Wave: A new family of trapdoor preimage sampleable functions

Thomas Debris-Alazard, Nicolas Sendrier and Jean-Pierre Tillich

Information Security Group, Royal Holloway,
University of London, UK

September 18, 2019
London-ish Lattice Meeting

Results

- The first code-based "hash-and-sign" that follows the GPV strategy (Trapdoor Preimage Sampleable functions) ;
- Security reduction to two problems (NP-complete) of coding theory:
  - Generic decoding of a linear code;
  - Distinguish between random codes and generalized $(U, U+V)$-codes.
- Key Size ≈3MB, signature size ≈13Kb, signing time ≈ 0.1s (non-optimized);
- Nice feature: uniform signatures through an efficient rejection sampling, one rejection every ≈ 100 signatures.

Digital signature scheme

$\boxed{\text{Unsecure channel}}$

$$\text{Alice: } \mathbf{m} \xrightarrow{\quad\downarrow\quad} \mathbf{m}' \text{: Bob}$$

Alice wants to ensure Bob that:

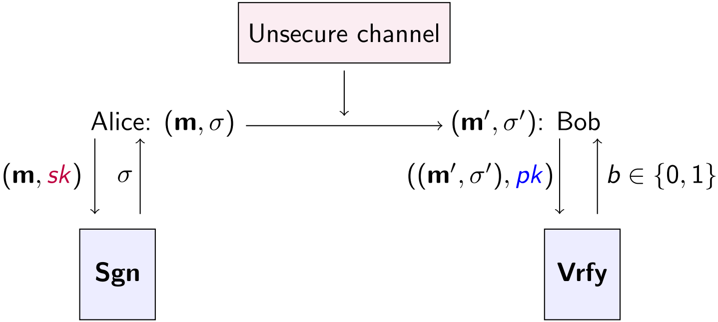
- $\mathbf{m}$ has not been corrupted ($\mathbf{m} = \mathbf{m}'$).
- $\mathbf{m}$ comes from Alice

$\rightarrow$ Idea: add a *signature* to $\mathbf{m}$

Digital signature scheme

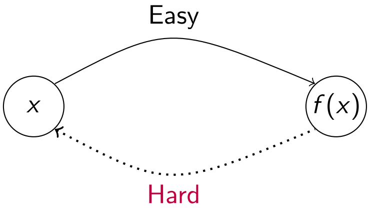
Alice first makes the following operations:

- Generation of (pk, sk).
- Send pk to *everyone*.



Full Domain Hash Signature

- f be a trapdoor one-way function



Easy with trap.

- To sign $\mathbf{m}$ one computes $\mathbf{y} = \mathcal{H}(\mathbf{m})$ (hash) and $\sigma \in f^{-1}(\mathbf{y})$.
  $\rightarrow$ It is required to invert f on all vectors (full domain).
- Verification $f(\sigma) = \mathcal{H}(\mathbf{m})$?

… with Bijective Trapdoors OW?

- Let f be a bijective trapdoor one-way function
- To sign $\mathbf{m}$, compute $\sigma = f^{-1}(\mathcal{H}(\mathbf{m}))$ ($\mathcal{H}$ hash function)

$\mathcal{H}(\mathbf{m})$ is uniform (ROM) $\Rightarrow \sigma$ is uniform too!

(no leakage)

Signature schemes DSA, RSA meet this nice feature

Hard condition to meet in code/lattice-based cryptography...

Gentry-Peikert-Vaikuntanathan (GPV) Approach

It is based on trapdoor one-way preimage sampleable function!

A family of trapdoor one way-functions $(f_a)_a$ and a distribution $\mathcal{D}$ such that

- $f_a(x)$ is uniformly distributed when $x \xleftarrow{\$} \mathcal{D}$,
- algorithm computing $x \leftarrow f_a^{-1}(y)$ with the trapdoor is distributed according to $\mathcal{D}$

Gentry-Peikert-Vaikuntanathan (GPV) Approach

It is based on trapdoor one-way preimage sampleable function!

A family of trapdoor one way-functions $(f_a)_a$ and a distribution $\mathcal{D}$ such that

- $f_a(x)$ is uniformly distributed when $x \xleftarrow{\$} \mathcal{D}$,
- algorithm computing $x \leftarrow f_a^{-1}(y)$ with the trapdoor is distributed according to $\mathcal{D}$

$$\mathcal{D} = \begin{cases} \textit{uniform over words of fixed Hamming weight} \text{ in our case} \\ \textit{gaussian} \text{ for lattices} \end{cases}$$

Trapdoor One-way of Wave

Our one-way will be ($|\cdot|$ Hamming weight)

$$f_{\mathbf{H}}: \begin{array}{ccc} \{\mathbf{e} \in \mathbb{F}_q^n : |\mathbf{e}| = w\} & \longrightarrow & \mathbb{F}_q^{n-k} \\ \mathbf{e} & \longmapsto & \mathbf{H}\mathbf{e}^{\intercal} \end{array}$$

Inverting $f_{\mathbf{H}}$ amounts to solve the following problem:

Problem (Syndrome Decoding with fixed weight)

Given $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$, *and an integer* w, *find* $\mathbf{e} \in \mathbb{F}_q^n$ *such that* $\mathbf{H}\mathbf{e}^{\intercal} = \mathbf{s}^{\intercal}$ *and* $|\mathbf{e}| = w$.

