Post-Quantum Cryptography in Practice: a Review

Adrian Valente

LASEC

January 18 2018

Cryptography in danger?

- Public-key crypto based on integer factorization and discrete log
- Broken by a quantum computer

Bricks of Crypto

- Symmetric Encryption
- Hash Functions
- PRNG
- KDF
- Signatures
- Key Establishment

Bricks of Crypto

Symmetric Encryption

Hash Functions

PRNG

KDF

DANGER!

Signatures

Key Establishment

Cryptography in danger?

- 2 primitives at stake: **signature schemes** and **key encapsulation mechanisms**
- Broken by Shor's algorithm in poly time on a quantum computer
- Even if no quantum computer: lack of diversity harmful

What is Post-Quantum Cryptography?

Post-Quantum Cryptography = quest for a KEM and signature scheme not broken by a quantum algorithm

The Post-Quantum Crypto Zoo

Five main classes of post-quantum cryptosystems:

- multivariate cryptosystems,
- lattice-based cryptosystems,
- code-based cryptosystems,
- hash-based signature schemes,
- supersingular elliptic curve isogeny Diffie-Hellman.

	Multivariate	Lattice	Codes	Hash	Isogeny
Signatures	Yes	Yes	Possible	Yes	Possible
KEM	Possible	Yes	Yes	No	Yes

Outline

Outline

For each class, we will see

- What problem is it based on?
- How does it work?
- How did it evolve?
- What are its performances?
- Some key points

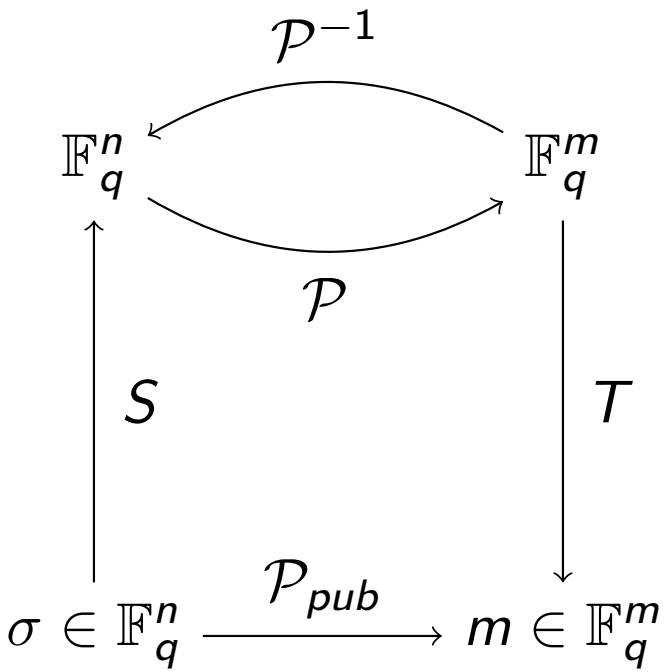
MQ-problem

Find the root of several multivariate polynomials.

- INPUT: $p_1, \cdots, p_m \in \mathbb{F}_q[X_1, \cdots, X_n]$, $m, n \in \mathbb{N}$, q a prime power.
- OUTPUT: $\mathbf{x}^* \in \mathbb{F}_q^n$ st. $p_1(\mathbf{x}^*) = \cdots = p_n(\mathbf{x}^*) = 0$.

