Introduction to Post-Quantum Cryptography

Politecnico di Torino, 11 Gennaio 2019

DARKMATTER

GUARDED BY GENIUS

Content

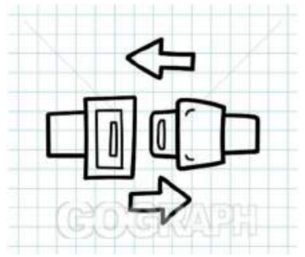
What is cryptography?

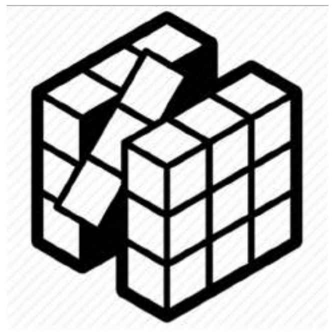
Two types of cryptography

- Symmetric crypto:
  - allows two parties to communicate secretly on a **public channel**, only if they exchanged a secret on a **private channel before**
  - Efficient both in hardware and software
  - Usually more trusted
- Asymmetric crypto:
  - allows two parties to communicate secretly on a **public channel**, even if **they never communicated before**
  - Significantly less efficient than symmetric crypto
  - Based on mathematical problems that are hard to solve
- Symmetric and asymmetric crypto are usually combined together

One-way functions… for kids

What my 2 years old baby can do, but not undo…





One-way functions

Informally:

A **one-way function** is a function that is **easy to compute** on every input, but **hard to invert** given the image of a random input (this property is called **pre-image resistance**).

Formally:

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is **one-way** if f can be computed by a polynomial time algorithm, but any polynomial time randomized algorithm F that attempts to compute a pseudo-inverse for f succeeds with negligible probability.

That is, for all randomized algorithms F, all positive integers c and all sufficiently large $n = \mathrm{length}(x)$

$$\Pr\left[f\left(F\big(f(x)\big)\right) = f(x)\right] < n^{-c}$$

where the probability is over the choice of x from the discrete uniform distribution on $\{0,1\}^n$, and the randomness of F.

One-way functions: examples

So far, it is not know if one-way functions exists. The following could be good candidates:

- Hash function (e.g. SHA-1, MD5, SHA-256,…)

$$y = H(x)$$

where x is a bit-string of any length and y of fixed length, and H might have also the properties of

  - *Second pre-image resistance*: given x_1 and $y_1 = H(x_1)$, is hard to find x_2 such that $y_1 = H(x_2)$
  - *Collision resistance*: is hard to find x_1, x_2 such that $H(x_1) = H(x_2)$

- Diffie-Hellman function

Discrete Logarithm problem

$$y = g^x$$

where g is the generator of a prime order cyclic group (finite field element or elliptic curve point)

- RSA function (one-way **trapdoor** function)

