Recent Advances in Decoding Random Binary Linear Codes – and Their Implications to Crypto

Alexander May

Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr-University Bochum

KALININGRAD SUMMER SCHOOL
JULY 2019

Linear Codes and Distance

Definition Linear Code

A linear code is a k-dimensional subspace of $\mathbb{F}_2^n$.

Represent via:

- **Generator matrix** G

$$C = \{\mathbf{x}G \in \mathbb{F}_2^n \mid \mathbf{x} \in \mathbb{F}_2^k\}, \text{ where } G \in \mathbb{F}_2^{k \times n}$$

- **Parity check matrix** H

$$C = \{\mathbf{c} \in \mathbb{F}_2^n \mid H\mathbf{c} = \mathbf{0}\}, \text{ where } H \in \mathbb{F}_2^{n-k \times n}$$

- **Random Code:** $G \in_R \mathbb{F}_2^{k \times n}$ respectively $H \in_R \mathbb{F}_2^{n-k \times n}$
  - Random codes are hard instances for decoding.
  - Crypto motivation: Scramble structured C in “random" SCT.
  - Good generic hardness criterion.

Bounded and Full Distance Decoding

Definition Distance

$d = \min_{\mathbf{c} \neq \mathbf{c}' \in C}\{\Delta(\mathbf{c}, \mathbf{c}')\}$, where Δ is the Hamming distance.

Remark: Unique decoding of $\mathbf{c} + \mathbf{e}$ when $\Delta(\mathbf{e}) \leq \frac{d-1}{2}$.

Definition Bounded Distance Decoding (BD)

Given : $H, \mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in C, \Delta(\mathbf{e}) \leq \frac{d-1}{2}$
Find : $\mathbf{e}$ and thus $\mathbf{c} = \mathbf{x} + \mathbf{e}$

Syndrome Decoding

- Syndrome $\mathbf{s} := H\mathbf{x} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}$.
- Bounded Distance is the usual case in crypto.

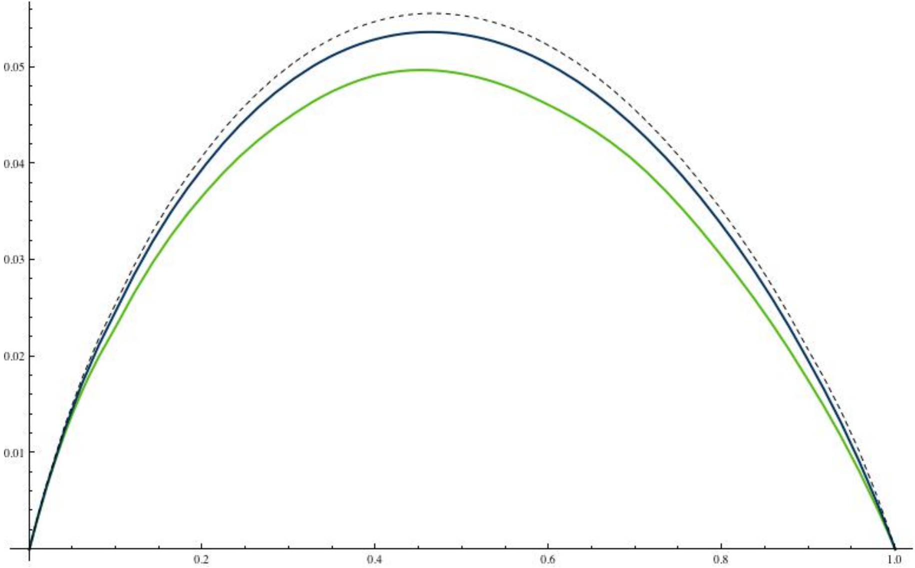
Definition Full Distance Decoding (FD)

Given : $H, \mathbf{x} \in \mathbb{F}_2^n$
Find : $\mathbf{c}$ with $\Delta(\mathbf{c}, \mathbf{x}) \leq d$

On Running Times

- Running time of any decoding algorithm is a function of (n, k, d).
- Look at map $\mathbb{F}_2^n \to \mathbb{F}_2^{n-k}$ with $\mathbf{e} \mapsto H\mathbf{e}$ with $\Delta(\mathbf{e}) \leq d$.
- Map cannot be injective unless $\binom{n}{d} < 2^{n-k}$.
- Write $\binom{n}{d} \approx 2^{H(\frac{d}{n})n}$, which yields
$$H(\tfrac{d}{n}) < 1 - \tfrac{k}{n}. \quad \text{(Gilbert-Varshamov bound)}$$
- For **random codes** this bound is sharp.
- Hence, we can directly link d to n, k.
- Running time becomes a function of n, k only.
- Since BD/FD decoding is NP-hard we expect running time
$$T(n, k) = 2^{f(\frac{k}{n})n}.$$
- For simplifying, we are mainly interested in $T(n) = \max_k\{T(n, k)\}$.

