Isochrony
~~Constant time~~ techniques for lattice-based signatures

Mélissa Rossi

Presentation based on

- **[CCS'2019]** joint work with Barthe, Belaïd, Espitau, Fouque and Tibouchi
- **[PQCRYPTO'20]** joint work with Howe, Prest and Ricosset

Workshop RISQ

24/03/2020

« Constant time » is a confusing term

Constant time does not mean constant execution time

« Constant time » is a confusing term

Constant time does not mean constant execution time

« Constant time »

The execution time **does not depend on the private key.**

➡ Not necessarily constant !

« Constant time » is a confusing term

Constant time does not mean constant execution time

« Constant time »

The execution time **does not depend on the private key.**

➡ Not necessarily constant !

Better say isochronous ?

- Howe et al. (2019/1411)

« Constant time » is a confusing term

Constant time does not mean constant execution time

« Constant time »

The execution time **does not depend on the private key.**

➡ Not necessarily constant !

Better say isochronous ?

- Howe et al. (2019/1411)

Assumption

$+, -, \times, /$ Constant time on integers

Gaussian Distributions and Lattice Based signatures I

Fiat-Shamir with aborts Family

- BLISS
- Crystals-Dilithium
- qTesla

Hash and Sign Family

- GPV
- Falcon

Gaussian Distributions and Lattice Based signatures I

Fiat-Shamir with aborts
Family

Gaussian Distributions and Lattice Based signatures I

Fiat-Shamir with aborts Family

Schnorr-like signatures with aborts

- Lyubashevsky (EC'12)

Based on SIS, LWE or variants

Fiat-Shamir with aborts Family

Schnorr-like signatures with aborts

- Lyubashevsky (EC'12)

Based on SIS, LWE or variants

Public parameter: $\mathbf{a}$

Secret key: $\mathbf{s}$ (short)

Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} \mod q$

Fiat-Shamir with aborts Family

Schnorr-like signatures with aborts

- Lyubashevsky (EC'12)

Based on SIS, LWE or variants

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}$ (short)
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} \mod q$

Sign(s, m):
1: do
2: $\quad \mathbf{y} \xleftarrow{\$} Y$
3: $\quad c \leftarrow H(\mathbf{ay}, m)$
4: $\quad \mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y}$
5: while Rejected($\mathbf{z}$)
6: return ($\mathbf{z}$, c)

Fiat-Shamir with aborts Family

Schnorr-like signatures with aborts

- Lyubashevsky (EC'12)

Based on SIS, LWE or variants

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}$ (short)
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} \mod q$

Sign(s, m):
1: do
2: $\mathbf{y} \xleftarrow{\$} Y$
3: $c \leftarrow H(\mathbf{ay}, m)$
4: $\mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y}$
5: while Rejected($\mathbf{z}$)
6: return $(\mathbf{z}, c)$

Verify($\mathbf{z}, c, \mathbf{t}, m$):
1: $\mathbf{v} \leftarrow \mathbf{a} \cdot \mathbf{z} - c \cdot \mathbf{t}$
2: return 1 if $c = H(\mathbf{v}, m)$ and $\mathbf{z}$ is small else 0

Fiat-Shamir with aborts Family

Schnorr-like signatures with aborts

- Lyubashevsky (EC'12)

Based on SIS, LWE or variants

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}$ (short)
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} \mod q$

Sign(s, m):
1: do
2: $\quad \mathbf{y} \xleftarrow{\$} Y$
3: $\quad c \leftarrow H(\mathbf{ay}, m)$
4: $\quad \mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y}$
5: while Rejected($\mathbf{z}$)
6: return ($\mathbf{z}$, c)

Why ?

Verify($\mathbf{z}, c, \mathbf{t}, m$):
1: $\mathbf{v} \leftarrow \mathbf{a} \cdot \mathbf{z} - c \cdot \mathbf{t}$
2: return 1 if $c = H(\mathbf{v}, m)$ and $\mathbf{z}$ is small else 0

Rejection Sampling Lemma

- Lyubashevsky (EUROCRYPT'12)

Also called Acceptance-Rejection method

Going from an actual distribution Y to an ideal target distribution X

Rejection Sampling Lemma

- Lyubashevsky (EUROCRYPT'12)

Also called Acceptance-Rejection method

Going from an actual distribution Y to an ideal target distribution X

To sample from a distribution X, with density f, one uses samples from the distribution Y, with density g as follows:

(1) Get a sample y from distribution Y

(2) Accept y as a sample drawn from X with probability $\frac{f(y)}{M \cdot g(y)}$ = $\frac{\text{Target}}{M \cdot \text{Actual}}$

reject otherwise

Rejection Sampling Lemma

- Lyubashevsky (EUROCRYPT'12)

Also called Acceptance-Rejection method

Going from an actual distribution Y to an ideal target distribution X

To sample from a distribution X, with density f, one uses samples from the distribution Y, with density g as follows:

(1) Get a sample y from distribution Y

(2) Accept y as a sample drawn from X with probability $\dfrac{f(y)}{M \cdot g(y)} = \dfrac{\text{Target}}{M \cdot \text{Actual}}$

reject otherwise

statistically close to

(if $M \cdot g(y) \geq f(y) \quad \forall y$)

(1) Get a sample x from distribution X

(2) Accept x with probability $\dfrac{1}{M}$,

reject otherwise

Gaussian Distributions and Lattice Based signatures I

Where are the Gaussian distributions in Fiat-Shamir with aborts signatures?

Key generation:

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}$
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s}$

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}, \mathbf{e}$
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$

Signature algorithm:

1: **do**
2: $\quad \mathbf{y} \xleftarrow{\$} Y$
3: $\quad c \leftarrow H(\mathbf{ay}, m)$
4: $\quad \mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y}$
5: **while** Rejected($\mathbf{z}$)
6: **return** ($\mathbf{z}, c$)

Gaussian Distributions and Lattice Based signatures I

Where are the Gaussian distributions in Fiat-Shamir with aborts signatures?

Key generation:

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}$
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s}$

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}, \mathbf{e}$
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$

Signature algorithm:

1: **do**
2: $\quad \mathbf{y} \xleftarrow{\$} Y$
3: $\quad c \leftarrow H(\mathbf{ay}, m)$
4: $\quad \mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y}$
5: **while Rejected(z)**
6: **return** $(\mathbf{z}, c)$

Gaussian Distributions and Lattice Based signatures I

Where are the Gaussian distributions in Fiat-Shamir with aborts signatures?

Key generation:

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}$
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s}$

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}, \mathbf{e}$
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$

Signature algorithm:

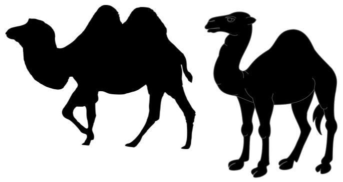
do
$\quad \mathbf{y} \xleftarrow{\$} Y$
$\quad c \leftarrow H(\mathbf{ay}, m)$
$\quad \mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y}$
while Rejected(z)
return $(\mathbf{z}, c)$

They were originally used for two reasons:

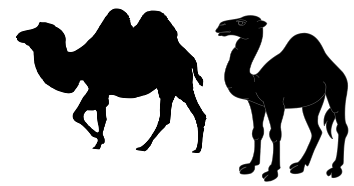
Security reductions

Performance

Gaussians lead to implementation vulnerabilities



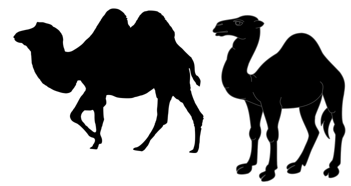
Gaussians lead to implementation vulnerabilities



- ➡ How to evaluate them ?

  by computing transcendental functions $\exp(\,.\,)$ and $\cosh(\,.\,)$

Gaussians lead to implementation vulnerabilities

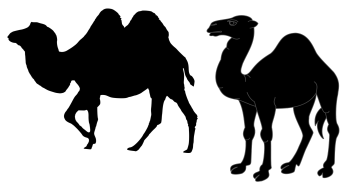


➡ How to evaluate them ?

by computing transcendental functions $\exp(\,.\,)$ and $\cosh(\,.\,)$

Timing vulnerabilities

Gaussians lead to implementation vulnerabilities



➡ How to evaluate them ?

by computing transcendental functions $\exp(\,.\,)$ and $\cosh(\,.\,)$

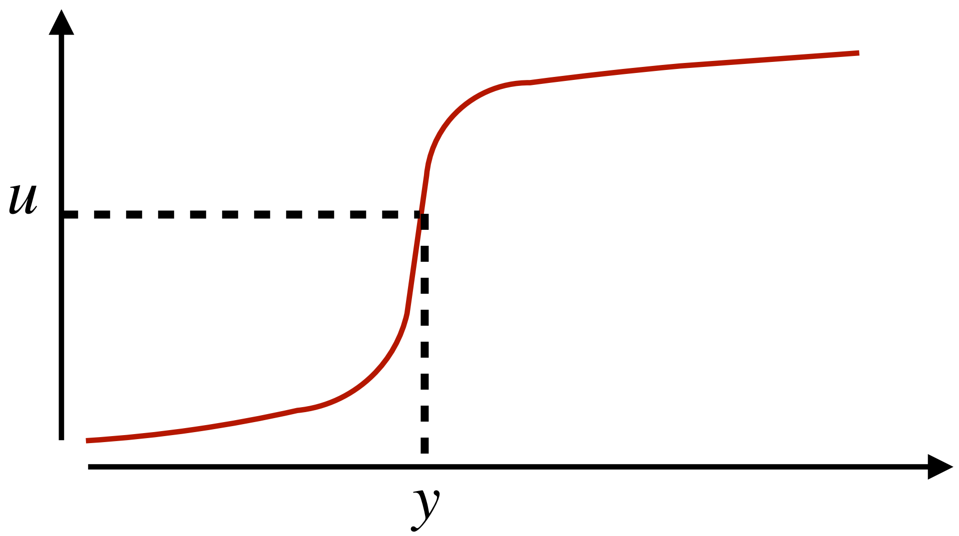
Timing vulnerabilities

➡ How to sample ?

