Ternary representation for large distance decoding in $\mathbb{F}_3$

Valerian HATEY and Kévin CARRIER

CY Cergy Paris Université
ENSEA (École Nationale Supérieure de l'Électronique et de ses Applications)

19 octobre 2023

The Wave signature

1. Digital signature submitted to new NIST competition **[Debris,Sendrier,Tillich,Asiacrypt2019]**
2. Code based signature in $\mathbb{F}_3$

Wave security

Problem (Syndrom decoding problem)

Given

- $\boldsymbol{H} \in \mathbb{F}_3^{(n-k)\times n}$ *following a uniform distribution*
- $\mathbf{s} \in \mathbb{F}_3^{n-k}$ *following a uniform distribution*
- $t \in [\![0, n]\!]$

Find $\mathbf{e} \in \mathbb{F}_3^n$ *such that* $\boldsymbol{H}\mathbf{e}^T = \mathbf{s}$ *and* $\Delta(\mathbf{e}) = t$.

Security (see **[Sendrier,PQCrypto2023]**)

- Large weight decoding $\Rightarrow$ **Forgery Attack**
- Small weight decoding $\Rightarrow$ **Key Recovery Attack**

Best large/small weight decoding algorithms

ISD (Information Set Decoding introduced by **[Prange,1962]**).

Usual representation technique in Hamming weight

Definition (Representation in Hamming weight)

Let $\mathbf{z} \in \mathbb{F}_q^n$ *such that* $\Delta(\mathbf{z}) = t$.
A (u, v)-*representation of* $\mathbf{z}$ *is a pair* $(\mathbf{x}, \mathbf{y})$ *such that*

- $\Delta(\mathbf{x}) = u$
- $\Delta(\mathbf{y}) = v$
- $\mathbf{z} = \mathbf{x} + \mathbf{y}$

$$\underset{\mathbf{z}}{\boxed{\quad t \quad}} = \underset{\mathbf{x}}{\boxed{\quad u \quad}} + \underset{\mathbf{y}}{\boxed{\quad v \quad}}$$

A more precise operator than the Hamming weight

In $\mathbb{F}_3$, the Hamming weight gives us :

- number of 0
- The sum of the number of 1 and 2

We can be more precise

Definition : Symbol counter operator $\blacktriangle(.)$

Let $\mathbf{x} \in \mathbb{F}_q^n$, we define

$$\blacktriangle(\mathbf{x}) := (t_0, \cdots, t_{q-1}) \in [\![0, n]\!]^q$$

With $t_i := Card(\{j \in [\![1, n]\!] \ : \ \mathbf{x}(j) = i\})$

Idea to improve the state of the art of decoding in $\mathbb{F}_3$ with $\blacktriangle(.)$

$\blacktriangle(.)$ tool idea for large weight decodings

- **Small weight** : small quantity of 1 and 2 $\Rightarrow$ negligible precision gain
- **Large weight** : large quantity of 1 and 2 $\Rightarrow$ interesting precision gain

Idea to improve the state of the art of large weight decoding in $\mathbb{F}_3$

In $\mathbb{F}_2$, $\blacktriangle(.) \Leftrightarrow \Delta(.)$
$\Rightarrow$ Adapting the best current ISDs in $\mathbb{F}_2$ in Hamming weight **[Both,May,2018]** to $\blacktriangle(.)$ in $\mathbb{F}_3$

Adaptation of the concept of representation to $\blacktriangle(.)$

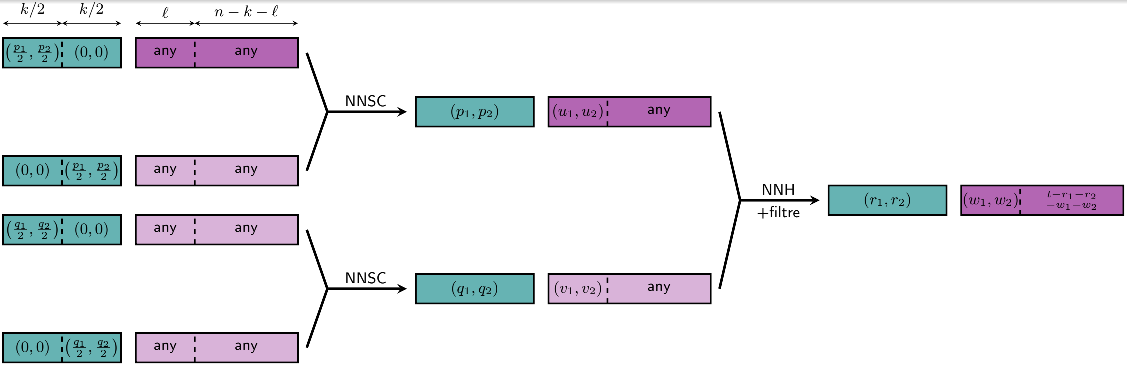
Definition (Representation with $\blacktriangle(.)$)

