An overview of WAVE and its implementation

Gustavo Banegas[1]

December 1, 2023

[1]Qualcomm SARL, France
gustavo@cryptme.in
gsouzaba@qti.qualcomm.com

Outline

Table of Contents

Why do we need?

Cryptoapocalypse

Algorithms for Quantum Computation:
Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e.

Why do we need?

Cryptoapocalypse

A fast quantum mechanical algorithm for database search

Lov K. Grover
3C-404A, AT&T Bell Labs
600 Mountain Avenue
Murray Hill NJ 07974
lkg@mhcnet.att.com

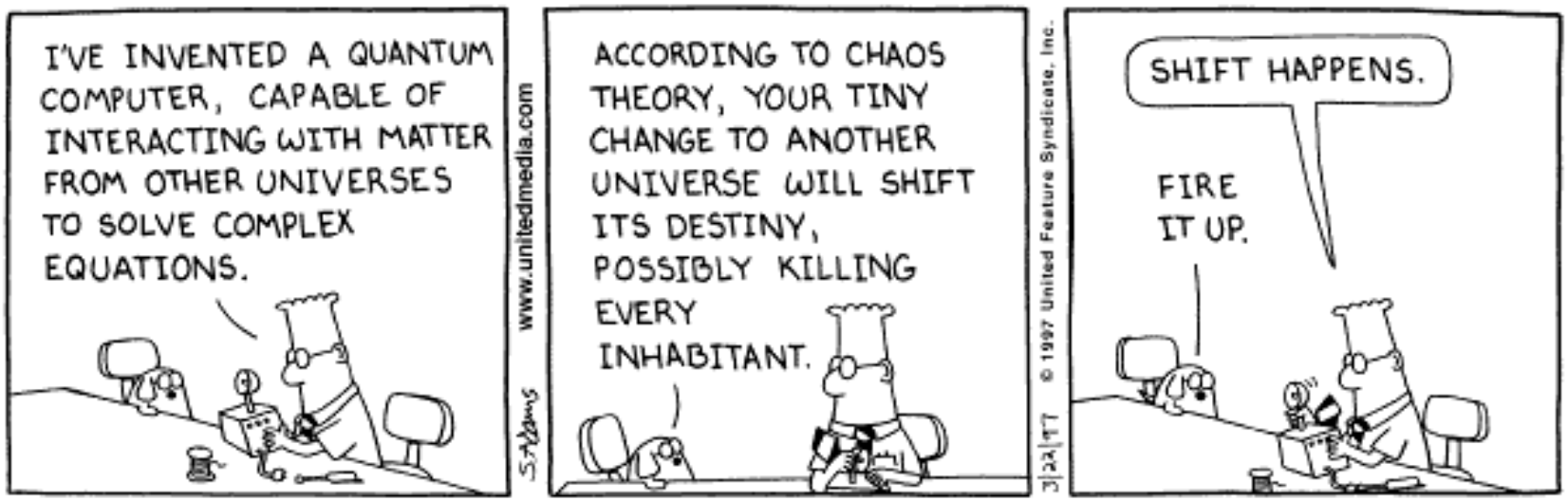
Summary

An unsorted database contains N records, of which just one satisfies a particular property. The problem is to identify that one record. Any classical algorithm, deterministic or probabilistic, will clearly take $O(N)$ steps since on the average it will have to examine a large fraction of the N records. Quantum mechanical systems can do several operations simultaneously due to their wave like properties. This paper gives an $O(\sqrt{N})$ step quantum mechanical algorithm for identifying that record. It is within a constant factor of the fastest possible quantum mechanical algorithm.

This paper applies quantum computing to a mundane problem in information processing and presents an algorithm that is significantly faster than any classical algorithm can be. The problem is this: there is an unsorted database containing N items out of which just one item satisfies a given condition - that one item has to be retrieved. Once an item is examined, it is possible to tell whether or not it satisfies the condition in one step. However, there does not exist any sorting on the database that would aid its selection. The most efficient classical algorithm for this is to examine the items in the database one by one. If an item satisfies the required condition stop; if it does not, keep track of this item so that it is not examined again. It is easily seen

Why do we need?

Cryptoapocalypse



Why do we need?

And other reasons:

- NIST called an "additional post-quantum signature";

Why do we need?

And other reasons:

- NIST called an "additional post-quantum signature";
- We need diversity of schemes based in different mathematical problems;

Why do we need?