Running Time graphically



The Way to go

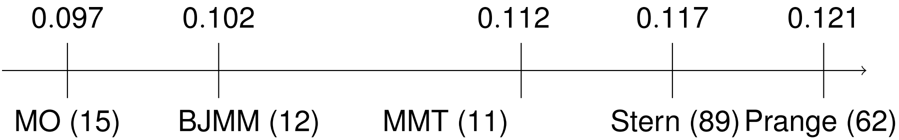


Figure: Full Distance decoding (FD)

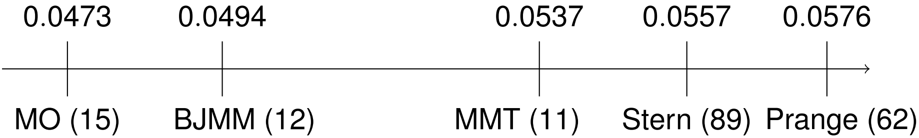


Figure: Bounded Distance decoding (BD)

Let's just start.

Goal: Solve $H\mathbf{e} = \mathbf{s}$ for small weight $\mathbf{e}$.
Assumption: Wlog we know $\omega := \Delta(\mathbf{e})$.

Algorithm Exhaustive Search

INPUT: H, $\mathbf{x}$, ω

1. For all $\mathbf{e} \in \mathbb{F}_2^n$ with $\Delta(\mathbf{e}) = \omega$: Check whether $H\mathbf{e} = \mathbf{s} = H\mathbf{x}$.

OUTPUT: $\mathbf{e}$

Running time: $T(n) = \binom{n}{\omega} \leq 2^{0.386n}$.

Allowed Transformations

Linear algebra transformation for $H\mathbf{e} = \mathbf{s}$.

1. **Column permutation**:
$$H\mathbf{e} = HPP^{-1}\mathbf{e} = \mathbf{s}$$
for some permutation matrix $P \in \mathbb{F}_2^{n\times n}$.
2. **Elementary row operations**:
$$GH\mathbf{e} = G\mathbf{s} =: \mathbf{s}'$$
for some invertible matrix $G \in \mathbb{F}_2^{n-k\times n-k}$.

Easy special cases:

1. **Quadratic case**: $H \in \mathbb{F}_2^{n\times n}$. Compute $\mathbf{e} = H^{-1}\mathbf{s}$.
2. **Any weight** $\Delta(\mathbf{e})$: Compute $GH\mathbf{e} = (H' \mid I_{n-k})\mathbf{e} = G\mathbf{s}$.

Remark: Hardness/unicity comes from under-defined + small weight.

Prange's algorithm (1962)

Idea: $(H' \mid I_{n-k})(\mathbf{e}_1 \mid\mid \mathbf{e}_2) = H'\mathbf{e}_1 + \mathbf{e}_2 = \mathbf{s}'$

Algorithm Prange

INPUT: H, $\mathbf{x}$, ω

REPEAT

1. Permute columns, construct systematic $(H' \mid I_{n-k})$. Fix $p < \omega$.
2. For all $\mathbf{e}_1 \in \mathbb{F}_2^k$ with $\Delta(\mathbf{e}_1) = p$:
    1. If $(\Delta(H'\mathbf{e}_1 + \mathbf{s}') = \omega - p)$, success.

UNTIL success

OUTPUT: Undo permutation of $\mathbf{e} = (\mathbf{e}_1 || H'\mathbf{e}_1 + \mathbf{s}')$.

Running time:

- Outer loop has success prob $\frac{\binom{k}{p}\binom{n-k}{\omega-p}}{\binom{n}{\omega}}$.
- Inner loop has running time $\binom{k}{p}$. Total: $\frac{\binom{n}{\omega}}{\binom{n-k}{\omega-p}}$, optimal for $p = 0$.
- Yields running time $T(n) = 2^{\frac{1}{17}n}$, with constant memory.