Example: $p_1 = x_1^2 + 3x_1x_2 - x_2 + 4, p_2 = \cdots$

Designing a multivariate scheme



History

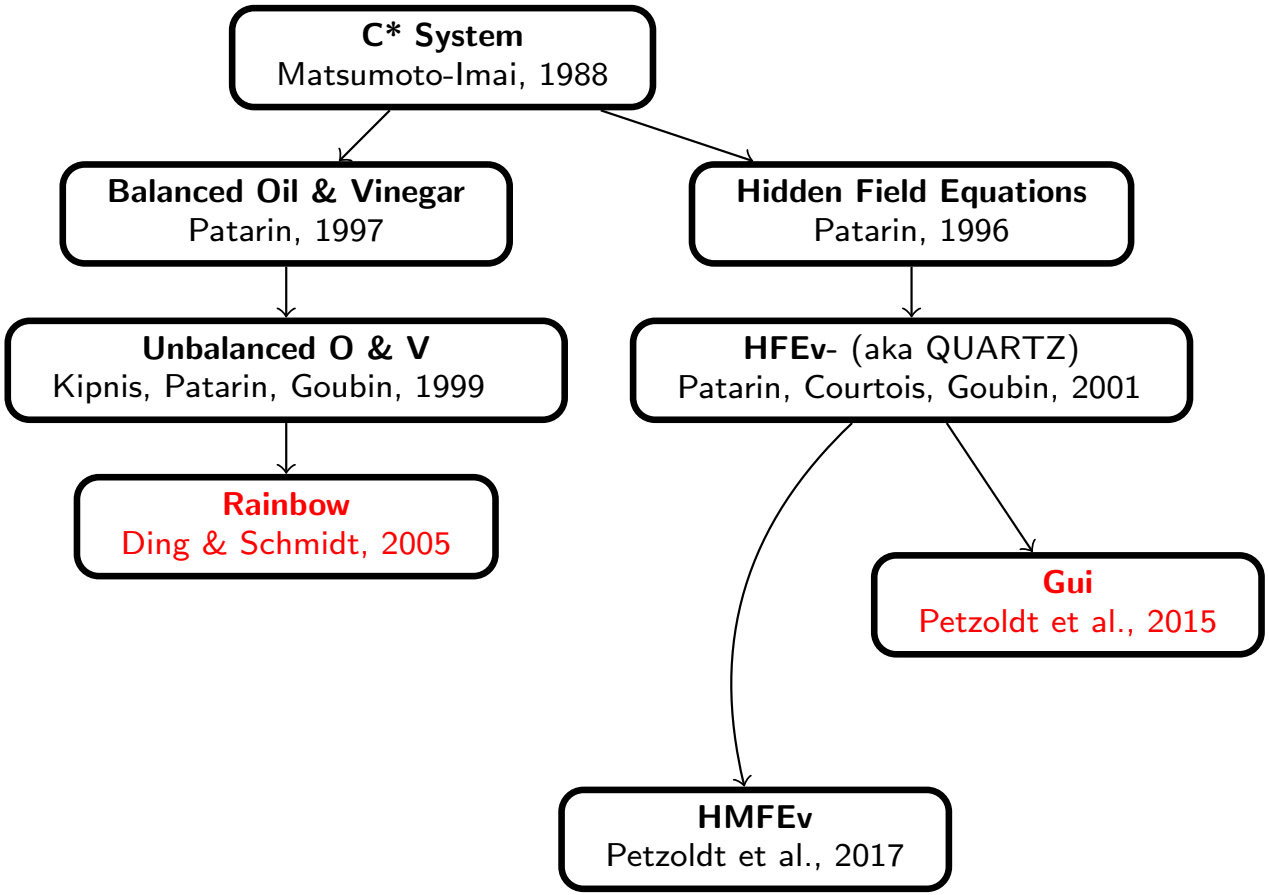


Figure: Evolution of multivariate cryptography

Performance

	Rainbow	Gui-96	HMFEv
Public Key Size (Kbytes)	15	61.6	22.5
Signature Size (bytes)	264	126	218
Signature Time (ms)	-	0.07	0.20
Verification Time (ms)	-	0.02	0.01

Table: Performance comparison of multivariate signature schemes for a security estimate of 80 bits

Key points

Very small signature sizes
Very simple implementation

Mostly useful for signatures
Long history of broken schemes, very developed crypt-analysis

Quite big public keys (more than 10 KB)
No formal security proof

LWE problem

- SETTING: - $p, n \in \mathbb{N}$,
  - $\mathbf{s} \in \mathbb{Z}_p^n$,
  - $\mathbf{a}_1, \cdots, \mathbf{a}_m \in \mathbb{Z}_p^n$ uniformly and independently sampled,
  - $e_1, \cdots, e_m \in \mathbb{Z}_p$, sampled from χ.
  - $\langle \mathbf{s}, \mathbf{a}_1 \rangle + e_1 = b_1 \pmod{p}, \cdots, \langle \mathbf{s}, \mathbf{a}_m \rangle + e_m = b_m \pmod{p}$
- INPUT: $\mathbf{a}_1, \cdots, \mathbf{a}_m,\ b_1, \cdots, b_m$
- OUTPUT: $\mathbf{s}$

Running an LWE-based cryptosystem

- Key generation: random $\mathbf{s} \in {\mathbb{Z}_p}^n$ and $\mathbf{A} \in {\mathbb{Z}_p}^{n \times n}$. Sample $\mathbf{e}$ from χ on ${\mathbb{Z}_p}^n$ and compute the values $\mathbf{b} = \mathbf{As} + \mathbf{e}$.
  Private key : $\mathbf{s}$,
  Public key : $(\mathbf{A}, \mathbf{b})$.

Example with $p = 32$, $n = 2$:
$\mathbf{A} = \left(\begin{smallmatrix} 16 & 9 \\ 23 & 0 \end{smallmatrix}\right)$, $\mathbf{s} = (5\ 27)^T$, $\mathbf{e} = (1, 31)^T$, $\mathbf{b} = (4\ 18)^T$.

Running an LWE-based cryptosystem

- Encryption: To encrypt a single bit $\mu \in \{0,1\}$: choose a random $\mathbf{x} \in \{0,1\}^n$, compute $\mathbf{u} = \mathbf{x}^T\mathbf{A}$ and $v = \mathbf{x}^T\mathbf{b} + \lfloor \frac{p}{2} \rceil \mu$.
  The ciphertext is $(\mathbf{u}, v)$.

Example: $mu = 1$, $\mathbf{x} = (1\ 0)^T$,
$\mathbf{u} = (16\ 9)$, $v = 4 + 16 = 20$.

Running an LWE-based cryptosystem

- Decryption: Compute $-\langle \mathbf{u}, \mathbf{s} \rangle + v = \mathbf{x}^T \mathbf{e} + \lfloor \frac{p}{2} \rceil \mu$, which permits to distinguish between the two possible values of μ provided that $|\mathbf{x}^T \mathbf{e}| \leq \lfloor \frac{p}{4} \rceil$.

Example: We compute $-3 + 30 = 17$ which should be near $\lfloor \frac{p}{2} \rceil \mu = 16$. Indeed, it is the case!

