PASS_{RS} : Practical Signatures from the Partial Fourier Recovery Problem

John M. Schanck

Institute for Quantum Computing, University of Waterloo
Security Innovation, Wilmington MA, USA

June 13, 2014

Joint work with:

Jeff Hoffstein, Jill Pipher, Joseph H. Silverman and William Whyte

Outline

- Multipoint evaluation and the Chinese remainder theorem
- Conjectured hard problem
- Signature scheme
- Benchmarks
- *Why post-quantum, why now?*

Multipoint Evaluation

Let $a = (a_1, a_2, \ldots, a_k) \in R^k$.

Multipoint Evaluation

Let $a = (a_1, a_2, \ldots, a_k) \in R^k$.
The map

$$\begin{aligned} \mathcal{E}_a : R[x] &\to R^k \\ \boldsymbol{f} &\mapsto (\boldsymbol{f}(a_1), \boldsymbol{f}(a_2), \ldots, \boldsymbol{f}(a_k)) \end{aligned}$$

is a ring homomorphism.

Multipoint Evaluation

Let $a = (a_1, a_2, \ldots, a_k) \in R^k$.
The map

$$\begin{aligned} \mathcal{E}_a : R[x] &\to R^k \\ \boldsymbol{f} &\mapsto (\boldsymbol{f}(a_1), \boldsymbol{f}(a_2), \ldots, \boldsymbol{f}(a_k)) \end{aligned}$$

is a ring homomorphism.

$$\mathcal{E}_a(\boldsymbol{f} \cdot \boldsymbol{g}) = (\boldsymbol{f}(a_1) \cdot \boldsymbol{g}(a_1), \ldots, \boldsymbol{f}(a_k) \cdot \boldsymbol{g}(a_k))$$

$$\mathcal{E}_a(\boldsymbol{f} + \boldsymbol{g}) = (\boldsymbol{f}(a_1) + \boldsymbol{g}(a_1), \ldots, \boldsymbol{f}(a_k) + \boldsymbol{g}(a_k))$$

Chinese Remainder Theorem

For any

$$\Phi = \phi_1 \phi_2 \dots \phi_k \in R[x]$$

ϕ_i coprime. The map

$$\begin{aligned} \mathcal{F}_\Phi : R[x]/\langle \Phi \rangle \ &\to R[x]/\langle \phi_1 \rangle \times R[x]/\langle \phi_2 \rangle \times \cdots \times R[x]/\langle \phi_k \rangle \\ \boldsymbol{f} &\mapsto (\boldsymbol{f} \pmod{\phi_1},\ \boldsymbol{f} \pmod{\phi_2}, \dots,\ \boldsymbol{f} \pmod{\phi_k}) \end{aligned}$$

is an isomorphism of rings.

Multipoint evaluation 2

Let

$$\Phi = (x - a_1)(x - a_2) \dots (x - a_n) \in R.$$

Then the map

$$\begin{aligned} \mathcal{F}_\Phi : R/\langle \Phi \rangle &\to R^n \\ \boldsymbol{f} &\mapsto (\boldsymbol{f}(a_1), \boldsymbol{f}(a_2), \dots, \boldsymbol{f}(a_n)) \end{aligned}$$

is an isomorphism of rings.

An important example

- Choose prime $n, q \in \mathbb{Z}$ such that $q \equiv 1 \pmod{n}$.
- There exists ω such that $\omega^n \equiv 1 \pmod{q}$.
- $\Phi = x^n - 1 = (x-1)(x-\omega^1)\dots(x-\omega^{n-1}) \pmod{q}$

The Discrete Fourier (or Number Theoretic) Transform over $\mathbb{Z}_q$:

$$\begin{aligned} \mathcal{F}_\Phi : R/\langle\Phi\rangle &\to R^n \\ \boldsymbol{f} &\mapsto \left(\boldsymbol{f}(1), \boldsymbol{f}(\omega^1), \dots, \boldsymbol{f}(\omega^{n-1})\right) \end{aligned}$$

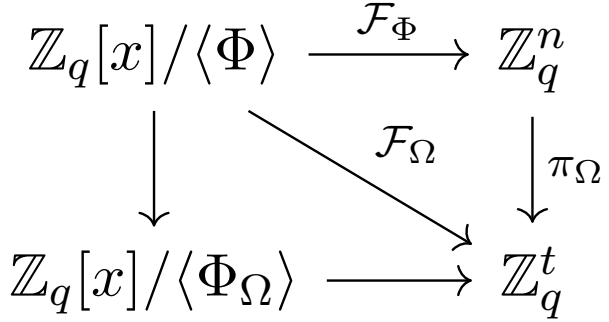
Partial Fourier Transform

- n, q as above. $\Phi = x^n - 1 \pmod{q}$.
- Let $\Omega \subset \{0, \ldots, n-1\}$, $|\Omega| = t \approx n/2$.
- $\Phi_\Omega = \prod_{i \in \Omega} (x - \omega^i) \pmod{q}$

$$\begin{array}{ccc} \mathbb{Z}_q[x]/\langle \Phi \rangle & \xrightarrow{\mathcal{F}_\Phi} & \mathbb{Z}_q^n \\ \downarrow & & \downarrow \pi_\Omega \\ \mathbb{Z}_q[x]/\langle \Phi_\Omega \rangle & \longrightarrow & \mathbb{Z}_q^t \end{array}$$

A *really* hard problem

Let $\boldsymbol{f}$ be drawn uniformly from $\mathbb{Z}_q[x]/\langle\Phi\rangle$.
Given $\mathcal{F}_\Omega(\boldsymbol{f})$, compute $\boldsymbol{f}$.