Let $\mathbf{z} \in \mathbb{F}_q^n$ *such that* $\blacktriangle(\mathbf{z}) = (t_0, t_1, t_2)$.
A (u_1, u_2, v_1, v_2)*-representation of* $\mathbf{z}$ *is a pair* $(\mathbf{x}, \mathbf{y})$ *such that*

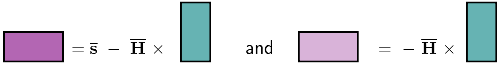
- $\blacktriangle(\mathbf{x}) = (u_0, u_1, u_2)$
- $\blacktriangle(\mathbf{y}) = (v_0, v_1, v_2)$
- $\mathbf{z} = \mathbf{x} + \mathbf{y}$

$$\underbrace{\boxed{(t_0, t_1, t_2)}}_{\mathbf{z}} = \underbrace{\boxed{(u_0, u_1, u_2)}}_{\mathbf{x}} + \underbrace{\boxed{(v_0, v_1, v_2)}}_{\mathbf{y}}$$

Two stages [Both,May,2018] with $\blacktriangle(.)$



Legend :



where $\overline{\mathbf{H}}|\mathbf{Id} = \mathbf{SHP}$ and $\overline{\mathbf{s}} = \mathbf{S}\mathbf{s}^\top$

Adaptation of the definition

What we need ?

1. Adapt the **Nearest Neighbor** search to $\blacktriangle(.)$
2. Count the **number of representations**

NNH Problem (Nearest Neighbor with Hamming weight)

Let two lists $\mathcal{L}_1$ and $\mathcal{L}_2$ of random elements of $\mathbb{F}_3^n$, and $t \in \mathbb{N}$, find :

$$\{(\mathbf{x}, \mathbf{y}) \in \mathcal{L}_1 \times \mathcal{L}_2 \text{ such that } \Delta(\mathbf{x} + \mathbf{y}) = t\}$$

NNSC Problem (Nearest Neighbor with Symbol Counter $\blacktriangle(.)$)

Let two lists $\mathcal{L}_1$ and $\mathcal{L}_2$ of random elements in $\mathbb{F}_3^n$, and $t_0, t_1, t_2 \in \mathbb{N}$ such that $t_0 + t_1 + t_2 = n$, find :

$$\{(\mathbf{x}, \mathbf{y}) \in \mathcal{L}_1 \times \mathcal{L}_2 \text{ such that } \blacktriangle(\mathbf{x} + \mathbf{y}) = (t_0, t_1, t_2)\}$$

Solving NNSC with Hamming weight

Let (t_0, t_1, t_2) be fixed.
Three ways to get back to Hamming's weight problem :

1. $\mathcal{L}_1 \times \mathcal{L}_2 \rightarrow (t_0, t_1, t_2) \rightarrow$ **Hamming weight** $t_1 + t_2$
2. $\mathcal{L}_1 \times (\mathcal{L}_2 + \mathbf{1}) \rightarrow (t_2, t_0, t_1) \rightarrow$ **Hamming weight** $t_0 + t_1$
3. $\mathcal{L}_1 \times (\mathcal{L}_2 + \mathbf{2}) \rightarrow (t_1, t_2, t_0) \rightarrow$ **Hamming weight** $t_0 + t_2$

The algorithm

At (t_0, t_1, t_2) fixed :

- Solve the **Nearest Neighbor in Hamming weight** which is the most efficient among the 3 previous ones using **[Carrier,2020]** optimal method.
- Filter the solutions $(\mathbf{x}, \mathbf{y}) \in \mathcal{L}_1 \times \mathcal{L}_2$ such that $\blacktriangle(\mathbf{x} + \mathbf{y}) = (t_0, t_1, t_2)$

Solving NNSC with $\blacktriangle(.)$

Adaptation of the Nearest Neighbor algorithm with random codes of **[Carrier,2020]** to $\blacktriangle(.)$

Principle

- We use hash functions that are the decoder of a random code $\mathcal{C}$.
- We use a hash table T indexed by $\mathcal{C}$.