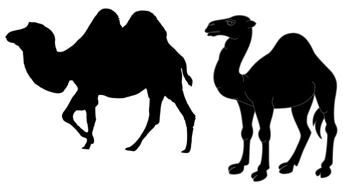
Many possibilities, ex :

Large Cumulative Distribution Tables (CDT)

u uniform in [0,1], recover y with a table of the red curve



Gaussians lead to implementation vulnerabilities



➡ How to evaluate them ?

by computing transcendental functions $\exp(\,.\,)$ and $\cosh(\,.\,)$

Timing vulnerabilities

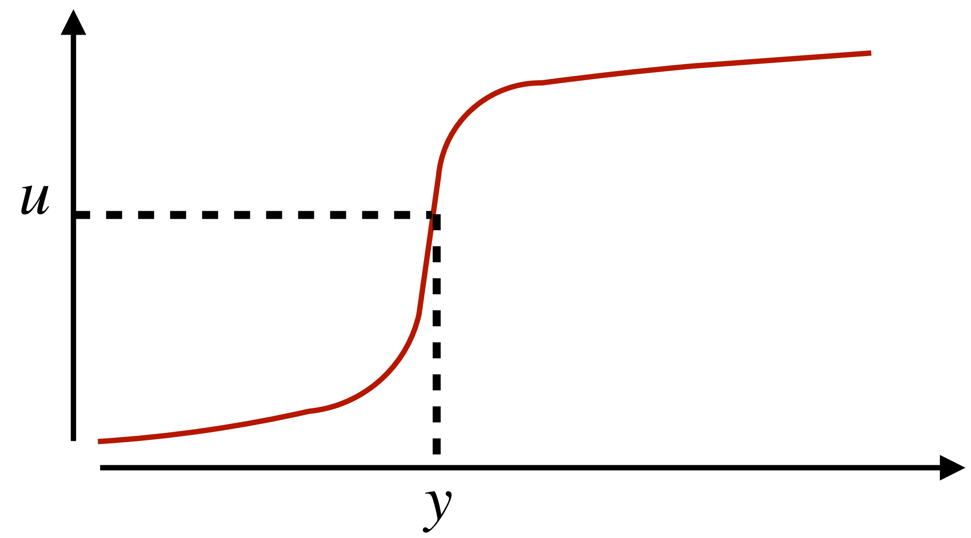
➡ How to sample ?

Many possibilities, ex :

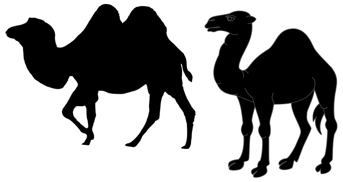
Large Cumulative Distribution Tables (CDT)

u uniform in [0,1], recover y with a table of the red curve

Binary search leaks y by memory access pattern



Gaussians lead to implementation vulnerabilities



➡ How to evaluate them ?

by computing transcendental functions $\exp(\,.\,)$ and $\cosh(\,.\,)$

Timing vulnerabilities

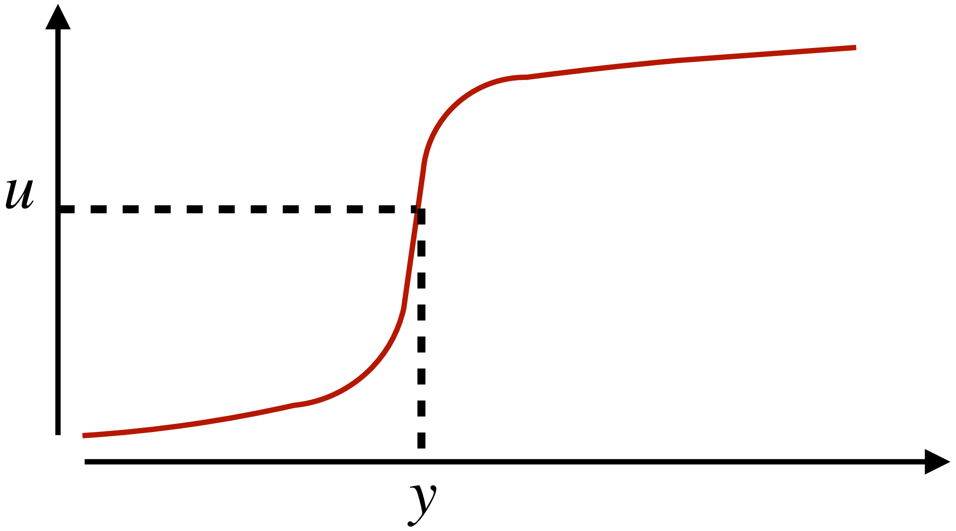
➡ How to sample ?

Many possibilities, ex :

Large Cumulative Distribution Tables (CDT)

u uniform in [0,1], recover y with a table of the red curve

Binary search leaks y by memory access pattern



Side channel vulnerabilities

Gaussian distributions are hard to implement securely

BLISS was the first practical implementation of a lattice based signature scheme.

BLISS
Fiat-Shamir with aborts
Bimodal Gaussians

- Ducas et al (CRYPTO'13)

Gaussian distributions are hard to implement securely

BLISS was the first practical implementation of a lattice based signature scheme.

BLISS
Fiat-Shamir with aborts
Bimodal Gaussians

- Ducas et al (CRYPTO'13)

Many side channel attacks
targeting Gaussian distributions (timing)

- Groot Bruinderink et al. CHES'2016
- Espitau et al. SAC'2016
- Pessl et al. ACM-CCS'2017
- Espitau et al. ACM-CCS'2017
- Bootle et al. ASIACRYPT'2018
- Barthe et al. ACM-CCS'2019

Gaussian distributions are hard to implement securely

BLISS was the first practical implementation of a lattice based signature scheme.

BLISS
Fiat-Shamir with aborts
Bimodal Gaussians

- Ducas et al (CRYPTO'13)

Many side channel attacks targeting Gaussian distributions (timing)

- Groot Bruinderink et al. CHES'2016
- Espitau et al. SAC'2016
- Pessl et al. ACM-CCS'2017
- Espitau et al. ACM-CCS'2017
- Bootle et al. ASIACRYPT'2018
- Barthe et al. ACM-CCS'2019

New designs: 

Gaussians are now avoided in Fiat-Shamir with aborts lattice signature schemes
… but there is a price to pay in terms of performance

Gaussian distributions are hard to implement securely

BLISS was the first practical implementation of a lattice based signature scheme.

BLISS
Fiat-Shamir with aborts
Bimodal Gaussians

- Ducas et al (CRYPTO'13)

Many side channel attacks targeting Gaussian distributions (timing)

- Groot Bruinderink et al. CHES'2016
- Espitau et al. SAC'2016
- Pessl et al. ACM-CCS'2017
- Espitau et al. ACM-CCS'2017
- Bootle et al. ASIACRYPT'2018
- Barthe et al. ACM-CCS'2019

New designs: 

Gaussians are now avoided in Fiat-Shamir with aborts lattice signature schemes … but there is a price to pay in terms of performance

Crystals-Dilithium: Only uniform distributions
- Ducas et al (NIST-PQC'17)

qTesla: Gaussian sampling only in the keygen
- Bindel et al (NIST-PQC'17)

Gaussian Distributions and Lattice Based signatures 2

Fiat-Shamir with abort
Signatures

Hash and Sign
Signatures

Hash and Sign Signatures

- Gentry, Peikert and Vaikuntanathan (STOC'08)

Gaussian Distributions and Lattice Based signatures 2

Hash and Sign Signatures

- Gentry, Peikert and Vaikuntanathan (STOC'08)

KeyGen()

1: Generate matrices $\mathbf{A}, \mathbf{B}$

such that $\begin{cases} \mathbf{BA} = \mathbf{0} \\ \mathbf{B} \text{ has small coefficients} \end{cases}$

2: $pk \leftarrow \mathbf{A}$

3: $sk \leftarrow \mathbf{B}$

Gaussian Distributions and Lattice Based signatures 2

Hash and Sign Signatures

- Gentry, Peikert and Vaikuntanathan (STOC'08)

KeyGen()

1: Generate matrices $\mathbf{A}, \mathbf{B}$

such that $\begin{cases} \mathbf{BA} = \mathbf{0} \\ \mathbf{B} \text{ has small coefficients} \end{cases}$

