A journey in the land of (hash-and-sign) lattice-based signatures

Thomas Prest

Thales Communications & Security

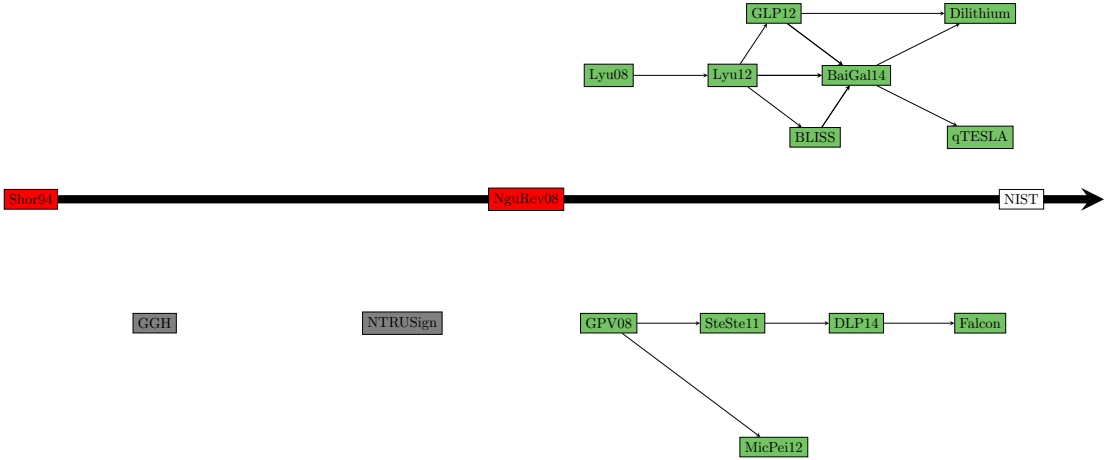
Lattice-based cryptography

Lattice-based cryptography in a nutshell [dPL17]:

> Every lattice-based cryptographic construction relies on the fact that when given a matrix $\mathbf{A}$ and a vector $\mathbf{y}$ over some ring R (such as $\mathbb{Z}_q$ or $\mathbb{Z}_q[X]/(X^d+1)$ with the usual addition and multiplication operations), it is hard to recover a vector $\mathbf{x}$ with small coefficients such that
>
> $$\mathbf{Ax} = \mathbf{y}.$$

Nice! Building signature schemes based on this principle should be easy, right?

A (non-exhaustive) timeline of lattice-based signature schemes



The early schemes

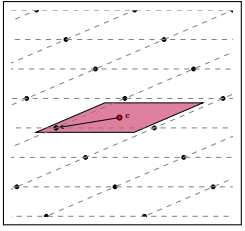
Early proposals of lattice-based signatures were GGH [GGH97] and NTRUSign [HHGP+03].

Alice		Bob
SK: $\mathbf{B} \in \mathbb{Z}_q^{m \times m}$ short s.t. $\mathbf{B}\mathbf{A}^t = \mathbf{0}$		PK: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$
Sign Compute $\mathbf{c}$ s.t. $\mathbf{c}\mathbf{A} = H(\mathsf{m})$ $\mathbf{v} \leftarrow \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$	$\xrightarrow{\mathbf{s}}$	Verify Accept iff - $H(\mathsf{m}) = \mathbf{s}\mathbf{A}$ - $\|\mathbf{s}\|$ is short

The early schemes

Early proposals of lattice-based signatures were GGH [GGH97] and NTRUSign [HHGP+03].

Alice		Bob
SK: $\mathbf{B} \in \mathbb{Z}_q^{m \times m}$ short s.t. $\mathbf{B}\mathbf{A}^t = \mathbf{0}$		PK: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$
Sign Compute $\mathbf{c}$ s.t. $\mathbf{c}\mathbf{A} = H(\mathbf{m})$ $\boxed{\mathbf{v} \leftarrow \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}}$ $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$	$\xrightarrow{\mathbf{s}}$	**Verify** Accept iff - $H(\mathbf{m}) = \mathbf{s}\mathbf{A}$ - $\|\mathbf{s}\|$ is short



Outputting $\mathbf{v} \leftarrow \lfloor \mathbf{c}\mathbf{B}^{-1} \rceil \mathbf{B}$ leaks the private key! [NR06]

Fiat-Shamir

The Fiat-Shamir heuristic: turns a zero-knowledge proof into a signature scheme.

Alice SK: $x \in \mathbb{Z}_q$		Bob PK: $g, h = g^x \in \mathbb{Z}_q$
$y \leftarrow \mathbb{Z}_q^\times$	$\xrightarrow{u = g^y}$	
		$c \leftarrow \mathbb{Z}_q^\times$
	$\xleftarrow{c}$	
$z \leftarrow y - cx$		
	$\xrightarrow{z}$	
		Accept iff $g^z h^c = u$

Fiat-Shamir

The Fiat-Shamir heuristic: turns a zero-knowledge proof into a signature scheme.

Alice		Bob
SK: $x \in \mathbb{Z}_q$		PK: $g, h = g^x \in \mathbb{Z}_q$
Sign		
$y \leftarrow \mathbb{Z}_q^\times$	$\xrightarrow{u = g^y}$	
$u \leftarrow g^y$		$c \leftarrow \mathbb{Z}_q^\times$
$c \leftarrow H(u, \mathsf{m})$	$\xleftarrow{c}$	
$z \leftarrow y - cx$		
	$\xrightarrow{c, z}$	**Verify**
		Accept iff $H(g^z h^c, \mathsf{m}) = c$

Fiat-Shamir

The Fiat-Shamir heuristic: turns a zero-knowledge proof into a signature scheme.

Alice		Bob
SK: $x \in \mathbb{Z}_q$		PK: $g, h = g^x \in \mathbb{Z}_q$
Sign		
$y \leftarrow \mathbb{Z}_q^\times$	$\xrightarrow{u = g^y}$	
$u \leftarrow g^y$		$c \leftarrow \mathbb{Z}_q^\times$
$c \leftarrow H(u, \mathsf{m})$	$\xleftarrow{c}$	
$z \leftarrow y - cx$		
	$\xrightarrow{c, z}$	**Verify**
		Accept iff $H(g^z h^c, \mathsf{m}) = c$

If one replaces:

- $g, x, h = g^x$ by matrices **A**, **S** short, **T** = **AS**
- $y, u = g^y$ by **y** short, **u** = **Ay**

One gets...

Fiat-Shamir over lattices using SIS

Alice		Bob
SK: $\mathbf{S} \in \mathbb{Z}_q^{m \times k}$ short		PK: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{T} = \mathbf{AS}$
Sign		
$\mathbf{y} \leftarrow \mathbb{Z}^m$ short		
$\mathbf{u} \leftarrow \mathbf{Ay}$		
$\mathbf{c} \leftarrow H(\mathbf{u}, \mathsf{m})$ short		
$\mathbf{z} \leftarrow \mathbf{y} + \mathbf{Sc}$		
	$\xrightarrow{\mathbf{c}, \mathbf{z}}$	**Verify**
		Accept iff
		- $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mathsf{m})$
		- $\|\mathbf{z}\|$ is short

Problem:

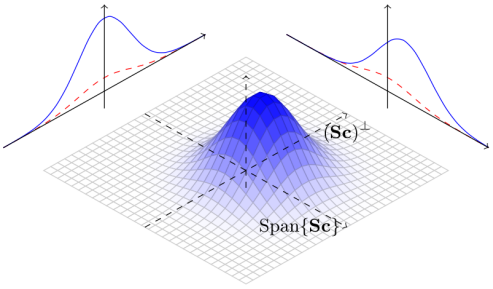
1. Proving the knowledge of a trapdoor (here $\mathbf{S}$) isn't hard.
2. It is hard to display this knowledge without leaking $\mathbf{S}$.
   Here, from several values $\mathbf{y}_i + \mathbf{Sz}_i$, one could recover $\mathbf{S}$ by statistic attacks.