History

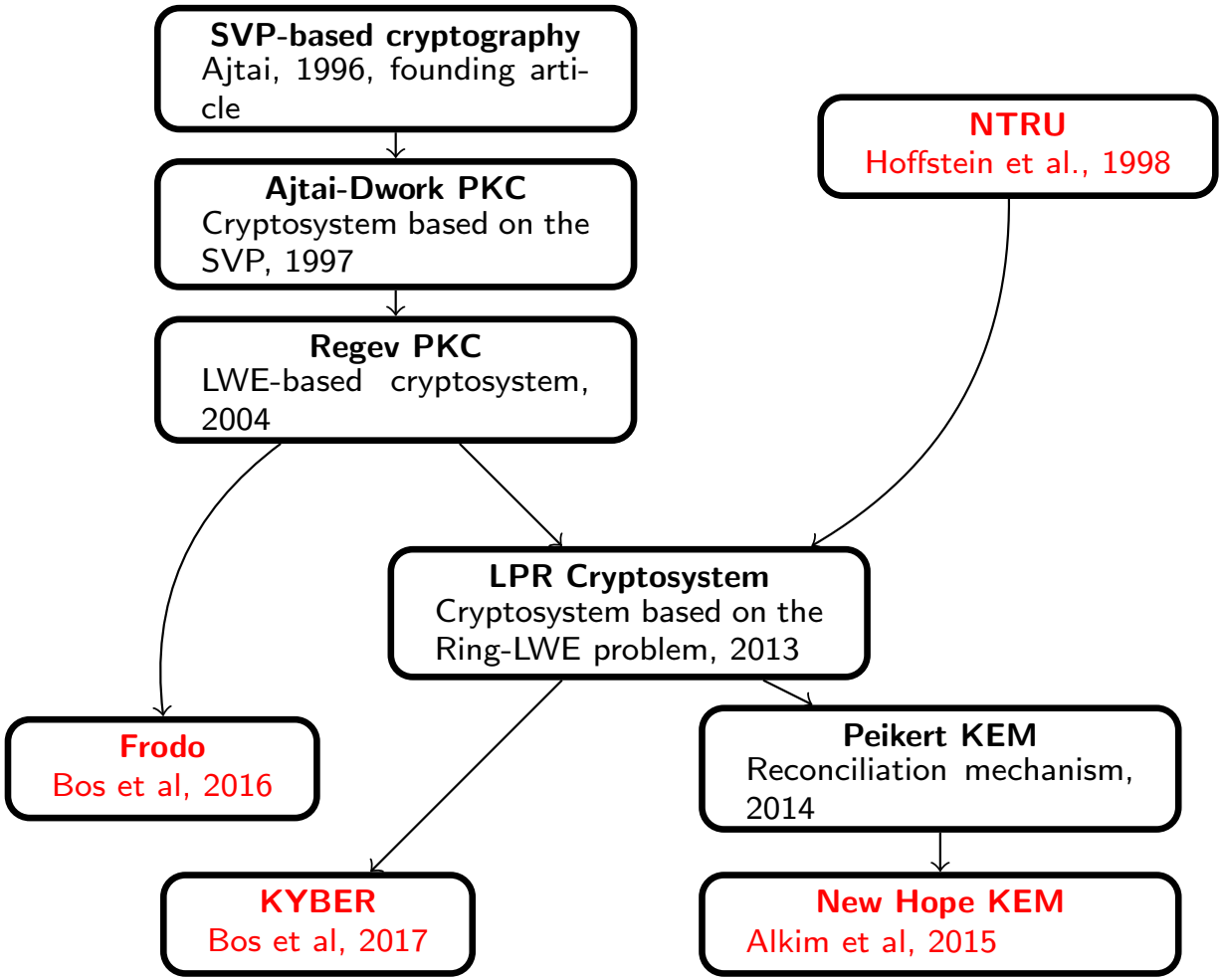


Figure: Evolution of lattice-based cryptosystems

Performance

	Kyber	New Hope	Frodo	NTRUEncrypt
Public Key Size (bytes)	1088	1824	11296	1027
Ciphertext size (Message size) (bytes)	1184 (32)	2048 (32)	11288	1022 (32)
Key generation time (ms)	0.15	0.11	1.13	3.02
Encryption time (ms)	0.19	0.16	1.34	0.78
Decryption time (ms)	0.21	0.04	0.13	0.16
Failure probability	2^{-142}	2^{-61}	$2^{-38.9}$	2^{-195}
Security estimate (bits)	161	128	130	128

Table: Performance comparison of lattice-based encryption algorithms

Key points

Versatile: both very good KEMs and signatures
Fast, small public keys and ciphertexts
Diversity of computational problems
Usually theoretical reductions

Algorithms have a probability of failure (can be made as small as desired)

Little understanding of cryptanalysis

Linear codes

Let $\mathbb{K} = \mathbb{F}_q$ be a finite field, and $E = \mathbb{K}^n$ be a vector space.