$\rightarrow$ Generic problem upon which all code-based cryptography relies

$\rightarrow$ Putting a trapdoor on $f_{\mathbf{H}}$ consists in putting a structure on $\mathbf{H}$!

Public-Key: $\mathbf{H}_{\text{pk}}$

Signature of $\mathcal{H}(\mathbf{m})$: $\mathbf{e}$ of weight w with $\mathbf{H}_{\text{pk}}\mathbf{e}^{\intercal} = \mathcal{H}(\mathbf{m})$.

Codes: Basic Definition

A code $\mathcal{C}$ is a subspace of $\mathbb{F}_q^n$

When $\mathcal{C}$ is of dimension k it is defined by a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of full-rank as:

$$\mathcal{C} \stackrel{\triangle}{=} \{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{H}\mathbf{c}^{\mathsf{T}} = \mathbf{0}\}$$

The Trapdoor(I)

$\mathbf{H}_{\text{pk}}$ parity-check matrix of a permuted generalized $(U, U+V)$ code:

- A permutation $\mathbf{P}$,
- Two codes U and V of length $n/2$,
- Four vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ over $\mathbb{F}_q^{n/2}$ such that

$$a_i d_i - b_i c_i \neq 0 \quad \text{and} \quad a_i c_i \neq 0$$

$$(\mathbf{a} \odot U + \mathbf{b} \odot V, \mathbf{c} \odot U + \mathbf{d} \odot V)\mathbf{P} \stackrel{\triangle}{=} \{(\mathbf{a} \odot \mathbf{u} + \mathbf{b} \odot \mathbf{v}, \mathbf{c} \odot \mathbf{u} + \mathbf{d} \odot \mathbf{v})\mathbf{P} : \mathbf{u} \in U, \mathbf{v} \in V\}$$

with

$$\mathbf{x} \odot \mathbf{y} \stackrel{\triangle}{=} (x_1 y_1, x_2 y_2, \cdots, x_{n/2} y_{n/2})$$

The Trapdoor(II)

Example of generalized $(U, U+V)$-code:

- $(U, U+V) \stackrel{\triangle}{=} \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- $(U+V, U-V) \stackrel{\triangle}{=} \{(\mathbf{u}+\mathbf{v}, \mathbf{u}-\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- ...
- More generally, for all $\mathbf{u} = (u_1, \cdots, u_{n/2}) \in U$ and $\mathbf{v} = (v_1, \cdots, v_{n/2}) \in V$:

$$\Big(\underbrace{u_1, u_2 \overbrace{+ v_2, \cdots \quad , u_{n/2} + v_{n/2}; \; u_1 + v_1, u_2 -}^{+n/2 \text{ symbols}} v_2, \cdots}_{} \quad , v_{n/2} - u_{n/2}\Big)$$

$$\xleftrightarrow{\qquad\qquad n/2 \qquad\qquad}$$

The Trapdoor(II)

Example of generalized $(U, U+V)$-code:

- $(U, U+V) \stackrel{\triangle}{=} \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- $(U+V, U-V) \stackrel{\triangle}{=} \{(\mathbf{u}+\mathbf{v}, \mathbf{u}-\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- ...
- More generally, for all $\mathbf{u} = (u_1, \cdots, u_{n/2}) \in U$ and $\mathbf{v} = (v_1, \cdots, v_{n/2}) \in V$:

$$\Big(\underbrace{u_1, u_2 \overbrace{+ v_2, \cdots, u_{n/2} + v_{n/2}; u_1 + v_1, u_2 -}^{+n/2 \text{ symbols}} v_2, \cdots, v_{n/2} - u_{n/2}}_{\text{first half: } n/2}\Big)$$

Proposition

Decide if a code is a permuted generalized $(U, U+V)$-code or not is NP-complete.

Security Reduction

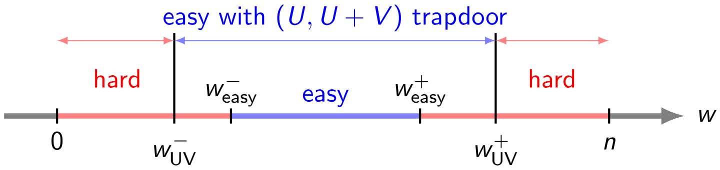
We reduce the security to two problems:

- Distinguishing between a permuted generalized $(U, U+V)$ code and a random code;
- Hardness of finding $\mathbf{e}$ of weight w s.t: $\mathbf{H}\mathbf{e}^{\intercal} = \mathbf{s}^{\intercal}$ (Syndrome Decoding).