Color code: blue $\Rightarrow$ small norm, burgundy $\Rightarrow$ big norm

Rejection Sampling

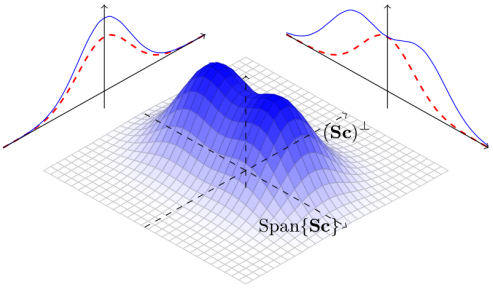
Idea: correct the start distribution f to obtain the desired distribution g

1. Sample $x \leftarrow f$
2. Accept x with probability $\frac{1}{M}\frac{g(x)}{f(x)}$, with $M \geq \max_{z \in Supp(g)} \frac{g(z)}{f(z)}$



[Lyu12]

$D_{\mathbb{Z}^m,\sigma,\mathbf{Sc}} \Rightarrow D_{\mathbb{Z}^m,\sigma,0}$



[DDLL13]

$\frac{1}{2}[D_{\mathbb{Z}^m,\sigma,\mathbf{Sc}} + D_{\mathbb{Z}^m,\sigma,-\mathbf{Sc}}] \Rightarrow D_{\mathbb{Z}^m,\sigma,0}$

Fiat-Shamir over lattices using SIS: end

Examples: [Lyu12, DDLL13].

Alice		Bob
SK: $\mathbf{S} \in \mathbb{Z}_q^{m \times k}$ short		PK: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{T} \leftarrow \mathbf{AS}$
Sign $\mathbf{y} \leftarrow \mathbb{Z}^m$ short $\mathbf{u} \leftarrow \mathbf{Ay}$ $\mathbf{c} \leftarrow H(\mathbf{u}, \mathbf{m})$ short $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{Sc}$ Rejection sampling over $\mathbf{z}$ so that $\mathbb{P}[\mathbf{z}] \propto D_{\mathbb{Z}^m, 0, \sigma}$	$\xrightarrow{\mathbf{c}, \mathbf{z}}$	**Verify** Accept iff - $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mathbf{m})$ - $\|\mathbf{z}\|$ is short

Color code: blue $\Rightarrow$ small norm, burgundy $\Rightarrow$ big norm

Fiat-Shamir over lattices using LWE

Examples: [Lyu09, GLP12, BG14, ABBD15].

Alice		Bob
SK: $\mathbf{S} \in \mathbb{Z}^{n \times n}, \mathbf{E} \in \mathbb{Z}^{m \times n}$ short		PK: $\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{T} \leftarrow \mathbf{AS} + \mathbf{E}$
Sign $\mathbf{y} \leftarrow \mathbb{Z}^m$ short $\mathbf{u} \leftarrow \mathbf{Ay}$ $\mathbf{c} \leftarrow H(\mathsf{msb}(\mathbf{u}), \mathsf{m})$ short $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{Sc}$ Rejection sampling over $\mathbf{z}$	$\xrightarrow{\mathbf{c}, \mathbf{z}}$	**Verify** Accept iff - $\mathbf{c} = H(\mathsf{msb}(\mathbf{Az} - \mathbf{Tc}), \mathsf{m})$ - $\|\mathbf{z}\|$ is short

Color code: blue $\Rightarrow$ small norm, burgundy $\Rightarrow$ big norm

Conclusion for Fiat-Shamir schemes

- Central idea: Fiat-Shamir with aborts
- Main differences between schemes:
  - The class of lattices used: standard, module, ring, NTRU, etc.
  - Distributions used for signing: uniform, Gaussian, bimodal Gaussian, hybrid, etc.

Provably secure hash-and-sign over lattices

Theoretical framework formalized in [GPV08].

Alice		Bob
SK: $\mathbf{B} \in \mathbb{Z}_q^{m \times m}$ short s.t. $\mathbf{B}\mathbf{A}^t = \mathbf{0}$		PK: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$
Sign Compute $\mathbf{c}$ s.t. $\mathbf{c}\mathbf{A} = H(\mathsf{m})$ Sample $\mathbf{v} \in \mathbb{Z}^m \mathbf{B}$ close to $\mathbf{c}$ $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$	$\xrightarrow{\mathbf{s}}$	**Verify** Accept iff - $H(\mathsf{m}) = \mathbf{s}\mathbf{A}$ - $\|\mathbf{s}\|$ is short

(!) $\mathbf{v}$ is securely sampled using a *trapdoor sampler*.

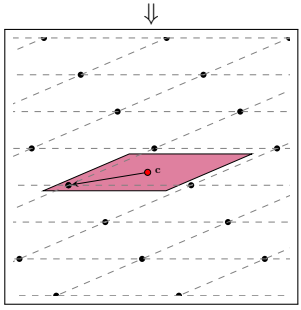
Trapdoor samplers: compute $\mathbf{z} \in \mathbb{Z}^n$ such that $\|(\mathbf{z} - \mathbf{t})\mathbf{B}\|$ is small

Approach 1

Algorithm 1 Round-off

Require: $\mathbf{B}$

1: **for** $j = n, \ldots, 1$ **do**
2: $\quad z_j \leftarrow \lfloor t_j \rceil$
3: **return** $\mathbf{z}$

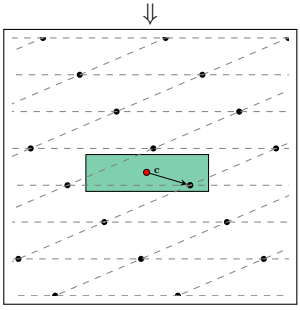


Approach 2

Algorithm 2 Nearest plane

Require: $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$

1: **for** $j = n, \ldots, 1$ **do**
2: $\quad \bar{t}_j \leftarrow t_j + \sum_{i>j}(t_i - z_i)L_{ij}$
3: $\quad z_j \leftarrow \lfloor \bar{t}_j \rceil$
4: **return** $\mathbf{z}$



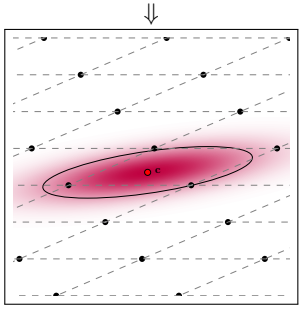
Trapdoor samplers: compute $\mathbf{z} \in \mathbb{Z}^n$ such that $\|(\mathbf{z} - \mathbf{t})\mathbf{B}\|$ is small

Approach 1

Algorithm 3 Randomized round-off

Require: $\mathbf{B}$

1: **for** $j = n, \ldots, 1$ **do**
2: $\quad z_j \leftarrow \lfloor t_j \rceil_\sigma$
3: **return** $\mathbf{z}$