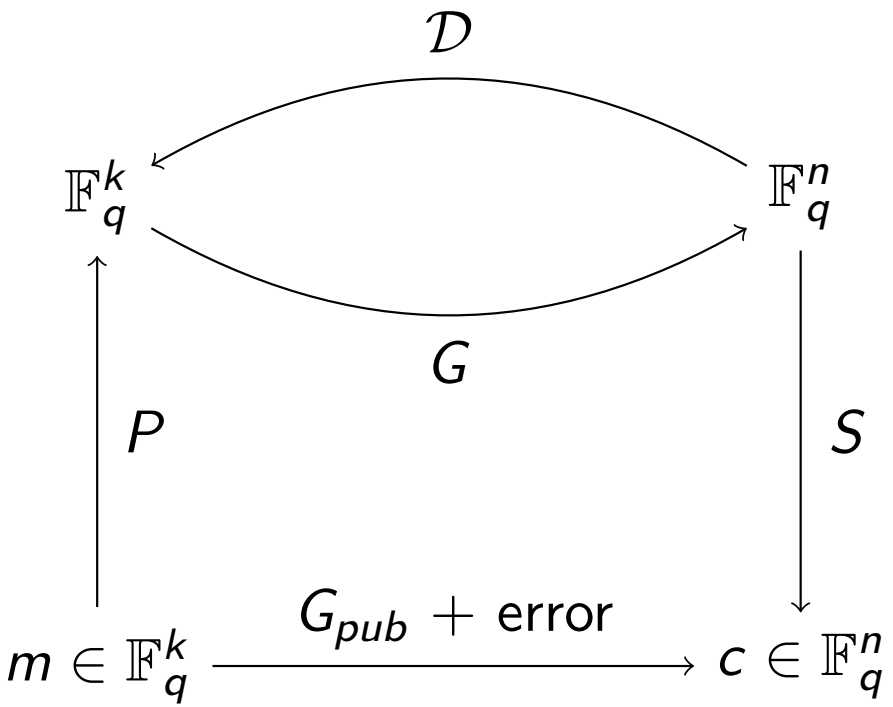
Definition

A $[n, k]$-linear code over E is a linear subspace $\mathcal{C}$ of E of dimension k.

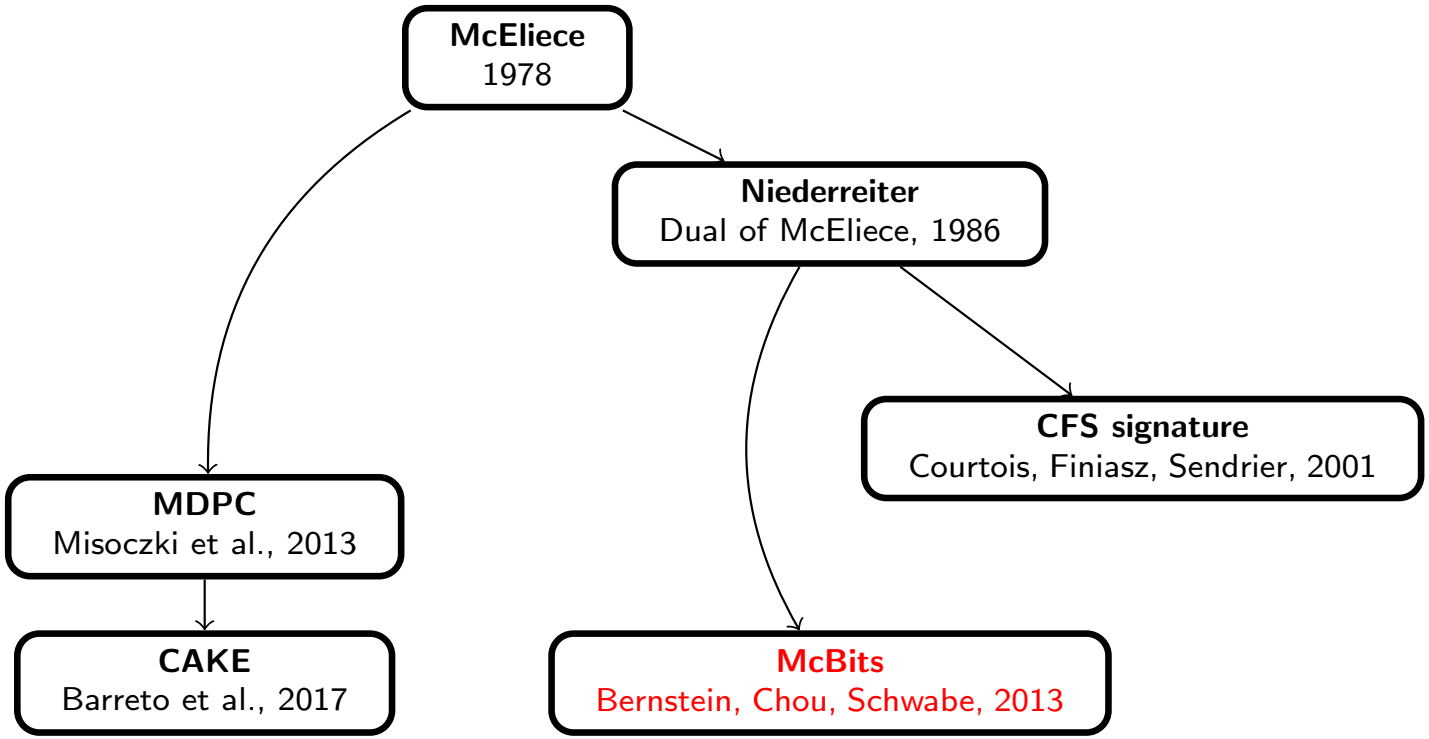
Definition

We say that $\mathcal{C}$ is t-error correcting if there exists a function $\mathcal{D} : E \to \mathcal{C}$ such that for all $x \in \mathcal{C}$, $e \in E$ such that $wt(e) \leq t$, then $\mathcal{D}(x + e) = x$.