2: $pk \leftarrow \mathbf{A}$

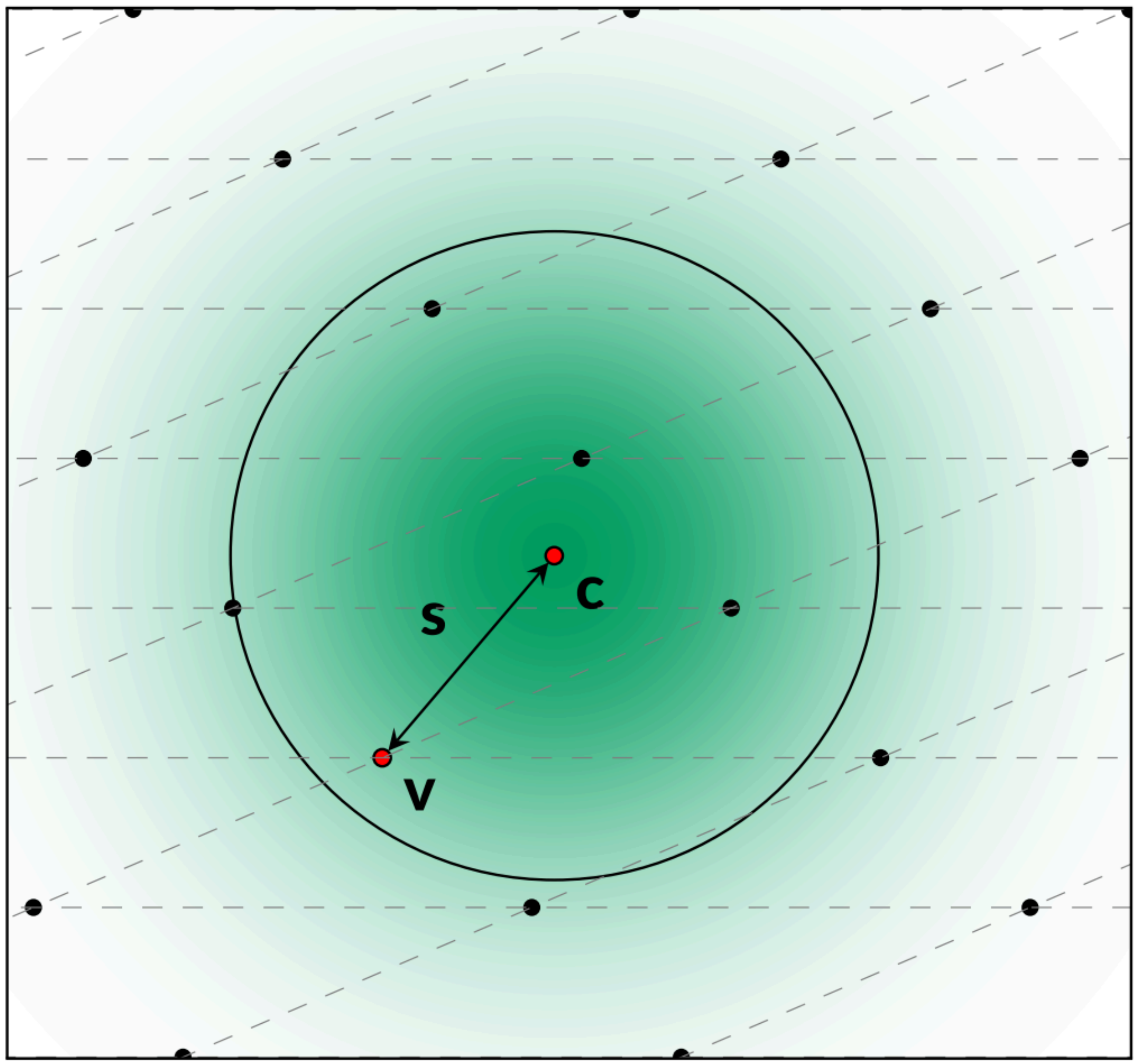
3: $sk \leftarrow \mathbf{B}$

Sign(m,sk)

1: Compute $\mathbf{c}$ such that $\mathbf{cA} = H(m)$

2: $\mathbf{v} \leftarrow$ a vector in $\Lambda(\mathbf{B})$ close to $\mathbf{c}$

3: $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$



Gaussian Distributions and Lattice Based signatures 2

Hash and Sign Signatures

- Gentry, Peikert and Vaikuntanathan (STOC'08)

KeyGen()

1: Generate matrices $\mathbf{A}, \mathbf{B}$

such that $\begin{cases} \mathbf{BA} = \mathbf{0} \\ \mathbf{B} \text{ has small coefficients} \end{cases}$

2: $pk \leftarrow \mathbf{A}$

3: $sk \leftarrow \mathbf{B}$

Sign(m,sk)

1: Compute $\mathbf{c}$ such that $\mathbf{cA} = H(m)$

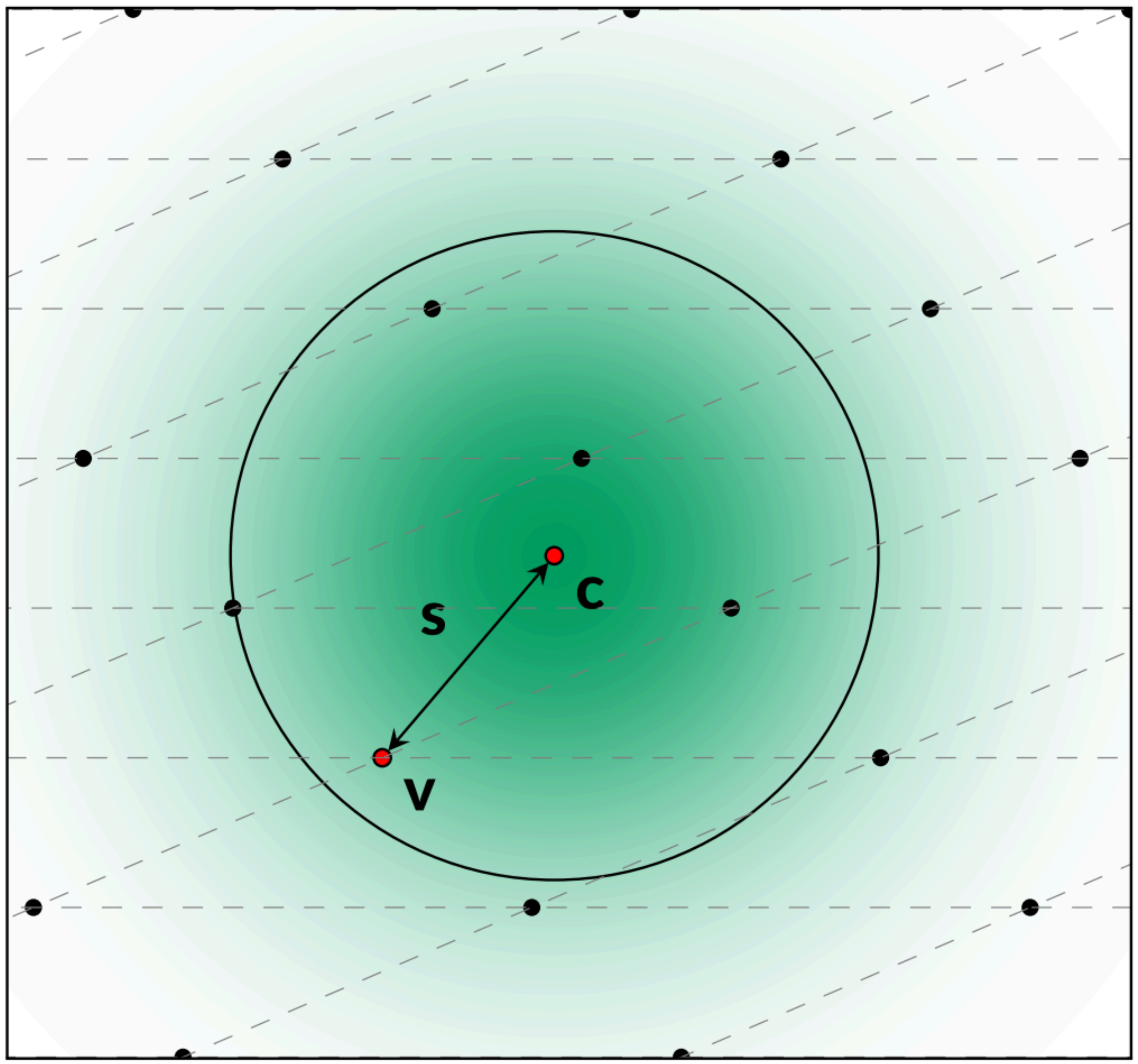
2: $\mathbf{v} \leftarrow$ a vector in $\Lambda(\mathbf{B})$ close to $\mathbf{c}$

3: $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$

Verify(m,pk,s)

Accept iff:

$\begin{cases} \mathbf{s} \text{ is short} \\ \mathbf{sA} = H(m) \end{cases}$



Gaussian Distributions and Lattice Based signatures 2

Hash and Sign Signatures

- Gentry, Peikert and Vaikuntanathan (STOC'08)

KeyGen()

1: Generate matrices $\mathbf{A}, \mathbf{B}$

such that $\begin{cases} \mathbf{BA} = \mathbf{0} \\ \mathbf{B} \text{ has small coefficients} \end{cases}$

2: $pk \leftarrow \mathbf{A}$

3: $sk \leftarrow \mathbf{B}$

Sign(m,sk)