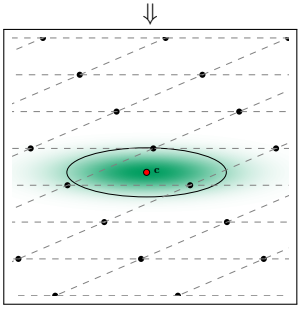


Approach 2

Algorithm 4 Randomized nearest plane

Require: $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$

1: **for** $j = n, \ldots, 1$ **do**
2: $\quad \bar{t}_j \leftarrow t_j + \sum_{i>j}(t_i - z_i)L_{ij}$
3: $\quad z_j \leftarrow \lfloor \bar{t}_j \rceil_\sigma$
4: **return** $\mathbf{z}$



Trapdoor samplers: compute $\mathbf{z} \in \mathbb{Z}^n$ such that $\|(\mathbf{z} - \mathbf{t})\mathbf{B}\|$ is small

Approach 1

Algorithm 5 Peikert's sampler

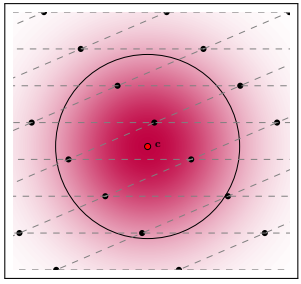
Require: $\mathbf{B}$

$\mathbf{x} \leftarrow \mathbf{C} \cdot \lfloor \mathbf{0} \rceil_\sigma$

for $j = n, \ldots, 1$ **do**

  $z_j \leftarrow \lfloor t_j - x_j \rceil_\sigma$

return $\mathbf{z}$



Approach 2

Algorithm 6 Klein's sampler

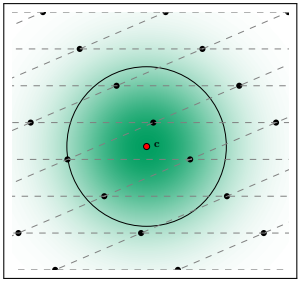
Require: $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$

for $j = n, \ldots, 1$ **do**

  $\bar{t}_j \leftarrow t_j + \sum_{i>j}(t_i - z_i)L_{ij}$

  $z_j \leftarrow \lfloor \bar{t}_j \rceil_{\sigma / \|\tilde{\mathbf{b}}_j\|}$

return $\mathbf{z}$



The GPV framework over NTRU lattices

Instantiation of the GPV framework over NTRU lattices [SS11, DLP14]

Simply take $\mathbf{A} = \left[\begin{array}{c|c} 1 & h \end{array}\right]$ and $\mathbf{B} = \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right]$, where:

$$\begin{array}{ccc} fG - gF & = & q \mod (x^n + 1) \\ h & = & gf^{-1} \mod (q, x^n + 1) \end{array} \quad (1)$$

Shortcomings:

1. Cumbersome key generation (slow, requires a lot of memory)
2. Signature generation is either:
   - with Klein's sampler, secure but slow: $O(n^2)$
   - with Peikert's sampler, less secure but fast: $O(n \log n)$
3. Use of floating-point arithmetic (FPA) $\Rightarrow$ which precision?
4. Parameters may be improved (?)

The rest of this talk: addressing these shortcomings.
The techniques also apply to the IBE of [DLP14].

I - The issue of the key generation

Key generation at a high-level:

1. generate small $f, g \in \mathbb{Z}[x]/(x^n + 1)$
2. solve the NTRU equation, i.e. find $F, G \in \mathbb{Z}[x]/(x^n + 1)$ such that

$$fG - gF = 1 \bmod (x^n + 1) \tag{2}$$

3. do simple stuff

Existing methods for step 2 were very cumbersome in time (∼1 second), memory (∼3 Mbytes) and implementation efforts (depends on who implements it).
Can we do better?

I - Exploiting the tower of rings structure

We have the following tower of rings:

$$\mathbb{Z} \subseteq \mathbb{Z}[x]/(x^2+1) \subseteq \cdots \subseteq \mathbb{Z}[x]/(x^{n/2}+1) \subseteq \mathbb{Z}[x]/(x^n+1)$$

and the field norm allows to "navigate" along this tower!

Let $\mathcal{Q}_n = \mathbb{Q}[x]/(x^n+1)$. The field norm N is defined by:

$$\begin{array}{rccc} \mathsf{N} & : & \mathcal{Q}_n & \to & \mathcal{Q}_{n/2} \\ & & f & \to & ff^\times \end{array} \tag{3}$$

where $f^\times$ denotes the Galois conjugate of f for the field extension $\mathcal{Q}_n/\mathcal{Q}_{n/2}$...
Or more simply in our case, $f^\times(x) = f(-x)$.

Fun fact: if we have this relationship over $\mathbb{Z}[x]/(x^{n/2}+1)$:

$$\mathsf{N}(f)G' - \mathsf{N}(g)F' = 1 \tag{4}$$

for some F', G', then we have this relationship over $\mathbb{Z}[x]/(x^n+1)$:

$$f(f^\times G') - g(g^\times F') = 1 \tag{5}$$

I - Outline of the new key generation algorithm

$$\begin{array}{ccc} \mathbb{Z}[x]/(x^n+1) & \ni & f, g \\ \cup\!\!\!| \\ \mathbb{Z}[x]/(x^{n/2}+1) \\ \cup\!\!\!| \\ \mathbb{Z}[x]/(x^{n/4}+1) \\ \cup\!\!\!| \\ \vdots \\ \cup\!\!\!| \\ \mathbb{Z} \end{array} \tag{6}$$

I - Outline of the new key generation algorithm

$$\begin{array}{ccc} \mathbb{Z}[x]/(x^n+1) & \ni & f, g \\ \cup\!\!\!| & & \downarrow \\ \mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) \\ \cup\!\!\!| & & \\ \mathbb{Z}[x]/(x^{n/4}+1) & & \\ \cup\!\!\!| & & \\ \vdots & & \\ \cup\!\!\!| & & \\ \mathbb{Z} & & \end{array} \tag{6}$$