Design of a code-based cryptosystem



History


McEliece
1978
Niederreiter
Dual of McEliece, 1986
CFS signature
Courtois, Finiasz, Sendrier, 2001
MDPC
Misoczki et al., 2013
CAKE
Barreto et al., 2017
McBits
Bernstein, Chou, Schwabe, 2013

	McBits
Public Key Size (Kilobytes)	311
Key Generation Time (ms)	169
Encryption Time (ms)	0.08
Decryption Time (ms)	0.27

Key points

Very deeply studied
Very fast

NOT easy to change the underlying code
Lots of broken cryptosystems

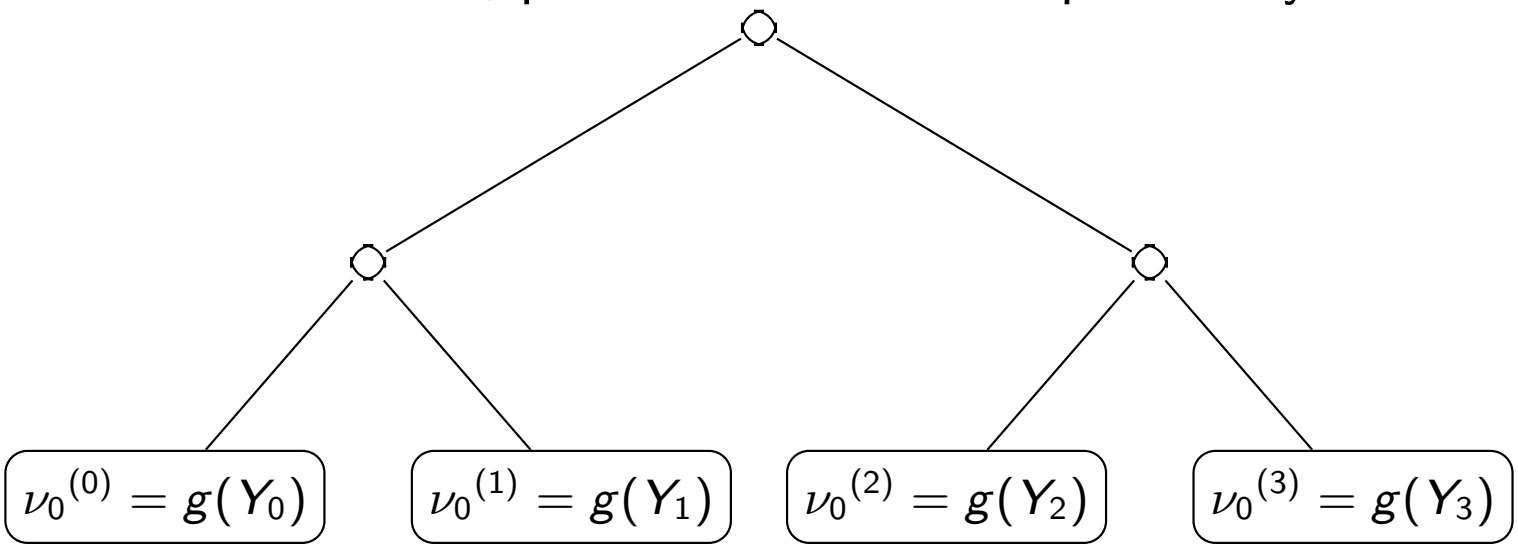
Very large public keys (sometimes 100KB)

One-time signatures

- Key Generation: $f : \{0,1\}^* \to \{0,1\}^n$ be a hash function. Let ${x_0}^{(0)}, {x_0}^{(1)}, {x_1}^{(0)}, {x_1}^{(1)}, \ldots, {x_{n-1}}^{(0)}, {x_{n-1}}^{(1)} \in \{0,1\}^n$ (=private key).
  Let $y_i^{(j)} = f(x_i^{(j)})$ (=public key)
- Signature: to sign $m \in \{0,1\}^n$, compute a hash $h = h_0 \cdots h_{n-1}$ of m and send $(x_i^{(h_i)})$
- Verification: Check that $f(s_i) = y_i^{(h_i)}$