Stern's algorithm (1989)

Meet in the Middle:

$$(H_1 \mid H_2 \mid I_{n-k})(\mathbf{e}_1||\mathbf{e}_2||\mathbf{e}_3) = H_1\mathbf{e}_1 + H_2\mathbf{e}_2 + \mathbf{e}_3 = \mathbf{s}'$$

Algorithm Stern

INPUT: H, $\mathbf{x}$, ω

REPEAT

1. Permute columns, construct systematic $(H_1 \mid H_2 \mid I_{n-k})$. Fix $p < \omega$.
2. For all $\mathbf{e}_1 \in \mathbb{F}_2^{\frac{k}{2}}$ with $\Delta(\mathbf{e}_1) = \frac{p}{2}$: Store $H_1\mathbf{e}_1$ in sorted L_1.
3. For all $\mathbf{e}_2 \in \mathbb{F}_2^{\frac{k}{2}}$ with $\Delta(\mathbf{e}_2) = \frac{p}{2}$: Store $H_2\mathbf{e}_2 + \mathbf{s}'$ in sorted L_2.
4. Search for elements in L_1, L_2 that differ by $\Delta(\mathbf{e}_3) = \omega - p$.

UNTIL success

OUTPUT: Undo permutation of $\mathbf{e} = (\mathbf{e}_1||\mathbf{e}_2||H_1\mathbf{e}_1 + H_2\mathbf{e}_2 + \mathbf{s}')$.

- Step 4: Look for vectors that completely match in ℓ coordinates.
- $T(n) = 2^{\frac{1}{18}}$, but requires memory to store L_1, L_2.

Representation Technique (Howgrave-Graham, Joux)

Meet in the Middle

- Split $\mathbf{e} = (\mathbf{e}_1 || \mathbf{e}_2)$ as $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_2^{\frac{k}{2}}$ with weight $\Delta(\mathbf{e}_i) = \frac{p}{2}$ each.
- Combination of $\mathbf{e}_1, \mathbf{e}_2$ is via concenation.
- Unique representation of $\mathbf{e}$ in terms of $\mathbf{e}_1, \mathbf{e}_2$.

Representation [May, Meurer, Thomae 2011]

- Split $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ as $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_2^k$ with weight $\Delta(\mathbf{e}_i) = \frac{p}{2}$ each.
- Combination of $\mathbf{e}_1, \mathbf{e}_2$ is via addition in $\mathbb{F}_2^k$.
- $\mathbf{e}$ has many representations as $\mathbf{e}_1 + \mathbf{e}_2$.

Example for $k = 8, p = 4$:

$$\begin{aligned}
(01101001) &= (01100000) + (00001001)\\
&= (01001000) + (00100001)\\
&= (01000001) + (00101000)\\
&= (00101000) + (01000001)\\
&= (00100001) + (01001000)\\
&= (00001001) + (01100000)
\end{aligned}$$

Pros and Cons of representations

Representation [MMT 2011, Asiacrypt 2011]

- Split $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ as $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_2^k$ with weight $\Delta(\mathbf{e}_i) = \frac{p}{2}$ each.
- **Disadvantages:**
  - List lengths of L_1, L_2 increases from $\binom{k/2}{p/2}$ to $\binom{k}{p/2}$.
  - Addition of $\mathbf{e}_1, \mathbf{e}_2$ usually yields Hamming weight smaller p.
- **Advantage:**
  - $\mathbf{e}$ has $\binom{p}{p/2} =: R$ representations as $\mathbf{e}_1 + \mathbf{e}_2$.
- Construct via Divide & Conquer only $\frac{1}{R}$-fraction of L_1, L_2.
- Since many solutions exist, it is easier to construct a special one.
- **Example:** Look only for $H_1\mathbf{e}_1, H_2\mathbf{e}_2 + \mathbf{s}'$ with last $\log(\frac{1}{R})$ coord. 0.
- Advantage (may) dominate whenever

$$\frac{\binom{k}{p/2}}{\binom{p}{p/2}} < \binom{k/2}{p/2}.$$

Result: Yields running time $2^{\frac{1}{19}n}$.

