Post-Quantum Secure Technologies

Daniël Kuijsters, Tim Weenink and Simona Samardjiska
Compumatica Secure Networks & Radboud Universiteit Nijmegen
October 4, 2018 (13:00 - 14:00)

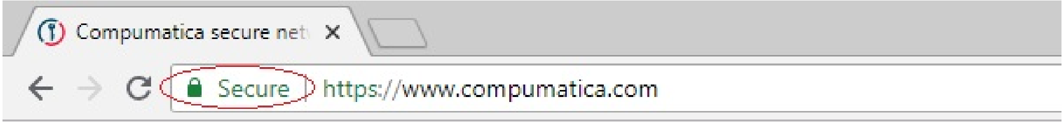
Radboud Universiteit

Compumatica SECURE NETWORKS

Cyber Security Week powered by The Hague Security Delta

www.cybersecurityweek.nl

Pre-Quantum Cryptography



This page is secure (valid HTTPS).

- Certificate - valid and trusted

  The connection to this site is using a valid, trusted server certificate issued by Let's Encrypt Authority X3.

  View certificate

- Connection - secure (strong TLS 1.2)

  The connection to this site is encrypted and authenticated using TLS 1.2 (a strong protocol), ECDHE_RSA with P-256 (a strong key exchange), and AES_128_GCM (a strong cipher).

The Threat - Shor (1994) and Grover (1996)

Cyber Security Week
powered by The Hague Security Delta

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

A fast quantum mechanical algorithm for database search

Lov K. Grover
3C-404A, Bell Labs
600 Mountain Avenue
Murray Hill NJ 07974
lkgrover@bell-labs.com

The Impact

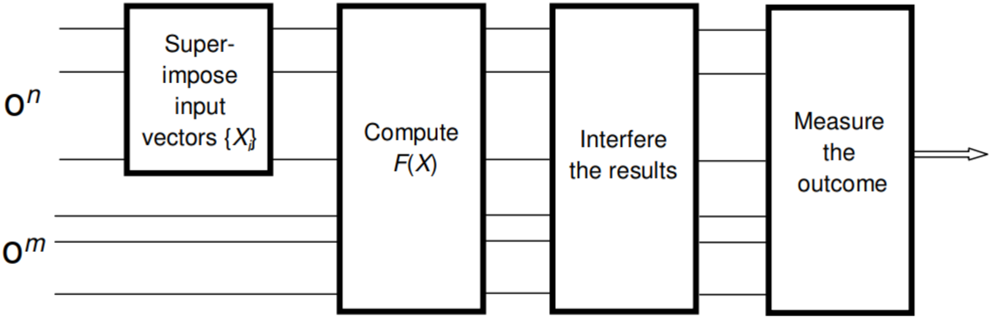
Shor's algorithm efficiently solves:

- Integer factorization problem - RSA is dead.
- The discrete logarithm problem in finite fields - DSA is dead.
- The discrete logarithm problem on elliptic curves - ECDH and ECDSA are dead.

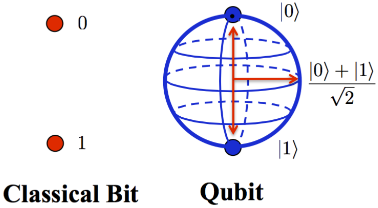
Grover's algorithm has an impact on the security of symmetric primitives:

- AES key size needs to be doubled.
- The output of hash functions needs to be doubled.

Quantum Computation - The Bigger Picture



Superposition - Single Qubit



In general, the state is described as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ are called **probability amplitudes** and $|\alpha|^2 + |\beta|^2 = 1$.

Superposition - Multiple Qubits

The case of two qubits

Their combined state is described as

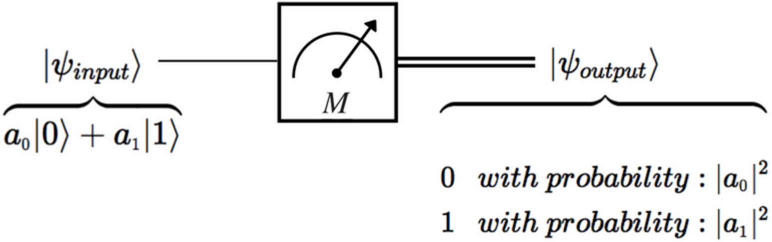
$$|\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle$$

where $\alpha_i \in \mathbb{C}$ for $i = 0, 1, 2, 3$ and $\sum_{i=0}^{3} |\alpha_i|^2 = 1$.

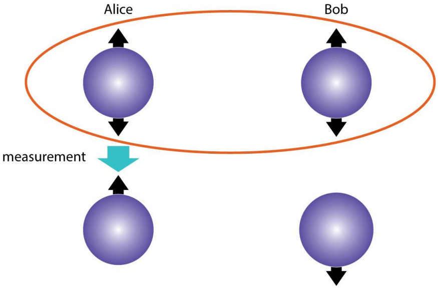
Quantum parallelism

All the possible combinations of "0" and "1" are processed at the same time.

Measurement

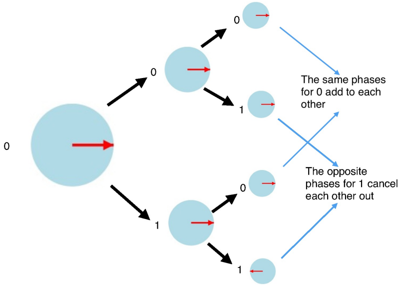


Entanglement



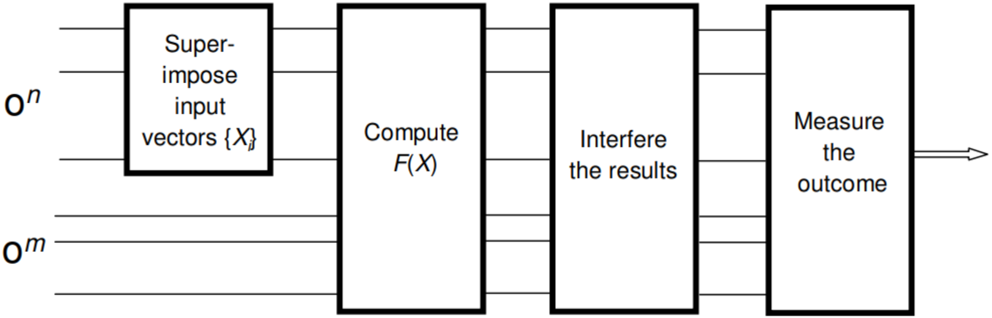
Required if a quantum algorithm is to offer an exponential speed-up over classical computation.

Interference



Used to amplify the state corresponding to the correct solution.

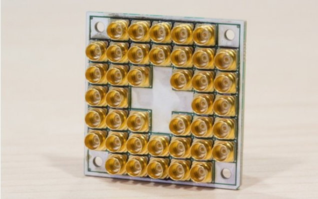
Quantum Computation - The Bigger Picture



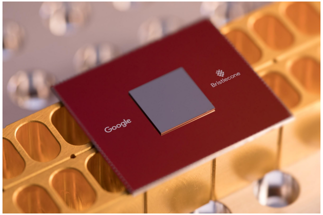
Should We Be Worried?

Cyber Security Week
powered by The Hague Security Delta

IBM's 50-qubit system (November 2017).



Intel's 49-qubit chip (January 2018).



Google's 72-qubit chip (March 2018).

Post-Quantum Cryptography

NIST Submissions			
	Signatures	KEM/Encryption	Overall
Lattice-based	5	21	26
Code-based	3	17	19
Multivariate	7	2	9
Hash-based	3		3
Other	2	5	7
Total	19	45	64

Already Involved

Cyber Security Week
powered by The Hague Security Delta

Infineon

utimaco®

QuantiCor

Microsoft® Research

ISARA

Deployment Issues

- Standardization.
- Secure implementations.
- Protocol compatibility.

NewHope

Key Exchange

NewHope

Published in 2016.

Post-quantum key exchange.

Partially developed in the Netherlands.

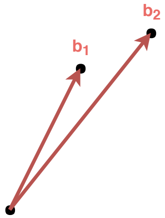
- RU Nijmegen.
- CWI Amsterdam.

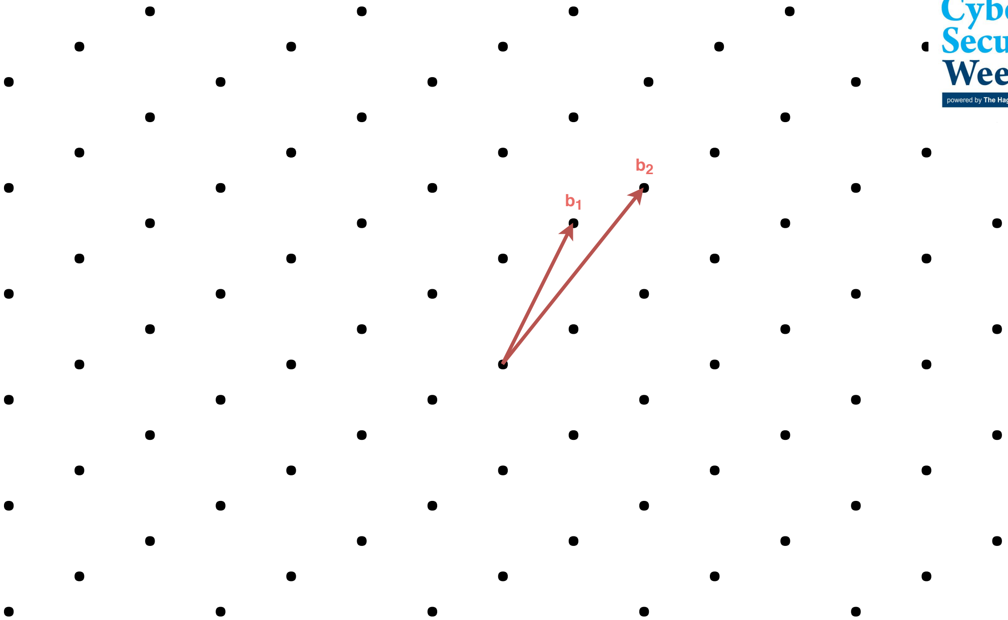
Facebook Internet Defense Prize winner (2016).

Implementation by Google.

Based on the hardness of lattice problems.

Lattices


b2
b1

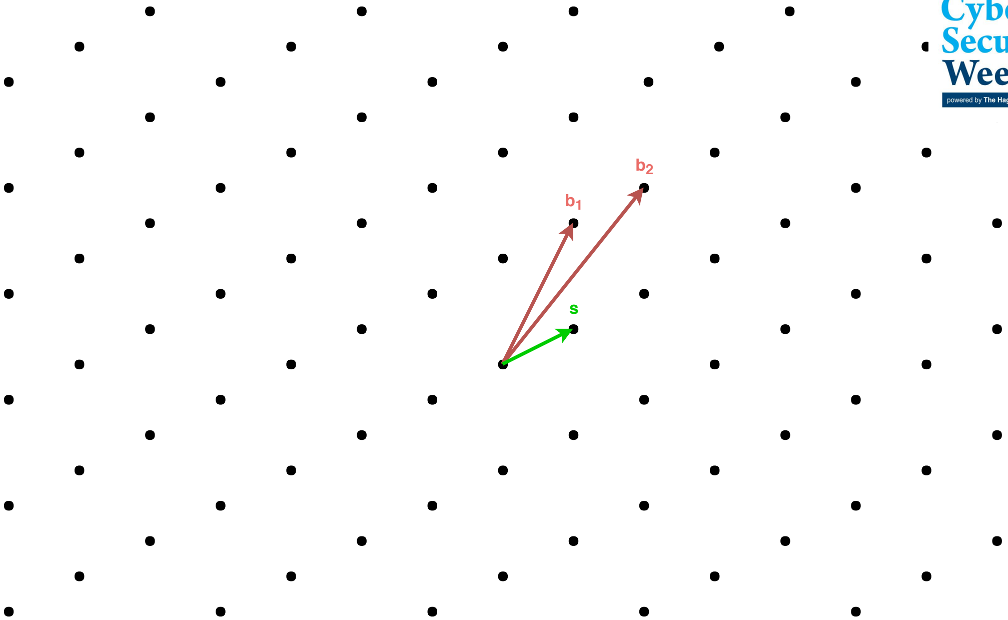


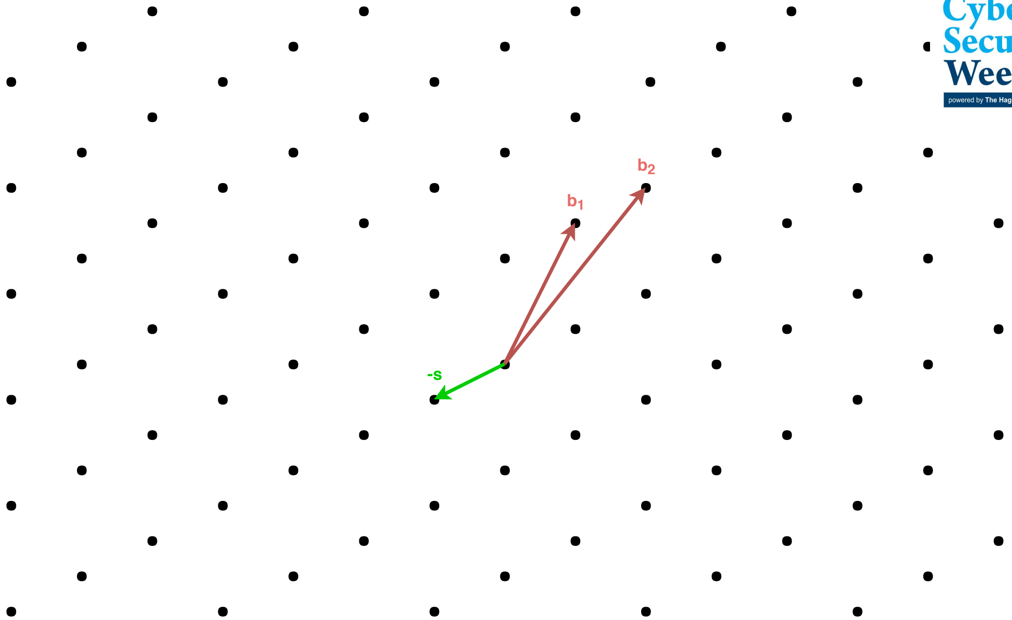
b_1
b_2

Lattice Problems

Shortest vector problem (SVP)