1: Compute $\mathbf{c}$ such that $\mathbf{cA} = H(m)$

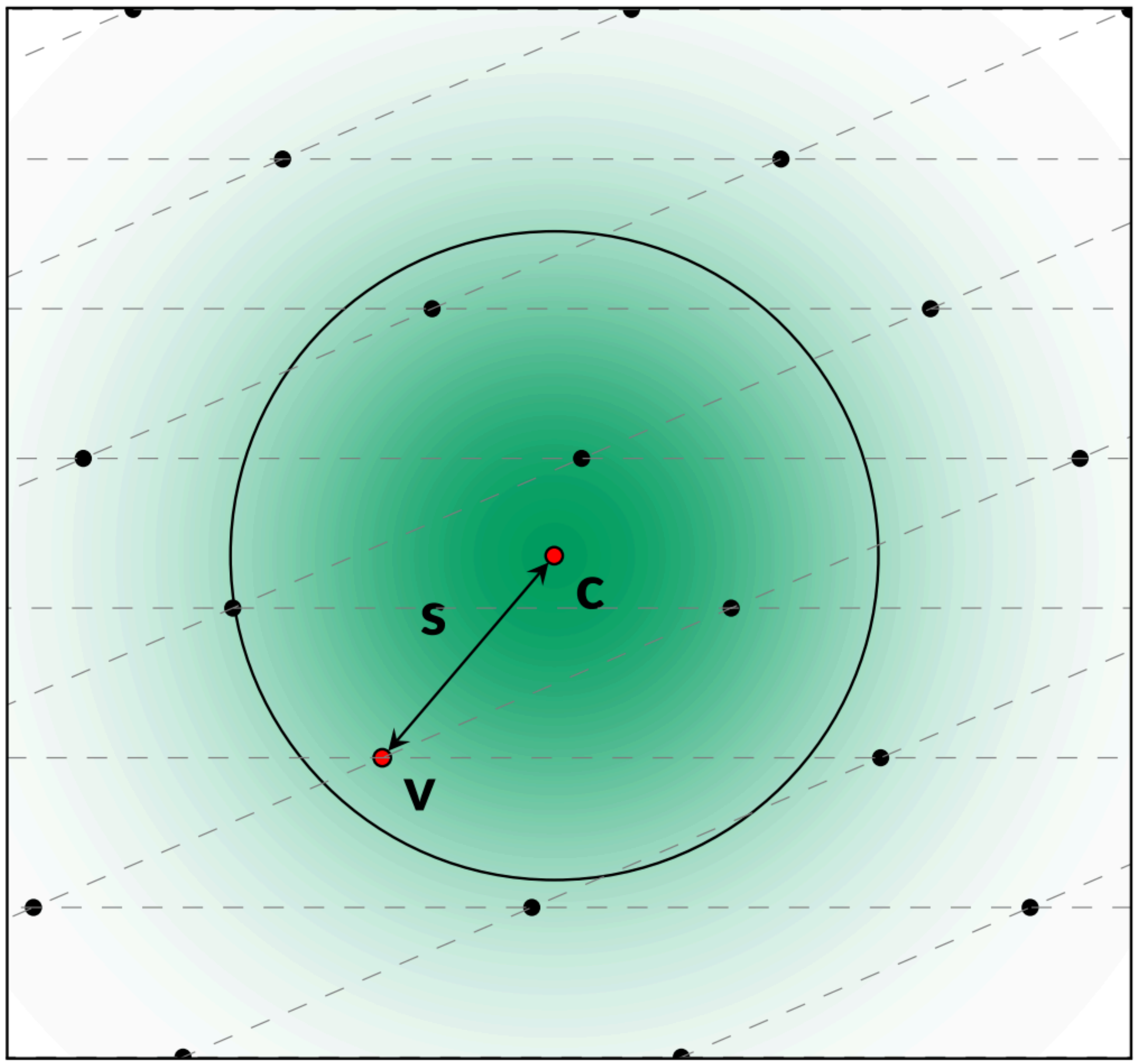
2: $\mathbf{v} \leftarrow$ a vector in $\Lambda(\mathbf{B})$ close to $\mathbf{c}$

3: $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$

Verify(m,pk,s)

Accept iff:

$\begin{cases} \mathbf{s} \text{ is short} \\ \mathbf{sA} = H(m) \end{cases}$



Gaussian distributions are not « implementation friendly »

Side channel attacks
targeting the Gaussian rejection sampling

- Espitau et al. SAC'2016
- Fouque et al EUROCRYPT'2019

Main takeaway of this presentation

Gaussian distributions can actually be

- ☑ Simpler to implement
- ☑ Provably resistant to timing attacks

Main takeaway of this presentation

Gaussian distributions can actually be

- ☑ Simpler to implement
- ☑ Provably resistant to timing attacks

1. Galactics: a polynomial approximation tool for Gaussians
2. Isochronous BLISS
3. Isochronous FALCON

Approximation tool

Evaluating a polynomial can be done isochronously

➡ A tool to use a polynomials instead of (bimodal) Gaussians

☑ Using Renyi divergence and Sobolev spaces

☑ Proof of concept developed in sage8.3

☑ Polynomials have integer coefficients (Lattice reduction)

Renyi divergence result

- Prest ASIACRYPT'17

Take two cryptographic schemes

- One with distribution $\mathscr{D}$
- One with an approximate distribution $\mathscr{D}'$ with the same support

For keeping the 'same' bit security for both schemes with at most 2^{64} signature queries (NIST suggestion),

$$\left\| 1 - \frac{\mathscr{D}'}{\mathscr{D}} \right\|_{\infty} \leq 2^{-45}.$$

Renyi divergence result

- Prest ASIACRYPT'17

Take two cryptographic schemes

- One with distribution $\mathscr{D}$
- One with an approximate distribution $\mathscr{D}'$ with the same support

For keeping the 'same' bit security for both schemes with at most 2^{64} signature queries (NIST suggestion),

$$\left\| 1 - \frac{\mathscr{D}'}{\mathscr{D}} \right\|_\infty \leq 2^{-45}.$$

$$\sqrt{\frac{2 \cdot \lambda}{2 \cdot (2 \cdot \lambda + 1)^2 \cdot Q_{\mathscr{D}}}}$$

Renyi divergence result

- Prest ASIACRYPT'17

Take two cryptographic schemes

- One with distribution $\mathscr{D}$
- One with an approximate distribution $\mathscr{D}'$ with the same support

For keeping the 'same' bit security for both schemes with at most 2^{64} signature queries (NIST suggestion),

$$\left\| 1 - \frac{\mathscr{D}'}{\mathscr{D}} \right\|_{\infty} \leq 2^{-45}.$$

$$\sqrt{\frac{2 \cdot \lambda}{2 \cdot (2 \cdot \lambda + 1)^2 \cdot Q_{\mathscr{D}}}}$$

$\exp(\,\cdot\,)$ $\cosh(\,\cdot\,)$

Goal : $\mathscr{D} =$  (a transcendental function on an interval)

$\mathscr{D}' =$ a polynomial

Other Polynomial Approximations

How to choose $\mathscr{D}'$?

- **Taylor development**

$$\mathscr{D}'(x) = \mathscr{D}(0) + \mathscr{D}^{(1)}(0) \cdot x + \cdots + \frac{\mathscr{D}^{(d)}(0)}{d!} \cdot x^d$$

Not precise enough

Other Polynomial Approximations

How to choose $\mathscr{D}'$?

- **Taylor development**

$$\mathscr{D}'(x) = \mathscr{D}(0) + \mathscr{D}^{(1)}(0) \cdot x + \cdots + \frac{\mathscr{D}^{(d)}(0)}{d!} \cdot x^d$$

Not precise enough

- **Padé approximants** (rational function approximation)

$$\mathscr{D}'(x) = \frac{P(x)}{Q(x)}$$

  - Prest ASIACRYPT'17

Two polynomials, higher degrees

Other Polynomial Approximations

How to choose $\mathscr{D}'$?

- **Taylor development**

$$\mathscr{D}'(x) = \mathscr{D}(0) + \mathscr{D}^{(1)}(0) \cdot x + \cdots + \frac{\mathscr{D}^{(d)}(0)}{d!} \cdot x^d$$

Not precise enough

- **Padé approximants** (rational function approximation)

$$\mathscr{D}'(x) = \frac{P(x)}{Q(x)}$$

  - Prest ASIACRYPT'17

Two polynomials, higher degrees

- **Minimax computations** : Sollya software package
  - Brisebarre and Chevillard IEEE'07
  - Chevillard, Joldes and Lauter ICMS'10
  - Zhao, Steinfeld and Sakzad 2018/1234

Floating point arithmetics

$$\mathscr{D}' = \arg \min_{\deg(P) \leq d} \left(\sup_{x \in I} \left(1 - \frac{P(x)}{\mathscr{D}(x)} \right) \right)$$

Our alternative Polynomial Approximation

How to choose $\mathscr{D}'$?

$\exp(\,\cdot\,)$ $\cosh(\,\cdot\,)$

Our alternative Polynomial Approximation

How to choose $\mathscr{D}'$?

$\exp(\cdot)$

$\cosh(\cdot)$

Transcendent functions $\longrightarrow$ Degree d polynomial in $\mathbb{R}[x]$ $\longrightarrow$ Degree d polynomial in $\mathbb{Z}[x]$ with η bit coefficients

Towards a polynomial approximation

We want $\left\| 1 - \dfrac{\mathscr{D}'}{\mathscr{D}} \right\|_\infty \leq 2^{-45}$ and $\mathscr{D}' \in \mathbb{Z}[x]$.

It corresponds to finding the closest element of 1 in $\mathbb{Z}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

Towards a polynomial approximation

We want $\left\| 1 - \frac{\mathscr{D}'}{\mathscr{D}} \right\|_\infty \leq 2^{-45}$ and $\mathscr{D}' \in \mathbb{Z}[x]$.

It corresponds to finding the closest element of 1 in $\mathbb{Z}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

In a first attempt, let us look for the closest element of 1 in $\mathbb{R}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

Towards a polynomial approximation

We want $\left\| 1 - \frac{\mathscr{D}'}{\mathscr{D}} \right\|_\infty \leq 2^{-45}$ and $\mathscr{D}' \in \mathbb{Z}[x]$.

It corresponds to finding the closest element of 1 in $\mathbb{Z}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

In a first attempt, let us look for the closest element of 1 in $\mathbb{R}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

An orthogonal projection could be great.

Towards a polynomial approximation

We want $\left\| 1 - \frac{\mathscr{D}'}{\mathscr{D}} \right\|_\infty \leq 2^{-45}$ and $\mathscr{D}' \in \mathbb{Z}[x]$.

It corresponds to finding the closest element of 1 in $\mathbb{Z}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

In a first attempt, let us look for the closest element of 1 in $\mathbb{R}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

An orthogonal projection could be great.

The $\|\cdot\|_\infty$ norm is not Euclidean, no easy projection on a subspace.

Towards a polynomial approximation

We want $\left\| 1 - \frac{\mathscr{D}'}{\mathscr{D}} \right\|_\infty \leq 2^{-45}$ and $\mathscr{D}' \in \mathbb{Z}[x]$.

It corresponds to finding the closest element of 1 in $\mathbb{Z}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

In a first attempt, let us look for the closest element of 1 in $\mathbb{R}[x]/\mathscr{D}(x)$ for the $\|\cdot\|_\infty$ norm.

An orthogonal projection could be great.

The $\|\cdot\|_\infty$ norm is not Euclidean, no easy projection on a subspace.

Over-approximate $\|\cdot\|_\infty$ by a Euclidean norm.