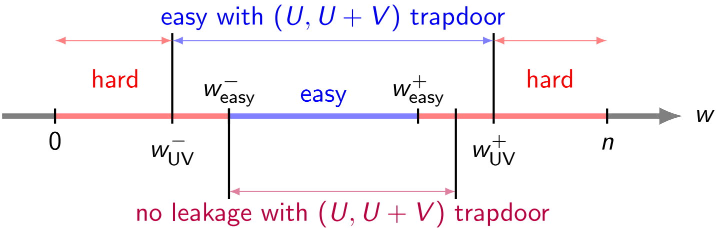
(both are NP-complete)

Hardness of Decoding

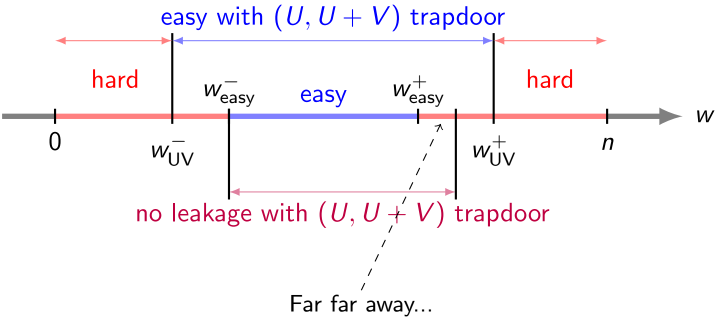
Hardness of Decoding



Hardness of Decoding



Hardness of Decoding



Prange Step

Given: $\mathbf{H}$ random of size $(n-k) \times n$, rank $n-k$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$;

Find: $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$.

Prange Step

Given: $\mathbf{H}$ random of size $(n-k) \times n$, rank $n-k$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$;

Find: $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{H}\mathbf{e}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}$.

Choose $n-k$ columns and split $\mathbf{H}$ and $\mathbf{e}$ as :

$$\mathbf{H} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix} \quad \text{and} \quad \mathbf{e} = (\mathbf{e}', \mathbf{e}'')$$

where $\mathbf{B} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ is non-singular and $\mathbf{e}'' \in \mathbb{F}_q^{n-k}$

$$\mathbf{H}\mathbf{e}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}} \iff \mathbf{A}\mathbf{e}'^{\mathsf{T}} + \mathbf{B}\mathbf{e}''^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}$$

$$\mathbf{e}'' = \mathbf{B}^{-1}\left(\mathbf{s}^{\mathsf{T}} - \mathbf{A}\mathbf{e}'^{\mathsf{T}}\right)$$

- $\mathbf{e}' \in \mathbb{F}_q^k$ free to choose,
- $\mathbf{e}'' \in \mathbb{F}_q^{n-k}$ uniformly distributed as $\mathbf{s}$ is uniform

Prange Step

Given: $\mathbf{H}$ random of size $(n-k) \times n$, rank $n-k$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$;

Find: $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$.

Prange Step

Given: $\mathbf{H}$ random of size $(n-k) \times n$, rank $n-k$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$;

Find: $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{He}^\intercal = \mathbf{s}^\intercal$.

$\mathbf{e}'$	$\mathbf{e}''$
k bits (to choose)	$n-k$ bits (function of $\mathbf{e}'$)

- $\mathbf{e}''$ follows a uniform law over $\mathbb{F}_q^{n-k}$, therefore $\forall \varepsilon > 0, \exists \alpha > 0$:

$$\mathbb{E}(|\mathbf{e}''|) = \frac{q-1}{q}(n-k) \quad ; \quad \mathbb{P}\left(\left||\mathbf{e}''| - \frac{q-1}{q}(n-k)\right| \geq \varepsilon n\right) = e^{-\alpha n}$$

Prange Step

Given: $\mathbf{H}$ random of size $(n-k) \times n$, rank $n-k$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$;

Find: $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{He}^\intercal = \mathbf{s}^\intercal$.

$\mathbf{e}'$	$\mathbf{e}''$
k bits (to choose)	$n-k$ bits (function of $\mathbf{e}'$)

- $\mathbf{e}''$ follows a uniform law over $\mathbb{F}_q^{n-k}$, therefore $\forall \varepsilon > 0, \exists \alpha > 0$:

$$\mathbb{E}(|\mathbf{e}''|) = \frac{q-1}{q}(n-k) \quad ; \quad \mathbb{P}\left(\left| |\mathbf{e}''| - \frac{q-1}{q}(n-k) \right| \geq \varepsilon n \right) = e^{-\alpha n}$$

- We get an error $\mathbf{e} = (\mathbf{e}', \mathbf{e}'')$ such that for some $\beta > 0$:

$$\mathbb{E}(|\mathbf{e}|) = \mathbb{E}(|\mathbf{e}'|) + \frac{q-1}{q}(n-k)$$

$$\mathbb{P}\left(|\mathbf{e}| \geq (1+\varepsilon)\left(\mathbb{E}(|\mathbf{e}'|) + \frac{q-1}{q}(n-k) \right) \right) = e^{-\beta n}$$

Prange Algorithm

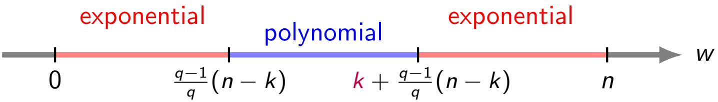
To reach an error of weight w:

repeat Prange Step until getting an error of weight w.