RSA/Factorization problem

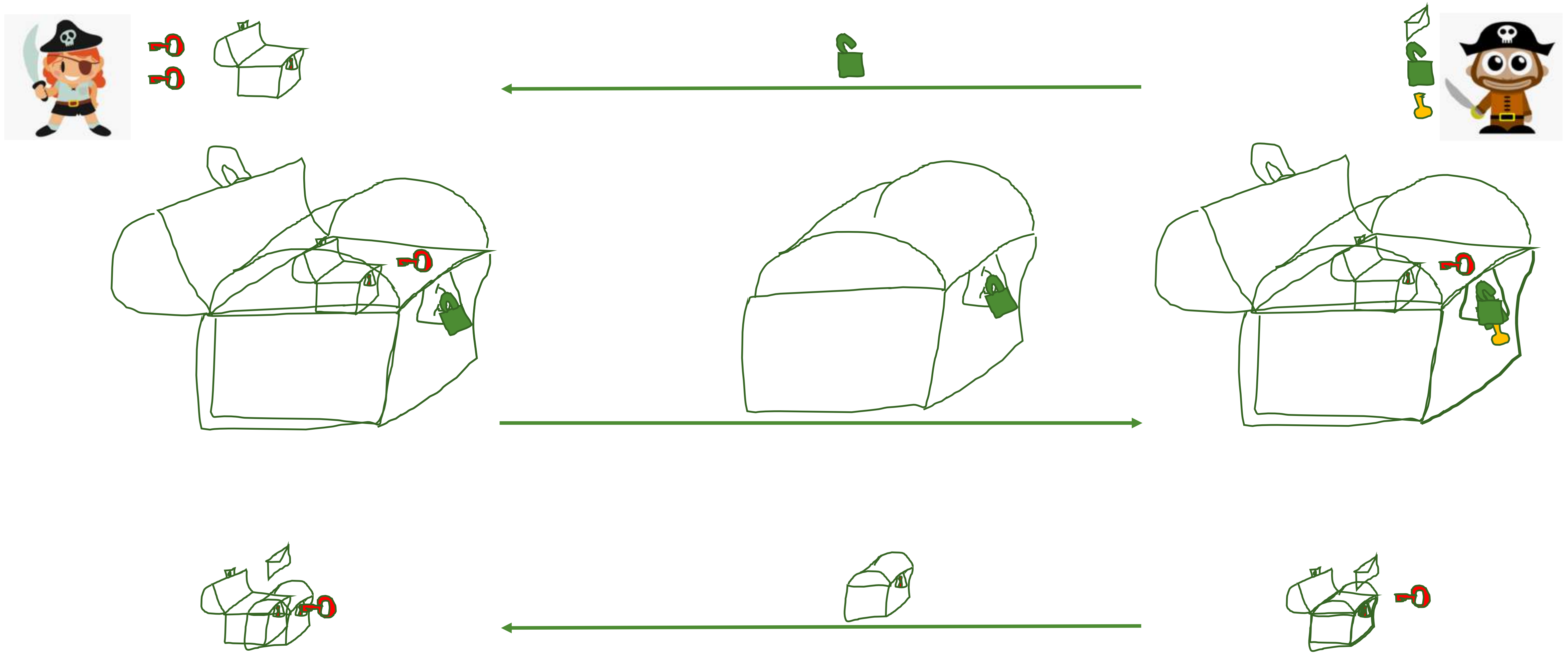
$$y = x^e \bmod N$$

where N is the product of two primes of roughly the same size

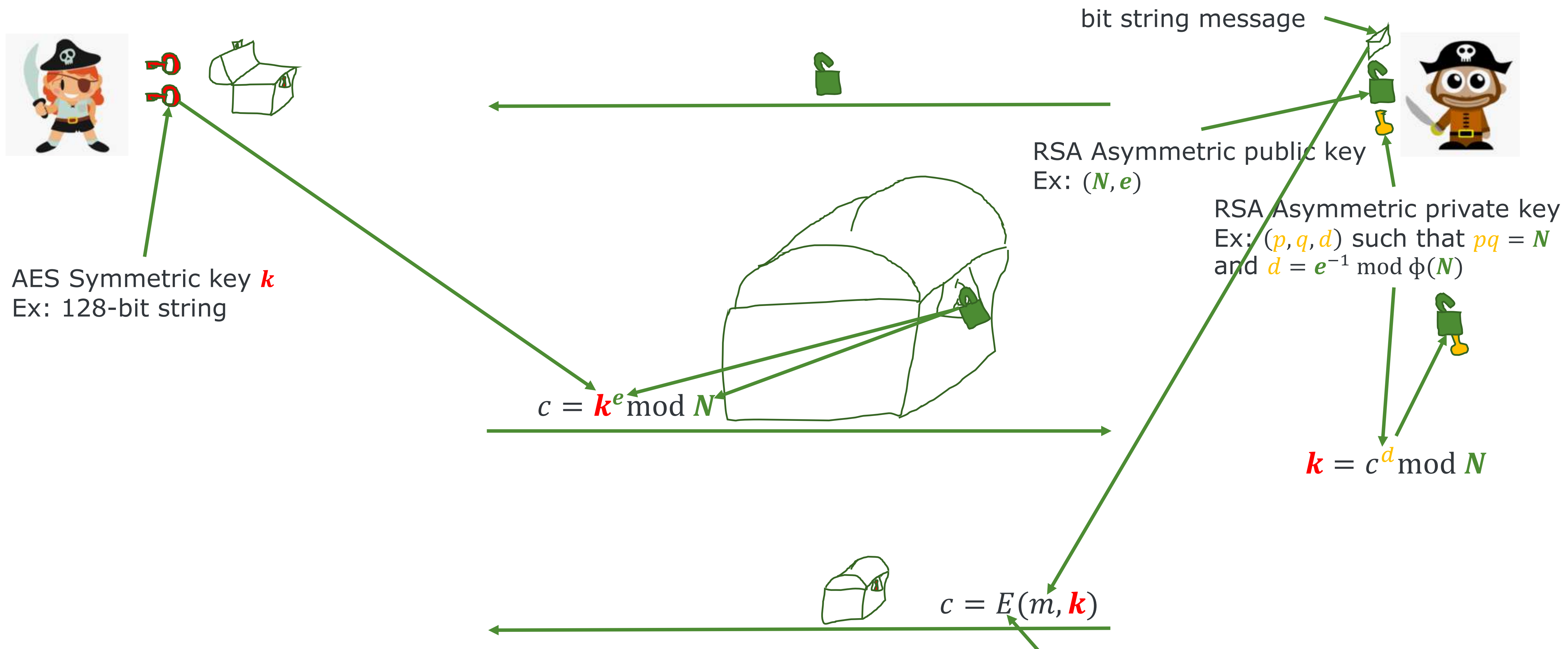
- Block ciphers (e.g. AES, DES, …), stream ciphers (e.g. A5/1, RC4, Salsa20,…) (**partially** one-way **compression** functions)