Given a lattice and basis, find the shortest non-zero vector.


b2
b1
s

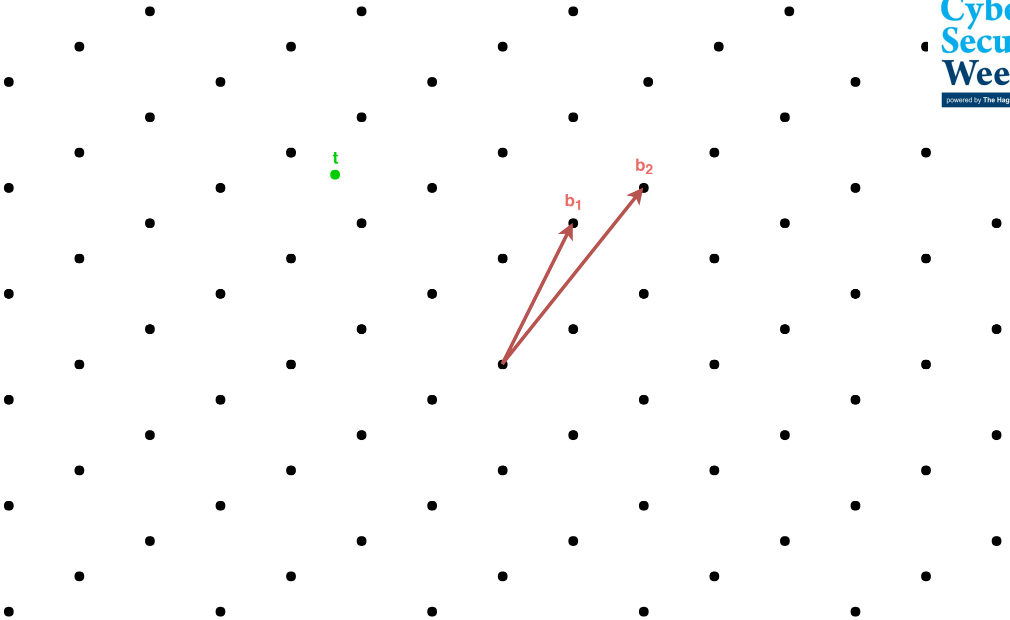


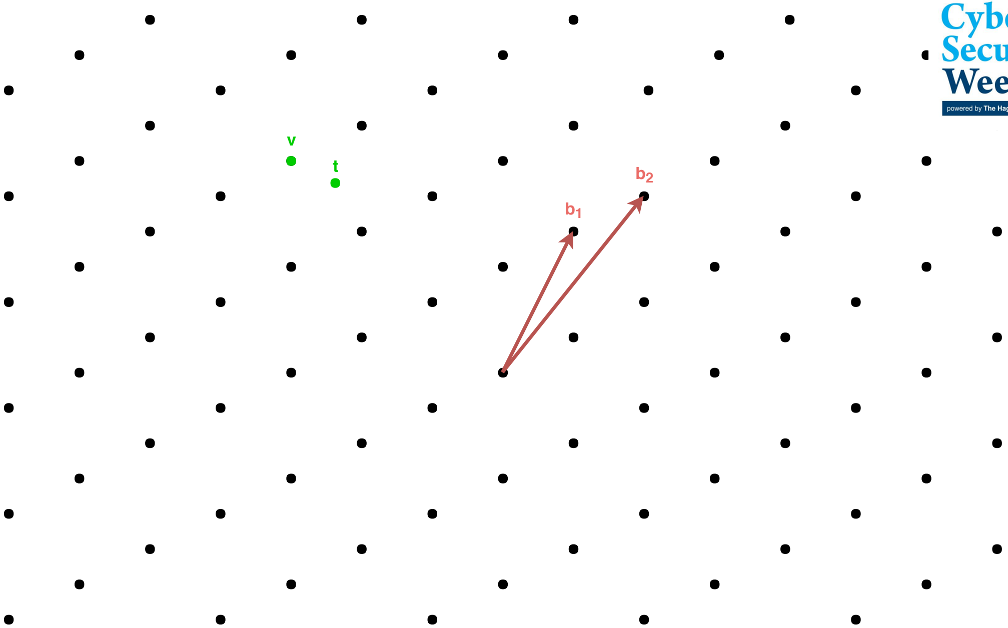
b2
b1
-s

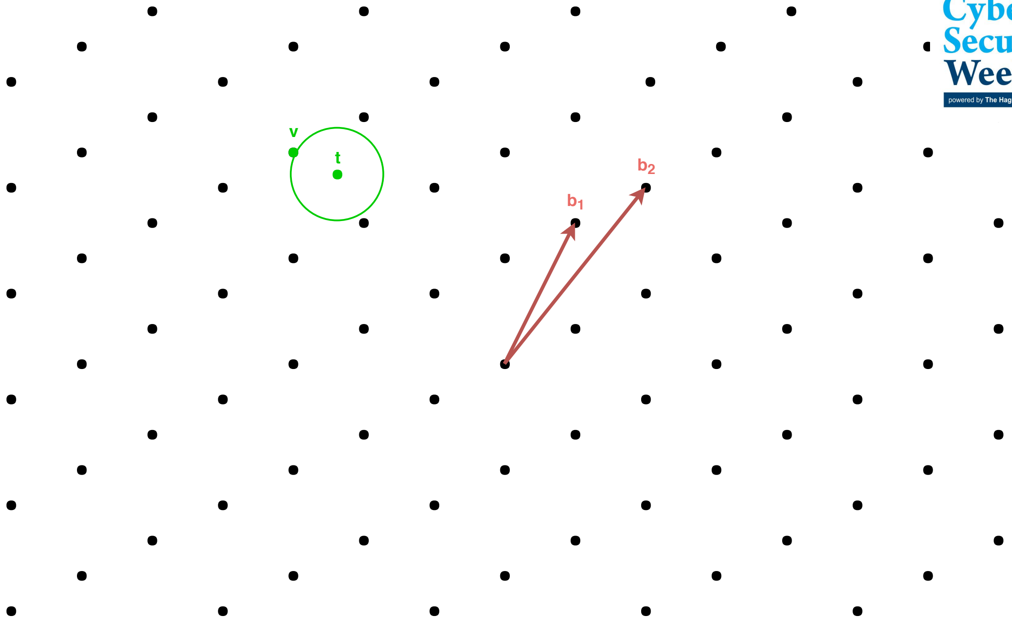
Lattice Problems

Closest vector problem (CVP)

Given a lattice, basis, and vector t, find the closest lattice point v to t.


t
b1
b2

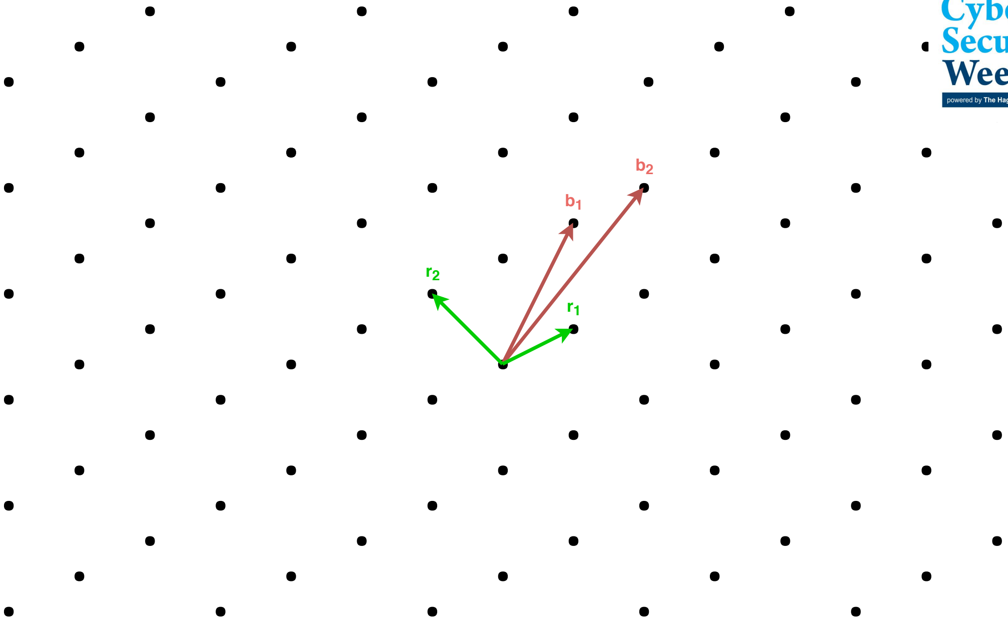

v
t
b1
b2

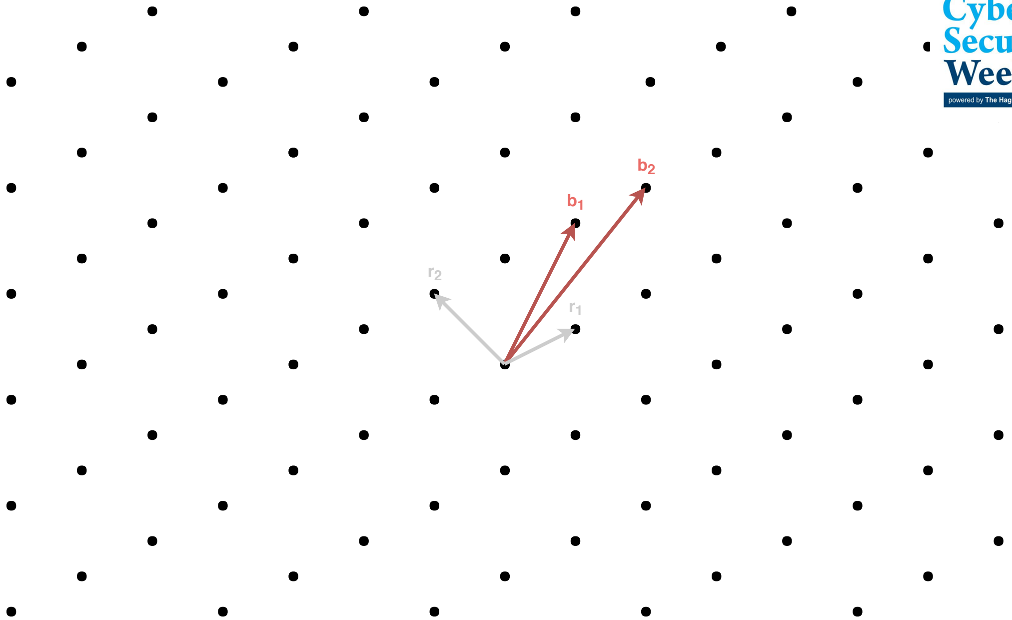

v
t
b_1
b_2

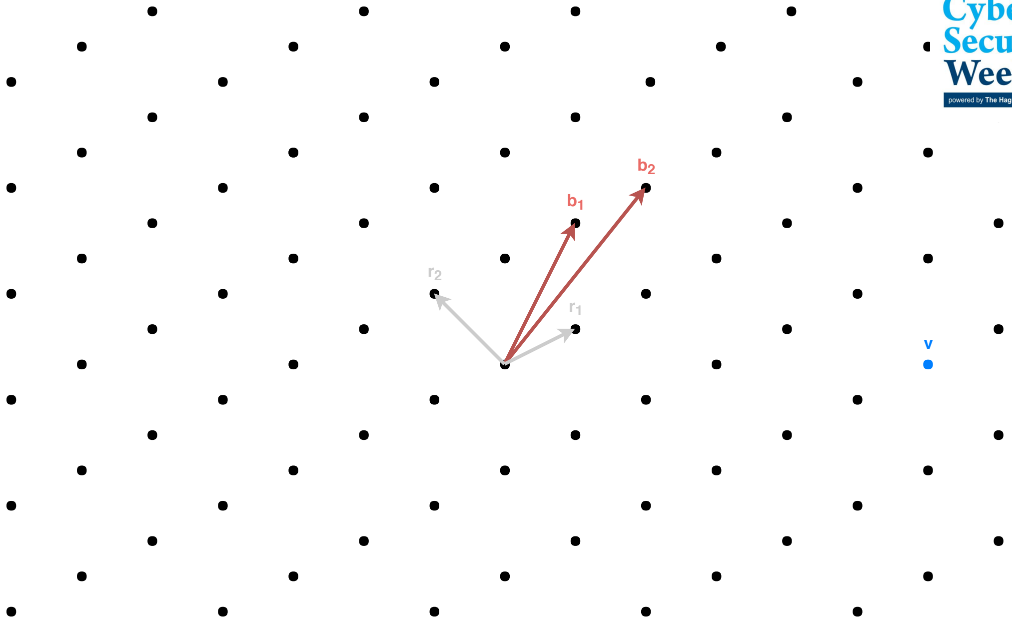
Encryption

Lattices can also be used for encryption.
Example: GGH cryptosystem.

