Applications of Information-set Decoding in Cryptanalysis

Christiane Peters

Technical University of Denmark

MSR Talk Series

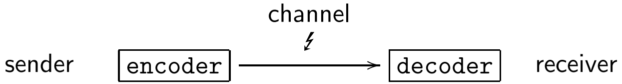
Redmond – March 14, 2013

Outline

Coding Theory

- The sender uses an encoder to transform a message into a codeword by adding redundancy.

- Goal: protect against errors in a noisy channel.



- The receiver uses a decoding algorithm to correct errors which might have occurred during transmission.

Linear encoding

- A message $\mathbf{m} \in \mathbf{F}_2^k$ is encoded into a codeword $\mathbf{x} \in \mathbf{F}_2^n$ which satisfies
$$H\mathbf{x} = \mathbf{0}$$
for an $r \times n$-matrix H where $r = n - k \geq 0$.

Example:

- Let $H = (A \mid I_r)$, then encoding $\mathbf{m} = (m_1, \ldots, m_k)$ into $\mathbf{x} = (x_1, \ldots, x_n)$ simply means setting
$$x_1 = m_1, \ldots, x_k = m_k$$
and then choosing the remaining x_i so that $H\mathbf{x} = \mathbf{0}$.

Error-correcting linear codes

The linear code C with parity-check matrix $H \in \mathbf{F}_2^{r \times n}$ consists of all codewords $\mathbf{x} \in \mathbf{F}_2^n$ such that $H\mathbf{x} = \mathbf{0}$.

Properties:

- The codewords in C form a linear subspace of dimension $n - r$ of $\mathbf{F}_2^n$.

- We say that C has length n and dimension $n - r$.

Example: Hamming code

A parity-check matrix for the $(7, 4, 3)$-Hamming code is given by

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

Example of a codeword: $\mathbf{x} = (1001100)$.

Hamming metric

- The Hamming distance of $\mathbf{x}, \mathbf{y} \in \mathbf{F}_2^n$ is

$$\mathrm{dist}(\mathbf{x}, \mathbf{y}) = \#\{i \in \{1, \dots, n\} \ : x_i \neq y_i\}.$$

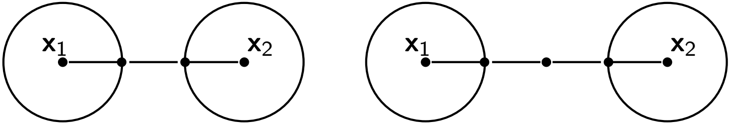
- The Hamming weight of a word $\mathbf{x} \in \mathbf{F}_2^n$ is

$$\mathrm{wt}(\mathbf{x}) = \#\{i \in \{1, \dots, n\} \ : x_i \neq 0\}.$$

- The minimum distance of a linear code C is defined as

$$d(C) = \min_{\substack{\mathbf{x}, \mathbf{y} \in C \\ \mathbf{x} \neq \mathbf{y}}} \mathrm{dist}(\mathbf{x}, \mathbf{y}) = \min_{\substack{\mathbf{x} \in C \\ \mathbf{x} \neq \mathbf{0}}} \mathrm{wt}(\mathbf{x}).$$

Minimum distance



code with $d = 3$ code with $d = 4$

Syndromes

- The syndrome of a vector $\mathbf{y}$ in $\mathbf{F}_2^n$ with respect to H is the vector $H\mathbf{y}$ in $\mathbf{F}_2^r$.

Given $\mathbf{y} = \mathbf{x} + \mathbf{e}$ for $\mathbf{x} \in C$ and $\mathbf{e} \in \mathbf{F}_2^n$. By linearity

$$H\mathbf{y} = H(\mathbf{x} + \mathbf{e}) = H\mathbf{x} + H\mathbf{e} = H\mathbf{e}$$

since $H\mathbf{x} = \mathbf{0}$.

- The space $\mathbf{F}_2^n$ can be partitioned into 2^r cosets $\mathbf{y} + C$.
- A word $\mathbf{e}$ of minimum weight in $\mathbf{y} + C$ is called coset leader.

$$\text{sender} \quad \boxed{\texttt{encoder}} \xrightarrow{\text{channel}} \boxed{\texttt{decoder}} \quad \text{receiver}$$

$$\mathbf{m} \hookrightarrow \mathbf{x} \underset{\text{error } \mathbf{e}}{\rightsquigarrow} \mathbf{x} + \mathbf{e} \longmapsto \mathbf{m}'$$

Decoding problem

Syndrome-decoding problem:

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

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

Decoding needs structure

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, convolutional codes, LDPC codes etc.

All those decoding algorithms use information on the structure of the code.

Generic decoding is hard

However, given a random binary matrix H,

Berlekamp, McEliece, van Tilborg (1978) showed that the general 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 $\approx n/\log n$.