And other reasons:

- NIST called an "additional post-quantum signature";
- We need diversity of schemes based in different mathematical problems;
- It is nice to have a scheme that you are into it!

What is code-based?

Linear codes

Code-based cryptography uses error correction codes to achieve cryptographic schemes.
The advantages are:

- We know that linear codes can achieve NP-Hard problem (Decoding);

What is code-based?

Linear codes

Code-based cryptography uses error correction codes to achieve cryptographic schemes.
The advantages are:

- We know that linear codes can achieve NP-Hard problem (Decoding);
- Most of the times, the schemes are fast;

What is code-based?

Linear codes

Code-based cryptography uses error correction codes to achieve cryptographic schemes.
The advantages are:

- We know that linear codes can achieve NP-Hard problem (Decoding);
- Most of the times, the schemes are fast;
- We have small ciphertext or small signatures.

What is code-based?

Which Schemes?

There are several signature schemes based in codes.
For example: Fuleeca, LESS, MEDS, Wave, and others.

The main problem is keys or signature sizes.

- Fuleeca - pk: 1,380 bytes, sig: 1,100 bytes.
- LESS - pk: 14,029 bytes, sig: 8,602 bytes.
- MEDS - pk: 9,923 bytes, sig: 9,896 bytes.
- WAVE - pk: 3,677,390 bytes, sig: 822 bytes.

What is code-based?

Which Schemes?

There are several signature schemes based in codes.
For example: Fuleeca, LESS, MEDS, Wave, and others.

The main problem is keys or signature sizes.

- Fuleeca - pk: 1,380 bytes, sig: 1,100 bytes.
- LESS - pk: 14,029 bytes, sig: 8,602 bytes.
- MEDS - pk: 9,923 bytes, sig: 9,896 bytes.
- WAVE - pk: 3,677,390 bytes, sig: 822 bytes.

What is code-based?

Which Schemes?

There are several signature schemes based in codes.
For example: Fuleeca, LESS, MEDS, Wave, and others.

The main problem is keys or signature sizes.

- Fuleeca - pk: 1,380 bytes, sig: 1,100 bytes.
- LESS - pk: 14,029 bytes, sig: 8,602 bytes.
- MEDS - pk: 9,923 bytes, sig: 9,896 bytes.
- WAVE - pk: 3,677,390 bytes, sig: 822 bytes.

Table of Contents

What is Wave?

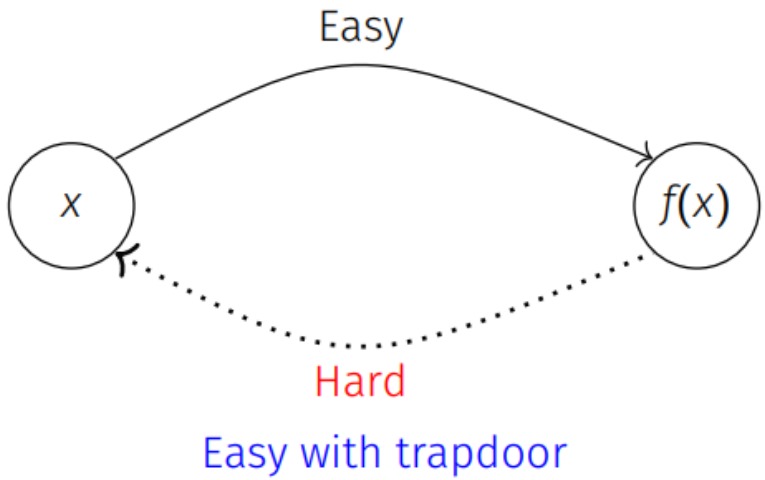
but first definitions...

- Consider a function H;
- f a trapdoor one-way function;

What is Wave?

but first definitions...

- Consider a function H;
- f a trapdoor one-way function;

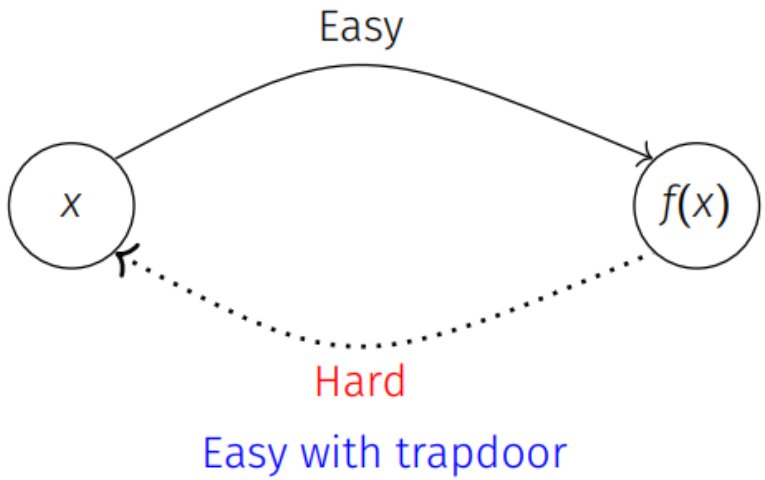


- To sign a message m:
  compute $\sigma \in f^{-1}(H(m))$.