When functional analysis meets cryptography

Sobolev H^2 norm

- Sobolev TAMS'1963

For u and v differentiable functions on I:

$$\langle u, v \rangle = \frac{1}{|I|} \int_I uv + |I| \int_I u'v'$$

Corresponding norm:

$$|u|_S^2 = \frac{1}{|I|} \int_I u^2 + |I| \int_I u'^2$$

Equivalence with $\|\cdot\|_\infty$:

$$\|u\|_\infty \leq \sqrt{2} \cdot |u|_S$$

Sobolev H^2 norm

- Sobolev TAMS'1963

For u and v differentiable functions on I:

$$\langle u, v \rangle = \frac{1}{|I|}\int_I uv + |I|\int_I u'v'$$

Corresponding norm:

$$|u|_S^2 = \frac{1}{|I|}\int_I u^2 + |I|\int_I u'^2$$

Equivalence with $\|\cdot\|_\infty$:

$$\|u\|_\infty \leq \sqrt{2}\cdot|u|_S$$

Let us look for the closest element of 1 in $\mathbb{R}[x]/\mathscr{D}(x)$ for the $|\cdot|_S$ norm.

Proof of concept (Available on Github at https://github.com/espitau/GALACTICS)

Proof of concept (Available on Github at https://github.com/espitau/GALACTICS)

Roadmap

1. Galactics: a polynomial approximation tool for Gaussians
2. **Isochronous BLISS**
3. Isochronous FALCON

BLISS Signature Scheme

- Ducas et al (CRYPTO'13)

Elements are polynomials in $\mathscr{R} = \dfrac{\mathbb{Z}[x]}{x^n + 1}$

Sign $(m, pk = \mathbf{a}, sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$:

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow \mathscr{D}_{\mathbb{Z}^n,\sigma}$

2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2$ $\quad\quad \zeta(q-2) = 1 \mod 2q$

3: $\mathbf{c} \leftarrow H(\lceil u \rfloor_d \mod p, m)$

4: choose a random bit b

5: $\mathbf{z}_1 \leftarrow (-1)^b \mathbf{c}\mathbf{s}_1 + \mathbf{y}_1$

6: $\mathbf{z}_2 \leftarrow (-1)^b \mathbf{c}\mathbf{s}_2 + \mathbf{y}_2$

7: restart except wp

$$\frac{1}{M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2}\right)}$$

8: return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

We polynomially approximate $\exp(\,.\,)$ and $\cosh(\,.\,)$ using the GALACTICS tool.

➡ Also used for the Gaussian Sampling

BLISS Signature Scheme

- Ducas et al (CRYPTO'13)

Elements are polynomials in $\mathscr{R} = \dfrac{\mathbb{Z}[x]}{x^n + 1}$

<u>Sign</u> $(m, pk = \mathbf{a}, sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$:

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow \mathscr{D}_{\mathbb{Z}^n, \sigma}$

2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2$ $\quad \zeta(q-2) = 1 \mod 2q$

3: $\mathbf{c} \leftarrow H(\lceil u \rfloor_d \mod p, m)$

4: choose a random bit b

5: $\mathbf{z}_1 \leftarrow (-1)^b \mathbf{c}\mathbf{s}_1 + \mathbf{y}_1$

6: $\mathbf{z}_2 \leftarrow (-1)^b \mathbf{c}\mathbf{s}_2 + \mathbf{y}_2$

7: restart except wp

$$\frac{1}{M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2}\right)}$$

8: return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

Gaussian sampling

We polynomially approximate $\exp(\,.\,)$ and $\cosh(\,.\,)$ using the GALACTICS tool.

➡ Also used for the Gaussian Sampling

BLISS Signature Scheme

- Ducas et al (CRYPTO'13)

Elements are polynomials in $\mathscr{R} = \dfrac{\mathbb{Z}[x]}{x^n + 1}$

Sign $(m, pk = \mathbf{a}, sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$:

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow \mathscr{D}_{\mathbb{Z}^n, \sigma}$

2: $\mathbf{u} \leftarrow \zeta \cdot \mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2$ $\quad \zeta(q-2) = 1 \mod 2q$

