \epsilon$.

Given n samples $(\mathbf{a}, y) = (\mathbf{a}, \langle \mathbf{x}, \mathbf{a} \rangle + e)$ try to recover $\mathbf{x}$ by solving the decoding problem:

$$\mathbf{y} = \mathbf{x}A + \mathbf{e}$$

with $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$, $\mathbf{y} = (y_1, \ldots, y_n)$ and $w = \epsilon n$.

ISD vs BKW

Meurer (Ph.D. thesis 2013) compares asymptotic cost of ISD against Blum–Kalai–Wasserman algorithm to solve LPN:

- BKW asymptotically superior to ISD; comparable for $\epsilon \leq 0.125$.
- For all practical instances $k = 128, \ldots, 1024$ and $\epsilon \leq 0.05$ ISD performs better than BKW; ISD need fewer oracle queries and (thus) less memory.
- Meurer needs 2^{72} bit operations to break the Levieil–Fouque parameters $k = 768$ and $\epsilon = 0.05$ (conjectured to achieve 100-bit security).
- Open problems: combine ISD and BKW.

Conclusion

Structural attacks

- Algebraic codes need subfields. GRS not secure.
- Alternatively, use quasi-cyclic MDPC codes. Don't overdo shrinking key size (keep the key space big enough).

Information-set-decoding

- Many variants; all of them have exponential running time.
- Useful to estimate security levels in code-based (and lattice-based?) cryptography.
- Scripts to estimate parameters:

  `https://bitbucket.org/cbcrypto/isdf2`

  `https://bitbucket.org/cbcrypto/isdfq`

Thank you for your attention!