The private vectors need to be short and (nearly) orthogonal.

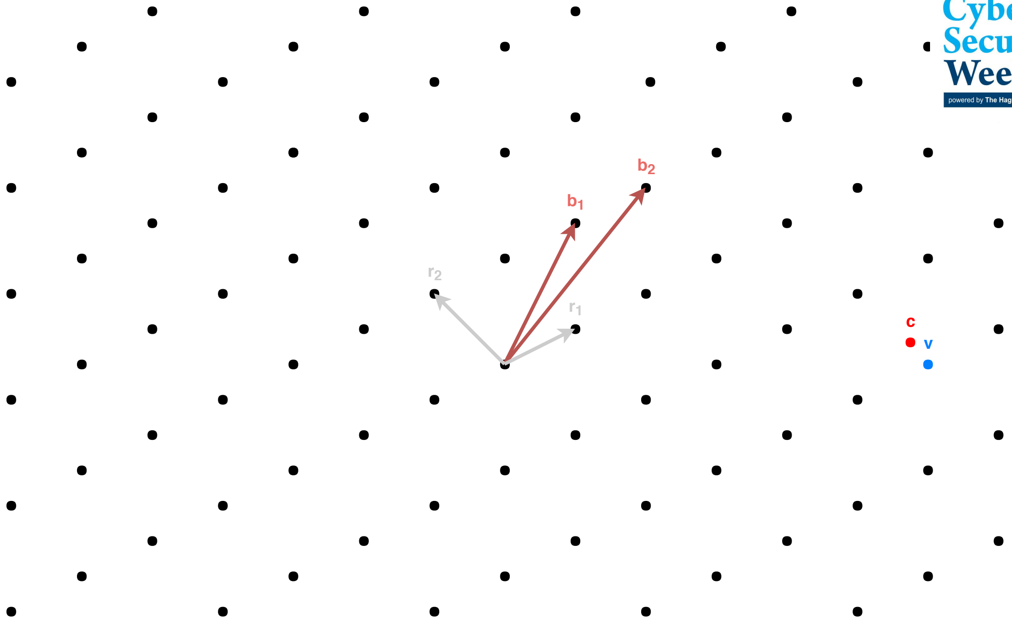

b2
b1
r2
r1


b2
b1
r2
r1

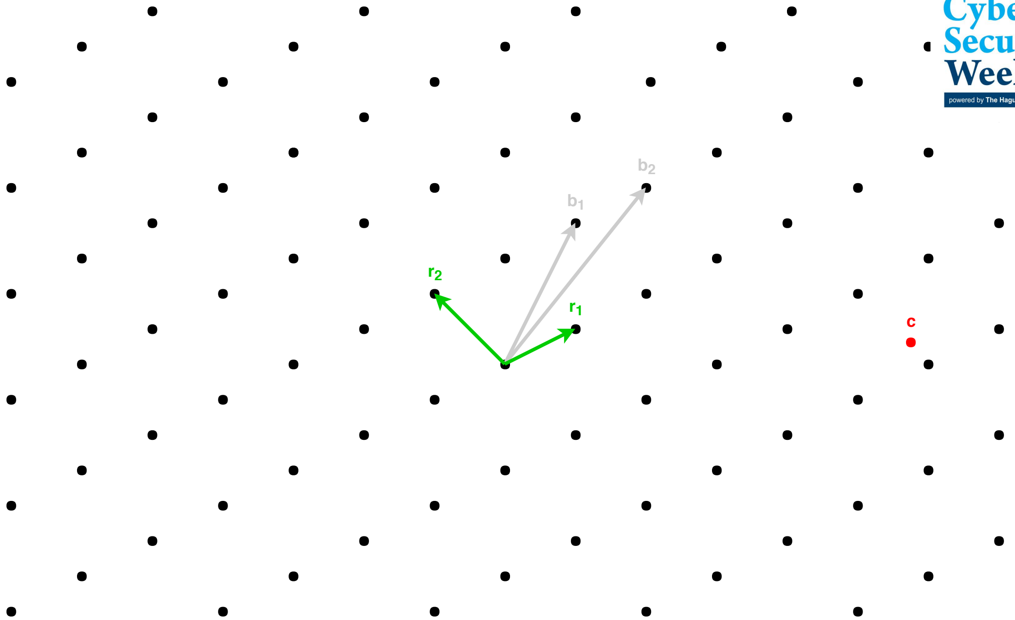


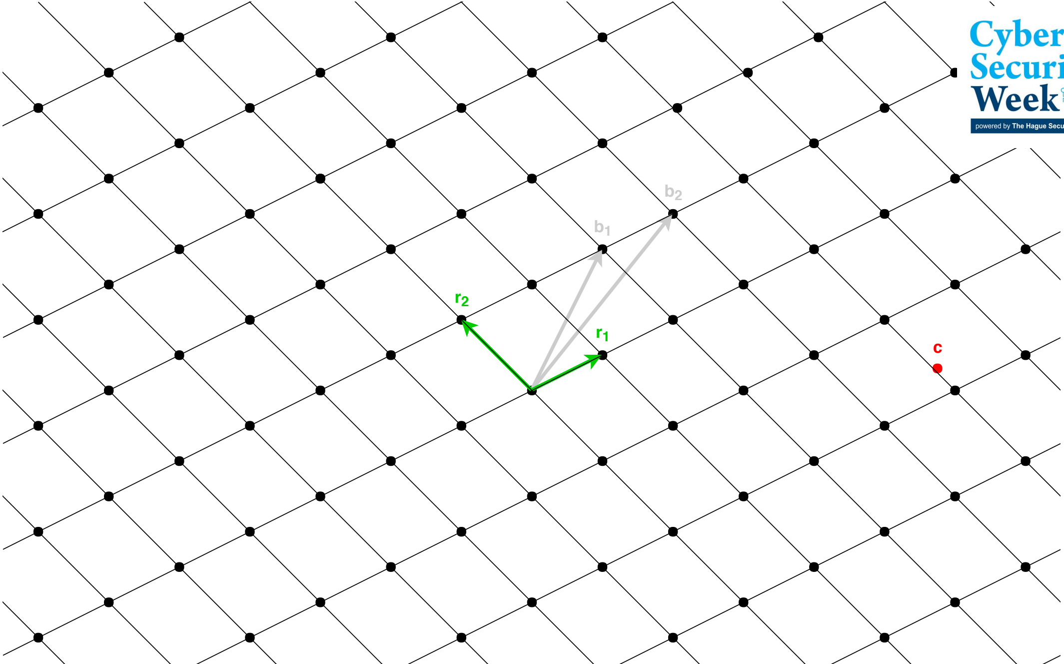
Cyber Security Week

powered by The Hague Security Delta

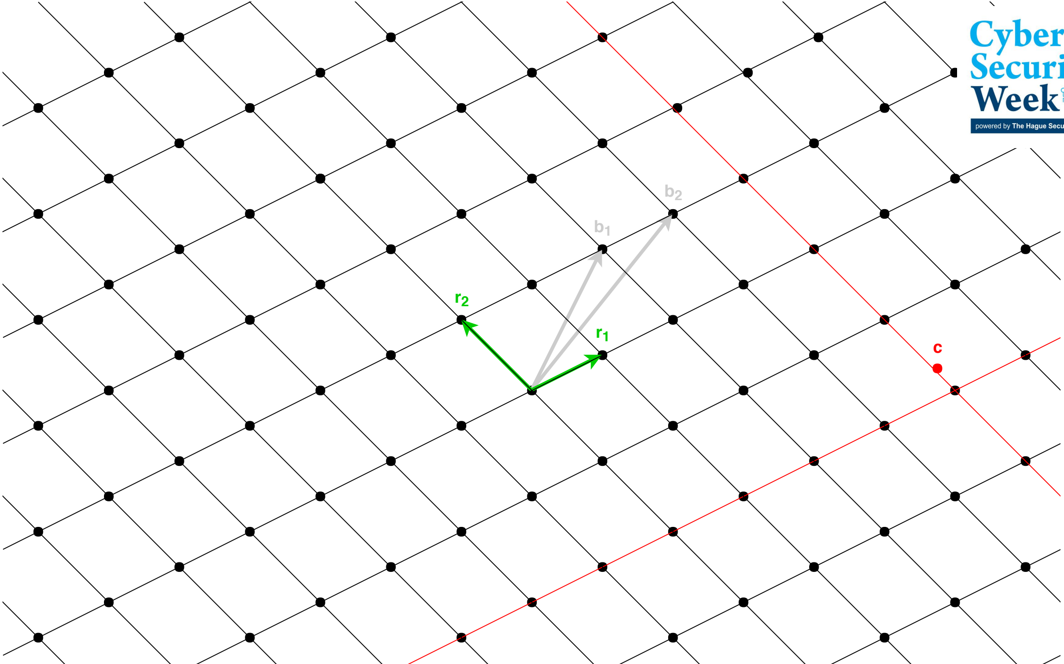


b_2
b_1
r_2
r_1
c
v

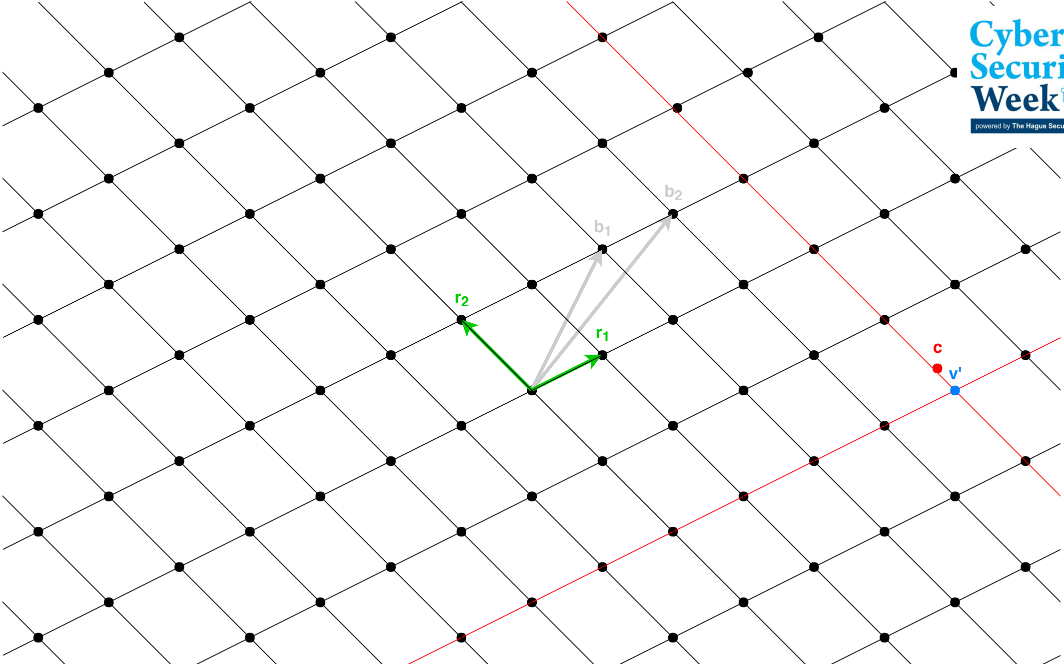

b_2
b_1
r_2
r_1
c



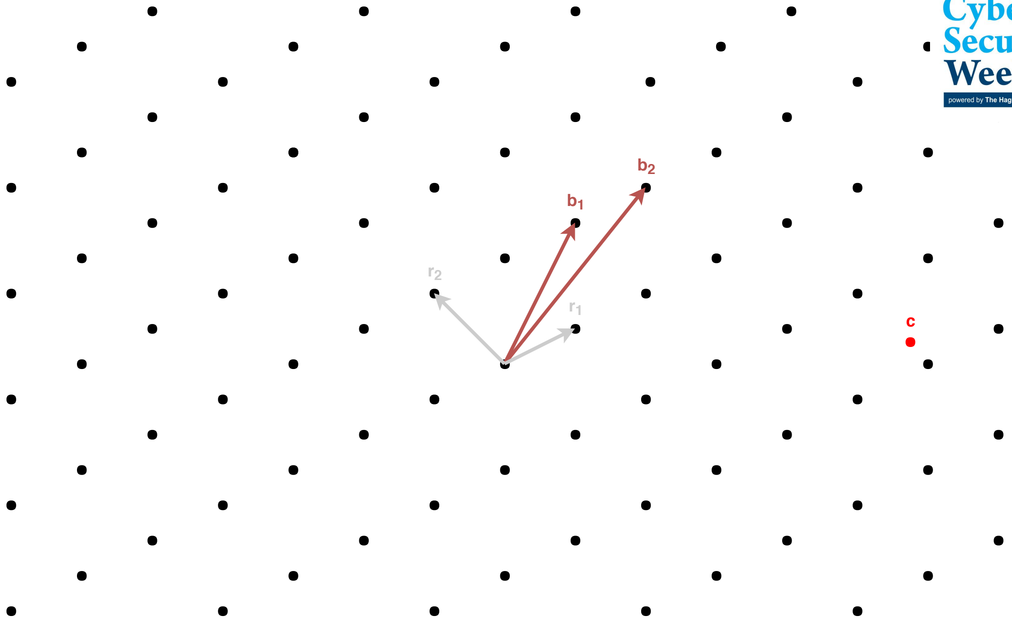
Cyber Security Week
powered by The Hague Security Delta

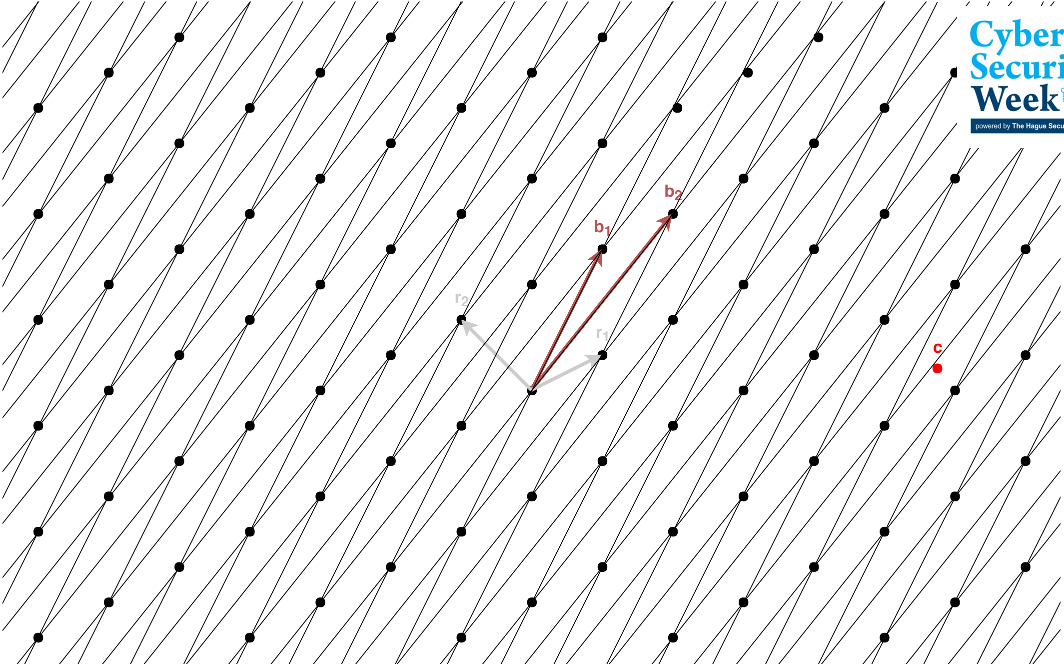


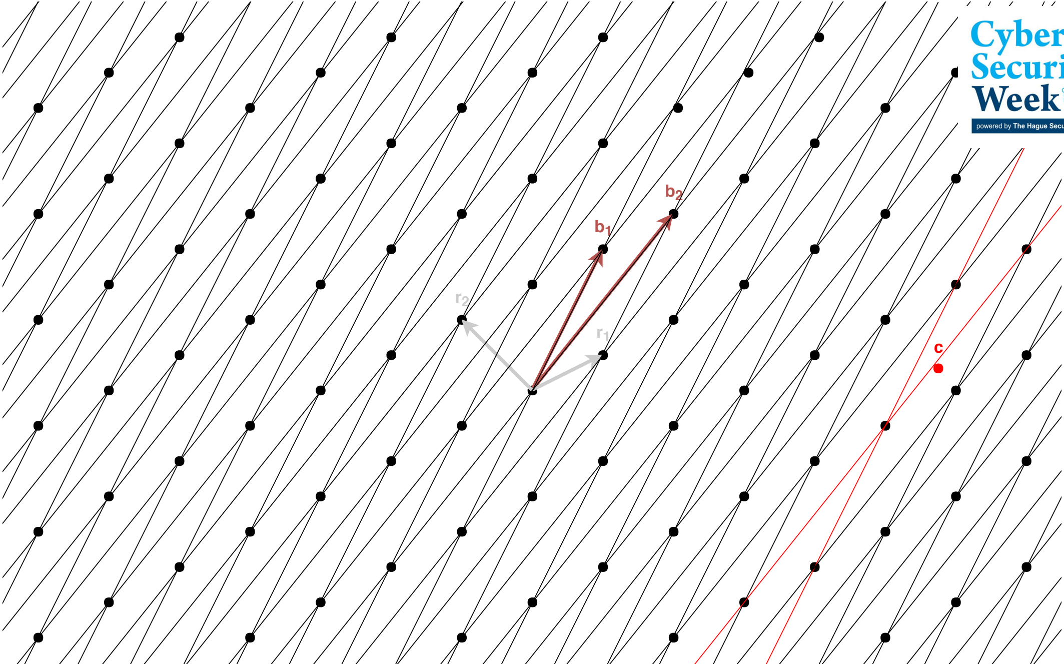
b2
b1
r2
r1
c



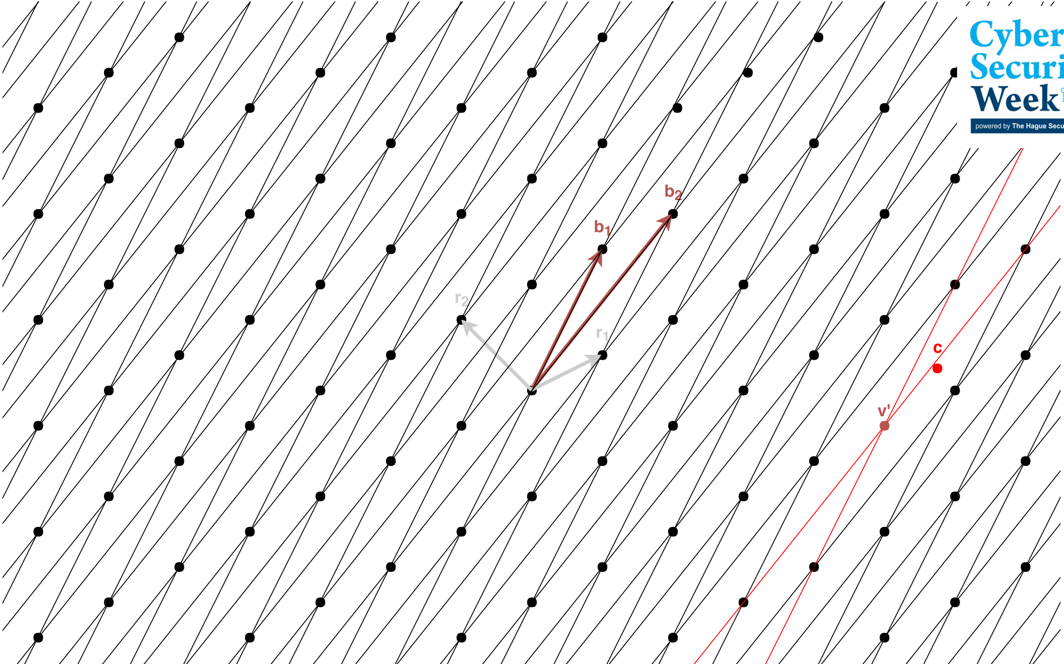
b2
b1
r2
r1
c
v'


b2
b1
r2
r1
c


b1
b2
r2
r1
c



b2
b1
r2
r1
c



Cyber Security Week

powered by The Hague Security Delta

Lattices

Hardness of lattices