$\mathbf{e}'$	$\mathbf{e}''$
k bits (to choose)	$n-k$ bits (function of $\mathbf{e}'$)

- $\mathbf{e}''$ follows a uniform law over $\mathbb{F}_q^{n-k}$
- Choice over $\mathbf{e}'$.

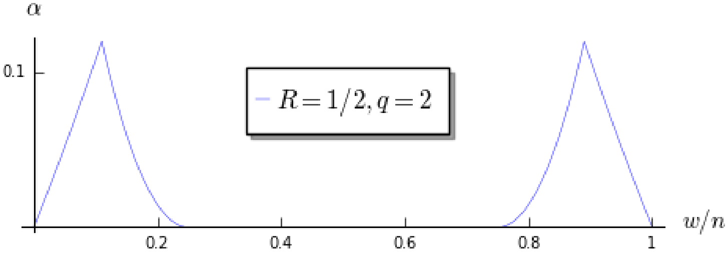
Figure: Complexity (number of calls) to reach some weight w



Exponent of the Prange Algorithm for $q = 2$

Complexity: $2^{\alpha n}$ where α function of w/n.

Figure: Exponent vs Relative Weight

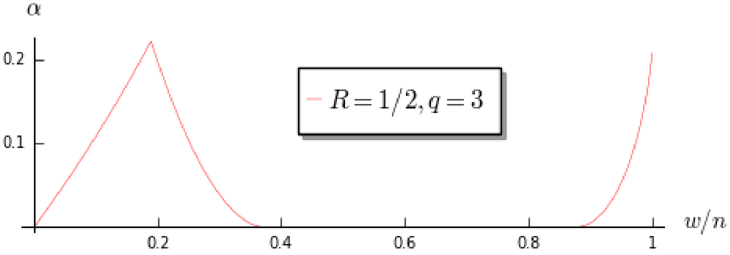


$$R = \frac{\text{dimension of the code}}{\text{length of the code}}$$

Exponent of the Prange Algorithm for $q = 3$

Complexity: $2^{\alpha n}$ where α function of w/n.

Figure: Exponent vs Relative Weight



$$R = \frac{\text{dimension of the code}}{\text{length of the code}}$$

Generic Decoding Algorithms

Coding theory has never come up with a polynomial algorithm outside the range $[\![\frac{q-1}{q}(n-k), k+\frac{q-1}{q}(n-k)]\!]$

Modern algorithms have decreased the exponent of Prange in the exponential areas of complexity

But not changed the range of polynomial complexity!

$\rightarrow$ Where is the worse case?

Worse Case for Generic Decoding Algorithm

When $w = \Theta(n)$, complexity is given by:

$$2^{c \cdot n(1+o(1))}$$

where c depends of k, w and q.

Key Size:

$$n \times R \times (1-R) \text{ where } c \times n = 128 \text{ and } R \stackrel{\triangle}{=} k/n$$

$$\longrightarrow \text{Goal} : \min_{k,w,q} \{n \times R \times (1-R) : n = 128/c\}$$

Worse Case for Generic Decoding Algorithm

When $w = \Theta(n)$, complexity is given by:

$$2^{c \cdot n(1+o(1))}$$

where c depends of k, w and q.

Key Size:

$$n \times R \times (1-R) \text{ where } c \times n = 128 \text{ and } R \stackrel{\triangle}{=} k/n$$

$$\longrightarrow \text{Goal : } \min_{k,w,q} \{ n \times R \times (1-R) : n = 128/c \}$$

- Usually: $q = 2$ and w equals to Gilbert-Varshamov bound (small weight),
- Recent work [BCDL19]: choose $q = 3$ and large weight.

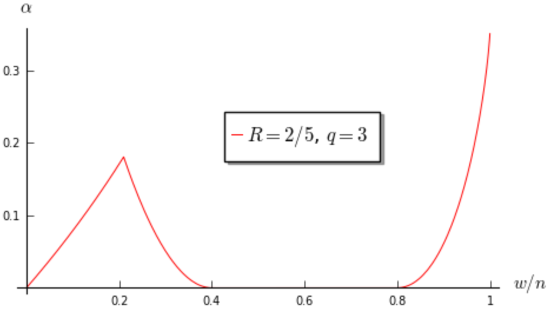
Minimum input sizes (in kbits) for a time complexity of 2^{128}

Algorithm	$q = 2$	$q = 3$ and $w/n > 1/2$
Prange	275	44
Dumer/Wagner	295	83
BJMM/Our algorithm	374	99

Exponent of the Prange Algorithm for $q = 3$

Complexity: $2^{\alpha n}$ where α function of w/n.

Figure: Exponent vs Relative Weight

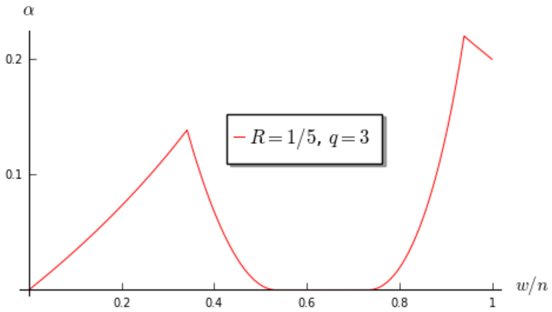


$$R = \frac{\text{dimension of the code}}{\text{length of the code}}$$

Exponent of the Prange Algorithm for $q = 3$

Complexity: $2^{\alpha n}$ where α function of w/n.