Code-based Cryptography

- McEliece proposed a public-key cryptosystem based on error-correcting codes in 1978.
- Secret key is a linear error-correcting code with an efficient decoding algorithm.
- Public key is a transformation of the secret inner code which is hard to decode.

A code-based cryptosystem

Consider Niederreiter's dual version of McEliece's cryptosystem.

- The public key is an $r \times n$ matrix H and an integer $w \geq 0$.

Encryption of a message $\mathbf{m}$:

1. Use a constant-weight-word encoder to convert message $\mathbf{m}$ into a word $\mathbf{e} \in \mathbf{F}_2^n$ of weight w.
2. Send the ciphertext $\mathbf{s} = H\mathbf{e}$.

Constant-weight-word encoding is a bijection Φ between messages of fixed length and the set of words of length n and weight w.

Secret key

Trapdoor one-way function: the public key H has a hidden Goppa-code structure allowing fast decoding of w errors:

$$H = MH'P$$

where

- H' is the parity-check matrix of a Goppa code Γ of length n and dimension $n - r$ and minimum distance $2w + 1$,
- M is a random $r \times r$ invertible matrix, and
- P is a random $n \times n$ permutation matrix.

The triple (H', M, P) forms the secret key.

Decryption

Decryption of a ciphertext $\mathbf{s} = H\mathbf{e}$ using the secret decomposition $H = MH'P$.

1. Compute $M^{-1}\mathbf{s} = H'P\mathbf{e}$.
2. Use the decoding algorithm for Γ to find the weight-w word $P\mathbf{e}$.
3. Compute $\mathbf{m}$ using $\Phi^{-1}(\mathbf{e})$ after multiplication with P^{-1}.

Conversions

- This is the "text-book" version of code-based crypto.
- Plaintexts are not randomized.
- Use CCA2-secure conversions by Kobara–Imai (PKC 2001) when implementing the systems.

Security assumptions

Key security

- relies on the difficulty of retrieving the secret code from the public code; i.e., decompose H into $MH'P$ to get specifications for a decoding algorithm for H'.

Single-target attacks

- Decryption security relies on hardness of the syndrome-decoding problem assuming that H does not leak information about its structure.

Security level

- A system has b-bit security if it takes at least 2^b bit operations to decrypt a single ciphertext.

1. Basics

2. Code-based Cryptography

3. Information-Set Decoding

4. Implications for Cryptography

Generic 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 Finiasz–Sendrier; 2010 P.;
2011 Bernstein–Lange–P.; 2011 May–Meurer–Thomae;
2012 Becker–Joux–May–Meurer.

ISD in Magma

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

Basic ISD algorithms (until year 1998) are implemented in Magma:

- `DecodingAttack`
- `McEliecesAttack`
- `LeeBrickellsAttack`
- `LeonsAttack`
- `SternsAttack`
- `CanteautChabaudsAttack`

Generic decoder

Build a decoder which gets as input

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

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

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

Problem

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

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

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

Row randomization

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

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

Can arbitrarily permute rows without changing the problem.

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

Row randomization

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

Can arbitrarily permute rows without changing the problem.

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

Column normalization

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

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

Can arbitrarily permute columns without changing the problem.

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

Column normalization

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

Can arbitrarily permute columns without changing the problem.

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

Information-set decoding

$$
\begin{array}{cccccccccc|c}
1 & 0 & 1 & \cdots & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\
0 & 1 & 1 & \cdots & 1 & 0 & 1 & 0 & \cdots & 0 & 1 \\
1 & 1 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 1 \\
 & & & & & & & & & & 0 \\
 & & & & & & & & \ddots & & \vdots \\
1 & 0 & 1 & \cdots & 1 & 0 & 0 & 0 & \cdots & 1 & 0 \\
\mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & & \mathbf{c}_k & \mathbf{c}_{k+1} & & & & \mathbf{c}_n & \mathbf{s} = \mathbf{c}_3 + \mathbf{c}_7 + \mathbf{c}_{28} + \mathbf{c}_{30} -
\end{array}
$$

Can add one row to another $\Rightarrow$ build identity matrix.

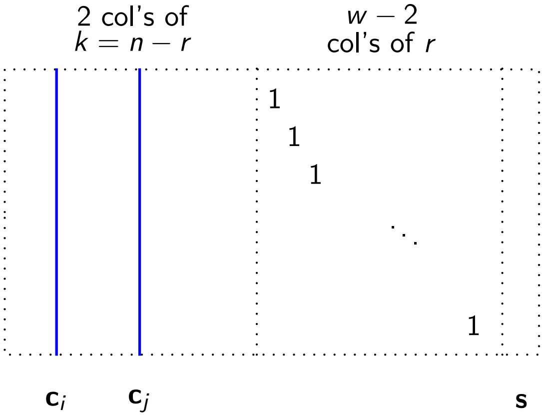
Goal: find w columns which xor $\mathbf{s}$.