What is Wave?

but first definitions...

- Consider a function H;
- f a trapdoor one-way function;



- To sign a message m:
  compute $\sigma \in f^{-1}(H(m))$.
- to verify (m, σ):
  check $f(\sigma) \stackrel{?}{=} H(m)$.

f needs to surjective.

What is Wave?

Code-based - Signature

We can have one-way functions in code-based cryptography.
For example:
$f_w : (c, e) \in \mathcal{C} \times \{e : |e| = w\} \to c + e$.
Inverting f_w is decoding $\mathcal{C}$ at distance w.

What is Wave?

Code-based - Signature

We can have one-way functions in code-based cryptography.
For example:
$f_w : (c, e) \in \mathcal{C} \times \{e : |e| = w\} \rightarrow c + e$.
Inverting f_w is decoding $\mathcal{C}$ at distance w.For some w, it is easy to invert f_w.

What is Wave?

Wave - Signature

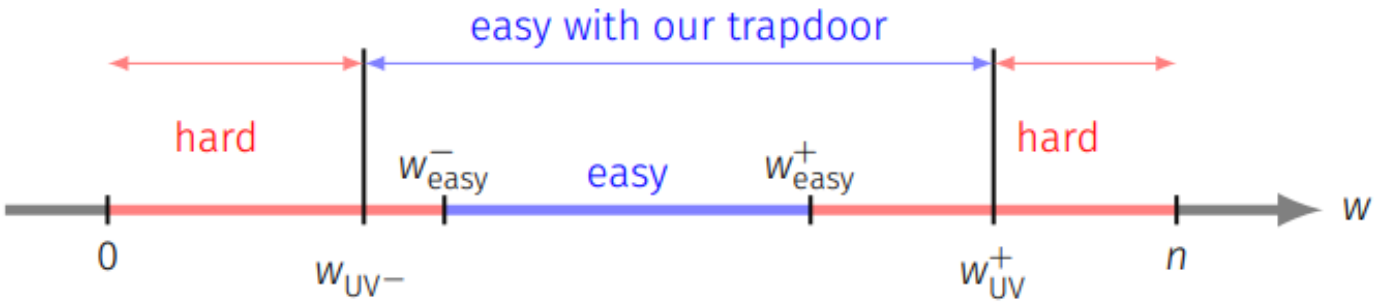
In WAVE, we take advantage of certain properties and that we have the control of the parameters to find a code $\mathcal{C}$ that is hard enough that cannot be decoded without the trapdoor function. In the end we can instantiate the signature as:

- Public Data: a hash function H, an $[n, k]$-code $\mathcal{C}$ and the distance w;
- Signing m:
  1. Hash: $m \to y = H(m)$;
  2. Decoding: find with a trapdoor $c \in \mathcal{C}$ such that $|y - c| = w$.
- Verifying:
  $c \in \mathcal{C}$ and $H(m) - c| = w$

What is Wave?

Wave - Signature

To achieve such security we need to define some parameters first.