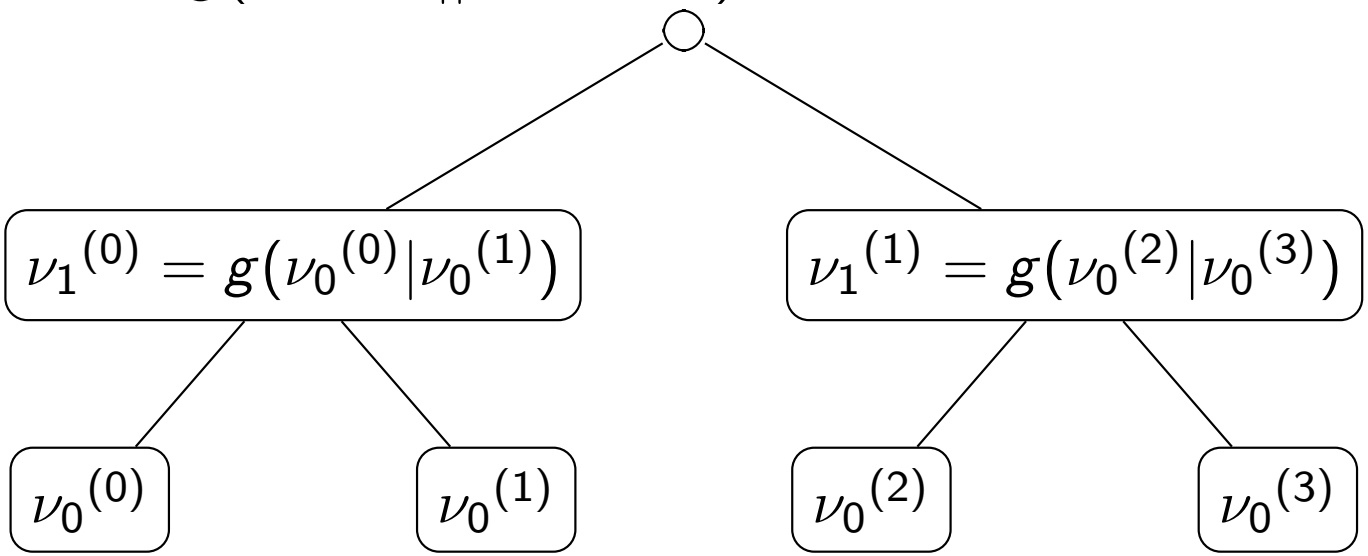
Merkle trees

Let $g : \{0,1\}^* \to \{0,1\}^n$ be a hash function.
First generate some one-time signature key-pairs:
$(X_0, Y_0), \cdots, (X_3, Y_3)$.
On the lowest level, put the hash of their public keys.

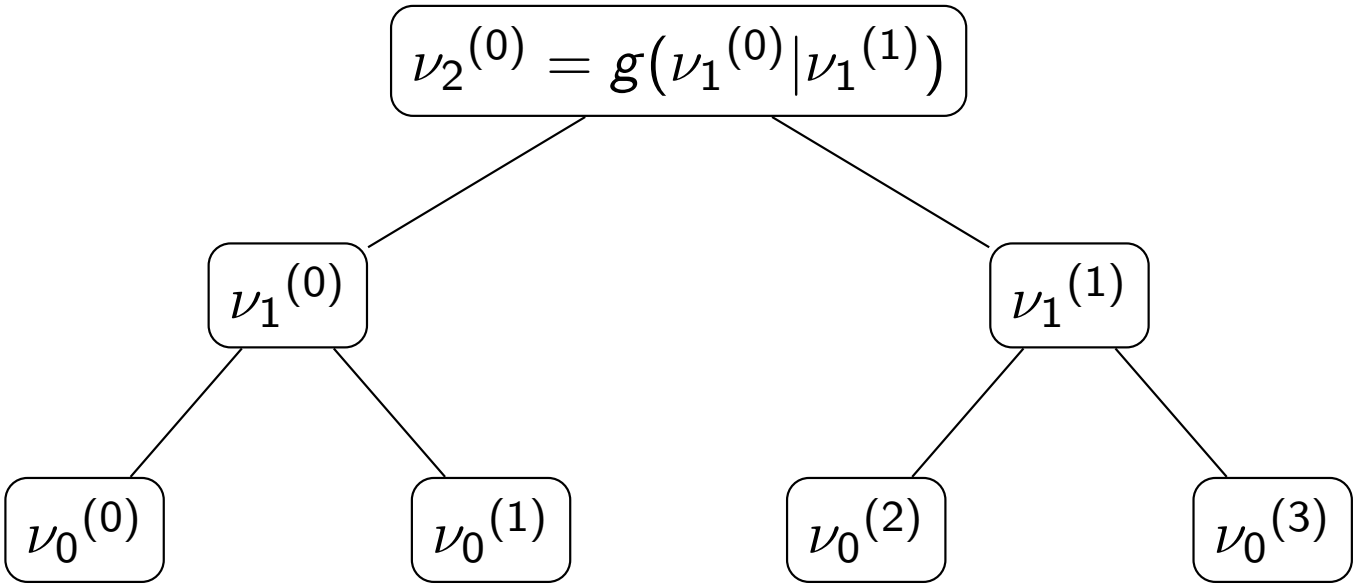


Merkle trees

At each level above, merge children with a hash function:
$\nu_i{}^{(j)} = g(\nu_{i-1}{}^{(2j)} || \nu_{i-1}{}^{(2j+1)})$