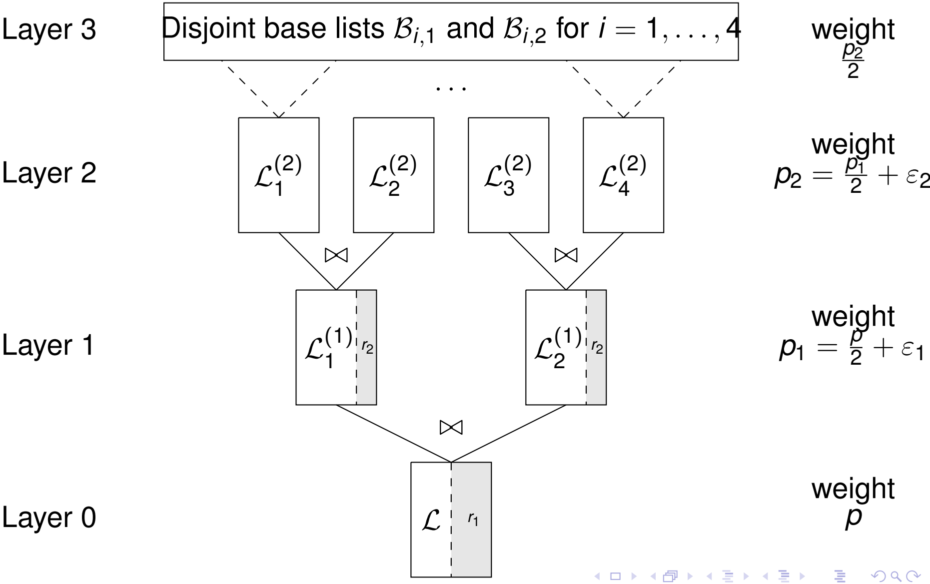
More representations (Becker,Joux,May,Meurer 2012)

Idea:

- Choose $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_2^k$ with weight $\Delta(\mathbf{e}_i) = \frac{p}{2} + \epsilon$ each.
- Choose ϵ such that ϵ 1-positions cancel on expectation.
- In MMT: $\binom{p}{p/2}$ representations of 1's as
$$1 = 1 + 0 = 0 + 1.$$
- Now: Additionally $\binom{k-p}{\epsilon}$ representations of 0's as
$$0 = 1 + 1 = 0 + 0.$$
- Paper subtitle:

  "How $1 + 1 = 0$ Improves Information Set Decoding".
- Yields $T(n) = 2^{\frac{1}{20}n}$.

How to construct special solutions



A word about memory

	Bounded Distance		Full Distance	
	time	space	time	space
Prange	0.05752	-	0.1208	-
Stern	0.05564	0.0135	0.1167	0.0318
Ball-collision	0.05559	0.0148	0.1164	0.0374
MMT	0.05364	0.0216	0.1116	0.0541
BJMM	0.04934	0.0286	0.1019	0.0769

Stern's algorithm (1989)

Meet in the Middle:

$$(H_1 \mid H_2 \mid I_{n-k})(\mathbf{e}_1||\mathbf{e}_2||\mathbf{e}_3) = H_1\mathbf{e}_1 + H_2\mathbf{e}_2 + \mathbf{e}_3 = \mathbf{s}'$$

Algorithm Stern

INPUT: H, $\mathbf{x}$, ω

REPEAT

1. Permute columns, construct systematic $(H_1 \mid H_2 \mid I_{n-k})$. Fix $p < \omega$.
2. For all $\mathbf{e}_1 \in \mathbb{F}_2^{\frac{k}{2}}$ with $\Delta(\mathbf{e}_1) = \frac{p}{2}$: Store $H_1\mathbf{e}_1$ in sorted L_1.
3. For all $\mathbf{e}_2 \in \mathbb{F}_2^{\frac{k}{2}}$ with $\Delta(\mathbf{e}_2) = \frac{p}{2}$: Store $H_2\mathbf{e}_2 + \mathbf{s}'$ in sorted L_2.
4. Search for elements in L_1, L_2 that differ by $\Delta(\mathbf{e}_3) = \omega - p$.

UNTIL success

OUTPUT: Undo permutation of $\mathbf{e} = (\mathbf{e}_1||\mathbf{e}_2||H_1\mathbf{e}_1 + H_2\mathbf{e}_2 + \mathbf{s}')$.

- Step 4: Look for vectors that completely match in ℓ coordinates.
- $T(n) = 2^{\frac{1}{18}}$, but requires memory to store L_1, L_2.

Nearest Neighbor Problem

Definition Nearest Neighbor Problem

Given : $L_1, L_2 \subset_R \mathbb{F}_2^n$ with $|L_i| = 2^{\lambda n}$
Find : all $(\mathbf{u}, \mathbf{v}) \in L_1 \times L_2$ with $\Delta(\mathbf{u}, \mathbf{v}) = \gamma n$.