Based on finding the shortest vector.

- Still hard when a lattice basis is given.

Different problems

There are more problems that are based on the hardness of lattice problems.

Learning With Errors (LWE)

Reductions

Solving random LWE instances is as hard as solving worst-case instances of certain lattice problems with a quantum computer.

Learning With Errors (LWE)

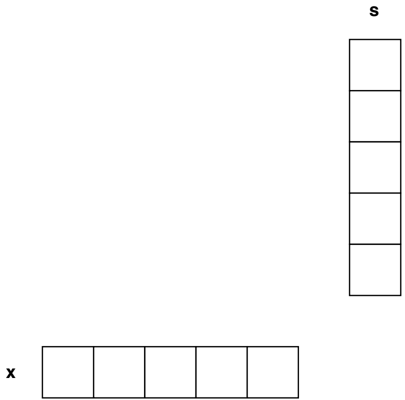
Challenge

Given an input matrix $A \in \mathbb{Z}_q^{m \times n}$ and output vector $b \in \mathbb{Z}_q^{m \times 1}$, find the secret vector $s \in \mathbb{Z}_q^{1 \times n}$.
The output vector b is the result of multiplying A and s, and adding a noise vector e afterwards.

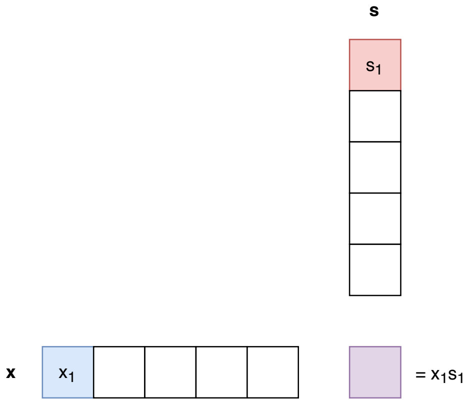
Example

We will start with an easy case: parity learning ($q = 2$).