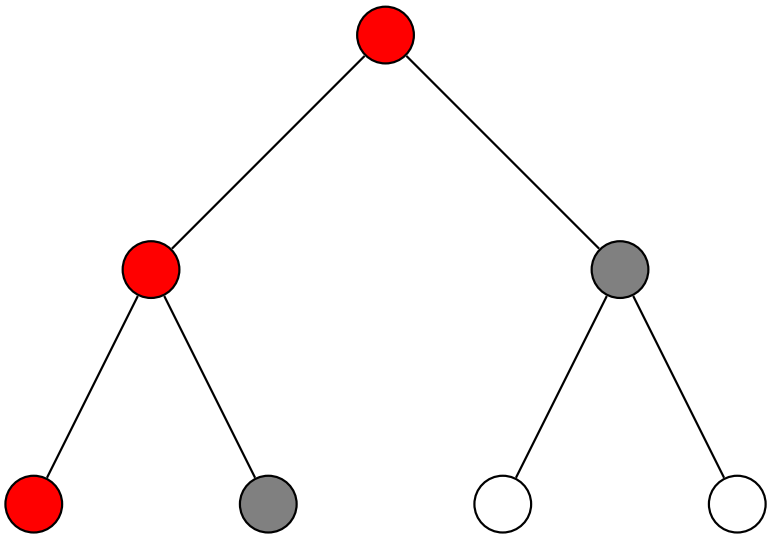


Merkle trees

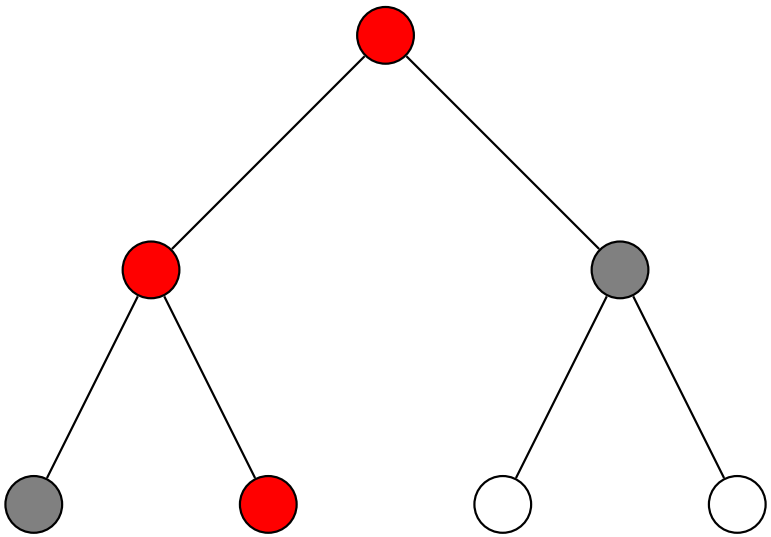


Signing & Verification

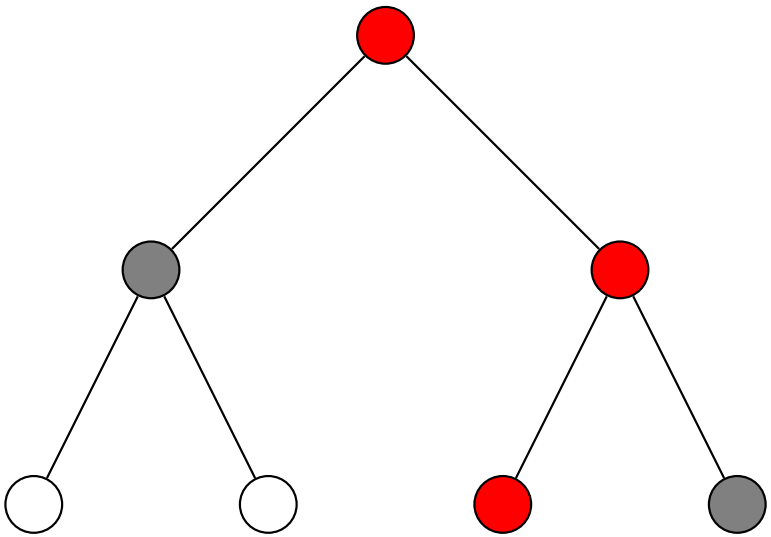
The signer has to provide an authentication path, for the verifier to compute all the hashes to the top (red path).



Signing & Verification



Signing & Verification



Stateful or stateless?

- Problem: order of traversal of the tree
- Maintaining a state is DANGEROUS
- SPHINCS : stateless hash-based signature scheme

Performance

	SPHINCS
Public Key Size (Kbytes)	1
Signature Size (bytes)	41000
Signature Time (Mcycles)	51.63
Verification Time (Mcycles)	1.45