3: $\mathbf{c} \leftarrow H(\lfloor u \rceil_d \mod p, m)$

4: choose a random bit b

5: $\mathbf{z}_1 \leftarrow (-1)^b \mathbf{c}\mathbf{s}_1 + \mathbf{y}_1$

6: $\mathbf{z}_2 \leftarrow (-1)^b \mathbf{c}\mathbf{s}_2 + \mathbf{y}_2$

7: restart except wp

$$\frac{1}{M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2}\right)}$$

8: return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

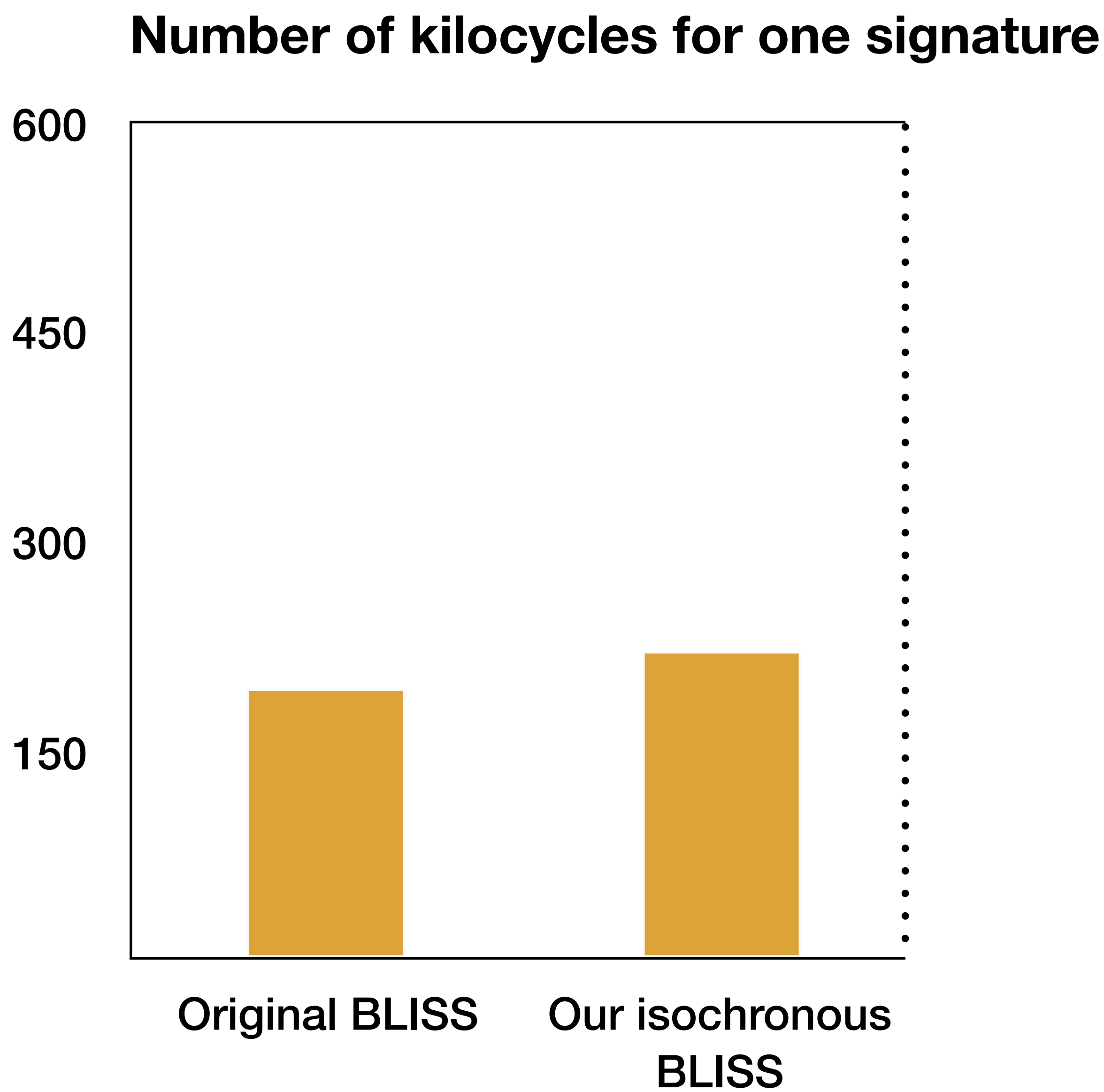
Gaussian sampling

Bimodal Gaussian rejection sampling

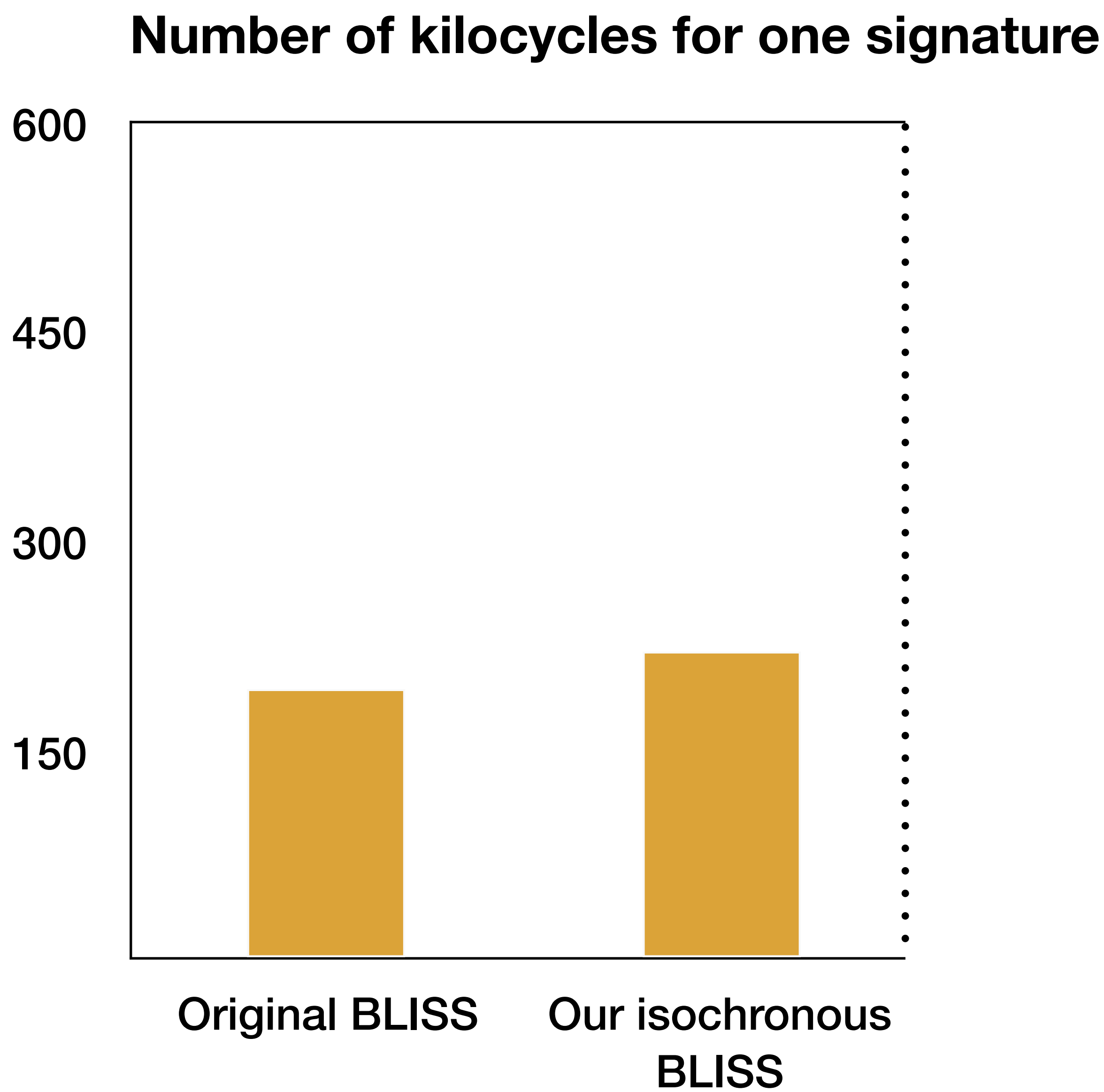
We polynomially approximate $\exp(\,.\,)$ and $\cosh(\,.\,)$ using the GALACTICS tool.

➡ Also used for the Gaussian Sampling

Performance (in kcycles)



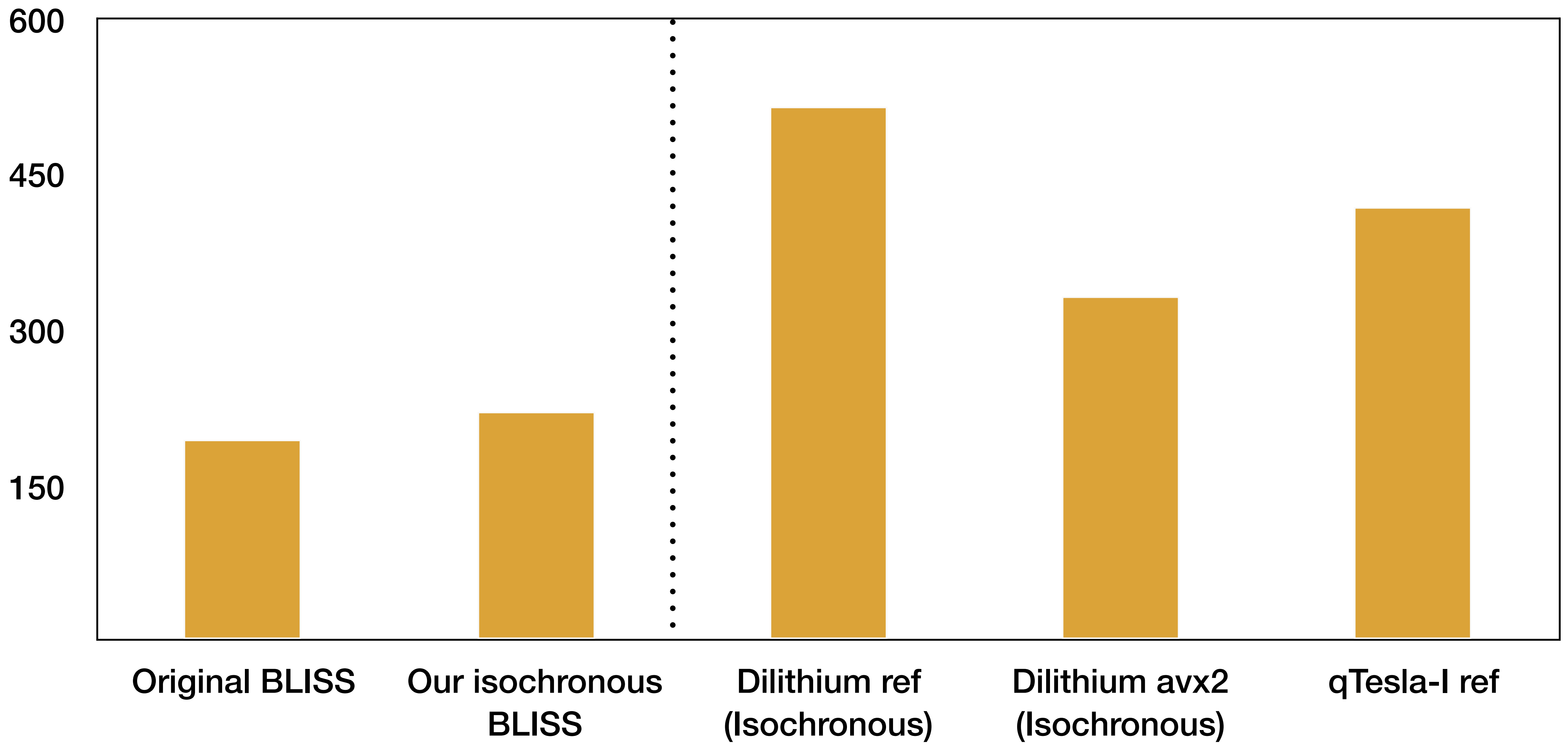
Performance (in kcycles)



The performance loss for isochrony is small

Performance (in kcycles)

Number of kilocycles for one signature



The performance loss for isochrony is small

Recall that:

Crystals-Dilithium: Only uniform distributions
qTesla: Gaussian sampling only in the keygen

and they are more conservative than BLISS

Roadmap

1. Galactics: a polynomial approximation tool for Gaussians
2. Isochronous BLISS
3. **Isochronous FALCON**

Falcon Gaussian sampler

Algorithm SampleZ(σ, μ)

Require: $\mu \in [0,1), \sigma \leq \sigma_0$

Ensure: $z \sim D_{\mathbb{Z},\sigma,\mu}$

1. $z_0 \leftarrow$ Basesampler()
2. $b \leftarrow \{0,1\}$ uniformly
3. $z \leftarrow (2b-1) \cdot z_0 + b$
4. $x \leftarrow -\dfrac{(z-\mu)^2}{2\sigma^2} + \dfrac{z_0^2}{2\sigma_0^2}$
5. Accept with probability $\exp(x)$
   Restart to 1. otherwise

Falcon Gaussian sampler

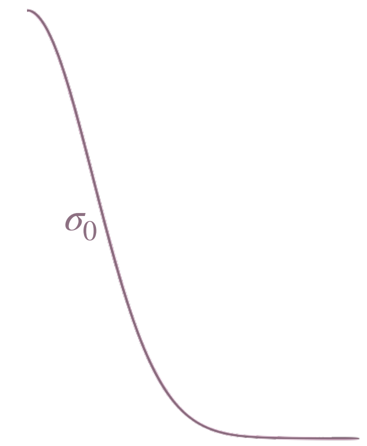
Algorithm SampleZ(σ, μ)

Require: $\mu \in [0,1), \sigma \leq \sigma_0$

Ensure: $z \sim D_{\mathbb{Z},\sigma,\mu}$

1. $z_0 \leftarrow$ Basesampler()
2. $b \leftarrow \{0,1\}$ uniformly
3. $z \leftarrow (2b-1)\cdot z_0 + b$
4. $x \leftarrow -\dfrac{(z-\mu)^2}{2\sigma^2} + \dfrac{z_0^2}{2\sigma_0^2}$
5. Accept with probability $\exp(x)$
   Restart to 1. otherwise

1.



Falcon Gaussian sampler

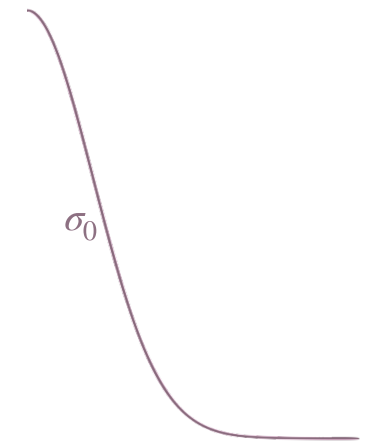
Algorithm SampleZ(σ, μ)

Require: $\mu \in [0,1), \sigma \leq \sigma_0$

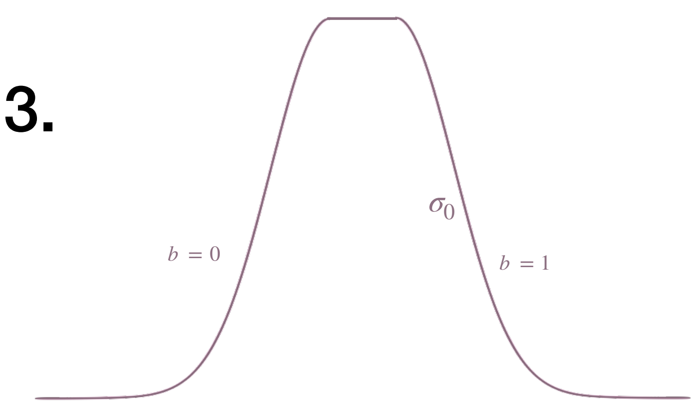
Ensure: $z \sim D_{\mathbb{Z},\sigma,\mu}$

1. $z_0 \leftarrow$ Basesampler()
2. $b \leftarrow \{0,1\}$ uniformly
3. $z \leftarrow (2b-1) \cdot z_0 + b$
4. $x \leftarrow -\dfrac{(z-\mu)^2}{2\sigma^2} + \dfrac{z_0^2}{2\sigma_0^2}$
5. Accept with probability $\exp(x)$
   Restart to 1. otherwise

1.