The hard problem we use

Let the coefficients of $\boldsymbol{f}$ be drawn uniformly from $\{-1, 0, 1\}$. Given $\mathcal{F}_\Omega(\boldsymbol{f})$, find $\boldsymbol{f}$ or another small polynomial, $\boldsymbol{f}'$ such that $\mathcal{F}_\Omega(\boldsymbol{f}) = \mathcal{F}_\Omega(\boldsymbol{f}')$.

Parameter definitions

n	Prime
q	Prime $\equiv 1 \pmod{n}$
ω	a primitive n^{th} root of unity in $\mathbb{Z}_q$
Ω	A subset of $\{0, \ldots, n-1\}$
t	$\lvert\Omega\rvert$
k	max-norm of noise polynomials
b	1-norm of challenge polynomials

Definitions

- Private key: $\boldsymbol{f} \in \mathbb{Z}_q[x]/\langle \Phi \rangle$, with small coefficients (i.e. $\in \{-1, 0, 1\}$).

Definitions

- Private key: $\boldsymbol{f} \in \mathbb{Z}_q[x]/\langle\Phi\rangle$, with small coefficients (i.e. $\in \{-1, 0, 1\}$).
- Public key: Ω, $\mathcal{F}_\Omega(\boldsymbol{f})$.

Definitions

- Private key: $\boldsymbol{f} \in \mathbb{Z}_q[x]/\langle \Phi \rangle$, with small coefficients (i.e. $\in \{-1, 0, 1\}$).
- Public key: Ω, $\mathcal{F}_\Omega(\boldsymbol{f})$.
- Noise polynomial: $\boldsymbol{y} \in \mathcal{B}^\infty(k)$.

Definitions

- Private key: $\boldsymbol{f} \in \mathbb{Z}_q[x]/\langle\Phi\rangle$, with small coefficients (i.e. $\in \{-1, 0, 1\}$).
- Public key: $\Omega,\ \mathcal{F}_\Omega(\boldsymbol{f})$.
- Noise polynomial: $\boldsymbol{y} \in \mathcal{B}^\infty(k)$.
- Commitment polynomial: $\boldsymbol{c} \in \mathcal{B}^1(b)$.

Definitions

- Private key: $\boldsymbol{f} \in \mathbb{Z}_q[x]/\langle\Phi\rangle$, with small coefficients (i.e. $\in \{-1, 0, 1\}$).
- Public key: $\Omega,\ \mathcal{F}_\Omega(\boldsymbol{f})$.
- Noise polynomial: $\boldsymbol{y} \in \mathcal{B}^\infty(k)$.
- Commitment polynomial: $\boldsymbol{c} \in \mathcal{B}^1(b)$.
- Signature: Point $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$ such that

$$\mathcal{F}_\Omega(\boldsymbol{z} - \boldsymbol{f}\boldsymbol{c}) = \mathcal{F}_\Omega(\boldsymbol{y}).$$

Algorithm 1 Sign

Input: $(\mu,\ \boldsymbol{f})$

1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^{\infty}(k)$
3: $\quad h \leftarrow \mathsf{Hash}(\mathcal{F}_{\Omega}(\boldsymbol{y}), \mu)$
4: $\quad \boldsymbol{c} \leftarrow \mathsf{FormatC}(h)$
5: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$
6: **until** $\boldsymbol{z} \in \mathcal{B}^{\infty}(k - b)$

Output: $(\boldsymbol{c}, \boldsymbol{z}, \mu)$

Algorithm 2 Sign

Input: $(\mu,\ \boldsymbol{f})$

1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^{\infty}(k)$
3: $\quad h \leftarrow \mathsf{Hash}(\mathcal{F}_{\Omega}(\boldsymbol{y}), \mu)$
4: $\quad \boldsymbol{c} \leftarrow \mathsf{FormatC}(h)$
5: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$
6: **until** $\boldsymbol{z} \in \mathcal{B}^{\infty}(k - b)$

Output: $(\boldsymbol{c}, \boldsymbol{z}, \mu)$

Algorithm 3 Sign

Input: $(\mu,\ \boldsymbol{f})$

1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^{\infty}(k)$
3: $\quad h \leftarrow \mathsf{Hash}(\mathcal{F}_{\Omega}(\boldsymbol{y}), \mu)$
4: $\quad \boldsymbol{c} \leftarrow \mathsf{FormatC}(h)$
5: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$
6: **until** $\boldsymbol{z} \in \mathcal{B}^{\infty}(k - b)$