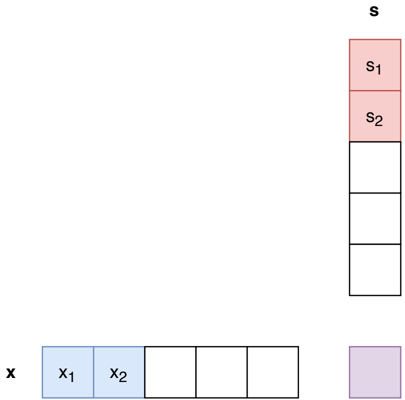
Parity Learning



Parity Learning

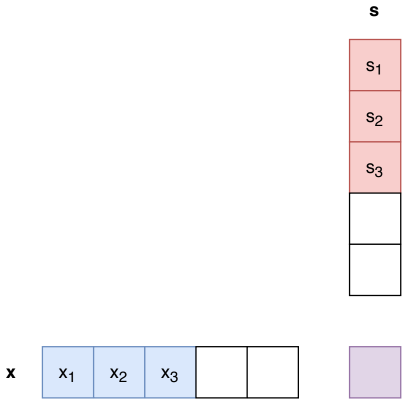


Parity Learning



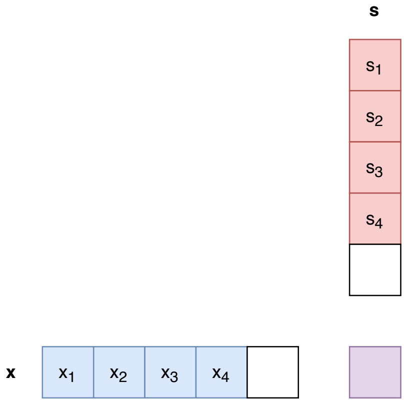
$= x_1 s_1 + x_2 s_2$

Parity Learning



$= x_1s_1 + x_2s_2 + x_3s_3$

Parity Learning



$= x_1 s_1 + x_2 s_2 + x_3 s_3 + x_4 s_4$

Parity Learning

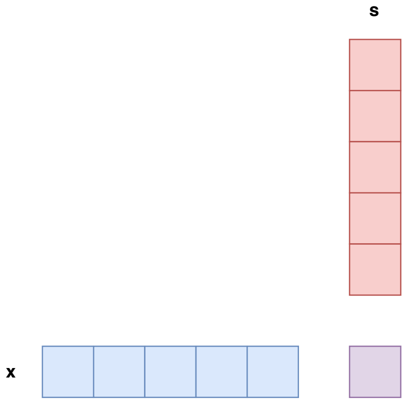
s

$$\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix}$$

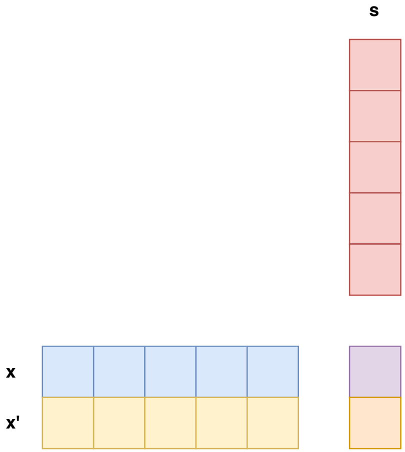
x $\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}$

$$= x_1 s_1 + x_2 s_2 + x_3 s_3 + x_4 s_4 + x_5 s_5$$

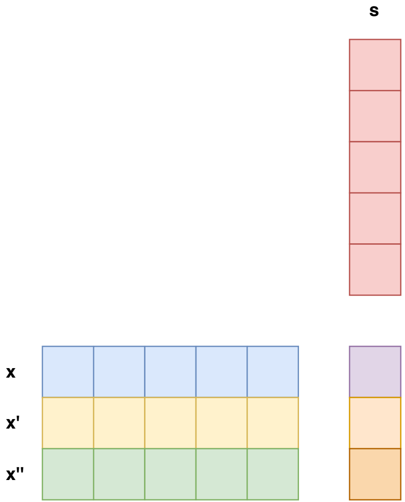
Parity Learning



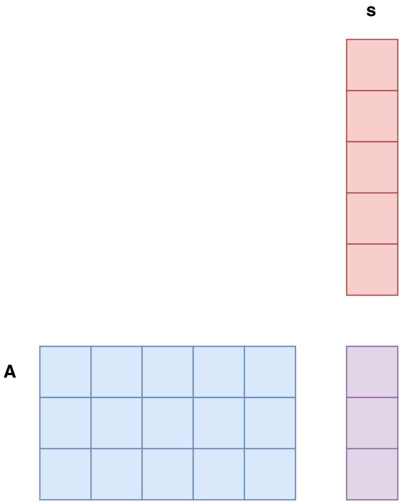
Parity Learning



Parity Learning



Parity Learning



Parity Learning

s

A

0	0	1	1	1
0	0	0	1	0
1	1	0	0	1

0
1
0

Parity Learning With Noise

$$\mathbf{s} = \begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix}$$

$$\mathbf{A} \quad \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{+} \overset{\mathbf{e}}{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}} = \overset{\mathbf{b}}{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}$$

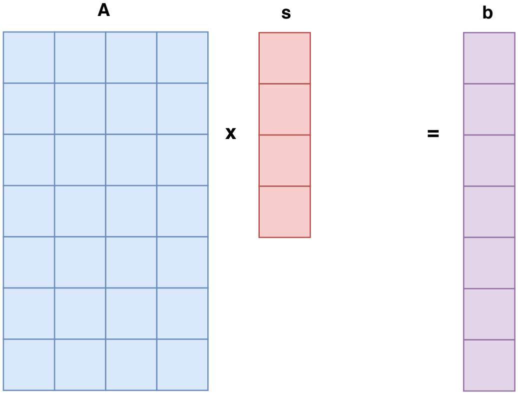
Learning With Errors (LWE)

We want to make it more difficult for quantum computers.
Instead of only 0's and 1's, we use $0, \ldots, q$ with a larger prime number q.

Example

In the following example, we will use $q = 13$.

Learning With Errors (LWE)



Learning With Errors (LWE)

$$\begin{array}{ccccc} \mathbf{A} & & \mathbf{s} & & \mathbf{b} \\ \begin{bmatrix} 5 & 9 & 6 & 10 \\ 7 & 1 & 0 & 6 \\ 8 & 9 & 2 & 11 \\ 12 & 7 & 3 & 10 \\ 3 & 8 & 1 & 1 \\ 2 & 11 & 8 & 4 \\ 9 & 7 & 7 & 0 \end{bmatrix} & \mathbf{x} & \begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix} & = & \begin{bmatrix} 4 \\ 0 \\ 2 \\ 8 \\ 2 \\ 12 \\ 10 \end{bmatrix} \end{array}$$

Learning With Errors (LWE)

$$\begin{array}{c} \mathbf{A} \\ \begin{bmatrix} 5 & 9 & 6 & 10 \\ 7 & 1 & 0 & 6 \\ 8 & 9 & 2 & 11 \\ 12 & 7 & 3 & 10 \\ 3 & 8 & 1 & 1 \\ 2 & 11 & 8 & 4 \\ 9 & 7 & 7 & 0 \end{bmatrix} \end{array} \mathbf{x} \begin{array}{c} \mathbf{s} \\ \begin{bmatrix} 2 \\ 7 \\ 3 \\ 3 \end{bmatrix} \end{array} = \begin{array}{c} \mathbf{b} \\ \begin{bmatrix} 4 \\ 0 \\ 1 \\ 8 \\ 3 \\ 0 \\ 10 \end{bmatrix} \end{array}$$

Learning With Errors (LWE)

$$\overset{\mathbf{A}}{\begin{bmatrix} 5 & 9 & 6 & 10 \\ 7 & 1 & 0 & 6 \\ 8 & 9 & 2 & 11 \\ 12 & 7 & 3 & 10 \\ 3 & 8 & 1 & 1 \\ 2 & 11 & 8 & 4 \\ 9 & 7 & 7 & 0 \end{bmatrix}} \times \overset{\mathbf{s}}{\begin{bmatrix} 2 \\ 7 \\ 3 \\ 3 \end{bmatrix}} + \overset{\mathbf{e}}{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ -1 \\ 0 \end{bmatrix}} = \overset{\mathbf{b}}{\begin{bmatrix} 4 \\ 0 \\ 2 \\ 8 \\ 2 \\ 12 \\ 10 \end{bmatrix}}$$

Learning With Errors (LWE)

$$\overset{\mathbf{A}}{\begin{bmatrix} 5 & 9 & 6 & 10 \\ 7 & 1 & 0 & 6 \\ 8 & 9 & 2 & 11 \\ 12 & 7 & 3 & 10 \\ 3 & 8 & 1 & 1 \\ 2 & 11 & 8 & 4 \\ 9 & 7 & 7 & 0 \end{bmatrix}} \mathbf{x} \overset{\mathbf{s}}{\begin{bmatrix} \\ \\ \\ \end{bmatrix}} + \overset{\mathbf{e}}{\begin{bmatrix} \\ \\ \\ \\ \\ \\ \end{bmatrix}} = \overset{\mathbf{b}}{\begin{bmatrix} 4 \\ 0 \\ 2 \\ 8 \\ 2 \\ 12 \\ 10 \end{bmatrix}}$$

Learning With Errors (LWE)

This example

$q = 13$, $n = 4$.

Real world parameters are much bigger.

NewHope

$q = 12289$, $n = 1024$.

Using LWE would give very large keys.

Therefore, NewHope uses Ring-LWE.

Ring-LWE

Idea

Use a polynomial a instead of matrix A.
Every coefficient of a is multiplied by every coefficient of s.
This results in a much smaller key size.

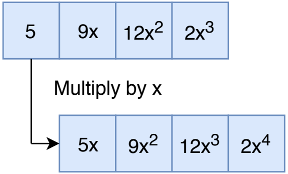
Example

Multiply $5 + 9x + 12x^2 + 2x^3$ from ring $\mathbb{Z}_{13}[x]/\langle x^4 + 1\rangle$ by x.

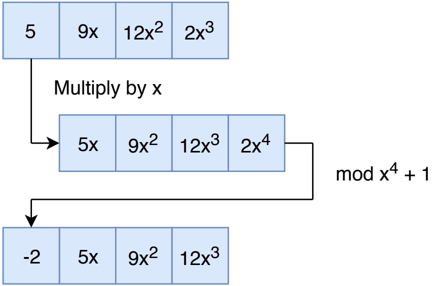
Ring-LWE

5	$9x$	$12x^2$	$2x^3$

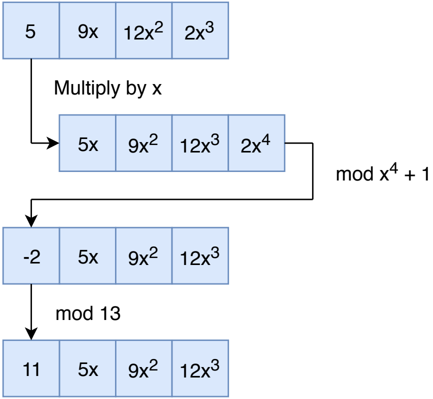
Ring-LWE



Ring-LWE



Ring-LWE



5
9x
12x2
2x3
Multiply by x
5x
9x2
12x3
2x4
mod x4 + 1
-2
5x
9x2
12x3
mod 13
11
5x
9x2
12x3

Ring-LWE

5	$9x$	$12x^2$	$2x^3$	**a**

x

2	$11x$	$7x^2$	$7x^3$	**s**

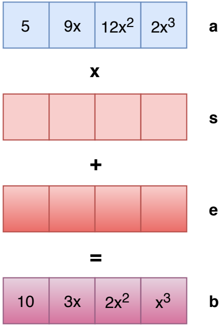
+

0	x	$-x^2$	x^3	**e**

=

10	$3x$	$2x^2$	x^3	**b**

Ring-LWE



NewHope

The secret vector s is sampled from the same distribution as e: binomial distribution centred around 0.

Let's look at (a simplified version of) the scheme.

NewHope: Simplified Scheme

Alice (server)		Bob (client)
Generate $a \in \mathbb{Z}_q^n, s, e \leftarrow \psi_{16}^n$		Generate $t, e', e'' \leftarrow \psi_{16}^n$
(Priv. key) s		(Priv. key) t
(Pub. key) $b = as + e$		
	$\xrightarrow{b,\, a}$	
		(Pub. key) $u = at + e'$
		Generate secret $\mathbf{k}$
		$c = bt + e'' + \mathbf{k}$
	$\xleftarrow{u,\, c}$	
$k' = c - us$		
$= bt + e'' + \mathbf{k} - (at + e')s$		
$= ast + et + e'' + \mathbf{k} - ats - e's$		
$= \mathbf{k} + et + e'' - e's$		
$\approx \mathbf{k}$		

NewHope

Disadvantages

Somewhat large messages ($\approx$ 2 KB each way).
Keys/noise should not be reused.
Ring structure is ignored in security analysis.

Overview

Scheme	Alice0 (ms)	Bob (ms)	Alice1 (ms)	Communication (bytes) A → B	B → A	Claimed security classical	quantum
RSA 3072-bit		0.09	4.49	387	384	128	
ECDH `nistp256`	0.366	0.698	0.331	32	32	128	
NewHope	0.112	0.164	0.034	1,824	2,048	229	206
SIDH	135	464	301	564	564	192	128
Frodo (LWE)	1.13	1.34	0.13	11,296	11,288	144	130

Google's experiment

Results experiment Google Chrome Canary

- Easy to implement
- Median connection latency only increased by a millisecond, however:
- Latency for the slowest 5% increased by 20ms
- Latency for the slowest 1% increased by 150ms
- Conclusion: data requirement of NewHope is moderately expensive for people on slower connections

Small latency is crucial for TLS.

Google's experiment

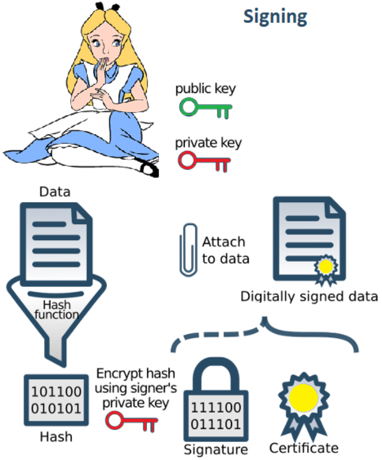
Migration to post-quantum

Hybrid method: use both classical and post-quantum.
Example: Google experimented with ECDH and NewHope.