Figure: Exponent vs Relative Weight



$$R = \frac{\text{dimension of the code}}{\text{length of the code}}$$

Our trapdoor

Our trapdoor consists in generalized $(U, U+V)$-codes.

Example:

- $(U, U+V) \triangleq \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- $(U+V, U-V) \triangleq \{(\mathbf{u}+\mathbf{v}, \mathbf{u}-\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- More generally, for all $\mathbf{u} = (u_1, \cdots, u_{n/2}) \in U$ and $\mathbf{v} = (v_1, \cdots, v_{n/2}) \in V$:

$$\Big(\underbrace{u_1, u_2 + v_2, \cdots \quad, u_{n/2} + v_{n/2}}_{n/2}; \overbrace{u_1 + v_1, u_2 - v_2}^{+n/2 \text{ bits}}, \cdots \quad, v_{n/2} - u_{n/2}\Big)$$

Our trapdoor

Our trapdoor consists in generalized $(U, U+V)$-codes.

Example:

- $(U, U+V) \stackrel{\triangle}{=} \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- $(U+V, U-V) \stackrel{\triangle}{=} \{(\mathbf{u}+\mathbf{v}, \mathbf{u}-\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}$;
- More generally, for all $\mathbf{u} = (u_1, \cdots, u_{n/2}) \in U$ and $\mathbf{v} = (v_1, \cdots, v_{n/2}) \in V$:

$$\underbrace{\left(u_1, \overbrace{u_2 + v_2, \cdots, u_{n/2} + v_{n/2}; u_1 + v_1, u_2 -}^{+n/2 \text{ bits}} v_2, \cdots, v_{n/2} - u_{n/2}\right)}_{n/2}$$

We will restrict in this talk our study to the case of:

$(U, U+V) -$ codes ; $q = 3$ with $\mathbb{F}_3 = \{-1, 0, 1\}$

$(U, U+V)$-decoder (I)

U (resp. V) random code of dimension k_U (resp. k_V) defined by $\mathbf{H}_U$ (resp. $\mathbf{H}_V$).

$\rightarrow$ The $(U, U+V)$-code is defined by:

$$\mathbf{H} \triangleq \begin{pmatrix} \mathbf{H}_U & \mathbf{0} \\ -\mathbf{H}_V & \mathbf{H}_V \end{pmatrix}$$

Let,

$$\mathbf{e} = (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V) \quad ; \quad \mathbf{s} = (\mathbf{s}_U, \mathbf{s}_V)$$

$$\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal \iff \begin{cases} \mathbf{H}_U \mathbf{e}_U^\intercal = \mathbf{s}_U^\intercal \\ \mathbf{H}_V \mathbf{e}_V^\intercal = \mathbf{s}_V^\intercal \end{cases}$$

$\rightarrow$ No gain when decoding independently with the Prange algorithm...

$(U, U+V)$-decoder (II)

We look for $\mathbf{e} = (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$ such that:

$$\mathbf{H}_U \mathbf{e}_U^\intercal = \mathbf{s}_U^\intercal \quad ; \quad \mathbf{H}_V \mathbf{e}_V^\intercal = \mathbf{s}_V^\intercal$$

$\rightarrow$ We use the Prange algorithm!

Polar code strategy:

(i) firstly to decode in V to get $\mathbf{e}_V$;
(ii) then to decode in U to get $\mathbf{e}_U$ using the knowledge of $\mathbf{e}_V$

We have the freedom to choose:

- k_V (dimension of V) symbols of $\mathbf{e}_V$;
- k_U (dimension of U) symbols of $\mathbf{e}_U$.

$(U, U+V)$-decoder (III)

We get a final error $\mathbf{e} = (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V) \in \mathbb{F}_3^n$ of shape:

$$\mathbf{e}_V = \begin{array}{|c|c|}\hline \mathbf{0} & \mathbf{e}''_V \\ \hline\end{array}$$

$$\mathbf{e} = \begin{array}{|c|c|c|c|}\hline \mathbf{e}'_U & \mathbf{e}''_U & \mathbf{e}'_U & \mathbf{e}''_U + \mathbf{e}''_V \\ \hline\end{array}$$

To reach an error of minimum weight:

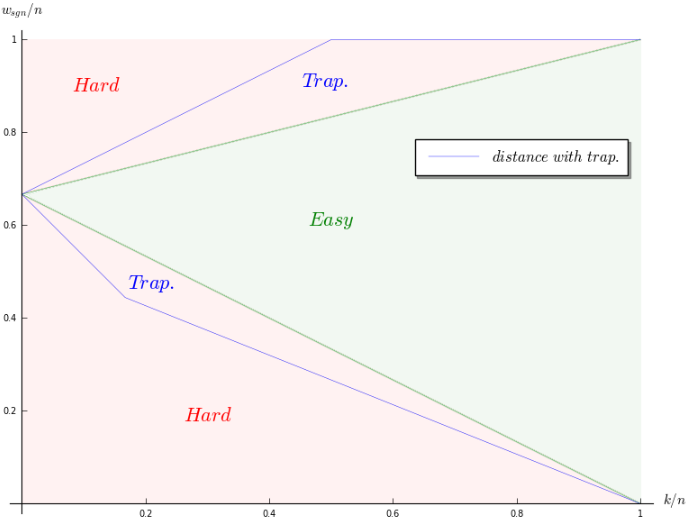
- Put as many 0's as possible in $\mathbf{e}'_U(i)$ (they are doubled in $\mathbf{e}$).

To reach an error of maximum weight

- Choose k_U symbols $\mathbf{e}_U(i)$ such that: $\begin{cases} \mathbf{e}_U(i) \neq 0 \\ \mathbf{e}_U(i) + \mathbf{e}_V(i) \neq 0 \end{cases}$

$\rightarrow$ Possible as $q = 3$ and do not depend of $\mathbf{e}_V(i)$!

Relative Distances of Signature



Achieving the Uniform Distribution(I)

Let,

$$\mathbf{e}^{\text{sgn}} \triangleq (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V) \quad (\text{resp. } \mathbf{e}^{\text{unif}} \triangleq (\mathbf{e}_1, \mathbf{e}_2))$$

be a signature (resp. be a uniform word of weight w).

We would like,

$$\mathbf{e}^{\text{sgn}} \sim \mathbf{e}^{\text{unif}}$$

We remark,

$$\begin{cases} \mathbf{e}_U \sim \mathbf{e}_1 \\ \mathbf{e}_V \sim \mathbf{e}_2 - \mathbf{e}_1 \end{cases}$$

But here,

$$\mathbf{e}_V = \text{Prange}\left(\mathbf{H}_V, \mathbf{s}_V\right)$$

In a first approximation we would like:

$$\mathbb{E}\left(|\mathbf{e}_V|\right) = \mathbb{E}\left(|\mathbf{e}_2 - \mathbf{e}_1|\right)$$

$\rightarrow$ How to adjust $\mathbb{E}\left(|\mathbf{e}_V|\right)$ with the Prange algorithm?

Achieving the Uniform Distribution(II)

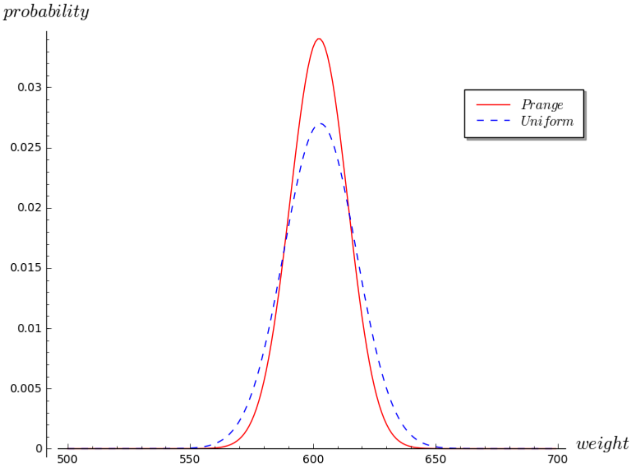
- We look for $\mathbb{E}(|\mathbf{e}_V|) = \mathbb{E}(|\mathbf{e}_2 - \mathbf{e}_1|)$ where $\mathbf{e}^{\text{unif}} \triangleq (\mathbf{e}_1, \mathbf{e}_2)$

$$\mathbf{e}_V = \underbrace{\boxed{\quad \mathbf{e}'_V \quad}}_{k_V \text{ bits}}\underbrace{\boxed{\quad \mathbf{e}''_V \quad}}_{n/2 - k_V \text{ bits}}$$

- $\mathbf{e}''_V$ follows a uniform law over $\mathbb{F}_3^{n/2-k}$: $\mathbb{E}(|\mathbf{e}''_V|) = \frac{2}{3}(n/2 - k_V)$
- $\mathbf{e}'_V$ such that: $\mathbb{E}\left(|\mathbf{e}'_V|\right) = (1-\alpha)k_V$ with a fixed α.

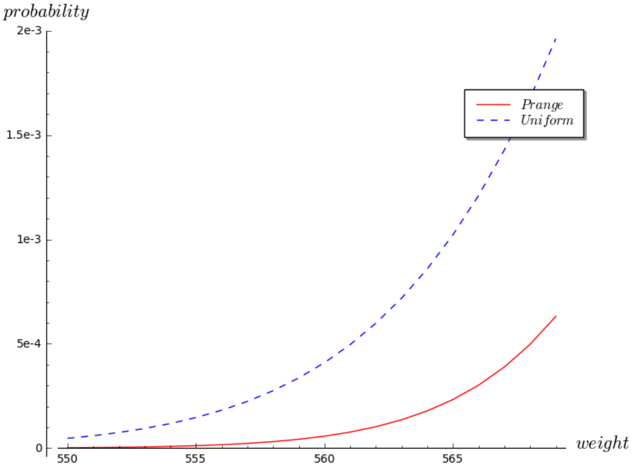
$\rightarrow$ Choose k_V such that: $(1-\alpha)k_V + \frac{2}{3}(n/2 - k_V) = \mathbb{E}\left(|\mathbf{e}_2 - \mathbf{e}_1|\right)$