Output: $(\boldsymbol{c}, \boldsymbol{z}, \mu)$

Algorithm 4 Sign

Input: $(\mu,\ \boldsymbol{f})$

1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^{\infty}(k)$
3: $\quad h \leftarrow \mathsf{Hash}(\mathcal{F}_{\Omega}(\boldsymbol{y}), \mu)$
4: $\quad \boldsymbol{c} \leftarrow \mathsf{FormatC}(h)$
5: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$
6: **until** $\boldsymbol{z} \in \mathcal{B}^{\infty}(k - b)$

Output: $(\boldsymbol{c}, \boldsymbol{z}, \mu)$

Algorithm 5 Sign

Input: $(\mu,\ \boldsymbol{f})$

1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^{\infty}(k)$
3: $\quad h \leftarrow \mathsf{Hash}(\mathcal{F}_{\Omega}(\boldsymbol{y}), \mu)$
4: $\quad \boldsymbol{c} \leftarrow \mathsf{FormatC}(h)$
5: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$
6: **until** $\boldsymbol{z} \in \mathcal{B}^{\infty}(k - b)$

Output: $(\boldsymbol{c}, \boldsymbol{z}, \mu)$

Algorithm 6 Sign

Input: $(\mu,\ \boldsymbol{f})$

1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^{\infty}(k)$
3: $\quad h \leftarrow \mathsf{Hash}(\mathcal{F}_{\Omega}(\boldsymbol{y}), \mu)$
4: $\quad \boldsymbol{c} \leftarrow \mathsf{FormatC}(h)$
5: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$
6: **until** $\boldsymbol{z} \in \mathcal{B}^{\infty}(k - b)$

Output: $(\boldsymbol{c}, \boldsymbol{z}, \mu)$

Algorithm 7 Sign

Input: $(\mu,\ \boldsymbol{f})$

1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^{\infty}(k)$
3: $\quad h \leftarrow \mathsf{Hash}(\mathcal{F}_{\Omega}(\boldsymbol{y}), \mu)$
4: $\quad \boldsymbol{c} \leftarrow \mathsf{FormatC}(h)$
5: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$
6: **until** $\boldsymbol{z} \in \mathcal{B}^{\infty}(k - b)$

Output: $(\boldsymbol{c}, \boldsymbol{z}, \mu)$

Verification

Algorithm 8 Verify

Input: $(\boldsymbol{c}, \boldsymbol{z}, \mu, \mathcal{F}_\Omega(\boldsymbol{f}))$

1: $result \leftarrow$ invalid
2: **if** $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$ **then**
3: $h' \leftarrow \mathsf{Hash}(\mathcal{F}_\Omega(\boldsymbol{z}) - \mathcal{F}_\Omega(\boldsymbol{f})\mathcal{F}_\Omega(\boldsymbol{c}), \mu)$ {Recall $\boldsymbol{z} = \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$}
4: $\boldsymbol{c}' \leftarrow \mathsf{FormatC}(h')$
5: **if** $\boldsymbol{c} = \boldsymbol{c}'$ **then**
6: $result \leftarrow$ valid
7: **end if**
8: **end if**

Output: $result$

Verification

Algorithm 9 Verify

Input: $(\boldsymbol{c}, \boldsymbol{z}, \mu, \mathcal{F}_\Omega(\boldsymbol{f}))$

1: $result \leftarrow$ invalid
2: **if** $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$ **then**
3: $h' \leftarrow \mathsf{Hash}(\mathcal{F}_\Omega(\boldsymbol{z}) - \mathcal{F}_\Omega(\boldsymbol{f})\mathcal{F}_\Omega(\boldsymbol{c}), \mu)$ {Recall $\boldsymbol{z} = \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$}
4: $\boldsymbol{c}' \leftarrow \mathsf{FormatC}(h')$
5: **if** $\boldsymbol{c} = \boldsymbol{c}'$ **then**
6: $result \leftarrow$ valid
7: **end if**
8: **end if**

Output: $result$

Verification

Algorithm 10 Verify

Input: $(\boldsymbol{c}, \boldsymbol{z}, \mu, \mathcal{F}_\Omega(\boldsymbol{f}))$

1: $result \leftarrow$ invalid
2: **if** $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$ **then**
3: $h' \leftarrow \mathsf{Hash}(\mathcal{F}_\Omega(\boldsymbol{z}) - \mathcal{F}_\Omega(\boldsymbol{f})\mathcal{F}_\Omega(\boldsymbol{c}), \mu)$ {Recall $\boldsymbol{z} = \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$}
4: $\boldsymbol{c}' \leftarrow \mathsf{FormatC}(h')$
5: **if** $\boldsymbol{c} = \boldsymbol{c}'$ **then**
6: $result \leftarrow$ valid
7: **end if**
8: **end if**

Output: $result$

Verification

Algorithm 11 Verify

Input: $(\boldsymbol{c}, \boldsymbol{z}, \mu, \mathcal{F}_\Omega(\boldsymbol{f}))$