3.



Falcon Gaussian sampler

Algorithm SampleZ(σ, μ)

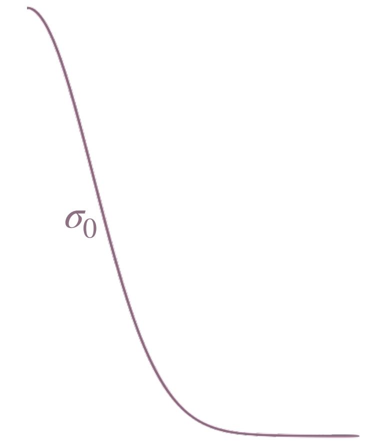
Require: $\mu \in [0,1), \sigma \leq \sigma_0$

Ensure: $z \sim D_{\mathbb{Z},\sigma,\mu}$

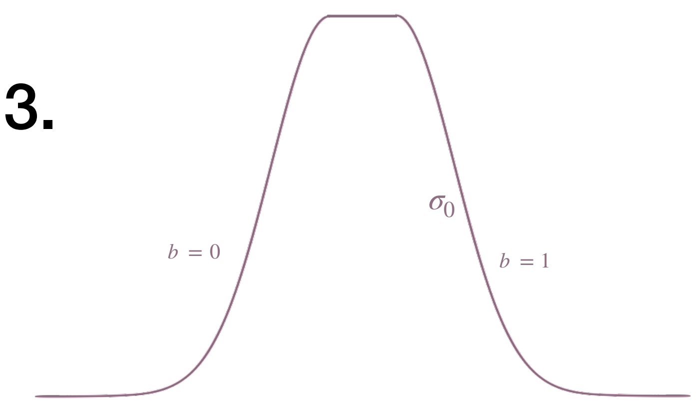
1. $z_0 \leftarrow$ Basesampler()
2. $b \leftarrow \{0,1\}$ uniformly
3. $z \leftarrow (2b-1)\cdot z_0 + b$
4. $x \leftarrow -\dfrac{(z-\mu)^2}{2\sigma^2} + \dfrac{z_0^2}{2\sigma_0^2}$
5. Accept with probability $\exp(x)$
   Restart to 1. otherwise

$$P_{\text{accept}} = \frac{\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)}{\exp\left(-\frac{z_0^2}{2\sigma_0^2}\right)}$$

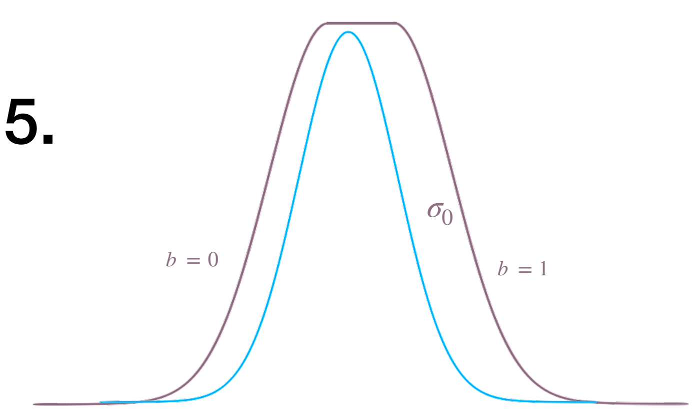
1.



3.



5.



Constant time Falcon Gaussian sampler

Algorithm SampleZ(σ, μ)

Require: $\mu \in [0,1), \sigma \leq \sigma_0$

Ensure: $z \sim D_{\mathbb{Z},\sigma,\mu}$

1. $z_0 \leftarrow$ Basesampler()
2. $b \leftarrow \{0,1\}$ uniformly
3. $z \leftarrow (2b-1) \cdot z_0 + b$
4. $x \leftarrow -\dfrac{(z-\mu)^2}{2\sigma^2} + \dfrac{z_0^2}{2\sigma_0^2}$
5. Accept with probability $\exp(x)$
   Restart to 1. otherwise

Constant time Falcon Gaussian sampler

Algorithm SampleZ(σ, μ)

Require: $\mu \in [0,1), \sigma \leq \sigma_0$

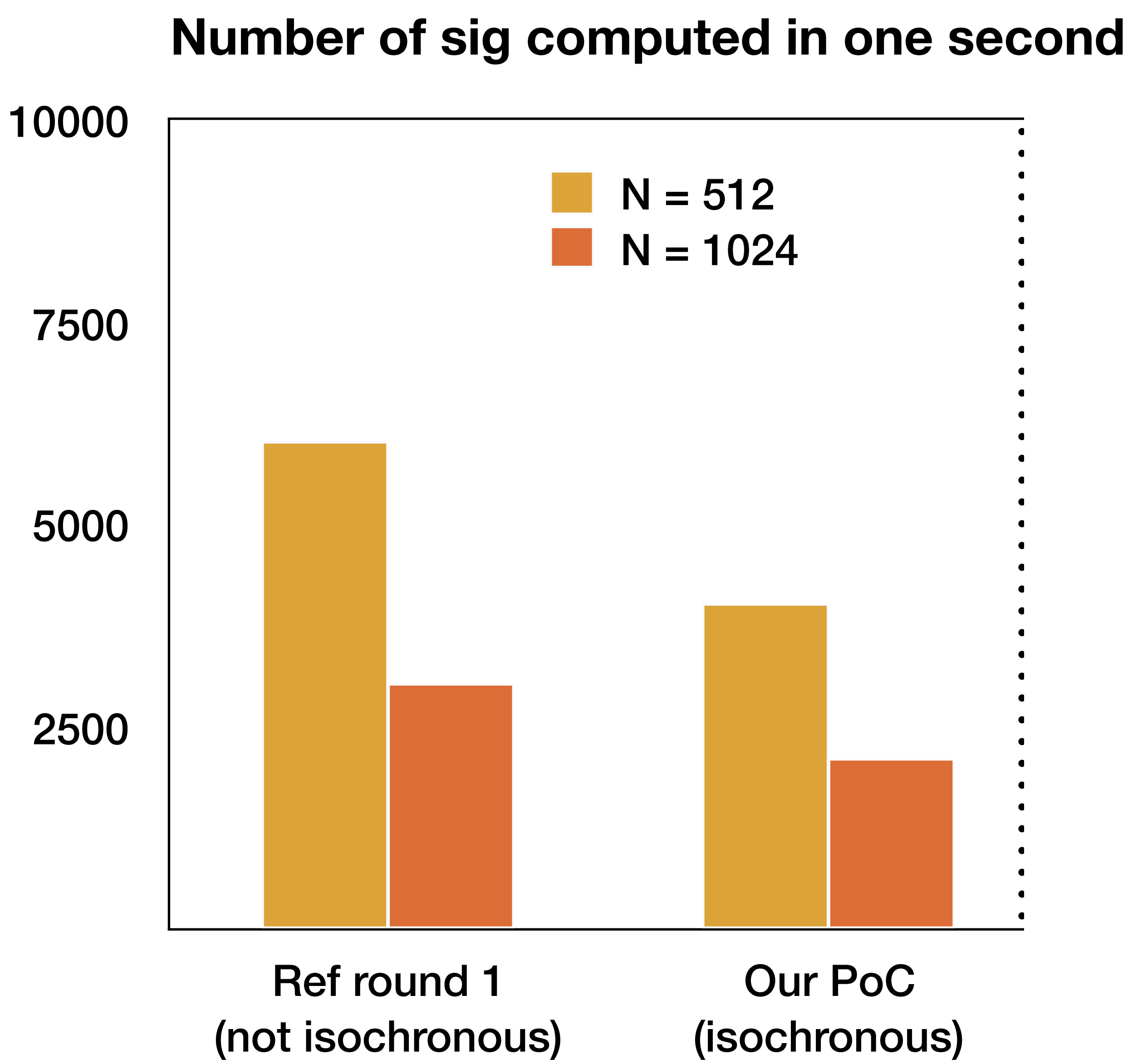
Ensure: $z \sim D_{\mathbb{Z},\sigma,\mu}$

1. $\boxed{z_0 \leftarrow \text{Basesampler()}}$
2. $b \leftarrow \{0,1\}$ uniformly
3. $z \leftarrow (2b - 1) \cdot z_0 + b$
4. $x \leftarrow -\dfrac{(z-\mu)^2}{2\sigma^2} + \dfrac{z_0^2}{2\sigma_0^2}$
5. Accept with probability $\boxed{\exp(x)}$
   Restart to 1. otherwise