Easy cases:

1. $\gamma = \frac{1}{2}$
   - Test every combination in $L_1 \times L_2$.
   - Run time $2^{2\lambda n(1+o(1))}$.
2. $\gamma = 0$
   - Sort lists and find matching pairs.
   - Run time $2^{\lambda n(1+o(1))}$.

Theorem May, Ozerov 2015

Nearest Neighbor can be solved in $2^{\frac{1}{1-\gamma}\lambda n(1+o(1))}$.

Main Idea of Nearest Neighbor

Observation: Nearest Neighbors are also **locally** near.

$\mathbf{u} \in \mathbf{L}_1$
$\mathbf{v} \in \mathbf{L}_2$

$\boxed{\mathbf{L}_1}\boxed{\mathbf{L}_2}$ ← size: $2^{\lambda n}$

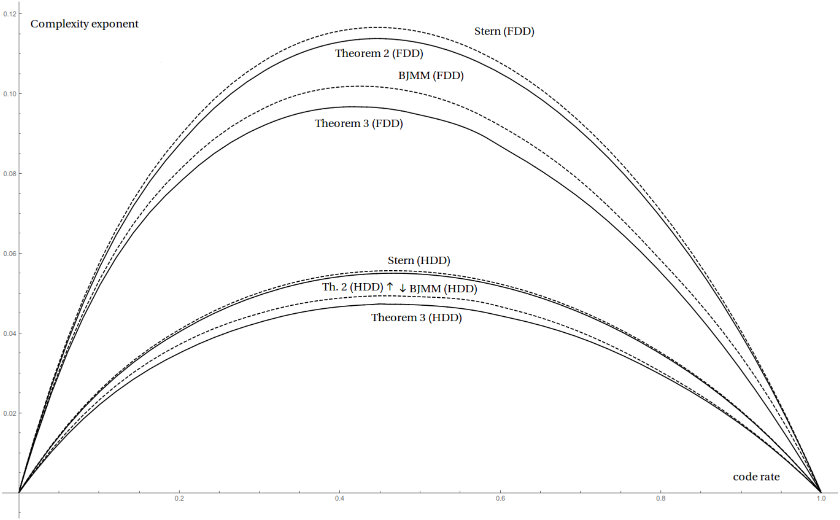
create exponentially many sublists
by choosing random partitions P

$\boxed{\mathbf{L}_1'}\boxed{\mathbf{L}_2'}$ $\boxed{\mathbf{L}_1'}\boxed{\mathbf{L}_2'}$ $\cdots$ $\boxed{\mathbf{L}_1'}\boxed{\mathbf{L}_2'}$ $\boxed{\mathbf{L}_1'}\boxed{\mathbf{L}_2'}$

For at least one sublist pair we have $(\mathbf{u}, \mathbf{v}) \in \mathbf{L}_1' \times \mathbf{L}_2'$ w.o.p.

- Filters out until L_1', L_2' reach polynomial size.
- Algorithm has quite large polynomial overheads.
- Yields $T(n) < 2^{\frac{1}{21}n}$ for Bounded Distance Decoding.

Improvements graphically



Asymptotical or Real?

Yann Hamdaoui and Nicolas Sendrier,
“A Non Asymptotic Analysis of Information Set Decoding", 2013

(n, k, d)	Stern	MMT	BJMM
(1024, 524, 50)	55.60	54.75	52.90
(2048, 1696, 32)	81.60	79.50	76.82

Conclusion

MMT, BJMM relevant for cryptographic keysizes! Breakpoint for MO?

But: The improvements asymptotically vanish for McEliece.

The LPN Problem and its Relation to Codes

Problem Learning Parities with Noise ($\text{LPN}_{n,p}$)

Given: $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i) \in \mathbb{F}_2^n \times \mathbb{F}_2$ with $\Pr[e_i = 1] = p$.
Find: $\mathbf{s} \in \mathbb{F}_2^n$

- Notation: $A\mathbf{s} = \mathbf{b} + \mathbf{e}$. For $p = 0$: Compute $\mathbf{s} = A^{-1}\mathbf{b}$.
- Best algorithm: BKW with time/sample/space $2^{\frac{n}{\log(\frac{n}{p})}}$.