1: $result \leftarrow$ invalid
2: **if** $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$ **then**
3: $h' \leftarrow \mathsf{Hash}(\mathcal{F}_\Omega(\boldsymbol{z}) - \mathcal{F}_\Omega(\boldsymbol{f})\mathcal{F}_\Omega(\boldsymbol{c}), \mu)$ {Recall $\boldsymbol{z} = \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$}
4: $\boldsymbol{c}' \leftarrow \mathsf{FormatC}(h')$
5: **if** $\boldsymbol{c} = \boldsymbol{c}'$ **then**
6: $result \leftarrow$ valid
7: **end if**
8: **end if**

Output: $result$

Verification

Algorithm 12 Verify

Input: $(\boldsymbol{c}, \boldsymbol{z}, \mu, \mathcal{F}_\Omega(\boldsymbol{f}))$

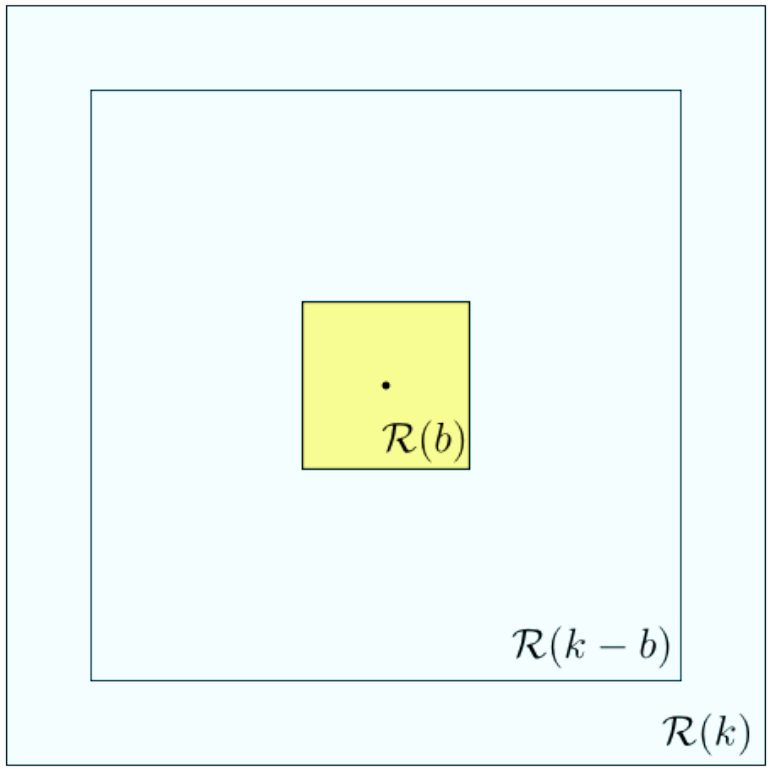
1: $result \leftarrow$ invalid
2: **if** $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$ **then**
3: $\quad h' \leftarrow \mathsf{Hash}(\mathcal{F}_\Omega(\boldsymbol{z}) - \mathcal{F}_\Omega(\boldsymbol{f})\mathcal{F}_\Omega(\boldsymbol{c}), \mu)$ {Recall $\boldsymbol{z} = \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$}
4: $\quad \boldsymbol{c}' \leftarrow \mathsf{FormatC}(h')$
5: $\quad$ **if** $\boldsymbol{c} = \boldsymbol{c}'$ **then**
6: $\quad\quad result \leftarrow$ valid
7: $\quad$ **end if**
8: **end if**

Output: $result$

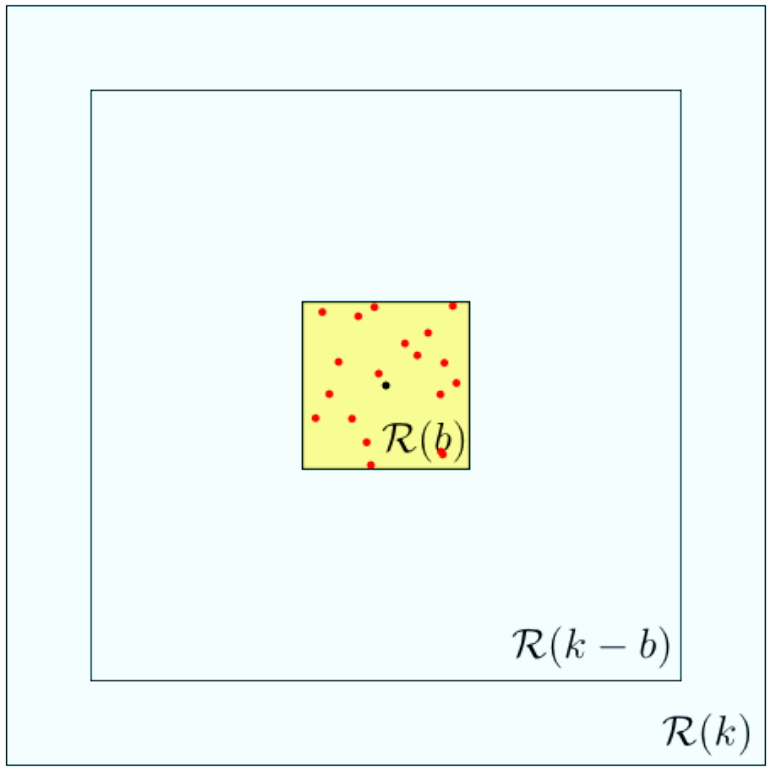
Transcript security

- The original PASS was broken due to information leakage in transcripts of message-signature pairs.
- Rejection sampling yields signature points which are independent of the key and uniformly distributed in $\mathcal{B}^{\infty}(k-b)$.