Isochrony and portability modifications

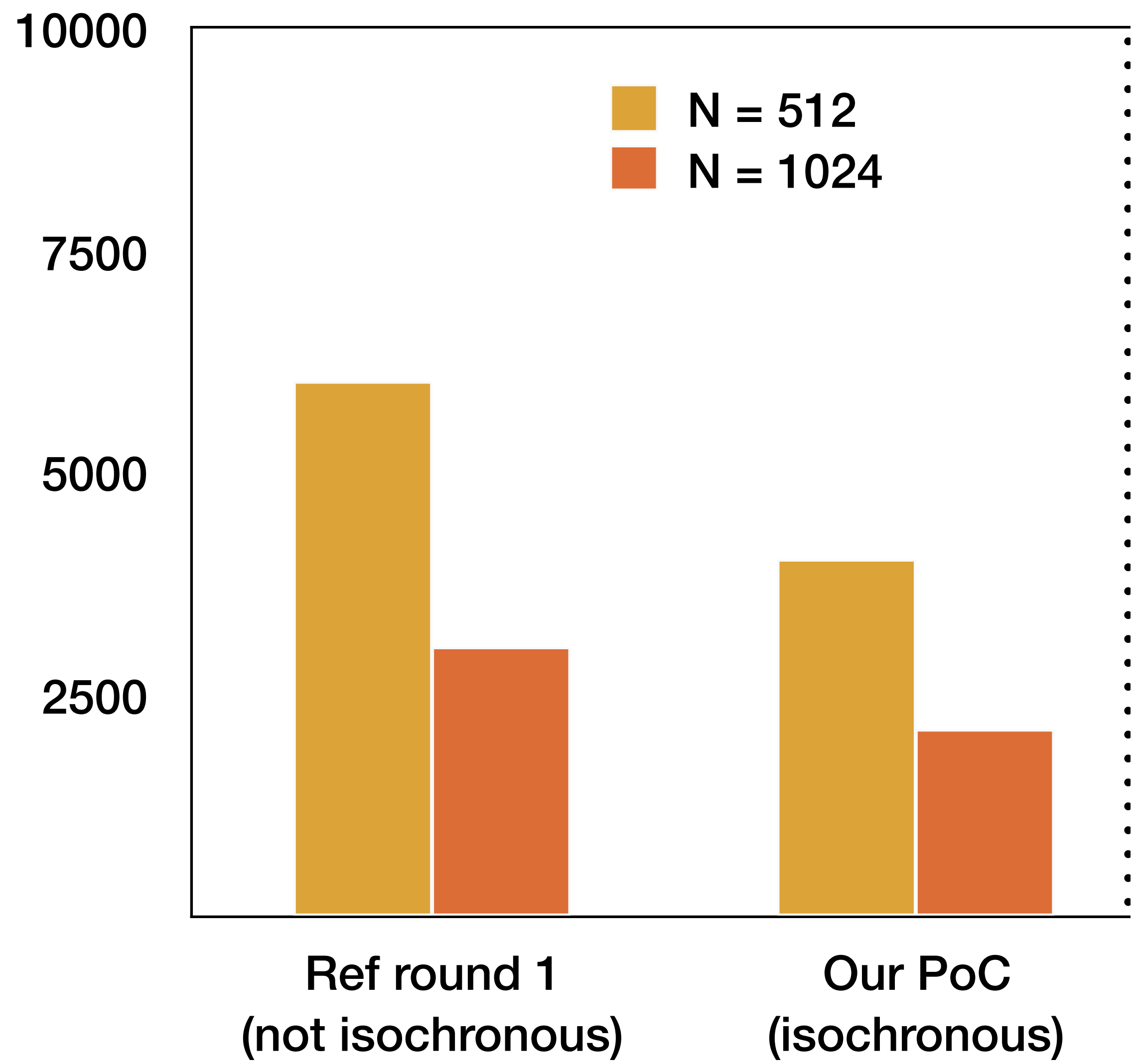
(1) Basesampler with a CDT
(2) Polynomial approximation for exp
(3) Make the number of iterations independent from the secret

Implementations



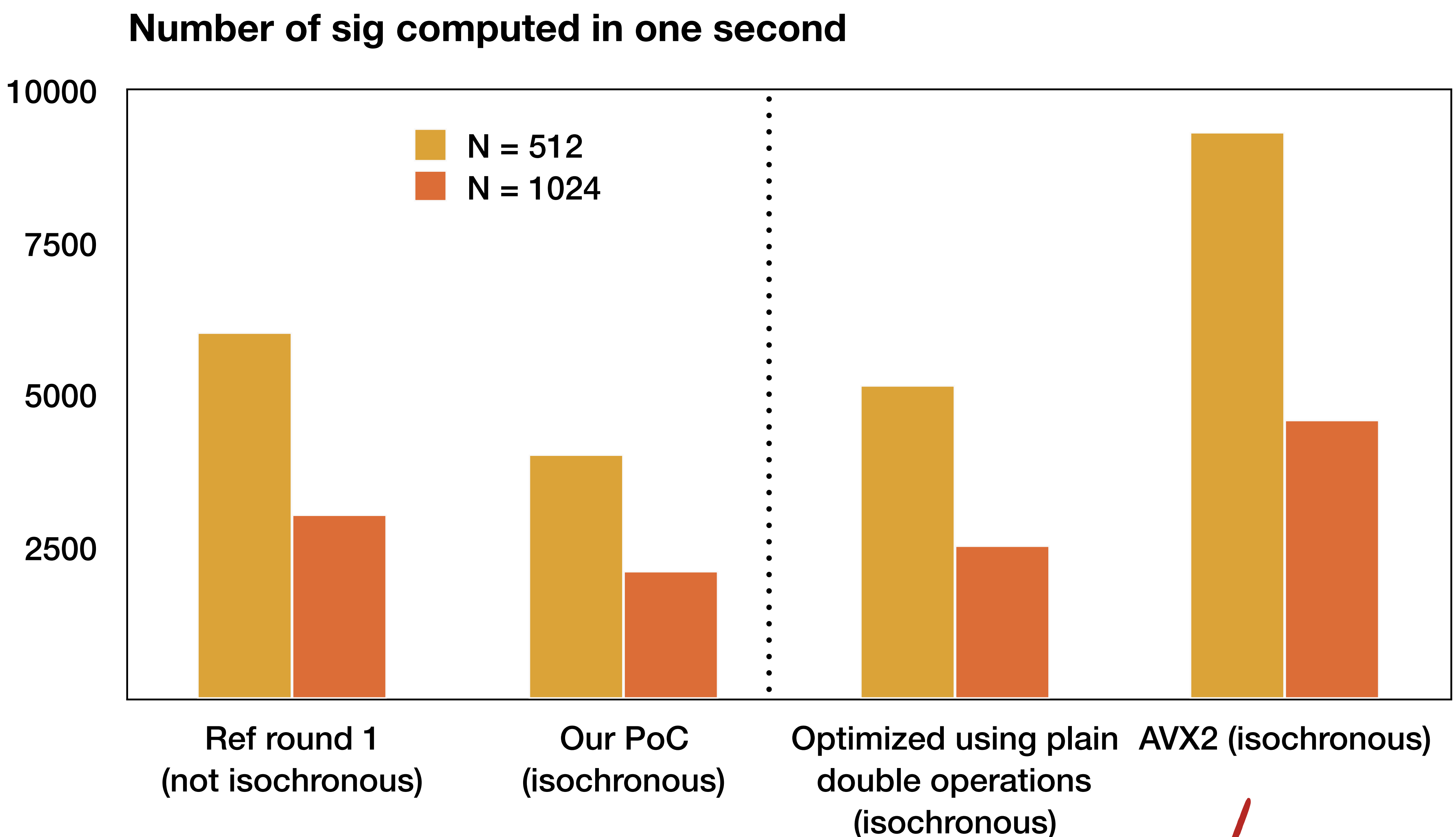
Implementations

Number of sig computed in one second



The performance loss for isochrony and portability
do not change the ranking of Falcon

Implementations



The performance loss for isochrony and portability do not change the ranking of Falcon

Recent implementations done by Thomas Pornin

See https://github.com/PQClean/PQClean/

And https://falcon-sign.info/falcon-impl-20190802.pdf

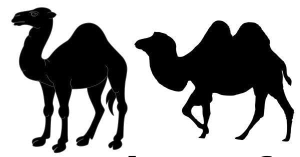
Implementations

Constant time and integers help Cortex M4 implementations

Falcon-512 (168 MHz)	Dynamic signatures (in milliseconds)	Memory (in bytes of extra RAM, not counting the key)
First M4 implementation (Oder et al. PQCRYPTO 2019)	479	50508
Recent Constant time and integers (Thomas Pornin) https://github.com/mupq/pqm4	243	36864

Conclusion

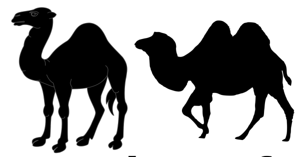
GALACTICS:



From transcendant functions to polynomials with integer coefficients

+ Simple rejection techniques for diminishing the CDT sizes

Conclusion

GALACTICS: 

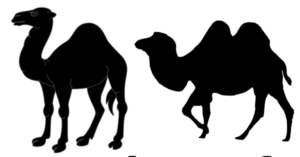
From transcendant functions to polynomials with integer coefficients

+ Simple rejection techniques for diminishing the CDT sizes

Helps to make BLISS isochronous and portable

Helps to make FALCON isochronous and portable

Conclusion

GALACTICS: 

From transcendant functions to polynomials with integer coefficients

+ Simple rejection techniques for diminishing the CDT sizes

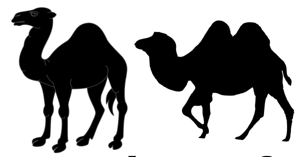
Helps to make BLISS isochronous and portable

Helps to make FALCON isochronous and portable

Gaussian distributions can now be

- ☑ Simpler to implement — portability
- ☑ Provably resistant to timing attacks

Conclusion

GALACTICS: 

From transcendant functions to polynomials with integer coefficients

+ Simple rejection techniques for diminishing the CDT sizes

Helps to make BLISS isochronous and portable

Helps to make FALCON isochronous and portable

Gaussian distributions can now be

- ☑ Simpler to implement — portability
- ☑ Provably resistant to timing attacks

Perspectives

Also helps for the **masking** countermeasure:

- ☑ See our CCS paper for the application to BLISS (without implementation)
- ☐ Ongoing work for Falcon

Questions ?