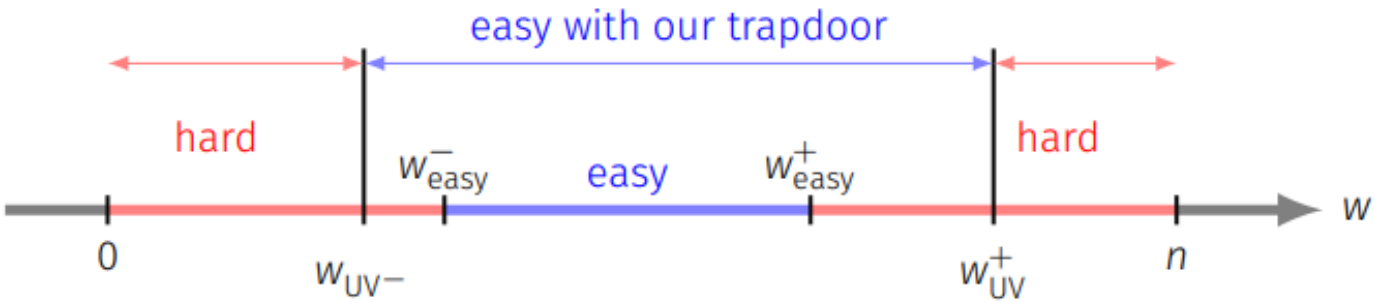


In the end, the trapdoor of WAVE is: U as $[n/2, k_u]$-code, V as $[n/2, k_v]$-code, and the vectors $b, c, d \in \mathbb{F}_q^{n/2}$ and a permutation π. Also, WAVE works in $\mathbb{F}_q$ where $q = 3$.

What is Wave?

Wave - Signature

To achieve such security we need to define some parameters first.



In the end, the trapdoor of WAVE is: U as $[n/2, k_u]$-code, V as $[n/2, k_v]$-code, and the vectors $b, c, d \in \mathbb{F}_q^{n/2}$ and a permutation π.
Also, WAVE works in $\mathbb{F}_q$ where $q = 3$.
The security assumption of WAVE:
It is hard to distinguish random and generalized $(U|U+V)$ codes;

What is Wave?

Wave - Signature

Given the secret-key/trapdoor: U, V, b, c, d and π. One can decode as the following:

1. Given $y \in \mathbb{F}_q^n$, break $y = (y_L | y_R)^\pi$;
2. Compute any $x_V \in V$ with Prange's Algorithm;
3. Using Prange's algorithm: compute $x_U \in U$ by choosing the k_U symbols $x_u(i)'s$ such that:

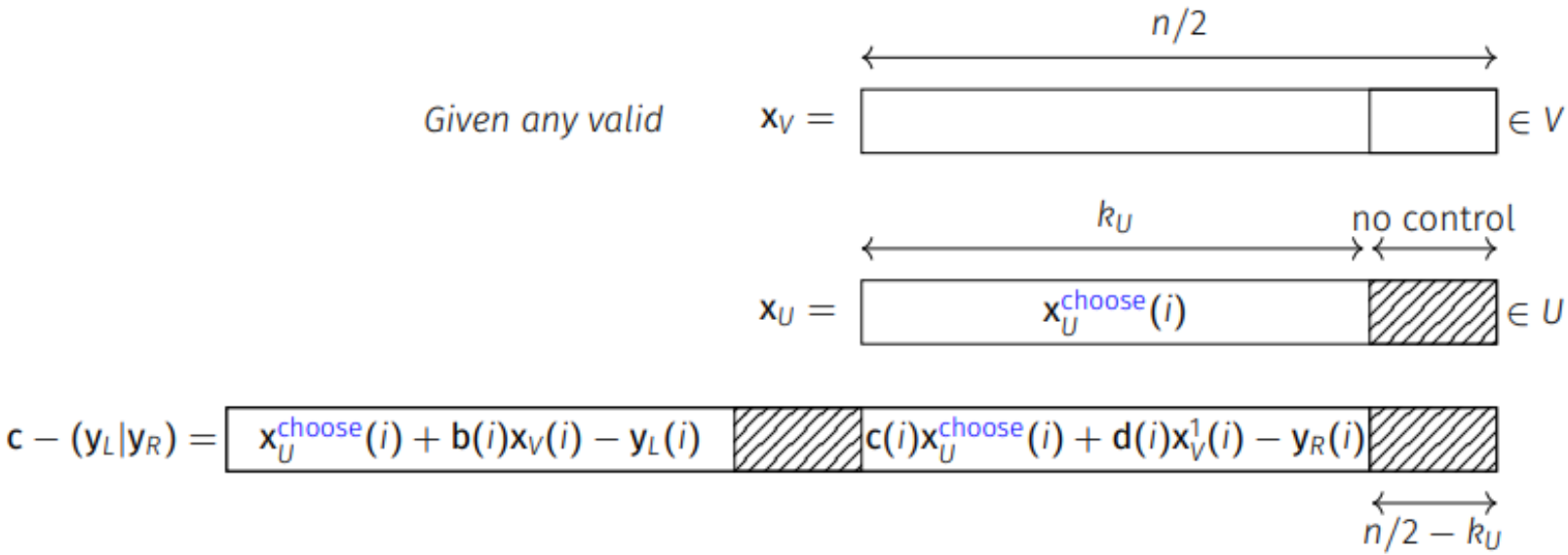
$$x_U(i) + b(i)x_V(i) \neq y_L(i)$$

$$c(i)x_U(i) + d(i)x_V(i) \neq y_R(i)$$

4. Return $c = (x_U + b \star x_V | c \star x_U + d \star x_V)^\pi$.

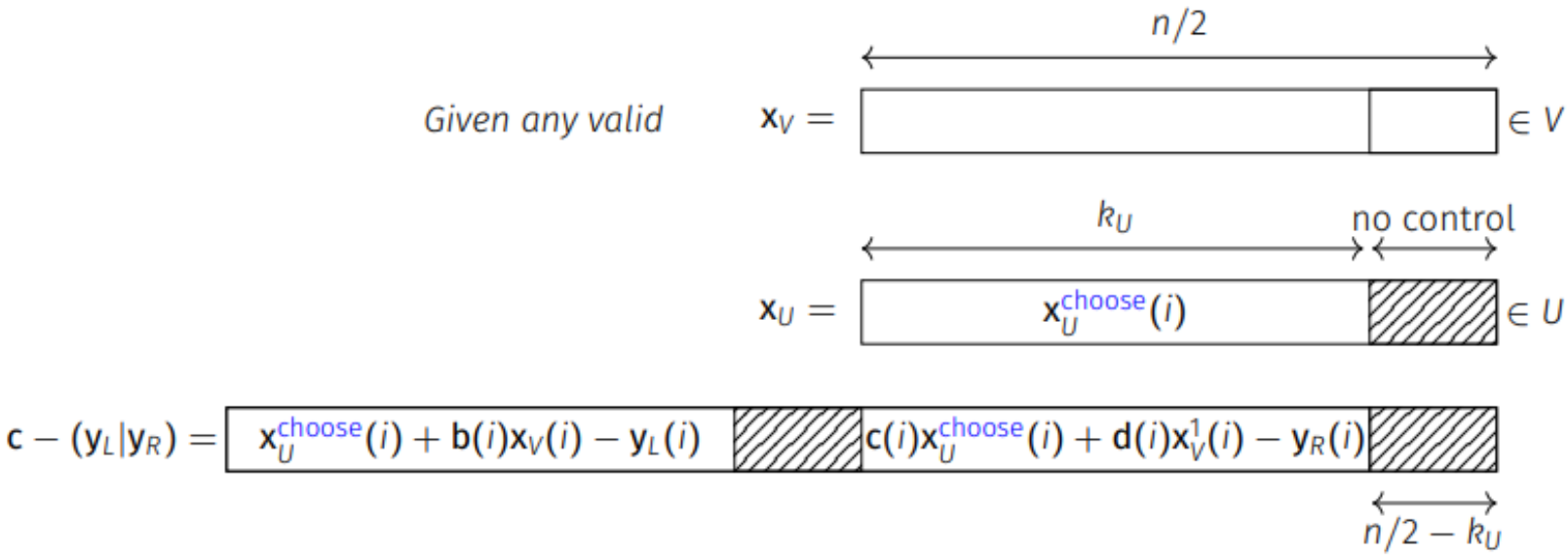
What is Wave?

Wave - Signature



What is Wave?