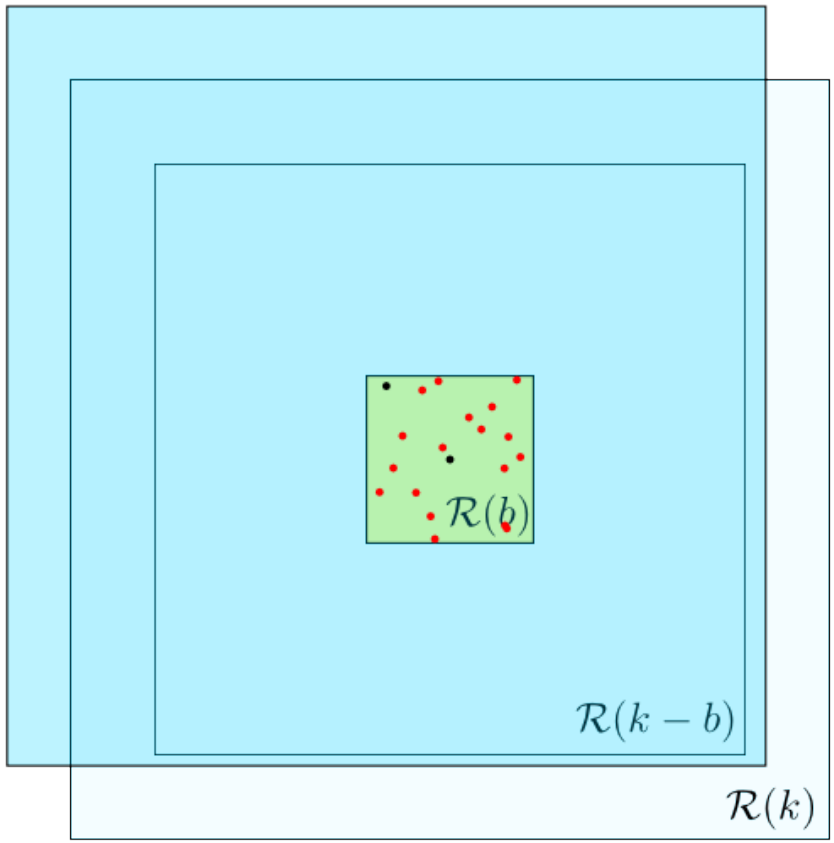
Transcript security



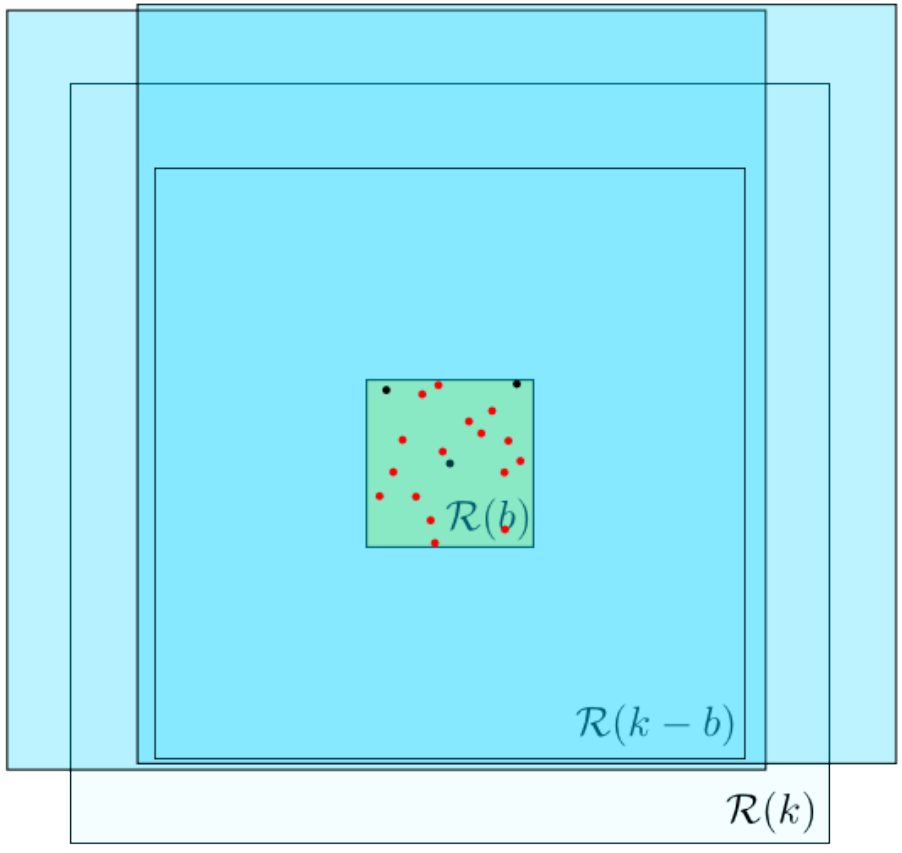
Transcript security



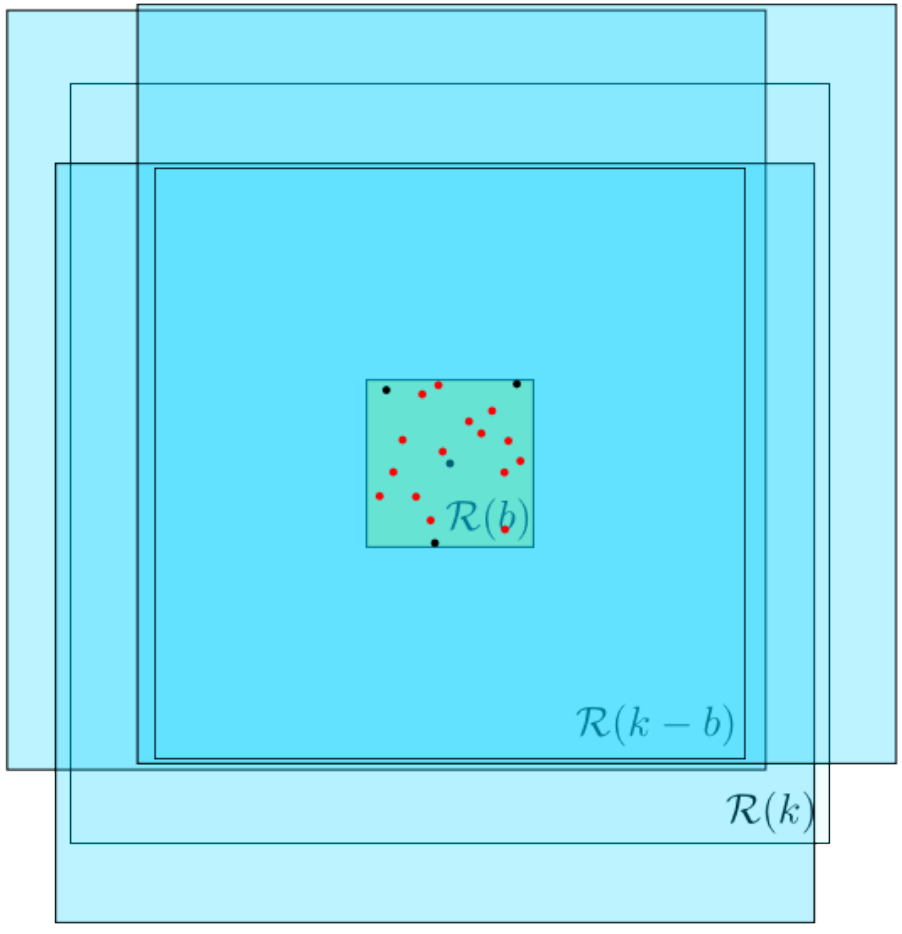
Transcript security



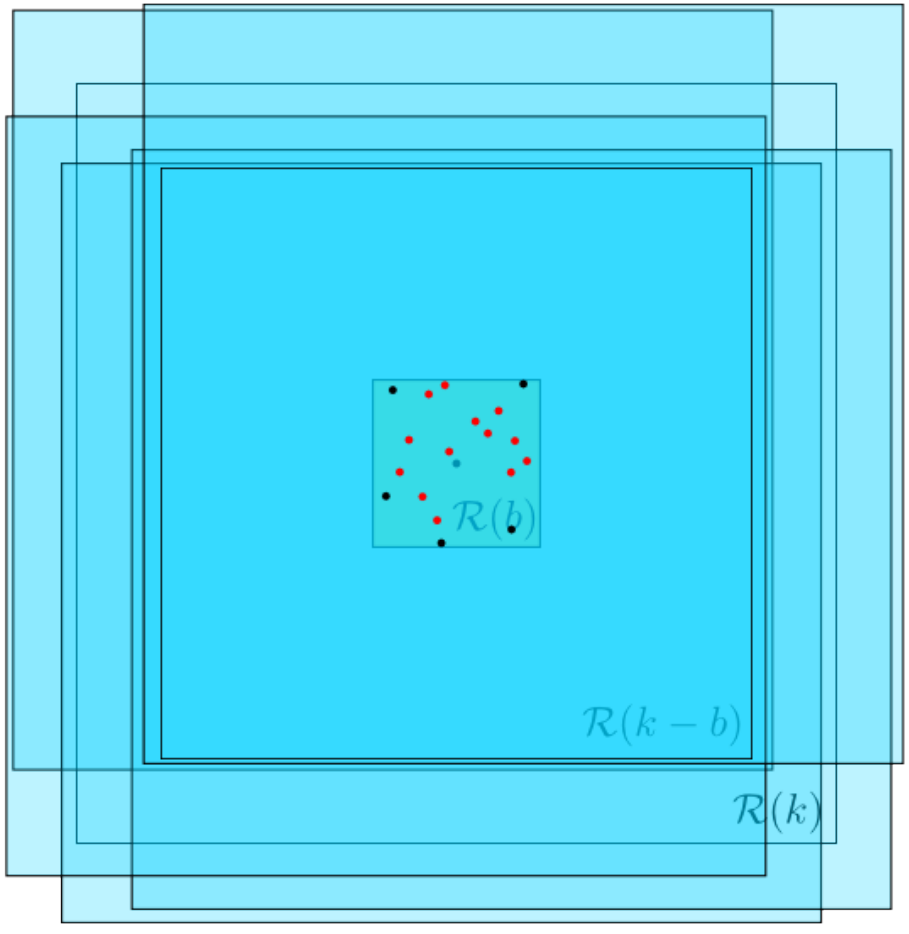
Transcript security



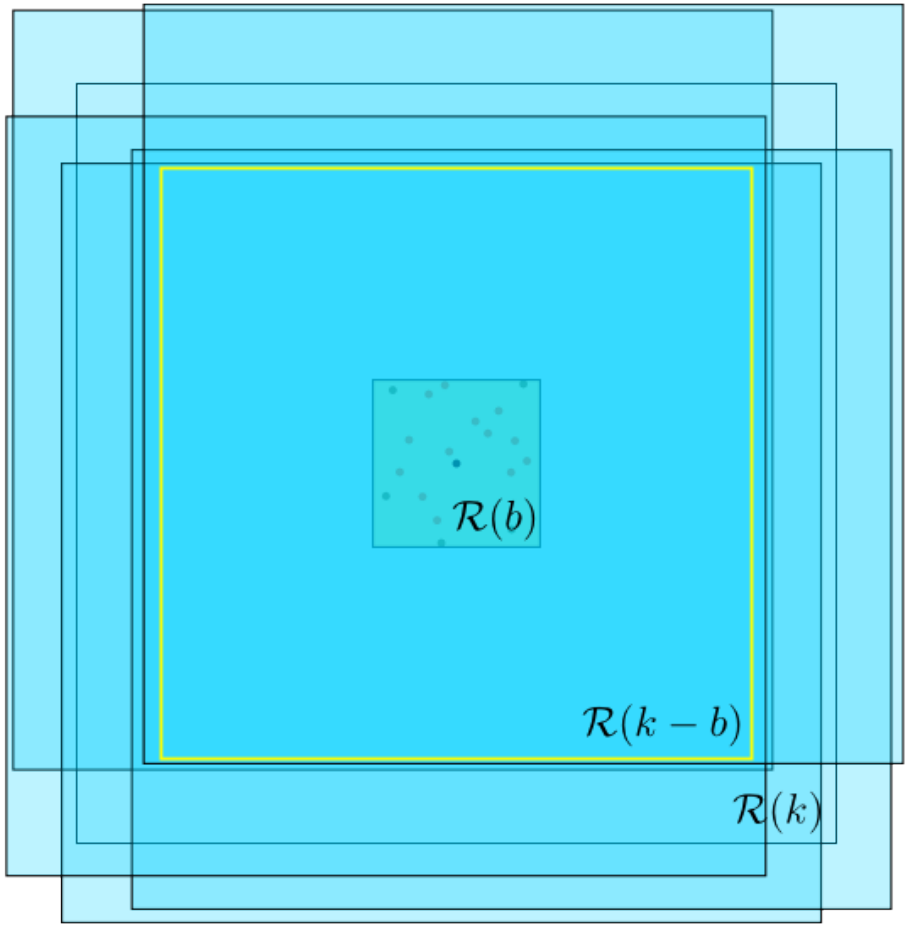
Transcript security



Transcript security



Transcript security



Lattice security

Find short vectors in the kernel (or some coset thereof) of a *Vandermonde* matrix

$$[\boldsymbol{A}]_{i,j} = \omega^{\Omega_i \cdot j}$$

Lattice security

$$\boldsymbol{A} = \begin{pmatrix} 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^2 & \omega^5 & \omega & \omega^4 \\ 1 & \omega^4 & \omega & \omega^5 & \omega^2 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix}$$

Lattice security

Vandermonde Matrix Small Integer Solution Problem:
Given n, q, Ω, $\boldsymbol{A}$, and β. Find $\boldsymbol{x}$ such that

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \quad \text{and} \quad |\boldsymbol{x}| \leq \beta$$

Inhomogeneous variant (also given $\mathcal{F}_\Omega(\boldsymbol{f})$:

$$\boldsymbol{A}\boldsymbol{x} = \mathcal{F}_\Omega(\boldsymbol{f}) \quad \text{and} \quad |\boldsymbol{x}| \leq \beta$$