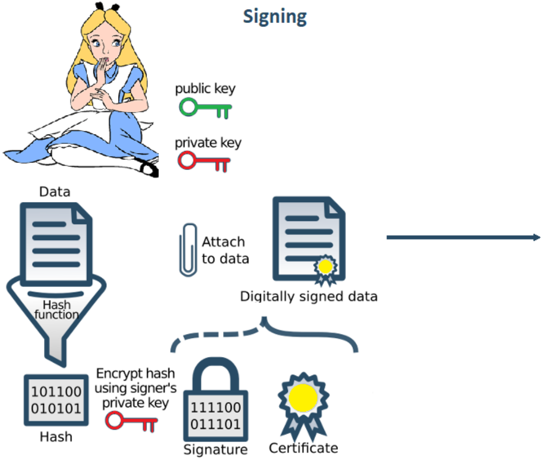
MQDSS

Digital Signature Scheme

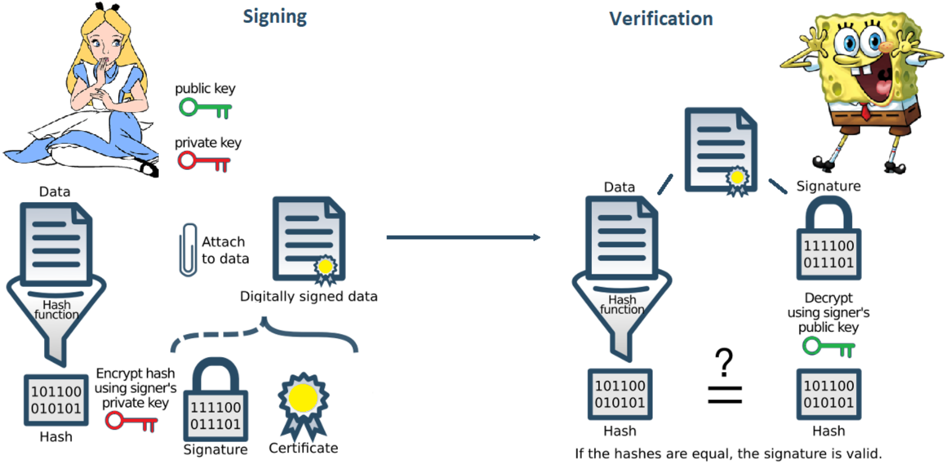
Digital Signatures



Digital Signatures



Digital Signatures



MQDSS in a nutshell

MQDSS

=

Multivariate **Q**uadratic **D**igital **S**ignature **S**cheme

MQDSS in a nutshell

MQDSS

=

Multivariate **Q**uadratic **D**igital **S**ignature **S**cheme

- First provably secure $\mathcal{MQ}$-based signature
  - Security proof in ROM
    (not the typical 'break and tweak' approach in $\mathcal{MQ}$ cryptography)
  - Reduction from (only!) $\mathcal{MQ}$ problem
  - Fiat-Shamir transform on $\mathcal{MQ}$-based 5-pass identification scheme

MQDSS in a nutshell

MQDSS

=

Multivariate **Q**uadratic **D**igital **S**ignature **S**cheme

- First provably secure $\mathcal{MQ}$-based signature
- Proposed ASIACRYPT 2016 [CHR$^+$16]
  - joint work with Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Peter Schwabe

MQDSS in a nutshell

MQDSS

=

Multivariate **Q**uadratic **D**igital **S**ignature **S**cheme

- First provably secure $\mathcal{MQ}$-based signature
- Proposed ASIACRYPT 2016 [CHR$^+$16]
- NIST candidate for standardization of Post-Quantum Cryptography
  - 'non-competition' started 30 Nov 2017

MQDSS in a nutshell

MQDSS

=

Multivariate **Q**uadratic **D**igital **S**ignature **S**cheme

- First provably secure $\mathcal{MQ}$-based signature
- Proposed ASIACRYPT 2016 [CHR$^+$16]
- NIST candidate for standardization of Post-Quantum Cryptography
- Parameters:

	k	public key (bytes)	secret key (bytes)	signature (bytes)
MQDSS-31-48	128	46	16	16534
MQDSS-31-64	192	64	24	34032

$\mathcal{MQ}$ (Multivariate Quadratic) Cryptosystems

no proof

MQ+IP+?

broken

- C*, sflash
- TTM, STS
- OV
- MQQ-SIG, MQQ-ENC
- QUARTZ
- PMI
- RSE, RSSE
- HFE, HFE-, HMFE
- ABC, ZHFE
- ELSA

Patched/new, still standing

- UOV, LUOV
- Rainbow, Cyclic Rainbow
- HFEv-, GUI, GeMSS
- DualModeMS

provable

MQ only

- QUAD stream cipher
- SSH ID schemes
- MQDSS (ROM)
- SOFIA (QROM)

$\mathcal{MQ}$ (Multivariate Quadratic) Cryptosystems

no proof

MQ+IP+?

broken

- C*, sflash
- TTM, STS
- OV
- MQQ-SIG, MQQ-ENC
- QUARTZ
- PMI
- RSE, RSSE
- HFE, HFE-, HMFE
- ABC, ZHFE
- ELSA

Patched/new, still standing

- UOV, LUOV
- Rainbow, Cyclic Rainbow
- HFEv-, GUI, GeMSS
- DualModeMS

signatures

provable

MQ only

- QUAD stream cipher
- SSH ID schemes
- MQDSS (ROM)
- SOFIA (QROM)

signatures

$\mathcal{MQ}$ problem

The function family $\mathcal{MQ}(n, m, \mathbb{F}_q)$: $\quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$

where $f_s(\mathbf{x}) = \sum_{i,j} a_{i,j}^{(s)} x_i x_j + \sum_i b_i^{(s)} x_i, \quad a_{i,j}^{(s)}, b_i^{(s)} \in \mathbb{F}_q$

$\mathcal{MQ}$ problem

The function family $\mathcal{MQ}(n, m, \mathbb{F}_q)$: $\quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$

$$\text{where } f_s(\mathbf{x}) = \sum_{i,j} a_{i,j}^{(s)} x_i x_j + \sum_i b_i^{(s)} x_i, \qquad a_{i,j}^{(s)},\ b_i^{(s)} \in \mathbb{F}_q$$

$\mathcal{MQ}$ problem

Given $\mathbf{F} \in \mathcal{MQ}(n, m, \mathbb{F}_q)$, $\mathbf{y} \in \mathbb{F}_q^m$, find – if any – $\mathbf{u} \in \mathbb{F}_q^n$ such that

$$\mathbf{F}(\mathbf{u}) = \mathbf{y}.$$

$\mathcal{MQ}$ problem

The function family $\mathcal{MQ}(n, m, \mathbb{F}_q)$: $\quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$

where $f_s(\mathbf{x}) = \sum_{i,j} a_{i,j}^{(s)} x_i x_j + \sum_i b_i^{(s)} x_i, \qquad a_{i,j}^{(s)}, \ b_i^{(s)} \in \mathbb{F}_q$

$\mathcal{MQ}$ problem

Given $\mathbf{F} \in \mathcal{MQ}(n, m, \mathbb{F}_q)$, $\mathbf{y} \in \mathbb{F}_q^m$, *find – if any –* $\mathbf{u} \in \mathbb{F}_q^n$ *such that*

$$\mathbf{F}(\mathbf{u}) = \mathbf{y}.$$

i.e., solve the system of equations:

$$\begin{cases} y_1 = & \sum_{i,j} a_{i,j}^{(1)} x_i x_j + \sum_i b_i^{(1)} x_i \\ & \vdots \\ y_m = & \sum_{i,j} a_{i,j}^{(m)} x_i x_j + \sum_i b_i^{(m)} x_i \end{cases}$$

$\mathcal{MQ}$ problem: numerical example

- Example parameters: $n = m = 3$, $\mathbb{F}_q = \mathbb{F}_5$

$\mathcal{MQ}$ problem: numerical example

- Example parameters: $n = m = 3$, $\mathbb{F}_q = \mathbb{F}_5$
- Random system of functions $\mathbf{F}$:

$$y_1 = 4x_1x_1 + 3x_1x_2 + 0x_1x_3 + x_2x_2 + 2x_2x_3 + x_3x_3 + 0x_1 + 2x_2 + 2x_3$$
$$y_2 = x_1x_1 + 2x_1x_2 + x_1x_3 + 0x_2x_2 + 3x_2x_3 + 4x_3x_3 + 0x_1 + 3x_2 + 2x_3$$
$$y_3 = 0x_1x_1 + x_1x_2 + 4x_1x_3 + 3x_2x_2 + 0x_2x_3 + x_3x_3 + 4x_1 + x_2 + 0x_3$$

$\mathcal{MQ}$ problem: numerical example

- Example parameters: $n = m = 3$, $\mathbb{F}_q = \mathbb{F}_5$
- Random system of functions $\mathbf{F}$:

$$
\begin{aligned}
y_1 &= 4x_1x_1 + 3x_1x_2 + 0x_1x_3 + x_2x_2 + 2x_2x_3 + x_3x_3 + 0x_1 + 2x_2 + 2x_3 \\
y_2 &= x_1x_1 + 2x_1x_2 + x_1x_3 + 0x_2x_2 + 3x_2x_3 + 4x_3x_3 + 0x_1 + 3x_2 + 2x_3 \\
y_3 &= 0x_1x_1 + x_1x_2 + 4x_1x_3 + 3x_2x_2 + 0x_2x_3 + x_3x_3 + 4x_1 + x_2 + 0x_3
\end{aligned}
$$

- 'Secret' input $\mathbf{x} = (1, 4, 3)$

$\mathcal{MQ}$ problem: numerical example

- Example parameters: $n = m = 3$, $\mathbb{F}_q = \mathbb{F}_5$
- Random system of functions $\mathbf{F}$:

$$
\begin{aligned}
y_1 &= 4x_1x_1 + 3x_1x_2 + 0x_1x_3 + x_2x_2 + 2x_2x_3 + x_3x_3 + 0x_1 + 2x_2 + 2x_3 \\
y_2 &= x_1x_1 + 2x_1x_2 + x_1x_3 + 0x_2x_2 + 3x_2x_3 + 4x_3x_3 + 0x_1 + 3x_2 + 2x_3 \\
y_3 &= 0x_1x_1 + x_1x_2 + 4x_1x_3 + 3x_2x_2 + 0x_2x_3 + x_3x_3 + 4x_1 + x_2 + 0x_3
\end{aligned}
$$

- 'Secret' input $\mathbf{x} = (1, 4, 3)$

$$
\begin{aligned}
y_1 &= 4 \cdot 1 \cdot 1 + 3 \cdot 1 \cdot 4 + 4 \cdot 4 + 2 \cdot 4 \cdot 3 + 3 \cdot 3 + 2 \cdot 4 + 2 \cdot 3 \\
y_2 &= 1 \cdot 1 + 2 \cdot 1 \cdot 4 + 1 \cdot 3 + 3 \cdot 4 \cdot 3 + 4 \cdot 3 \cdot 3 + 3 \cdot 4 + 2 \cdot 3 \\
y_3 &= 1 \cdot 4 + 4 \cdot 1 \cdot 3 + 3 \cdot 4 \cdot 4 + 3 \cdot 3 + 4 \cdot 1 + 4
\end{aligned}
$$

$\mathcal{MQ}$ problem: numerical example

- Example parameters: $n = m = 3$, $\mathbb{F}_q = \mathbb{F}_5$
- Random system of functions $\mathbf{F}$:

$$
\begin{aligned}
y_1 &= 4x_1x_1 + 3x_1x_2 + 0x_1x_3 + x_2x_2 + 2x_2x_3 + x_3x_3 + 0x_1 + 2x_2 + 2x_3 \\
y_2 &= x_1x_1 + 2x_1x_2 + x_1x_3 + 0x_2x_2 + 3x_2x_3 + 4x_3x_3 + 0x_1 + 3x_2 + 2x_3 \\
y_3 &= 0x_1x_1 + x_1x_2 + 4x_1x_3 + 3x_2x_2 + 0x_2x_3 + x_3x_3 + 4x_1 + x_2 + 0x_3
\end{aligned}
$$

- 'Secret' input $\mathbf{x} = (1, 4, 3)$

$$
\begin{aligned}
y_1 &= 4 \cdot 1 \cdot 1 + 3 \cdot 1 \cdot 4 + 4 \cdot 4 + 2 \cdot 4 \cdot 3 + 3 \cdot 3 + 2 \cdot 4 + 2 \cdot 3 = 79 \equiv 4 \\
y_2 &= 1 \cdot 1 + 2 \cdot 1 \cdot 4 + 1 \cdot 3 + 3 \cdot 4 \cdot 3 + 4 \cdot 3 \cdot 3 + 3 \cdot 4 + 2 \cdot 3 = 102 \equiv 2 \\
y_3 &= 1 \cdot 4 + 4 \cdot 1 \cdot 3 + 3 \cdot 4 \cdot 4 + 3 \cdot 3 + 4 \cdot 1 + 4 = 81 \equiv 1
\end{aligned}
$$