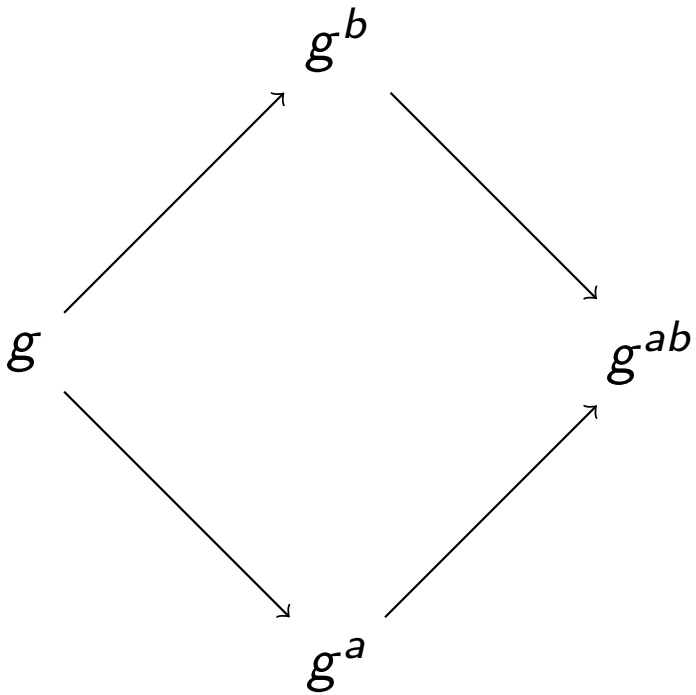
Key points

Very strong security guarantees

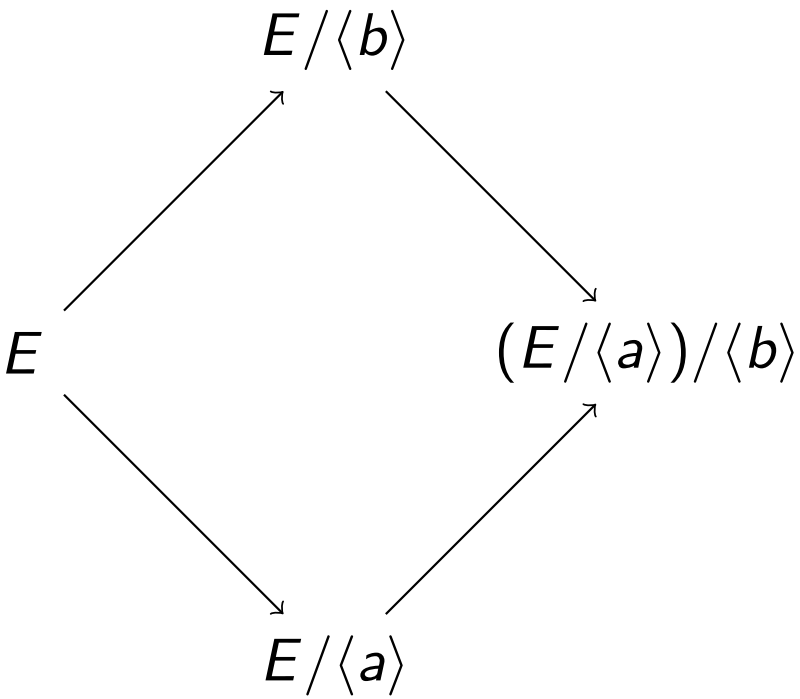
Problem of state

Huge signatures
Limited number of signatures for a public key

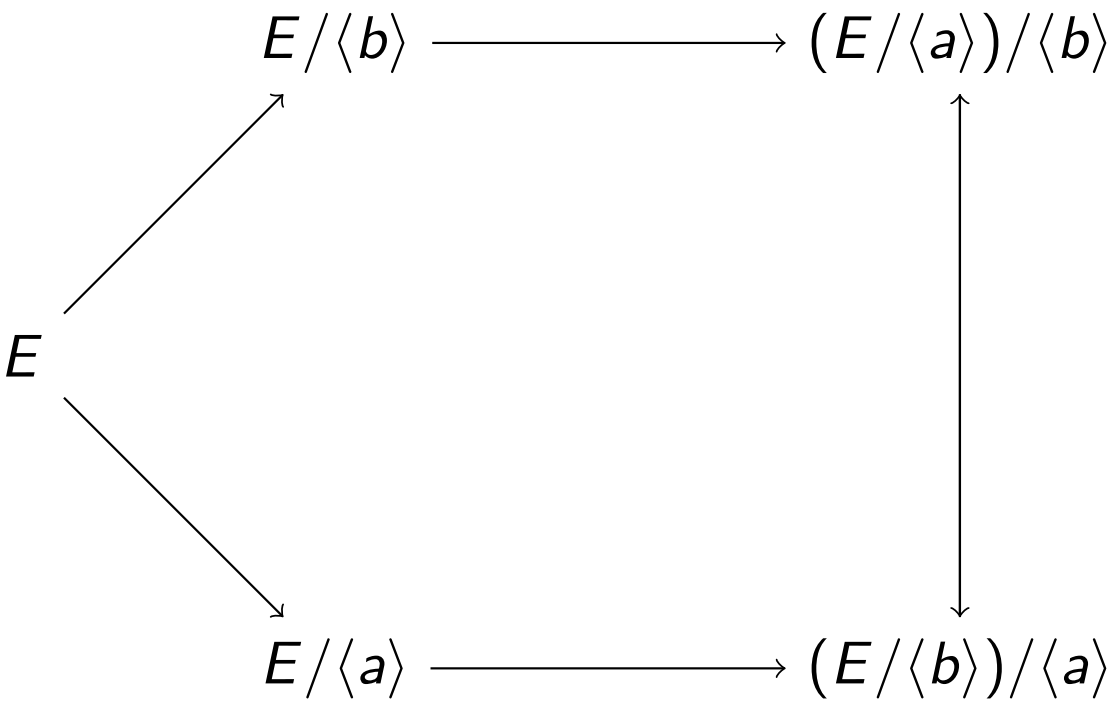
Classical Diffie-Hellman



Supersingular Isogeny Diffie-Hellman



Supersingular Isogeny Diffie-Hellman



Performance

	FJ 2011	CLN 2016
Messages size (bytes)	1164	576
Round 1 computation time (ms)	346	274
Round 2 computation time (ms)	589	618
Round 3 computation time (ms)	281	259

Key points

Most competitive message/public key sizes

So far, only for key establishment
Very new and seldom studied field

A bit slow

Towards a new standard

- Nov. 2017: submission deadline
- April 2018: first conference
- 3-5 years: analysis phase
- 2 years later: drafts ready

Submissions received

A total of 82 submissions received

	Signatures	KEM/Encryption	Overall
Lattice-based	4	24	28
Code-based	5	19	24
Multi-variate	7	6	13
Hash-based	4		4
Other	3	10	13
Total	**23**	**59**	**82**

Figure: Summary of submissions received by NIST (excerpt from *The Ship Has Sailed*, presentation by Dustin Moody at ASIACRYPT 2017)

A word for each class

Multivariate	Simple
Lattice	Versatile
Code	Deep
Hash	Strong
SIDH	???

And now...

- A very important process
- Hopefully a new dawn for public-key crypto
- Lots of new mathematical ideas
- Lots of new attacks
- So... A lot of work to do!

Thank you!

Thank you!