I - Outline of the new key generation algorithm

$$\begin{array}{ccc} \mathbb{Z}[x]/(x^n+1) & \ni & f, g \\ \cup\!\!\!| & & \downarrow \\ \mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) \\ \cup\!\!\!| & & \downarrow \\ \mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) \\ \cup\!\!\!| & & \\ \vdots & & \\ \cup\!\!\!| & & \\ \mathbb{Z} & & \end{array} \tag{6}$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g \\
\cup\!\!\!| & & \downarrow \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) \\
\cup\!\!\!| & & \downarrow \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) \\
\cup\!\!\!| & & \downarrow \\
\vdots & \vdots & \vdots \\
\cup\!\!\!| & & \\
\mathbb{Z} & &
\end{array}
\tag{6}
$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g \\
\cup\!\!\downarrow & & \downarrow \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) \\
\cup\!\!\downarrow & & \downarrow \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) \\
\cup\!\!\downarrow & & \downarrow \\
\vdots & \vdots & \vdots \\
\cup\!\!\downarrow & & \downarrow \\
\mathbb{Z} & \ni & \mathsf{N}^\ell(f), \mathsf{N}^\ell(g)
\end{array}
\tag{6}
$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g & & \\
\cup\downarrow & & \downarrow & & \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) & & \\
\cup\downarrow & & \downarrow & & \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) & & \\
\cup\downarrow & & \downarrow & & \\
\vdots & \vdots & \vdots & & \\
\cup\downarrow & & \downarrow & & \\
\mathbb{Z} & \ni & \mathsf{N}^\ell(f), \mathsf{N}^\ell(g) & \rightarrow & F^{[\ell]}, G^{[\ell]}
\end{array}
\tag{6}
$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g & & \\
\cup\!\!\downarrow & & \downarrow & & \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) & & \\
\cup\!\!\downarrow & & \downarrow & & \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) & & \\
\cup\!\!\downarrow & & \downarrow & & \\
\vdots & \vdots & \vdots & & \vdots \\
\cup\!\!\downarrow & & \downarrow & & \uparrow \\
\mathbb{Z} & \ni & \mathsf{N}^\ell(f), \mathsf{N}^\ell(g) & \rightarrow & F^{[\ell]}, G^{[\ell]}
\end{array}
\tag{6}
$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g & & \\
\cup\!\!\downarrow & & \downarrow & & \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) & & \\
\cup\!\!\downarrow & & \downarrow & & \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) & \rightarrow & F^{[2]}, G^{[2]} \\
\cup\!\!\downarrow & & \downarrow & & \uparrow \\
\vdots & \vdots & \vdots & & \vdots \\
\cup\!\!\downarrow & & \downarrow & & \uparrow \\
\mathbb{Z} & \ni & \mathsf{N}^\ell(f), \mathsf{N}^\ell(g) & \rightarrow & F^{[\ell]}, G^{[\ell]}
\end{array}
\tag{6}
$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g & & \\
\cup\!\!\!| & & \downarrow & & \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) & \to & F^{[1]}, G^{[1]} \\
\cup\!\!\!| & & \downarrow & & \uparrow \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) & \to & F^{[2]}, G^{[2]} \\
\cup\!\!\!| & & \downarrow & & \uparrow \\
\vdots & \vdots & \vdots & & \vdots \\
\cup\!\!\!| & & \downarrow & & \uparrow \\
\mathbb{Z} & \ni & \mathsf{N}^\ell(f), \mathsf{N}^\ell(g) & \to & F^{[\ell]}, G^{[\ell]}
\end{array}
\tag{6}
$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g & \to & F, G \\
\cup\!\!\!| & & \downarrow & & \uparrow \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) & \to & F^{[1]}, G^{[1]} \\
\cup\!\!\!| & & \downarrow & & \uparrow \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) & \to & F^{[2]}, G^{[2]} \\
\cup\!\!\!| & & \downarrow & & \uparrow \\
\vdots & \vdots & \vdots & & \vdots \\
\cup\!\!\!| & & \downarrow & & \uparrow \\
\mathbb{Z} & \ni & \mathsf{N}^\ell(f), \mathsf{N}^\ell(g) & \to & F^{[\ell]}, G^{[\ell]}
\end{array}
\tag{6}
$$

I - Outline of the new key generation algorithm

$$
\begin{array}{ccccc}
\mathbb{Z}[x]/(x^n+1) & \ni & f, g & \to & F, G \\
\cup\downarrow & & \downarrow & & \uparrow \\
\mathbb{Z}[x]/(x^{n/2}+1) & \ni & \mathsf{N}(f), \mathsf{N}(g) & \to & F^{[1]}, G^{[1]} \\
\cup\downarrow & & \downarrow & & \uparrow \\
\mathbb{Z}[x]/(x^{n/4}+1) & \ni & \mathsf{N}^2(f), \mathsf{N}^2(g) & \to & F^{[2]}, G^{[2]} \\
\cup\downarrow & & \downarrow & & \uparrow \\
\vdots & \vdots & \vdots & & \vdots \\
\cup\downarrow & & \downarrow & & \uparrow \\
\mathbb{Z} & \ni & \mathsf{N}^\ell(f), \mathsf{N}^\ell(g) & \to & F^{[\ell]}, G^{[\ell]}
\end{array}
\tag{6}
$$

At each lower level:

- The coefficients grow (in bitsize) by a factor 2...
- ... but the number of coefficients is divided by 2.

We gain in practice:

- a factor 100 in memory consumption (30KBytes $\Rightarrow$ key generation on embedded devices!)
- a factor 10 in time

Extends the techniques of "overstretched NTRU attacks" [ABD16, CJL16, KF17], but in a constructive way!

II - Fast Fourier Sampling [DP16]

Klein's sampler interprets $\mathbb{Q}[x]/(x^n+1)$ as a $\mathbb{Q}$-linear space of dimension n

$$\mathbf{B} = \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right] \in \mathbb{Z}[x]/(x^n+1)^{2\times 2} \qquad \Rightarrow \qquad \left[\begin{array}{c|c} \mathcal{C}(g) & -\mathcal{C}(f) \\ \hline \mathcal{C}(G) & -\mathcal{C}(F) \end{array}\right] \in \mathbb{Z}^{2n\times 2n}$$

$\Rightarrow$ completely ignores the rich algebraic structure of $\mathbb{Q}[x]/(x^n+1)$!

II - Fast Fourier Sampling [DP16]

Klein's sampler interprets $\mathbb{Q}[x]/(x^n+1)$ as a $\mathbb{Q}$-linear space of dimension n

$$\mathbf{B} = \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right] \in \mathbb{Z}[x]/(x^n+1)^{2\times 2} \qquad \Rightarrow \qquad \left[\begin{array}{c|c} \mathcal{C}(g) & -\mathcal{C}(f) \\ \hline \mathcal{C}(G) & -\mathcal{C}(F) \end{array}\right] \in \mathbb{Z}^{2n\times 2n}$$

$\Rightarrow$ completely ignores the rich algebraic structure of $\mathbb{Q}[x]/(x^n+1)$!

Splitting polynomials between their odd and even coefficients yields this chain of space isomorphisms:

$$\mathbb{Q}^n \cong (\mathbb{Q}[x]/(x^2+1))^{n/2} \cong \ldots \cong (\mathbb{Q}[x]/(x^{n/2}+1))^2 \cong \mathbb{Q}[x]/(x^n+1) \tag{7}$$

We will take advantage of this chain of isomorphisms to devise a *recursive* variant of Klein's sampler.

We reformulate the problem that our signature algorithm solves. Given:

- a challenge $t_0, t_1 \in \mathbb{Q}[x]/(x^n + 1)$,
- the secret basis $\mathbf{B} \in \mathbb{Z}[x]/(x^n + 1)^{2\times 2}$ (and its GSO),

sample $z_0, z_1 \in \mathbb{Z}[x]/(x^n + 1)$ such that $(z_0, z_1) \cdot \mathbf{B}$ is close to $(t_0, t_1) \cdot \mathbf{B}$.

What if we first sample z_1 so that $(0, z_1) \cdot \mathbf{B}$ is close to $(0, t_1) \cdot \mathbf{B}$, and then adaptively sample z_0?

- Sure, this is a generalization of Klein's sampler over the field $\mathbb{Q}[x]/(x^n + 1)$ instead of $\mathbb{Q}$.

We reformulate the problem that our signature algorithm solves. Given:

- a challenge $t_0, t_1 \in \mathbb{Q}[x]/(x^n + 1)$,
- the secret basis $\mathbf{B} \in \mathbb{Z}[x]/(x^n + 1)^{2\times 2}$ (and its GSO),

sample $z_0, z_1 \in \mathbb{Z}[x]/(x^n + 1)$ such that $(z_0, z_1) \cdot \mathbf{B}$ is close to $(t_0, t_1) \cdot \mathbf{B}$.