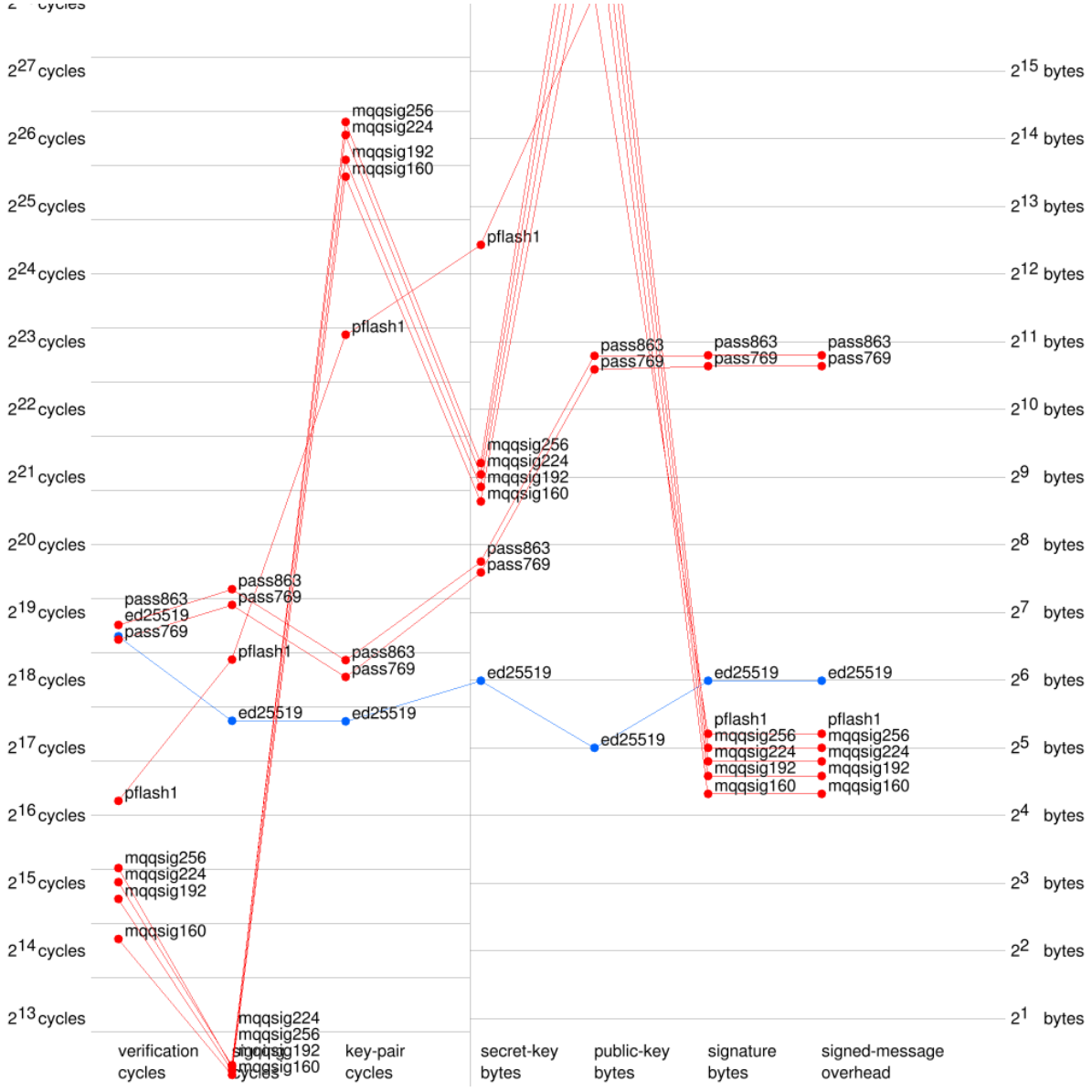
Parameters

n	769	863	1153
q	$\approx 2^{20}$	$\approx 2^{20}$	$\approx 2^{20}$
k	$2^{15}-1$	$2^{15}-1$	$2^{15}-1$
b	29	28	36
t	386	444	600
$\Pr[Accept]$	0.49	0.48	0.72
UniqueSVP gap	1.0075	1.0069	1.0052
ApproxSVP factor	1.0081	1.0077	1.0054
Lattice security factor	1.0084	1.0078	1.0058
Entropy of $\boldsymbol{c}$	200	200	260
Bit-security bound	< 100	< 100	≤ 130

Benchmarks

See http://bench.cr.yp.to/results-sign.html

Benchmarks


2^{27} cycles
2^{26} cycles
2^{25} cycles
2^{24} cycles
2^{23} cycles
2^{22} cycles
2^{21} cycles
2^{20} cycles
2^{19} cycles
2^{18} cycles
2^{17} cycles
2^{16} cycles
2^{15} cycles
2^{14} cycles
2^{13} cycles
2^{15} bytes
2^{14} bytes
2^{13} bytes
2^{12} bytes
2^{11} bytes
2^{10} bytes
2^{9} bytes
2^{8} bytes
2^{7} bytes
2^{6} bytes
2^{5} bytes
2^{4} bytes
2^{3} bytes
2^{2} bytes
2^{1} bytes
mqqsig256
mqqsig224
mqqsig192
mqqsig160
pflash1
pass863
pass769
ed25519
verification cycles
key-pair cycles
secret-key bytes
public-key bytes
signature bytes
signed-message overhead

Thanks!