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.

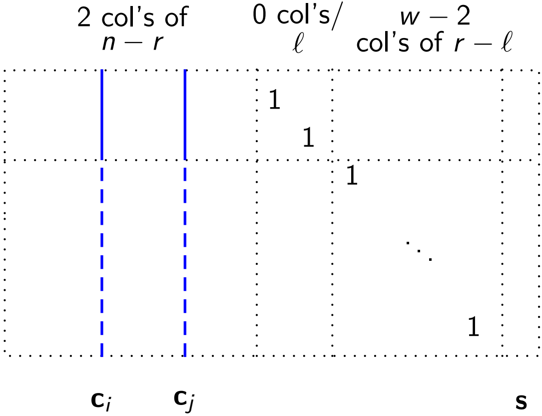
- Check for each pair (i, j) with $1 < i < j \le n - r$ if $\mathbf{s} + \mathbf{c}_i + \mathbf{c}_j$ has weight $w - p$.

- 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

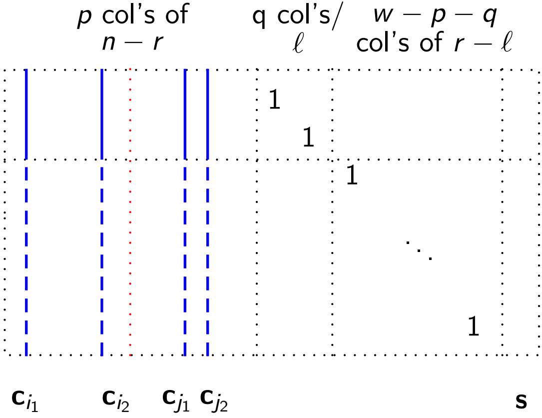
- Split first $n - r$ columns in two disjoint sets of equal size; draw $\mathbf{c}_i$'s from the left, $\mathbf{c}_j$'s from the right set.

- 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

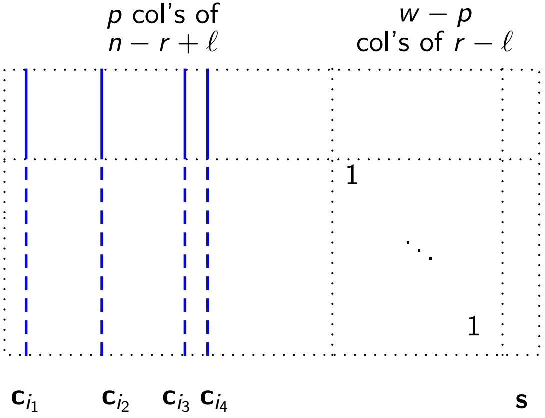
Bernstein, Lange, P. (2011):

- 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



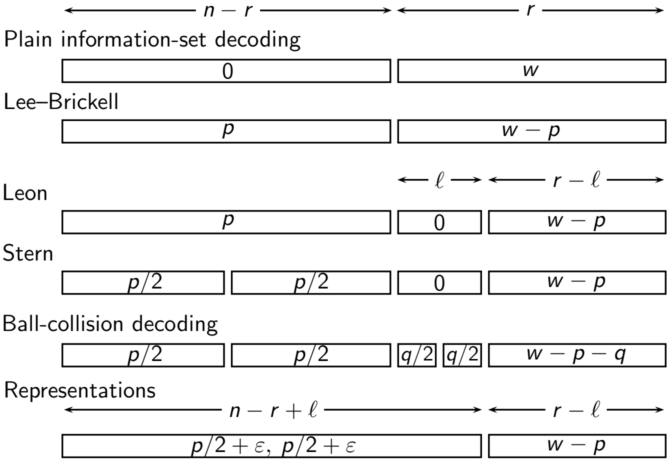
- Only partial Gauss elimination.
- Consider selected sums of p columns out of $n - r + \ell$.

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

2	2	0	46

  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.

Key sizes

- Suggestion: for 128-bit security of the McEliece cryptosystem take a binary Goppa code with $n = 2960$, $r = 672$, and $w = 57$ errors.

- The public-key size here is $187kB$ for 128-bit security against ISD.

Challenges

Go to

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

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

Inspired by `latticechallenge.org` project at TU Darmstadt.

- Want: cryptanalytic benchmarks.
- Build confidence in new setups (e.g., wild McEliece).

Conclusion

- Many variants of information-set-decoding algorithms.
- All of them have exponential running time.
- Useful to estimate security levels in code-based (and lattice-based?) cryptography.
- Simple Pari/GP script and more sophisticated C-code using GMP/MPFR/MPFI to estimate parameters:

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

Thank you for your attention!