What if we first sample z_1 so that $(0, z_1) \cdot \mathbf{B}$ is close to $(0, t_1) \cdot \mathbf{B}$, and then adaptively sample z_0?

- Sure, this is a generalization of Klein's sampler over the field $\mathbb{Q}[x]/(x^n + 1)$ instead of $\mathbb{Q}$.

Problem: sampling z_1 boils down to making $z_1 g$ close to $t_1 g$ for a given $g \in \mathbb{Q}[x]/(x^n + 1)$.
How to do that optimally without completely breaking the structure?

- Break $\mathbb{Q}[x]/(x^n + 1)$ into $\mathbb{Q}[x]/(x^{n/2} + 1)^2$!
- By splitting in odds/even coefficients, z_1, t_1 can be seen as elements of $\mathbb{Q}[x]/(x^{n/2} + 1)^2$.
- Similarly, g can be seen as an element of $\mathbb{Q}[x]/(x^{n/2} + 1)^{2\times 2}$ (because it is actually an endomorphism).

We reformulate the problem that our signature algorithm solves. Given:

- a challenge $t_0, t_1 \in \mathbb{Q}[x]/(x^n + 1)$,
- the secret basis $\mathbf{B} \in \mathbb{Z}[x]/(x^n + 1)^{2\times 2}$ (and its GSO),

sample $z_0, z_1 \in \mathbb{Z}[x]/(x^n + 1)$ such that $(z_0, z_1) \cdot \mathbf{B}$ is close to $(t_0, t_1) \cdot \mathbf{B}$.

What if we first sample z_1 so that $(0, z_1) \cdot \mathbf{B}$ is close to $(0, t_1) \cdot \mathbf{B}$, and then adaptively sample z_0?

- Sure, this is a generalization of Klein's sampler over the field $\mathbb{Q}[x]/(x^n + 1)$ instead of $\mathbb{Q}$.

Problem: sampling z_1 boils down to making $z_1 g$ close to $t_1 g$ for a given $g \in \mathbb{Q}[x]/(x^n + 1)$.
How to do that optimally without completely breaking the structure?

- Break $\mathbb{Q}[x]/(x^n + 1)$ into $\mathbb{Q}[x]/(x^{n/2} + 1)^2$!
- By splitting in odds/even coefficients, z_1, t_1 can be seen as elements of $\mathbb{Q}[x]/(x^{n/2} + 1)^2$.
- Similarly, g can be seen as an element of $\mathbb{Q}[x]/(x^{n/2} + 1)^{2\times 2}$ (because it is actually an endomorphism).

We now are in a situation identical to the beginning, but over a smaller subfield $\Rightarrow$ recursion!
Now we can find vectors as short (or close) as Klein's sampler would, but in time $O(n \log n)$.

IV - Improving the required precision with the Rényi divergence

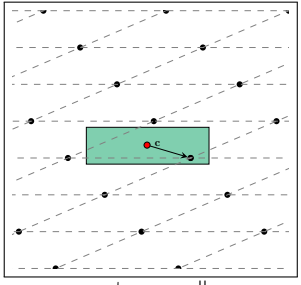
What is the proper way to evaluate the required precision of the FPA operations?

A tool which was extremely useful to us: the Rényi divergence [Ré61].

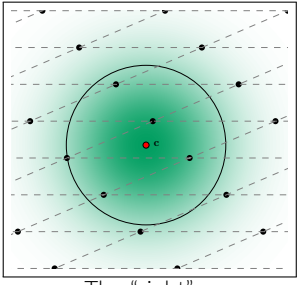
- Statistical distance analysis $\Rightarrow$ FPA operations require a precision of $\lambda + \text{polylog}(n, \ldots)$ bits.
- Rényi divergence analysis $\Rightarrow$ FPA operations require a precision of $\log_2(q_s)/2 + \text{polylog}(n, \ldots)$ bits, where q_s is the number of public queries

In NIST's CFP, $\log_2(q_s) \leq 64 \Rightarrow$ taking a precision of 53 bits is (provably!) sufficient.

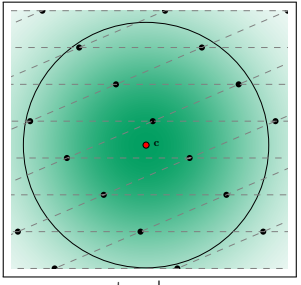
IV - Improving the standard deviation with the Rényi divergence



σ too small



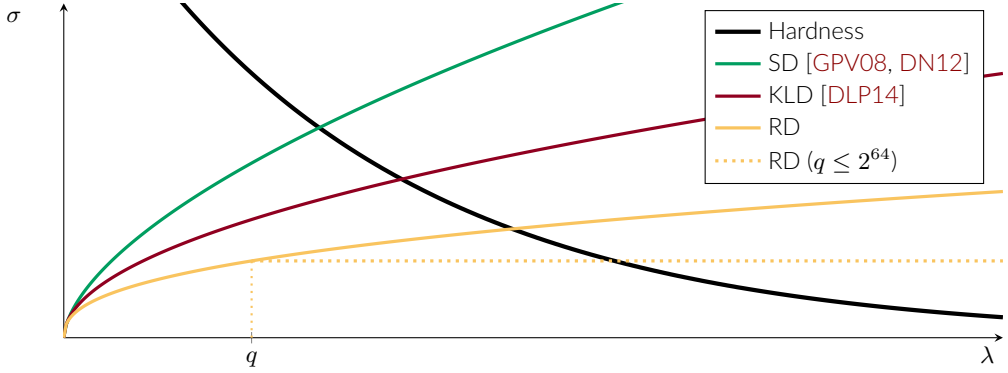
The "right" σ



σ too large

1. σ too large $\Rightarrow$ the trapdoor sampler is useless in a cryptographic context.
2. σ too small $\Rightarrow$ the trapdoor sampler does not behave like a perfect Gaussian.

IV - Improving the standard deviation with the Rényi divergence



The adequate value for σ is at the intersection of the hardness curve (constraint ❶) and the SD/KLD/RD curve (constraint ❷).

- A Rényi divergence-based analysis proves to be much more efficient than an SD/KLD-based one.
- Interesting fact: in practice, σ is not conditioned by λ but by q.

In practice, we gain about 30 bits of security (compared to the SD).

Falcon

The product of all these improvements is the signature scheme Falcon, which stands for

Fast Fourier **l**attice-based **co**mpact signatures over **N**TRU

Reference implementation on an Intel® Core® i7-6567U @3.3 GHz (all sizes in bytes):

NIST lvl	degree	keygen (ms)	keygen (RAM)	sign/s	vrfy/s	pub length	sig length
1	512	6.98	14336	6081.9	37175.3	897	617.38
2-3	768	12.69	27648	3547.9	20637.7	1441	993.91
4-5	1024	19.64	28672	3072.5	17697.4	1793	1233.29

The website:

`https://falcon-sign.info`

Authors: Fouque, Hoffstein, Kirchner, Lyubashevsky, Pornin, Prest, Ricosset, Seiler, Whyte, Zhang.

Fiat-Shamir vs Hash-and-sign

Fiat-Shamir schemes:

- Hard security proofs in the QROM
- Easy to protect against SCA ($\Rightarrow$ large signatures)
- Can avoid floating-point arithmetic

Hash-and-sign schemes:

- Easy security proof in the QROM [BDF+11]
- Hard to protect against SCA
- Small signatures
- Currently uses floating-point arithmetic
- Allows advanced constructions (ABE, (H)IBE, etc.)