- 'Public' output $\mathbf{y} = (4, 2, 1)$

Hardness of $\mathcal{MQ}$

- Assume overdefined systems: $m \geqslant n$, $m \in \mathcal{O}(n)$
- **State of the art: Algebraic techniques with exhaustive search**

Hardness of $\mathcal{MQ}$

- Assume overdefined systems: $m \geqslant n,\ m \in \mathcal{O}(n)$
- **State of the art: Algebraic techniques with exhaustive search**
  - FXL [YC05], HybridF5 [BFS15], BooleanSolve [BFSS13], Crossbred [JV17]

$$\mathcal{O}(2^{0.792n}) \text{ for } q = 2, \text{ and } \mathcal{O}(2^{0.873n}) \text{ for } q > 2$$

Hardness of $\mathcal{MQ}$

- Assume overdefined systems: $m \geqslant n,\ m \in \mathcal{O}(n)$
- **State of the art: Algebraic techniques with exhaustive search**
  - FXL [YC05], HybridF5 [BFS15], BooleanSolve [BFSS13], Crossbred [JV17]

    $$\mathcal{O}(2^{0.792n}) \text{ for } q = 2, \text{ and } \mathcal{O}(2^{0.873n}) \text{ for } q > 2$$

  - Grover-ized quantum algorithms [FHK+17, BY18]

    $$\mathcal{O}(2^{0.462n}) \text{ for } q = 2$$

Hardness of $\mathcal{MQ}$

- Assume overdefined systems: $m \geqslant n$, $m \in \mathcal{O}(n)$
- **State of the art: Algebraic techniques with exhaustive search**
  - FXL [YC05], HybridF5 [BFS15], BooleanSolve [BFSS13], Crossbred [JV17]

    $$\mathcal{O}(2^{0.792n}) \text{ for } q = 2, \text{ and } \mathcal{O}(2^{0.873n}) \text{ for } q > 2$$

  - Grover-ized quantum algorithms [FHK+17, BY18]

    $$\mathcal{O}(2^{0.462n}) \text{ for } q = 2$$

  - Analysis in terms of classical gates, quantum gates, circuit depth [CHR+17]

Canonical Identification Schemes

$$
\begin{array}{lcr}
\mathcal{P}, \mathsf{sk} & & \mathcal{V}, \mathsf{pk} \\
\hline
\mathsf{com} \leftarrow_R \mathcal{P}_0(\mathsf{sk}) & & \\
 & \xrightarrow{\mathsf{com}} & \\
 & & \mathsf{ch} \leftarrow_R \mathsf{ChS}(1^k) \\
 & \xleftarrow{\mathsf{ch}} & \\
\mathsf{resp} \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}) & & \\
 & \xrightarrow{\mathsf{resp}} & \\
 & & b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})
\end{array}
$$

Informally:

1. Prover commits to some (randomized) value derived from sk
2. Verifier picks a challenge ‘ch’
3. Prover computes response ‘resp’
4. Verifier checks if response matches challenge

The Fiat-Shamir transform

IDS

$\mathcal{P}$		$\mathcal{V}$
$\mathsf{com} \leftarrow_R \mathcal{P}_0\ (\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \leftarrow_R \mathsf{ChS}\ (1^k)$
$\mathsf{resp} \leftarrow \mathcal{P}_1\ (\mathsf{sk}, \mathsf{com}, \mathsf{ch})$	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathsf{Vf}\ (\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

The Fiat-Shamir transform

IDS

$$
\begin{array}{lcl}
\mathcal{P}^r & & \mathcal{V}^r \\
\hline
\mathsf{com} \leftarrow_R \mathcal{P}_0{}^r(\mathsf{sk}) & & \\
& \xrightarrow{\quad\mathsf{com}\quad} & \\
& & \mathsf{ch} \leftarrow_R \mathsf{ChS}^r(1^k) \\
& \xleftarrow{\quad\mathsf{ch}\quad} & \\
\mathsf{resp} \leftarrow \mathcal{P}_1{}^r(\mathsf{sk}, \mathsf{com}, \mathsf{ch}) & & \\
& \xrightarrow{\quad\mathsf{resp}\quad} & \\
& & b \leftarrow \mathsf{Vf}^r(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})
\end{array}
$$

The Fiat-Shamir transform

IDS

$\mathcal{P}^r$		$\mathcal{V}^r$
$\mathsf{com} \leftarrow_R {\mathcal{P}_0}^r(\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
$\mathsf{ch} \leftarrow_R \mathsf{ChS}^r(1^k)$	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow {\mathcal{P}_1}^r(\mathsf{sk}, \mathsf{com}, \mathsf{ch})$	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathsf{Vf}^r(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

$\downarrow \quad \downarrow \quad \downarrow$

FS signature

Signer
$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$
$\mathsf{ch} \leftarrow H(m, \mathsf{com})$
$\mathsf{resp} \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch})$
output : $\sigma = (\mathsf{com}, \mathsf{resp})$

Verifier
$\mathsf{ch} \leftarrow H(m, \mathsf{com})$
$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$
output : b

A generalization: FS transform on 5-pass IDS

IDS

$\mathcal{P}$		$\mathcal{V}$
$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}_1}$	$\mathsf{ch}_1 \leftarrow_R \mathsf{ChS}_1(1^k)$
$\mathsf{resp}_1 \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1)$	$\xrightarrow{\mathsf{resp}_1}$	
	$\xleftarrow{\mathsf{ch}_2}$	$\mathsf{ch}_2 \leftarrow_R \mathsf{ChS}_2(1^k)$
$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2)$	$\xrightarrow{\mathsf{resp}_2}$	
		$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2)$

$\downarrow \quad \downarrow \quad \downarrow$

FS signature

Signer	**Verifier**
$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$	
$\mathsf{ch}_1 \leftarrow H_1(m, \mathsf{com})$	$\mathsf{ch}_1 \leftarrow H_1(m, \mathsf{com})$
$\mathsf{resp}_1 \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1)$	$\mathsf{ch}_2 \leftarrow H_2(m, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$
$\mathsf{ch}_2 \leftarrow H_2(m, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$	$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2)$
$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2)$	
	output : b
output : $\sigma = (\mathsf{com}, \mathsf{resp}_1, \mathsf{resp}_2)$	

A generalization: FS transform on 5-pass IDS

IDS

$\mathcal{P}$		
$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}_1}$	$\mathsf{ch}_1 \leftarrow_R \mathsf{ChS}$
$\mathsf{resp}_1 \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1)$	$\xrightarrow{\mathsf{resp}_1}$	
	$\xleftarrow{\mathsf{ch}_2}$	$\mathsf{ch}_2 \leftarrow_R \mathsf{ChS}$
$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2)$	$\xrightarrow{\mathsf{resp}_2}$	
		$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2)$

- Zero-Knowledge
  - no leakage of secret
- Sound
  - no cheating
- With extractor
  - secret extractable from valid transcripts

FS signature

Signer

$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$

$\mathsf{ch}_1 \leftarrow H_1(m, \mathsf{com})$

$\mathsf{resp}_1 \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1)$

$\mathsf{ch}_2 \leftarrow H_2(m, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$

$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2)$

output : $\sigma = (\mathsf{com}, \mathsf{resp}_1, \mathsf{resp}_2)$

Verifier

$\mathsf{ch}_1 \leftarrow H_1(m, \mathsf{com})$

$\mathsf{ch}_2 \leftarrow H_2(m, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$

$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2)$

output : b

A generalization: FS transform on 5-pass IDS

IDS

$\mathcal{P}$

$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}_1}$	$\mathsf{ch}_1 \leftarrow_R \mathsf{ChS}$
$\mathsf{resp}_1 \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1)$	$\xrightarrow{\mathsf{resp}_1}$	
	$\xleftarrow{\mathsf{ch}_2}$	$\mathsf{ch}_2 \leftarrow_R \mathsf{ChS}$
$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2)$	$\xrightarrow{\mathsf{resp}_2}$	
		$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2)$

- Zero-Knowledge
  - no leakage of secret
- Sound
  - no cheating
- With extractor
  - secret extractable from valid transcripts

FS signature

Signer

$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$

$\mathsf{ch}_1 \leftarrow H_1(m, \mathsf{com})$

$\mathsf{resp}_1 \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1)$

$\mathsf{ch}_2 \leftarrow H_2(m, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$

$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2)$

output : $\sigma = (\mathsf{com}, \mathsf{resp}_1, \mathsf{resp}_2)$

EU-CMA

$\mathsf{ch}_1 \leftarrow H_1(m, \mathsf{com})$

$\mathsf{ch}_2 \leftarrow H_2(m, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$

$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2)$

output : b

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
  - Analogously, consider DLP: $s = r_0 + r_1 \Rightarrow g^s = g^{r_0} \cdot g^{r_1}$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x}, \mathbf{y}) = \mathbf{F}(\mathbf{x} + \mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x}, \mathbf{y}) = \mathbf{F}(\mathbf{x} + \mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

$$\mathbf{s} = \mathbf{r}_0 + \mathbf{r}_1 \quad \Rightarrow \quad \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0, \mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1)$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x}, \mathbf{y}) = \mathbf{F}(\mathbf{x} + \mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

$$\begin{aligned} \mathbf{s} &= \mathbf{r}_0 + \mathbf{r}_1 \quad \Rightarrow \quad \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0, \mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1) \\ \mathbf{r}_0 &= \mathbf{t}_0 + \mathbf{t}_1 \quad \Rightarrow \quad \mathbf{G}(\mathbf{r}_0, \mathbf{r}_1) = \mathbf{G}(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1, \mathbf{r}_1) \end{aligned}$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x}, \mathbf{y}) = \mathbf{F}(\mathbf{x} + \mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

$$
\begin{aligned}
\mathbf{s} &= \mathbf{r}_0 + \mathbf{r}_1 &&\Rightarrow \quad \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0, \mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1) \\
\mathbf{r}_0 &= \mathbf{t}_0 + \mathbf{t}_1 &&\Rightarrow \quad \mathbf{G}(\mathbf{r}_0, \mathbf{r}_1) = \mathbf{G}(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1, \mathbf{r}_1) \\
\mathbf{F}(\mathbf{r}_0) &= \mathbf{e}_0 + \mathbf{e}_1 &&
\end{aligned}
$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

Cyber Security Week powered by The Hague Security Delta

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x},\mathbf{y}) = \mathbf{F}(\mathbf{x}+\mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

$$\begin{array}{rcccccl} \mathbf{s} = & \mathbf{r}_0 & + & \mathbf{r}_1 & \Rightarrow & & \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1) \\ \mathbf{r}_0 = & \mathbf{t}_0 & + & \mathbf{t}_1 & \Rightarrow & & \mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) = \mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) \\ \mathbf{F}(\mathbf{r}_0) = & \mathbf{e}_0 & + & \mathbf{e}_1 & & & \end{array}$$

$$\begin{aligned} \mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{e}_0 &= \mathbf{F}(\mathbf{s}) - \mathbf{F}(\mathbf{r}_1) - \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) - \mathbf{e}_1 \\ \mathbf{t}_0 &= \mathbf{r}_0 - \mathbf{t}_1 \\ \mathbf{e}_0 &= \mathbf{F}(\mathbf{r}_0) - \mathbf{e}_1 \end{aligned}$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x},\mathbf{y}) = \mathbf{F}(\mathbf{x}+\mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

$$\begin{aligned}
\mathbf{s} &= \mathbf{r}_0 + \mathbf{r}_1 &\Rightarrow\quad & \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1)\\
\alpha\mathbf{r}_0 &= \mathbf{t}_0 + \mathbf{t}_1 &\Rightarrow\quad & \alpha\mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) = \mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1)\\
\alpha\mathbf{F}(\mathbf{r}_0) &= \mathbf{e}_0 + \mathbf{e}_1 & &
\end{aligned}$$

$$\begin{aligned}
\mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{e}_0 &= \alpha(\mathbf{F}(\mathbf{s}) - \mathbf{F}(\mathbf{r}_1)) - \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) - \mathbf{e}_1\\
\mathbf{t}_0 &= \alpha\mathbf{r}_0 - \mathbf{t}_1\\
\mathbf{e}_0 &= \alpha\mathbf{F}(\mathbf{r}_0) - \mathbf{e}_1
\end{aligned}$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x},\mathbf{y}) = \mathbf{F}(\mathbf{x}+\mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

$$\begin{aligned}
\mathbf{s} &= \boxed{\mathbf{r}_0} + \boxed{\mathbf{r}_1} &&\Rightarrow\quad \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1)\\
\alpha\mathbf{r}_0 &= \mathbf{t}_0 + \mathbf{t}_1 &&\Rightarrow\quad \alpha\mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) = \mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1)\\
\alpha\mathbf{F}(\mathbf{r}_0) &= \mathbf{e}_0 + \mathbf{e}_1
\end{aligned}$$