1. $\forall \mathbf{x} \in \mathcal{L}_1$, $\forall \mathbf{c} \in \mathcal{C}$ such that $\blacktriangle(\mathbf{x}+\mathbf{c}) = (u_0, u_1, u_2)$, append $\mathbf{x}$ to $T[c]$.
2. $\forall \mathbf{y} \in \mathcal{L}_2$, $\forall \mathbf{c} \in \mathcal{C}$, such that $\blacktriangle(\mathbf{y}-\mathbf{c}) = (v_0, v_1, v_2)$, test the candidate (pairs hashed to the same value).
3. Iterate with new code $\mathcal{C}$

About $u_i's$ and $v_i's$

Parameters that are chosen to optimize complexity

Algorithm complexity

Complexity

$$C = \min_{u_1,u_2,v_1,v_2} \left(\frac{C_{iter}}{P_{succ}}\right)$$

- C_{iter} **the complexity of an iteration** can be computed efficiently
- P_{succ} **the success probability** can be computed efficiently using Gröebner basis techniques

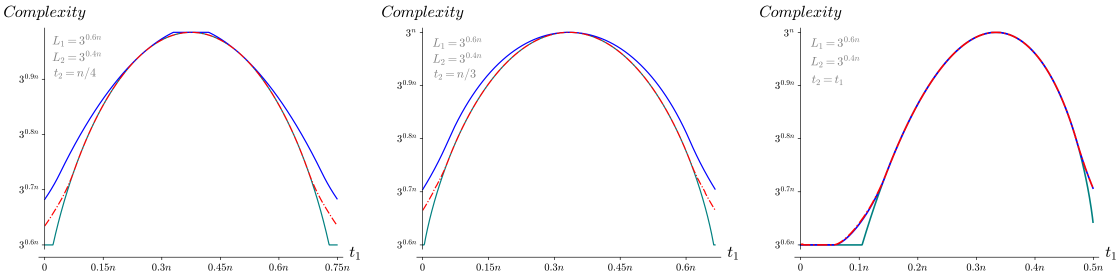
Calculation

We can compute efficiently the number of representation of a fixed vector (same technique with Gröebner basis)

One problem

To get C, we must optimize the parameters $(u_i, v_i) \Rightarrow$ numerical optimization $\Rightarrow$ expensive
(We start to better understand these parameters).

Comparison of complexities



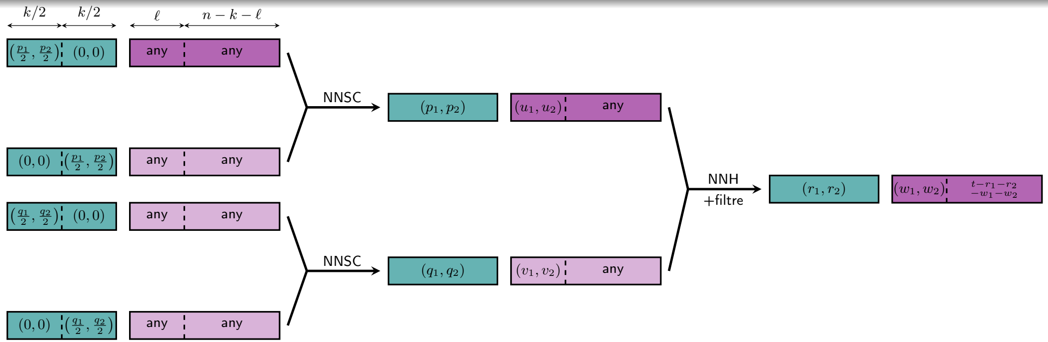
Red curve : complexity for solving NNSC problem using $\blacktriangle(.)$
Blue curve : complexity for solving NNSC problem using Hamming weight
Green curve : complexity lower bound

Result

Comparison of the complexity of several algorithms for ternary decoding with high weight $t = 0.98n$ and rate $k = n/2$.

Algorithm	Complexity
[Prange,1962]	0.12072
[Dumer,1991]	0.09091
[Stern,1988],[May,Ozerov,2015]	0.08491
[Bricout,Chailloux,Debris,Lequesne,2019]	$\mathbf{0.06535}$
This work	0.07239

Increase the depth of [Both,May,2018]



- **[Both,May,2018]** uses 4 stages. **[Bricout,Chailloux,Debris,Lequesne,2019]** uses 7 stages. **Our work** 2 stages.
- The number of parameters to optimize explode.
- We are better understanding the intern parameters of our Nearest Neighbor by getting inspiration from **[Carrier,2020]**.