Conclusion

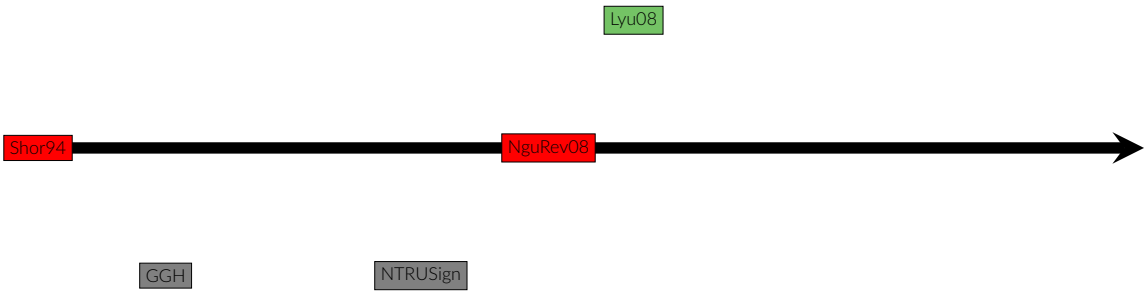
Conclusion

GGH

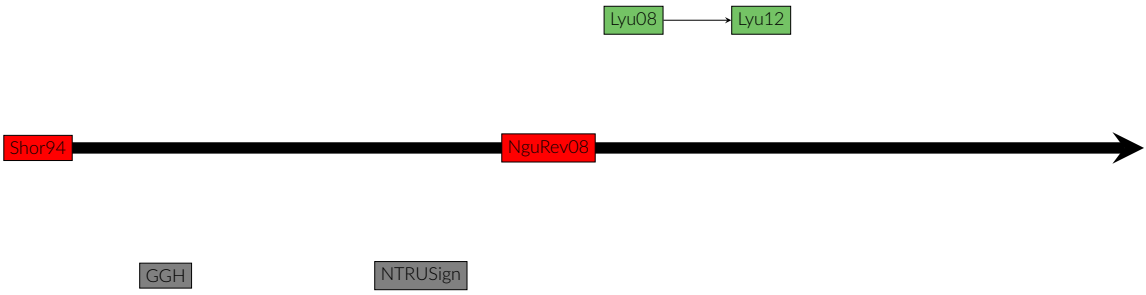
Conclusion

Conclusion

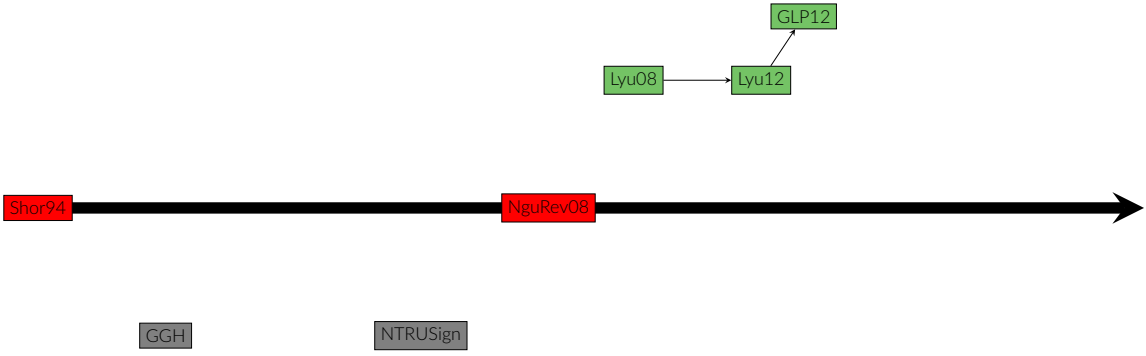
Conclusion



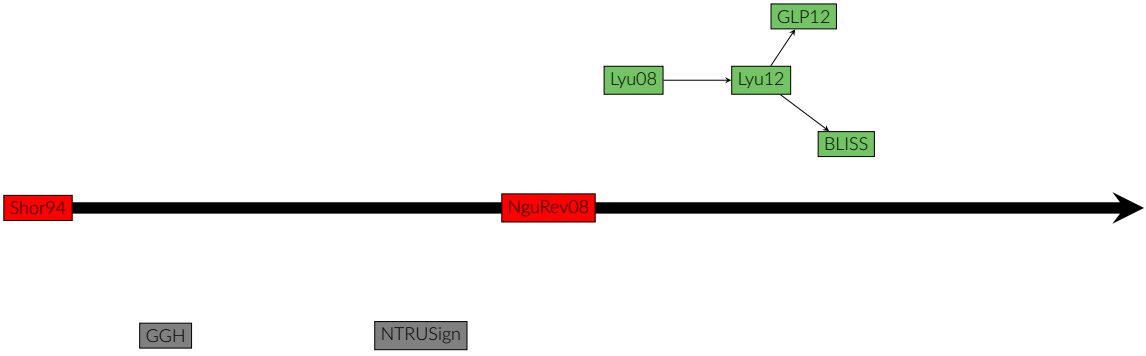
Conclusion



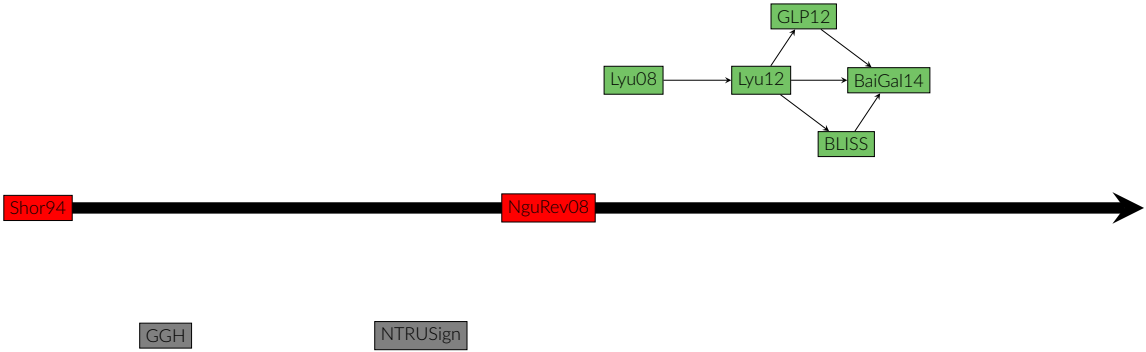
Conclusion



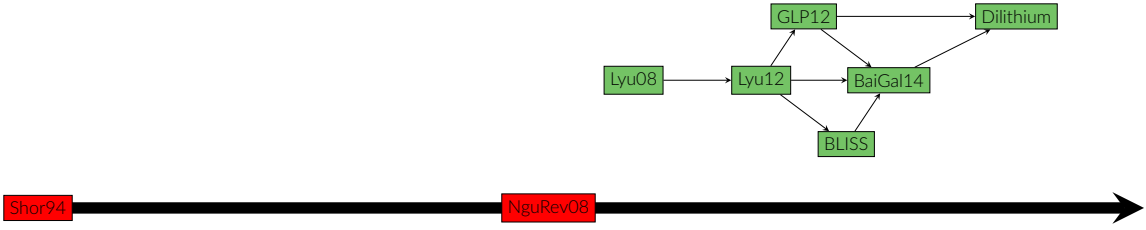
Conclusion



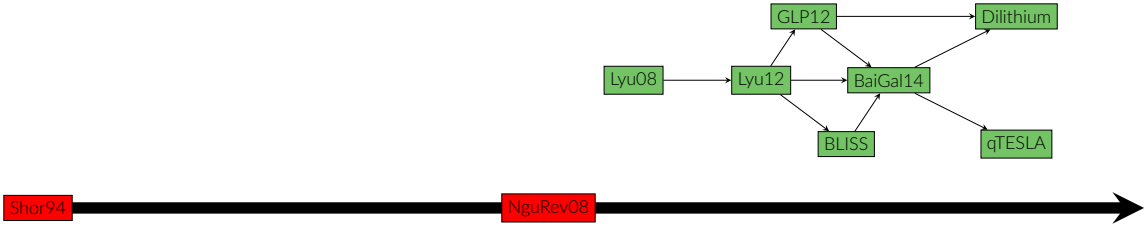
Conclusion


GLP12
Lyu08
Lyu12
BaiGal14
BLISS
Shor94
NguRev08
GGH
NTRUSign

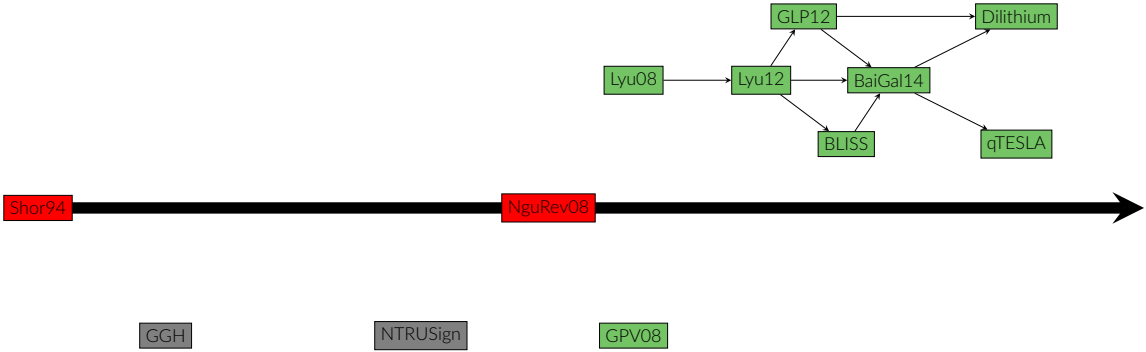
Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
BLISS
Shor94
NguRev08
GGH
NTRUSign

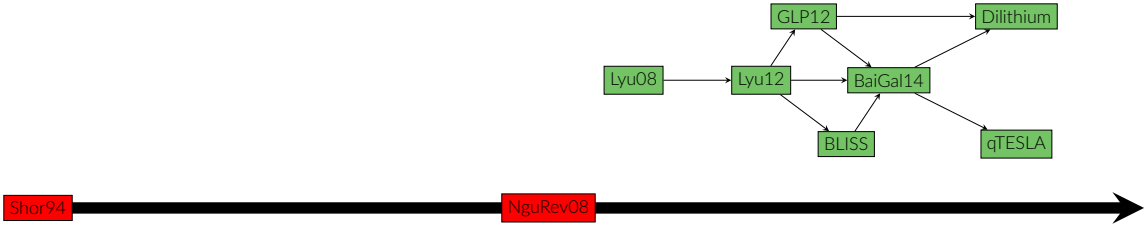
Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
BLISS
qTESLA
Shor94
NguRev08
GGH
NTRUSign

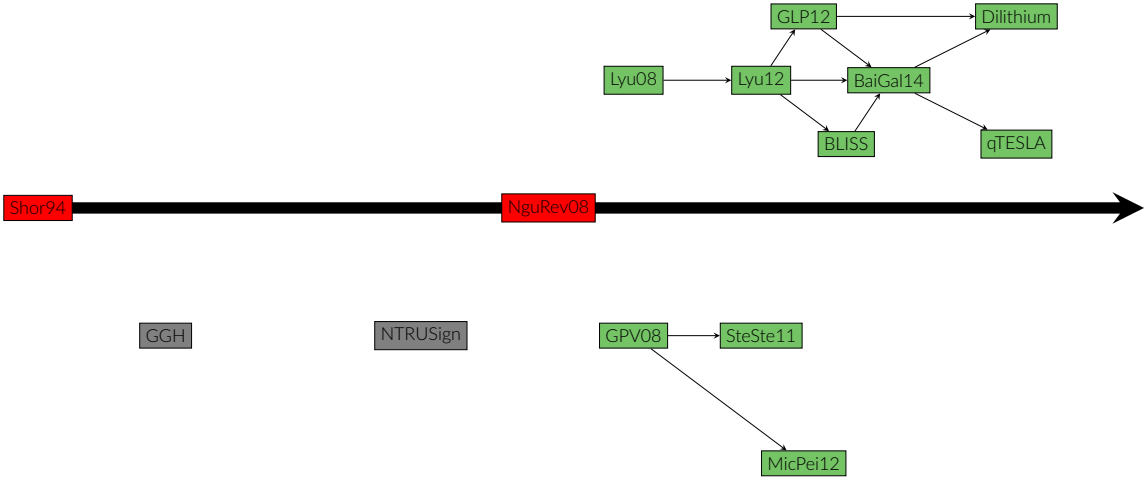
Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
BLISS
qTESLA
Shor94
NguRev08
GGH
NTRUSign
GPV08

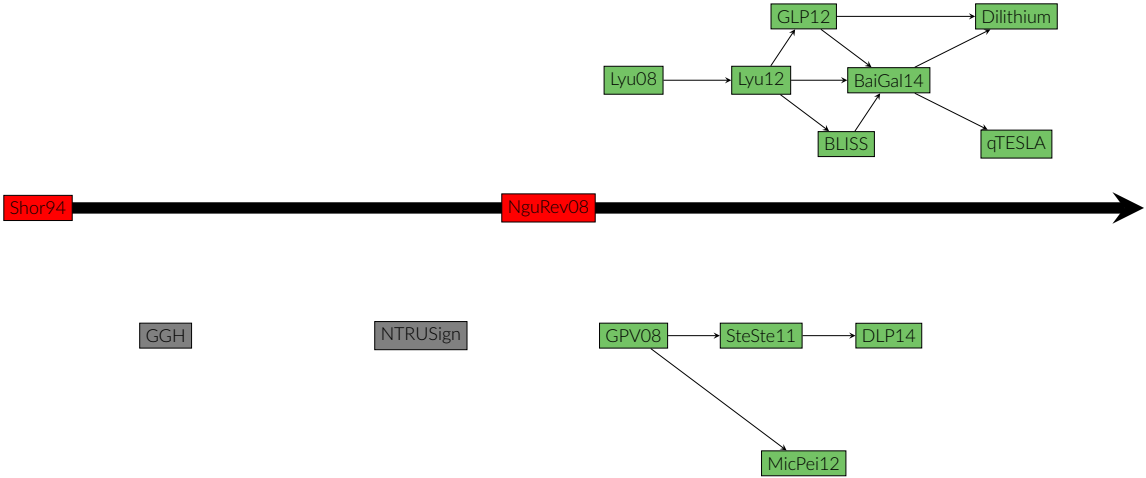
Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
BLISS
qTESLA
Shor94
NguRev08
GGH
NTRUSign
GPV08
SteSte11