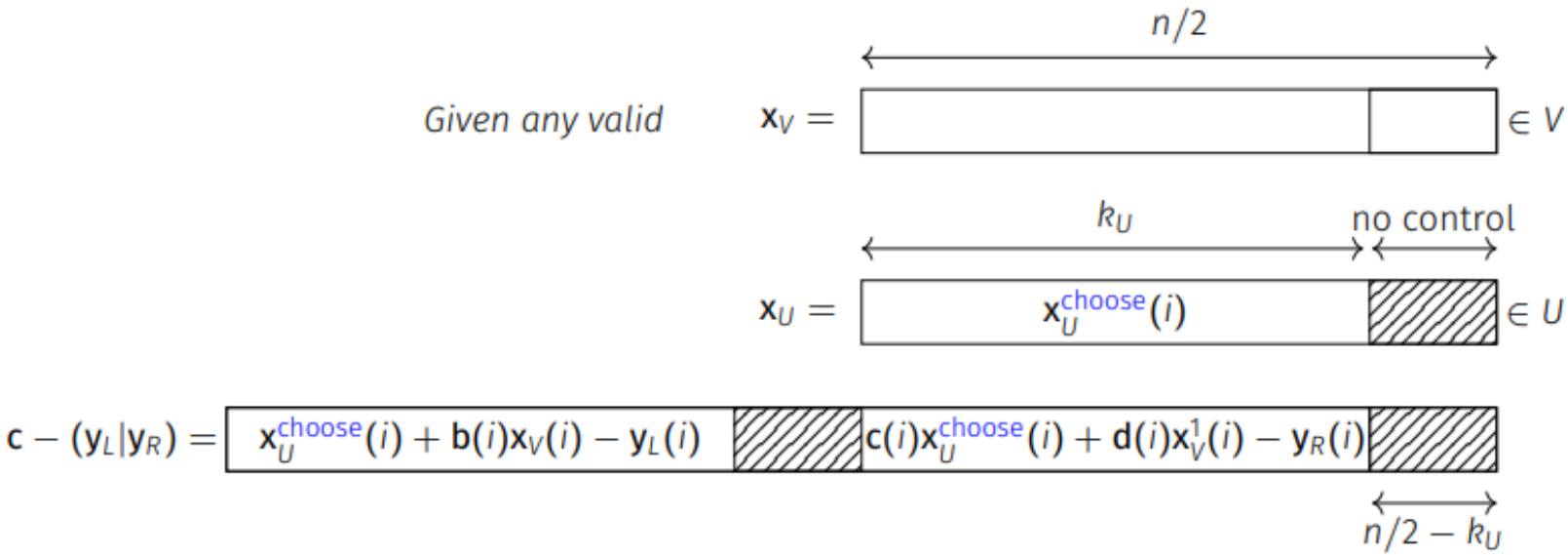
Wave - Signature



The typical distance w will be:

What is Wave?

Wave - Signature



The typical distance w will be:

$$w = 2k_u + 2\frac{q-1}{q} > w_{easy}^{+} = (k_U + k_v) + \frac{q-1}{q}(n - (k_U + k_v))$$

The key generation looks like this:

Output: $\begin{cases} \mathsf{pk} = \mathbf{M} \in \mathbb{F}_3^{k\times(n-k)}, \\ \mathsf{sk} = (\mathbf{H}_U, \mathbf{G}_V, \pi, \mathbf{b}, \mathbf{c}) \in \mathbb{F}_3^{(n/2-k_U)\times n/2} \times \mathbb{F}_3^{k_V\times n/2} \times \mathfrak{S}_n \times \mathbb{F}_3^{n/2} \times (\mathbb{F}_3\backslash\{0\})^{n/2}. \end{cases}$

1: $\mathbf{G}_V \xleftarrow{\$} \mathbb{F}_3^{k_V\times n/2}$
2: $\mathbf{H}_V \leftarrow \mathsf{Orthogonal}(\mathbf{G}_V)$ ▷ $\mathbf{H}_V \in \mathbb{F}_3^{(n/2-k_V)\times n/2}$ of full rank such that $\mathbf{G}_V\mathbf{H}_V^\top = 0$
3: $\mathbf{H}_U \xleftarrow{\$} \mathbb{F}_3^{(n/2-k_U)\times n/2}$
4: $\mathbf{b} \xleftarrow{\$} \mathbb{F}_3^{n/2}$
5: $\mathbf{c} \xleftarrow{\$} (\mathbb{F}_3 \setminus \{0\})^{n/2}$ ▷ Coordinates of $\mathbf{c}$ are non-zero
6: $\mathbf{d} \leftarrow \mathbf{1} + \mathbf{b} \star \mathbf{c}$
7: $\mathbf{H} \leftarrow \left(\begin{array}{c|c} \mathbf{d}\star\mathbf{H}_U & -\mathbf{b}\star\mathbf{H}_U \\ \hline -\mathbf{c}\star\mathbf{H}_V & \mathbf{H}_V \end{array}\right)$
8: $\sigma \xleftarrow{\$} \mathfrak{S}_n$
9: $((\mathbf{Id}_k \mid \mathbf{R}), \pi) \leftarrow \mathsf{SystGaussElim}(\mathbf{H}, \sigma)$
10: $\mathbf{M} \leftarrow \mathbf{M}(\mathbf{R})$
11: **return** $(\mathsf{pk} = \mathbf{M}, \mathsf{sk} = (\mathbf{H}_U, \mathbf{G}_V, \pi, \mathbf{b}, \mathbf{c}))$

The signature looks like this:

Input: a message $\mathbf{m}$, $\mathsf{sk} = (\mathbf{H}_U, \mathbf{G}_V, \pi, \mathbf{b}, \mathbf{c})$
Output: $(\mathbf{s}, \mathtt{salt})$

1: **repeat**
2: $\quad \mathtt{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$ ▷ $\lambda \in \{128, 192, 256\}$ denotes the security level
3: $\quad \mathbf{x} \leftarrow \mathsf{Hash}(\mathbf{m} \parallel \mathtt{salt})$ ▷ $\mathbf{x} \in \mathbb{F}_3^{n-k}$
4: $\quad \mathbf{y} = (\mathbf{y}_L \parallel \mathbf{y}_R) \leftarrow (\mathbf{x} \parallel \mathbf{0}^k)^{\pi^{-1}}$ ▷ $\mathbf{y}_L, \mathbf{y}_R \in \mathbb{F}_3^{n/2}$
5: $\quad \mathbf{y}_V \leftarrow \mathbf{y}_R - \mathbf{c} \star \mathbf{y}_L$
6: $\quad \mathbf{e}_V \leftarrow \mathsf{Decode}_V(\mathbf{y}_V, \mathbf{G}_V)$
7: $\quad \mathbf{y}_U \leftarrow \mathbf{y}_L - \mathbf{b} \star \mathbf{y}_V$ ▷ simplification of $\mathbf{y}_U = (\mathbf{1} + \mathbf{c} \star \mathbf{b}) \star \mathbf{y}_L - \mathbf{b} \star \mathbf{y}_R$
8: $\quad \mathbf{e}_U \leftarrow \mathsf{Decode}_U(\mathbf{y}_U, \mathbf{e}_V, \mathbf{b}, \mathbf{c}, \mathbf{H}_U)$
9: $\quad \mathbf{e}_L \leftarrow \mathbf{e}_U + \mathbf{b} \star \mathbf{e}_V$
10: $\quad \mathbf{e}_R \leftarrow \mathbf{c} \star \mathbf{e}_L + \mathbf{e}_V$ ▷ simplification of $\mathbf{e}_R = \mathbf{c} \star \mathbf{e}_U + (\mathbf{1} + \mathbf{b} \star \mathbf{c}) \star \mathbf{e}_V$
11: **until** $(|\mathbf{e}_V|, n/2 - w + |\mathbf{e}_L \star \mathbf{e}_R|) \in \mathsf{Accept}$
12: $\mathbf{e} \leftarrow (\mathbf{e}_L \parallel \mathbf{e}_R)^{\pi}$
13: $\mathbf{s} \leftarrow \mathbf{e}_{[n-k,n)}$
14: **return** $(\mathbf{s}, \mathtt{salt})$

The verification looks like this:

Input: $\mathbf{m} \in \{0,1\}^*$, $\sigma = (\mathtt{salt}, \mathbf{s}) \in \{0,1\}^{2\lambda} \times \mathbb{F}_3^k$, $\mathbf{M} \in \mathbb{F}_3^{k \times (n-k)}$
Output: True or False

1: $\mathbf{x} \leftarrow \mathsf{Hash}(\mathbf{m} \,\|\, \mathtt{salt})$
2: **for** $0 \le i < (k-1)/2$ **do** ▷ handle entries of $\mathbf{s}$ in pairs
3: $\quad(\widehat{\mathbf{s}}(2i), \widehat{\mathbf{s}}(2i+1)) \leftarrow (\mathbf{s}(2i) + \mathbf{s}(2i+1), \mathbf{s}(2i) - \mathbf{s}(2i+1))$ ▷ $(2i, 2i+1)$-th entries of $\widehat{\mathbf{s}}$
4: $\quad$**if** $\widehat{\mathbf{s}}(2i) = 1$ **then**
5: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \mathrm{row}(\mathbf{M}, 2i)$ ▷ add $2i$-th row of $\mathbf{M}$
6: $\quad$**else if** $\widehat{\mathbf{s}}(2i) = 2$ **then**
7: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} - \mathrm{row}(\mathbf{M}, 2i)$ ▷ subtract $2i$-th row of $\mathbf{M}$
8: $\quad$**if** $\widehat{\mathbf{s}}(2i+1) = 1$ **then**
9: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \mathrm{row}(\mathbf{M}, 2i+1)$ ▷ add $(2i+1)$-th row of $\mathbf{M}$
10: $\quad$**else if** $\widehat{\mathbf{s}}(2i+1) = 2$ **then**
11: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} - \mathrm{row}(\mathbf{M}, 2i+1)$ ▷ subtract $(2i+1)$-th row of $\mathbf{M}$
12: **if** k is odd **then** ▷ handle last entry if necessary
13: $\quad$**if** $\mathbf{s}(k-1) = 1$ **then**
14: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \mathrm{row}(\mathbf{M}, k-1)$
15: $\quad$**else if** $\mathbf{s}(k-1) = 2$ **then**
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} - \mathrm{row}(\mathbf{M}, k-1)$
17: **return** $|\mathbf{s}| + |\mathbf{x}| = w$

Table of Contents

How do we represent $\mathbb{F}_3$?