Algorithm Gauss

1. REPEAT
   1. Take n fresh samples. Compute $\mathbf{s}' = A^{-1}\mathbf{b}$.
2. UNTIL $\mathbf{s}' = \mathbf{s}$

Theorem

Gauss runs in time/sample complexity $\left(\frac{1}{1-p}\right)^n$ and poly space.

Proof: $\Pr[\text{Iteration of REPEAT successful}] = (1-p)^n$.

Getting the samples down.

Algorithm POOLED GAUSS (Esser, Kübler, May – Crypto 2017)

1. Choose a pool of $\Theta(n^2)$ samples.
2. REPEAT
   1. Take n samples from the pool. Compute $\mathbf{s}' = A^{-1}\mathbf{b}$.
3. UNTIL $\mathbf{s}' = \mathbf{s}$

Theorem

POOLED GAUSS runs in time $\left(\frac{1}{1-p}\right)^n$ with poly samples/space.

Theorem

POOLED GAUSS **quantumly** runs in $\left(\frac{1}{1-p}\right)^{\frac{n}{2}}$ with poly samples/space.

Corollary

Let $p(n) \to 0$. Then POOLED GAUSS runs in e^{pn}.

Decoding LPN with Preprocessing

Algorithm LPN with Preprocessing

INPUT: $\mathrm{LPN}_{n,p}$ instance

1. **Modify**: Use many samples to produce pool of dim-reduced ones. Results in $\mathrm{LPN}_{n',p'}$ instance with $n' < n$ and $p' \geq p$, e.g. use BKW.
2. **Decode**: Use decoding to solve $\mathrm{LPN}_{n',p'}$, e.g. POOLED GAUSS.
3. **Complete**: Recover rest of **s**, e.g. via enumeration or iterating.

Yields HYBRID algorithm that optimally uses space.

- For polynomial space: Put all efforts in **Decode**.
- For arbitrary space: Put all efforts in **Modify**.

Bit Complexity Estimates for Memory $\leq 2^{60}$

Largest RAM today: IBM 20-Petaflops with 1.6PB $< 2^{54}$ bits.

Table: `Well-Pooled MMT` or **`Hybrid`**

p	n							
	256	384	448	512	576	640	768	1280
$\frac{1}{\sqrt{n}}$	43	49	51	52	54	56	59	70
0.05	39	48	52	56	60	64	72	108
0.125	**58**	81	91	102	113	123	144	230
0.25	**65**	**124**	153	172	192	211	250	406
0.4	**85**	**153**	**186**	**219**	**251**	**286**	**357**	584

Bit Complexity Estimates for Memory $\leq 2^{60}$

Table: `Quantum Hybrid`

p	n							
	256	384	448	512	576	640	768	1280
$\frac{1}{\sqrt{n}}$	33	37	39	40	42	43	46	54
0.05	30	37	40	42	45	48	53	73
0.125	56	57	63	69	75	81	93	140
0.25	**63**	89	101	112	123	135	158	248
0.4	**76**	**121**	**144**	163	181	198	234	373

NIST Security Levels

Table: $p = \frac{1}{8}$

n	Classic	Quantum
670	**128**	84
1060	**192**	120
1410	**256**	152
460	93	**64**
630	121	**80**
1150	208	**128**

Table: $p = \frac{1}{4}$

n	Classic	Quantum
370	**128**	80
580	**192**	124
790	**256**	161
260	97	**64**
370	128	**80**
610	203	**128**

Experiments

Table: Solved instances

Algorithm	n	p	Pool	BKW	Decode	**Total**
WP MMT	243	0.125	6.73 d	-	8.34 d	**15.07 d**
WP MMT	135	0.25	5.65 d	-	8.19 d	**13.84 d**
HYBRID	135	0.25	2.21 d	1.72 h	3.41 d	**5.69 d**

Conclusions and Questions

- Improvement for BD
$$2^{\frac{1}{17}n} \rightarrow 2^{\frac{1}{18}n} \rightarrow 2^{\frac{1}{19}n} \rightarrow 2^{\frac{1}{20}n} \rightarrow 2^{\frac{1}{21}n}.$$
- Extensions to codes over $\mathbb{F}_q$ possible, but less effective.
- More applications of representations, nearest neighbors?
- May threaten McEliece security. Implementations?
- LPN with $n = 512, p = \frac{1}{4}$ or even $p = \frac{1}{8}$ seems (practically) secure.
- Generalization of LPN to LWE decoding only good for small error.
- Cryptanalysis: **Real implementations + extrapolation.**
- There is a need for small memory algorithms.