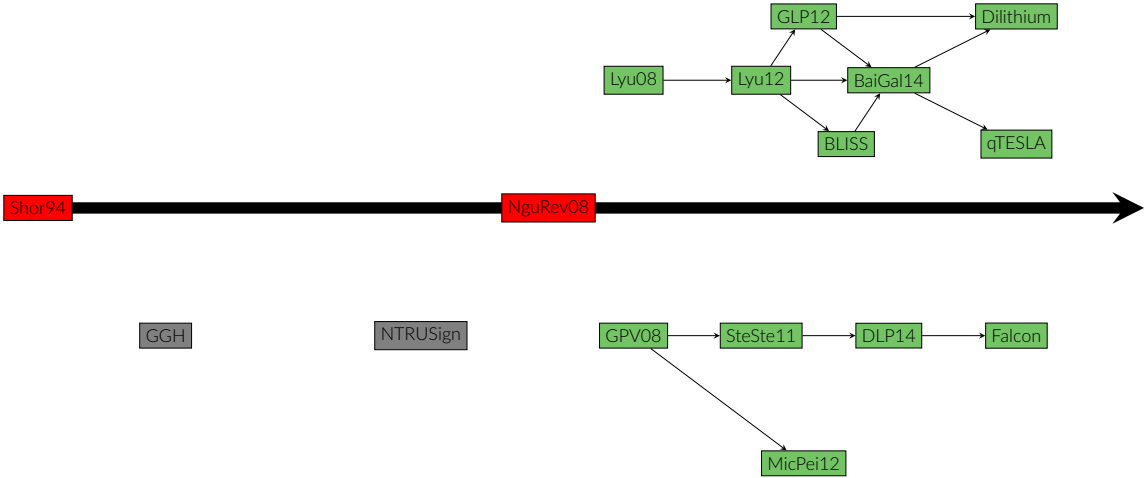
Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
BLISS
qTESLA
Shor94
NguRev08
GGH
NTRUSign
GPV08
SteSte11
MicPei12

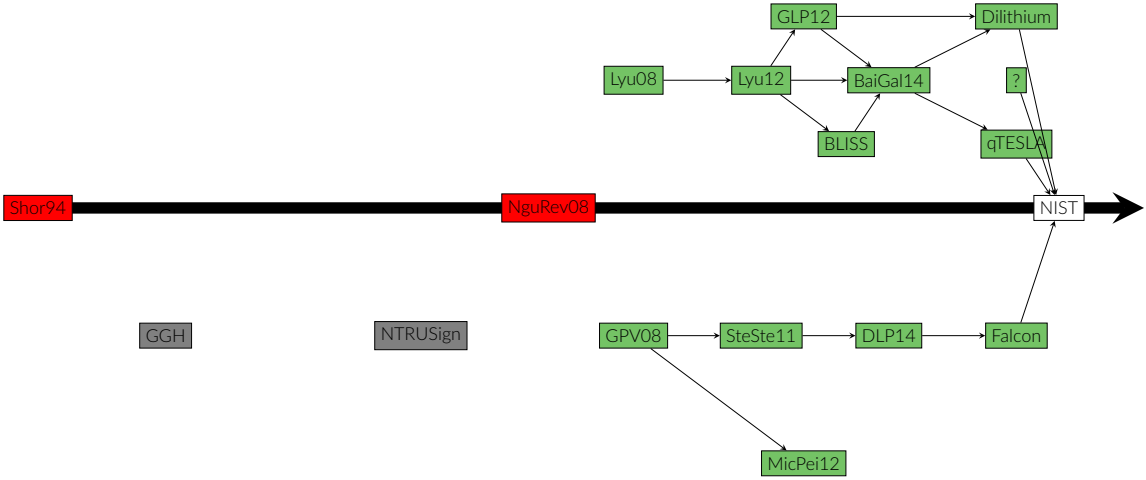
Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
BLISS
qTESLA
Shor94
NguRev08
GGH
NTRUSign
GPV08
SteSte11
DLP14
MicPei12

Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
BLISS
qTESLA
Shor94
NguRev08
GGH
NTRUSign
GPV08
SteSte11
DLP14
Falcon
MicPei12

Conclusion


GLP12
Dilithium
Lyu08
Lyu12
BaiGal14
?
BLISS
qTESLA
Shor94
NguRev08
NIST
GGH
NTRUSign
GPV08
SteSte11
DLP14
Falcon
MicPei12

Thanks!


Erdem Alkim, Nina Bindel, Johannes Buchmann, and Özgür Dagdelen.
TESLA: Tightly-secure efficient signatures from standard lattices.
Cryptology ePrint Archive, Report 2015/755, 2015.
http://eprint.iacr.org/2015/755.


Martin R. Albrecht, Shi Bai, and Léo Ducas.
A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes.
In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Heidelberg, August 2016.


Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry.
Random oracles in a quantum world.
In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011.


Shi Bai and Steven D. Galbraith.
An improved compression technique for signatures based on learning with errors.
In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 28–47. Springer, Heidelberg, February 2014.


Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee.

An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low level encoding of zero.
Cryptology ePrint Archive, Report 2016/139, 2016.
http://eprint.iacr.org/2016/139.


Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky.
Lattice signatures and bimodal Gaussians.
In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, August 2013.


Léo Ducas, Vadim Lyubashevsky, and Thomas Prest.
Efficient identity-based encryption over NTRU lattices.
In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 22–41. Springer, Heidelberg, December 2014.


Léo Ducas and Phong Q. Nguyen.
Faster Gaussian lattice sampling using lazy floating-point arithmetic.
In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 415–432. Springer, Heidelberg, December 2012.


Léo Ducas and Thomas Prest.
Fast fourier orthogonalization.

In Sergei A. Abramov, Eugene V. Zima, and Xiao-Shan Gao, editors, *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, Waterloo, ON, Canada, July 19-22, 2016*, pages 191–198. ACM, 2016.


Rafaël del Pino and Vadim Lyubashevsky.
Amortization with fewer equations for proving knowledge of small secrets.
In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 365–394. Springer, Heidelberg, August 2017.


Oded Goldreich, Shafi Goldwasser, and Shai Halevi.
Public-key cryptosystems from lattice reduction problems.
In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 112–131. Springer, Heidelberg, August 1997.


Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann.
Practical lattice-based cryptography: A signature scheme for embedded systems.
In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, Heidelberg, September 2012.


Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan.
Trapdoors for hard lattices and new cryptographic constructions.
In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.


Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte.
NTRUSIGN: Digital signatures using the NTRU lattice.
In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. Springer, Heidelberg, April 2003.


Paul Kirchner and Pierre-Alain Fouque.
Revisiting lattice attacks on overstretched NTRU parameters.
In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 3–26. Springer, Heidelberg, May 2017.


Vadim Lyubashevsky.
Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures.
In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.


Vadim Lyubashevsky.
Lattice signatures without trapdoors.
In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012.


Phong Q. Nguyen and Oded Regev.
Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures.

In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 271–288. Springer, Heidelberg, May / June 2006.

Alfréd Rényi.
On measures of entropy and information.
In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561, Berkeley, Calif., 1961. University of California Press.

Damien Stehlé and Ron Steinfeld.
Making NTRU as secure as worst-case problems over ideal lattices.
In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47. Springer, Heidelberg, May 2011.