Commit

$$\boxed{\begin{aligned}
\mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{e}_0\\
\mathbf{t}_0\\
\mathbf{e}_0
\end{aligned}}
\begin{aligned}
&= \alpha(\mathbf{F}(\mathbf{s}) - \mathbf{F}(\mathbf{r}_1)) - \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) - \mathbf{e}_1\\
&= \alpha\mathbf{r}_0 - \mathbf{t}_1\\
&= \alpha\mathbf{F}(\mathbf{r}_0) - \mathbf{e}_1
\end{aligned}$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x},\mathbf{y}) = \mathbf{F}(\mathbf{x}+\mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

If ch =0, reveal

$$\begin{aligned} \mathbf{s} &= \boxed{\mathbf{r}_0} + \boxed{\mathbf{r}_1} &\Rightarrow\quad & \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1) \\ \alpha\mathbf{r}_0 &= \mathbf{t}_0 + \mathbf{t}_1 &\Rightarrow\quad & \alpha\mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) = \mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) \\ \alpha\mathbf{F}(\mathbf{r}_0) &= \mathbf{e}_0 + \mathbf{e}_1 & & \end{aligned}$$

Commit

$$\begin{aligned} \mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{e}_0 &= \alpha(\mathbf{F}(\mathbf{s}) - \mathbf{F}(\mathbf{r}_1)) - \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) - \mathbf{e}_1 \\ \mathbf{t}_0 &\stackrel{?}{=} \alpha\mathbf{r}_0 - \mathbf{t}_1 \\ \mathbf{e}_0 &\stackrel{?}{=} \alpha\mathbf{F}(\mathbf{r}_0) - \mathbf{e}_1 \end{aligned}$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

- **Key technique:** cut-and-choose for $\mathcal{MQ}$
- **Key tool:** Bilinear map $\mathbf{G}(\mathbf{x},\mathbf{y}) = \mathbf{F}(\mathbf{x}+\mathbf{y}) - \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})$
- **The idea:**

If ch $=0$, reveal **If ch $=1$, reveal**

$$
\begin{array}{rcccccl}
\mathbf{s} & = & \mathbf{r}_0 & + & \mathbf{r}_1 & \Rightarrow & \mathbf{F}(\mathbf{s}) = \mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) + \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1) \\
\alpha\mathbf{r}_0 & = & \mathbf{t}_0 & + & \mathbf{t}_1 & \Rightarrow & \alpha\mathbf{G}(\mathbf{r}_0,\mathbf{r}_1) = \mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) \\
\alpha\mathbf{F}(\mathbf{r}_0) & = & \mathbf{e}_0 & + & \mathbf{e}_1 & &
\end{array}
$$

Commit

$$
\begin{array}{rcl}
\mathbf{G}(\mathbf{t}_0,\mathbf{r}_1) + \mathbf{e}_0 & \stackrel{?}{=} & \alpha(\mathbf{F}(\mathbf{s}) - \mathbf{F}(\mathbf{r}_1)) - \mathbf{G}(\mathbf{t}_1,\mathbf{r}_1) - \mathbf{e}_1 \\
\mathbf{t}_0 & \stackrel{?}{=} & \alpha\mathbf{r}_0 - \mathbf{t}_1 \\
\mathbf{e}_0 & \stackrel{?}{=} & \alpha\mathbf{F}(\mathbf{r}_0) - \mathbf{e}_1
\end{array}
$$

Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS [KSH11]

$\mathcal{P}(\mathbf{F},\mathbf{v},\mathbf{s})$		$\mathcal{V}(\mathbf{F},\mathbf{v})$
$\mathbf{r}_0,\mathbf{t}_0 \leftarrow_R \mathbb{F}_q^n, \mathbf{e}_0 \leftarrow_R \mathbb{F}_q^m$		
$\mathbf{r}_1 \leftarrow \mathbf{s}-\mathbf{r}_0$		
$c_0 \leftarrow Com(\mathbf{r}_0,\mathbf{t}_0,\mathbf{e}_0)$		
$c_1 \leftarrow Com(\mathbf{r}_1,\mathbf{G}(\mathbf{t}_0,\mathbf{r}_1)+\mathbf{e}_0)$	$\xrightarrow{(c_0,c_1)}$	
	$\xleftarrow{\alpha}$	$\alpha \leftarrow_R \mathbb{F}_q$
$\mathbf{t}_1 \leftarrow \alpha\mathbf{r}_0-\mathbf{t}_0$		
$\mathbf{e}_1 \leftarrow \alpha\mathbf{F}(\mathbf{r}_0)-\mathbf{e}_0$	$\xrightarrow{\text{resp}_1=(\mathbf{t}_1,\mathbf{e}_1)}$	
	$\xleftarrow{\text{ch}_2}$	$\text{ch}_2 \leftarrow_R \{0,1\}$
If $\text{ch}_2=0,\ \text{resp}_2 \leftarrow \mathbf{r}_0$		
Else $\text{resp}_2 \leftarrow \mathbf{r}_1$	$\xrightarrow{\text{resp}_2}$	
		If $\text{ch}_2=0$, Parse $\text{resp}_2=\mathbf{r}_0$, check
		$c_0 \stackrel{?}{=} Com(\mathbf{r}_0,\alpha\mathbf{r}_0-\mathbf{t}_1,\alpha\mathbf{F}(\mathbf{r}_0)-\mathbf{e}_1)$
		Else Parse $\text{resp}_2=\mathbf{r}_1$, check
		$c_1 \stackrel{?}{=} Com(\mathbf{r}_1,\alpha(\mathbf{v}-\mathbf{F}(\mathbf{r}_1))-\mathbf{G}(\mathbf{t}_1,\mathbf{r}_1)-\mathbf{e}_1)$

MQDSS

KGen()

$\mathsf{sk} \leftarrow_R \{0,1\}^k$,

Generate seeds from sk, expand to $\mathbf{F}, \mathbf{s}$

$\mathbf{v} \leftarrow \mathbf{F}(\mathbf{s})$, $\mathsf{pk} := (S_{\mathbf{F}}, \mathbf{v})$

Sign(sk, M)

Obtain $\mathbf{F}, \mathbf{s}, \mathbf{v}$, pk as in KGen

Sample r vectors $\mathbf{r}$, $\mathbf{t}$, $\mathbf{e}$ from seed, M

Perform r parallel rounds of IDS

$\mathsf{com} = (\mathsf{com}_0, \mathsf{com}_1) \leftarrow \mathcal{P}_0(\mathsf{sk})$

$\sigma_0 \leftarrow \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}||\ldots||\mathsf{com}^{(r)})$

$\mathsf{ch}_1 \leftarrow H_1(M, \sigma_0)$

$\sigma_1 \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_1)$

$\mathsf{ch}_2 \leftarrow H_2(M, \sigma_0, \mathsf{ch}_1, \sigma_1)$

$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{sk}, \sigma_0, \mathsf{ch}_1, \sigma_1, \mathsf{ch}_2)$

$\sigma_2 \leftarrow (\mathsf{resp}_2,$ non reconstruct. com parts$)$

output : $\sigma = (\sigma_0, \sigma_1, \sigma_2)$

Vf(pk, σ, M)

Expand seed $S_{\mathbf{F}}$ to $\mathbf{F}$

$\mathsf{ch}_1 \leftarrow H_1(M, \sigma_0)$

$\mathsf{ch}_2 \leftarrow H_2(M, \sigma_0, \mathsf{ch}_1, \sigma_1)$

Reconstruct missing commitments

$\sigma_0' \leftarrow \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}||\ldots||\mathsf{com}^{(r)})$

output : $\sigma_0' == \sigma_0$

Implementation considerations

- SHAKE-256 for commitments / hashes
  - Match output length to k

Implementation considerations

- SHAKE-256 for commitments / hashes
  - Match output length to k
- Mathematically straight-forward
  - Multiplications and additions in $\mathbb{F}_{31}$
  - Fast arithmetic

Implementation considerations

- SHAKE-256 for commitments / hashes
  - Match output length to k
- Mathematically straight-forward
  - Multiplications and additions in $\mathbb{F}_{31}$
  - Fast arithmetic
- Very natural internal parallelism

Implementation considerations

- SHAKE-256 for commitments / hashes
  - Match output length to k
- Mathematically straight-forward
  - Multiplications and additions in $\mathbb{F}_{31}$
  - Fast arithmetic
- Very natural internal parallelism
- Naively constant-time and slow
  - But still constant-time when optimized

Implementation considerations

- SHAKE-256 for commitments / hashes
  - Match output length to k
- Mathematically straight-forward
  - Multiplications and additions in $\mathbb{F}_{31}$
  - Fast arithmetic
- Very natural internal parallelism
- Naively constant-time and slow
  - But still constant-time when optimized
- Expanding $\mathbf{F}$ is memory-intensive (134 KiB)
  - Problematic on small devices

MQDSS parameters and performance

- Parameters for NIST ‘non-competition’:
  - (Loose reduction $\Rightarrow$ consider best known attacks)

	security (bits)	public key (bytes)	secret key (bytes)	signature (bytes)
MQDSS-31-48	128	46	16	16.5K
MQDSS-31-64	192	64	24	34K

MQDSS parameters and performance

- Parameters for NIST 'non-competition':
  - (Loose reduction $\Rightarrow$ consider best known attacks)

	security (bits)	public key (bytes)	secret key (bytes)	signature (bytes)
MQDSS-31-48	128	46	16	16.5K
MQDSS-31-64	192	64	24	34K

- Benchmarking on 3.5 GHz Intel Core i7-4770K CPU

	keygen	signing	verification
MQDSS-31-48	1 302K	26 500K	19 674K
MQDSS-31-64	2 769K	84 615K	63 210K

MQDSS parameters and performance

- Parameters for NIST 'non-competition':
  - (Loose reduction $\Rightarrow$ consider best known attacks)

	security (bits)	public key (bytes)	secret key (bytes)	signature (bytes)
MQDSS-31-48	128	46	16	16.5K
MQDSS-31-64	192	64	24	34K

- Other $\mathcal{MQ}$ based NIST candidates (128 bits security)

	public key	secret key	signature
Rainbow-Ia	148.5K	97.9K	64
DualModeMS	528	1.8M	32K
Gui-184	416.3K	19.1K	45

MQDSS parameters and performance

- Parameters for NIST 'non-competition':
  - (Loose reduction $\Rightarrow$ consider best known attacks)

	security (bits)	public key (bytes)	secret key (bytes)	signature (bytes)
MQDSS-31-48	128	46	16	16.5K
MQDSS-31-64	192	64	24	34K

- Other NIST candidates (128 bits security)

	public key	secret key	signature
Dilithium	1.2K	2.8K	2K
qTesla-p-I	14.8K	4.6K	2.8K
Sphincs+128s	32	64	8K
Picnic-L1-FS	32	16	34K

References I

Magali Bardet, Jean-Charles Faugère, and Bruno Salvy.
On the complexity of the F5 Gröbner basis algorithm.
Journal of Symbolic Computation, 70:49–70, 2015.
https://hal.inria.fr/hal-01064519/document.

Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer.
On the complexity of solving quadratic boolean systems.
Journal of Complexity, 29(1):53–75, 2013.
www-polsys.lip6.fr/~jcf/Papers/BFSS12.pdf.

Daniel J. Bernstein and Bo-Yin Yang.
Asymptotically faster quantum algorithms to solve multivariate quadratic equations.
In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 487–506, Cham, 2018. Springer International Publishing.

References II

Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe.
From 5-pass $\mathcal{MQ}$-based identification to $\mathcal{MQ}$-based signatures.
In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, volume 10032 of *LNCS*, pages 135–165. Springer, 2016.
http://eprint.iacr.org/2016/708.

Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe.
MQDSS.
Submission to NIST's post-quantum crypto standardization project, 2017.

Jean-Charles Faugère, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret.
Fast quantum algorithm for solving multivariate quadratic equations.
IACR Cryptology ePrint Archive, 2017:1236, 2017.

References III

Antoine Joux and Vanessa Vitse.
A crossbred algorithm for solving boolean polynomial systems.
Cryptology ePrint Archive, Report 2017/372, 2017.
http://eprint.iacr.org/2017/372.

Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari.
Public-key identification schemes based on multivariate quadratic polynomials.
In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *LNCS*, pages 706–723. Springer, 2011.
https://www.iacr.org/archive/crypto2011/68410703/68410703.pdf.

Bo-Yin Yang and Jiun-Ming Chen.
All in the XL family: Theory and practice.
In Choon sik Park and Seongtaek Chee, editors, *Information Security and Cryptology – ICISC 2004*, pages 67–86. Springer, 2005.
http://by.iis.sinica.edu.tw/by-publ/recent/xxl.pdf.

CONTACT

Daniël Kuijsters

- daniel.kuijsters@compumatica.com

Tim Weenink

- tim.weenink@compumatica.com

Simona Samardjiska

- simonas@cs.ru.nl

Cyber Security Week

powered by **The Hague Security Delta**