Representation of $\mathbb{F}_3$

We can see the elements of $\mathbb{F}_3$ as $\{0, 1, 2\}$ and if we represent them in a binary form, we end up with $00, 01, 10$, respectively. How do we represent this? Is this the best form?

How do we represent F_3?

Representation of F_3

Well, from the literature, the best way of representing the an element from $\mathbb{F}_3$ is as a composition of two elements as (a_h, a_ℓ), and then represent each element as:

$$0 \longleftrightarrow (0,0), \quad 1 \longleftrightarrow (0,1), \quad 2 \longleftrightarrow (1,1).$$

How do we represent F_3?

Representation of F_3

Well, from the literature, the best way of representing the an element from $\mathbb{F}_3$ is as a composition of two elements as (a_h, a_ℓ), and then represent each element as:

$$0 \longleftrightarrow (0,0), \quad 1 \longleftrightarrow (0,1), \quad 2 \longleftrightarrow (1,1).$$

The main reason is because of the arithmetic. In this way we can use bitslice and use less operations.

How do we represent F_3?

Operations in F_3

Addition and subtraction in $\mathbb{F}_3$ requires 7 binary operations, negation in $\mathbb{F}_3$ requires 1 binary operation, and multiplication in $\mathbb{F}_3$ requires 3 binary operations. A ternary vector $x = (x_i)_{0 \le i < n}$ where $x_i \longleftrightarrow (x_{i,h}, x_{i,\ell})$ is represented by (x_h, x_ℓ) with $x_h = (x_{i,h})_{0 \le i < n}$ and $x_\ell = (x_{i,\ell})_{0 \le i < n}$.

$$
\begin{aligned}
c = a + b \quad &\Longleftrightarrow \quad \begin{cases} c_h &= (a_\ell \oplus b_h) \,\&\, (a_h \oplus b_\ell) \\ c_\ell &= (a_\ell \oplus b_\ell) \mid (a_h \oplus b_\ell \oplus b_h) \end{cases} \\
c = a - b \quad &\Longleftrightarrow \quad \begin{cases} c_h &= (a_\ell \oplus b_\ell \oplus b_h) \,\&\, (a_h \oplus b_\ell) \\ c_\ell &= (a_\ell \oplus b_\ell) \mid (a_h \oplus b_h) \end{cases} \\
c = -a \quad &\Longleftrightarrow \quad \begin{cases} c_h &= a_h \oplus a_\ell \\ c_\ell &= a_\ell \end{cases} \\
c = a \star b \quad &\Longleftrightarrow \quad \begin{cases} c_h &= (a_h \oplus b_h) \,\&\, a_\ell \,\&\, b_\ell \\ c_\ell &= a_\ell \,\&\, b_\ell. \end{cases}
\end{aligned}
$$

Other aspects of implementation

Constant-time implementation

We have several functions that need to be constant time. For example:
Gauss Elimination in $\mathbb{F}_3$, the decoders, and the random sampling.
For the Gauss elimination, we did a small tweak for the reduce and normalize operations.

$$\mathit{normalize}_\ell(\mathrm{x}) = (1 + x_\ell - x_\ell^2) \cdot \mathrm{x}$$

and

$$\mathit{reduce}_\ell(\mathrm{x}, \mathrm{y}) = (x_\ell^2 \cdot \mathrm{x} + (1 - x_\ell^2)) \cdot \mathrm{y}, (1 - x_\ell^2 - x_l y_l) \cdot \mathrm{x} + x_\ell^2 \cdot \mathrm{y}$$

Open Problems

Are we done?

I believe that we are far away from optimize WAVE. Here we have some of the possible tweaks.

- Decrease the key sizes;
- Verify the parameters;
- Speed up the Gaussian Elimination (partially is done using 4 Russians method by Marianna);
- Hardware implementation of Wave.

Questions

Thank you for your attention.
Questions?
gustavo@cryptme.in