Prange vs Uniform Distribution for V



$$\mathbb{P}(\text{accept}) = \min_{j} \frac{\mathbb{P}(|\mathbf{e}_V| = i)}{\mathbb{P}(|\mathbf{e}_2 - \mathbf{e}_1| = j)}$$

Prange vs Uniform Distribution for V



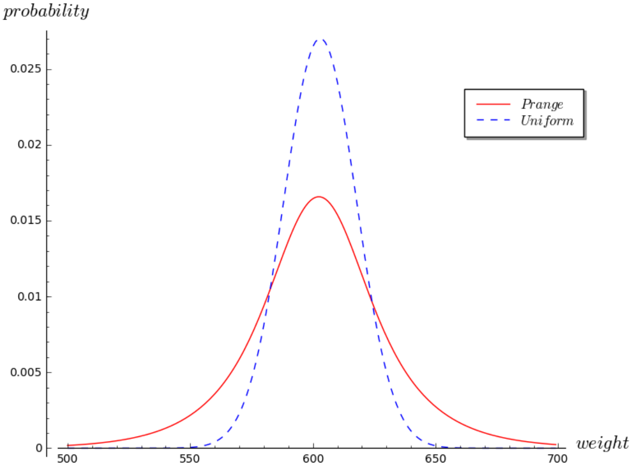
$$\mathbb{P}(\text{accept}) = \min_j \frac{\mathbb{P}(|\mathbf{e}_V| = i)}{\mathbb{P}(|\mathbf{e}_2 - \mathbf{e}_1| = j)}$$

Achieving the Uniform Distribution(III)

$$\mathbf{e}_V = \underbrace{\boxed{\quad \mathbf{e}'_V \quad}}_{k_V \text{ bits}}\underbrace{\boxed{\quad \mathbf{e}''_V \quad}}_{n/2 - k_V \text{ bits}}$$

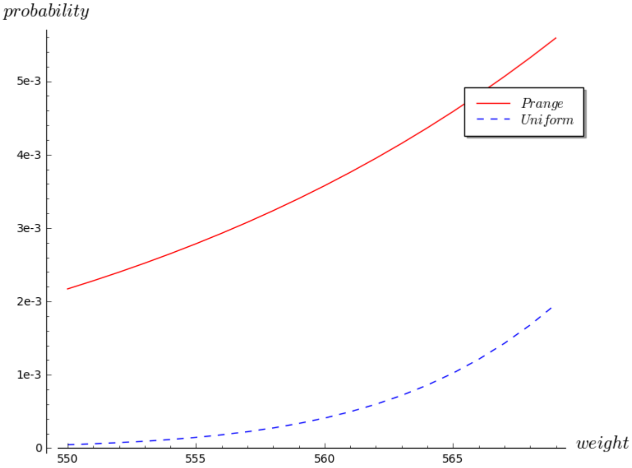
- $\mathbf{e}''_V$ follows a uniform law: its variance is fixed
- Choose $\mathbf{e}'_V$ such that: $\mathbb{E}\left(|\mathbf{e}'_V|\right) = (1-\alpha)k_V$ and high variance!

Prange vs Uniform Distribution for V



Now we can sometimes reject some outputs of the Prange algorithm!

Prange vs Uniform Distribution for V



Now we can sometimes reject some outputs of the Prange algorithm!

Achieving the Uniform Distribution(IV)

By making a rejection sampling on $|\mathbf{e}_V|$:

$$\text{“accept } |\mathbf{e}_V| = i\text{” with probability: } \frac{1}{M}\frac{\mathbb{P}(|\mathbf{e}_2 - \mathbf{e}_1| = i)}{\mathbb{P}(|\mathbf{e}_V| = i)}$$

with

$$M \triangleq \max_j \frac{\mathbb{P}(|\mathbf{e}_2 - \mathbf{e}_1| = j)}{\mathbb{P}(|\mathbf{e}_V| = j)}$$

$$\rightarrow \text{ This ensures } |\mathbf{e}_V| \sim |\mathbf{e}_1 - \mathbf{e}_2| \tag{1}$$

Distribution of the Prange algorithm is only function of the weight:

$$\mathbb{P}(\text{Prange}(\cdot) = \mathbf{e} \mid |\text{Prange}(\cdot)| = |\mathbf{e}|) = \frac{1}{\#\{\mathbf{x} : |\mathbf{x}| = |\mathbf{e}|\}}$$

$$\rightarrow \text{Combined with (1) it gives: } \mathbf{e}_V \sim \mathbf{e}_2 - \mathbf{e}_1$$

Achieving the Uniform Distribution(V)

To end, rejection sampling on $|\mathbf{e}_U|$ which gives:

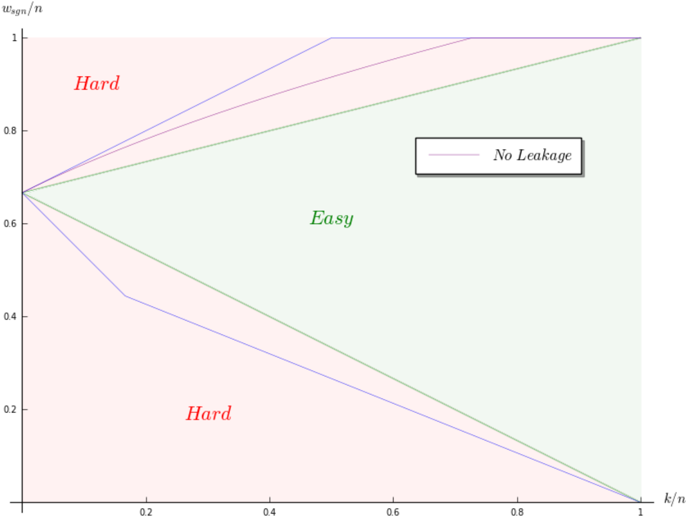
Distribution of signatures = Uniform over words of weight w

$\rightarrow$ Impossible attack with the knowledge of signatures!

With our parameter:

$$\mathbb{P}(\text{a reject}) \approx 0.01$$

Relative Distance with No Leakage



Security Model: a Strong One

Any adversary can have access to:

- q_{sign} signatures $(\mathbf{m}, \sigma)$ of its choice;
- q_{hash} hash results $\mathcal{H}(\mathbf{m})$.

$\rightarrow$ His goal: produce one signature he did not request!

The Decoding Problem

Problem (DOOM – Decoding One Out of Many)

Instance : $\mathbf{H}$; $\mathbf{s}_1, \cdots, \mathbf{s}_N$; w
Output : $(\mathbf{e}, i)$ *with* $|\mathbf{e}| = w$ *such that* $\mathbf{H}\mathbf{e}^{\intercal} = \mathbf{s}_i^{\intercal}$

Computational success in time t of breaking DOOM:

$$Succ_{DOOM}^{N}(t) = \max_{|\mathcal{A}| \leq t} \left\{ Succ_{DOOM}^{N}(\mathcal{A}) \right\}$$

where $Succ_{DOOM}^{N}(\mathcal{A})$ is the probability for $\mathcal{A}$ to break DOOM.

Security Reduction

- $\rho\left(\mathcal{D}_0, \mathcal{D}_1\right)$: statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$;
- $\rho_c\left(\mathcal{D}_0, \mathcal{D}_1\right)(t) = \max\limits_{|\mathcal{A}| \leq t} \left\{\mathbb{P}\left(\mathcal{A}(\mathcal{D}_0) = 0\right) - \mathbb{P}\left(\mathcal{A}(\mathcal{D}_1) = 0\right)\right\}$

Theorem (Security Reduction)

When $\mathcal{H}$ is a random function, we have for all time t:

$$\text{Security}^{\text{Wave}}(t, q_{\text{hash}}, q_{\text{sign}}) \leq 2Succ_{\mathrm{DOOM}}^{q_{\text{hash}}}(t_c) + \rho_c\left(\textbf{Random Code}, \textbf{Permuted Gen. } (U, U+V)\textbf{-code}\right)(t_c) + q_{\text{sign}}\,\rho\left(\textbf{Signature}, \textbf{Uniform}_w\right) + \frac{1}{2} q_{\text{hash}} \sqrt{\rho\left(\mathbf{H}_{\text{pk}}\mathbf{e}_w^{\mathsf{T}}, \mathbf{s}_{\text{unif}}^{\mathsf{T}}\right)}$$

where $t_c = t + O\left(q_{\text{hash}} \cdot n^2\right)$.

- $\sqrt{\rho\left(\mathbf{H}_{\text{pk}}\mathbf{e}_w^{\mathsf{T}}, \mathbf{s}_{\text{unif}}^{\mathsf{T}}\right)} = \text{negligible}()$ (left-over hash lemma)
- $\rho\left(\textbf{Signature}, \textbf{Uniform}_w\right) = 0$ (rejection sampling)

Conclusion

- The first code-based "hash-and-sign" based on NP-complete problems that strictly follows the GPV strategy;

Ongoing Work:

- We generalized decoding algorithms in $\mathbb{F}_3$ for high weights;
- Best algorithms to distinguish $(U, U+V)$-codes and random codes: decoding algorithms;
- Hope to remove the rejection sampling

  $\rightarrow$ Many degrees of freedom in the Prange algorithm!

Conclusion

- The first code-based "hash-and-sign" based on NP-complete problems that strictly follows the GPV strategy;

Ongoing Work:

- We generalized decoding algorithms in $\mathbb{F}_3$ for high weights;
- Best algorithms to distinguish $(U, U+V)$-codes and random codes: decoding algorithms;
- Hope to remove the rejection sampling

  $\rightarrow$ Many degrees of freedom in the Prange algorithm!

Thank You!