$$y = E(x, k)$$

Symmetric and asymmetric cryptography combined, with pirates…



Symmetric and asymmetric cryptography combined, in practice



bit string message

RSA Asymmetric public key
Ex: $(\boldsymbol{N}, \boldsymbol{e})$

RSA Asymmetric private key
Ex: (p, q, d) such that $pq = \boldsymbol{N}$
and $d = \boldsymbol{e}^{-1} \bmod \phi(\boldsymbol{N})$

AES Symmetric key $\boldsymbol{k}$
Ex: 128-bit string

$$c = \boldsymbol{k}^{e} \bmod \boldsymbol{N}$$

$$\boldsymbol{k} = c^{d} \bmod \boldsymbol{N}$$

$$c = E(m, \boldsymbol{k})$$

$E(.,.)$ is a block cipher
Ex: AES128

Public Key Encryption (PKE)

- A *public key encryption scheme* (PKE) is a scheme with public and private keys, where we can encrypt a message using the public key and decrypt using the private key.
- **EXAMPLE:** $c = k^e \bmod N$ with “padding” (RSA-OAEP)
- **PROBLEM:** usually k is much smaller than N (e.g., if k is 128 bits then N is 3096 bits)
  - This implies that if e is too small than $k^e \bmod N$ is easy to invert.
  - Then it is necessary to “pad” k in order to make it a random number $< N$.
  - Unfortunately, security proofs involving padding schemes are not easy.
- **SOLUTION:** use KEMs…

Key Encapsulation Method (KEM)

- A *key encapsulation method* (KEM) is a scheme with public and private keys, where we can use the public key to create a ciphertext (encapsulation) containing a randomly chosen symmetric key. We can decrypt the ciphertext using the private key.
- **EXAMPLE:**
  - Bob (after receiving (N, e) from Alice)
    - Generate random $x \in \{1, \ldots, N\}$
    - $k = \mathrm{Hash}(x)$
    - $c = x^e \bmod N$
    - Send c
  - Alice
    - $x = c^d \bmod N$
    - $k = \mathrm{Hash}(x)$

Key Exchange (KEX)

- A *key exchange protocol* (KEX) is a protocol that allows Alice and Bob to agree on a shared symmetric key.
- **EXAMPLE:**
  - Bob
    - Chooses random $b \in \{1, \dots, \mathrm{Order}(g)\}$
    - Sends $B = g^b$ to Alice
  - Alice
    - Chooses random $a \in \{1, \dots, \mathrm{Order}(g)\}$
    - Sends $\mathrm{A} = g^a$ to Bob
  - Both Alice and Bob compute $g^{ab} = (g^a)^b = (g^b)^a$ and use it as a shared secret.
- **NOTE:** There exist ways to transform a KEX into a KEM, but not always is possible.

Digital signatures

- A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents.
- A valid digital signature gives a recipient reason to believe that
  - the message was created by a known sender (authentication),
  - the sender cannot deny having sent the message (non-repudiation),
  - the message was not altered in transit (integrity).

Digital signatures: RSA-based example

- Generate public and private key as usual
- To sign:
  - $h = Hash(m)$
  - $s = h^{d} \bmod N$, using the public N and the private d
- To verify
  - $h = Hash(m)$
  - check that $s^{e} = h \bmod N$, using the public N and e

Basically works as PKE, but adding a hash and inverting public with private key.

Digital signatures: discrete log (non-trapdoor) example

Schnorr Identification Protocol :

Goal convince the verifier that
he is interacting with someone
who knows the secret key
that corresponds to the prover’s public key

$o = \mathrm{Order}(g)$, secret $x \in \{1, \dots, o\}$, public $y = g^x$

```
        Prover                                   Verifier

Choose random $k \in \{1, \dots, o\}$

Compute $s = g^k$

                          --- $s$ --> Choose random $c \in \{1, \dots, o\}$

Compute $r = k - xc$      <-- $c$ ---

                          --- $r$ --> check $g^r y^c = s$
```

Fiat-Shamir Transform

Signer

```
Choose random $k \in \{1, \dots, o\}$
$s = g^k$
$c = \mathrm{Hash}(s||\mathrm{message})$
$r = k - xc$
Signature $= (r, c)$
```

Verifier

```
$s' = g^r y^c$
$c' = \mathrm{Hash}(s'||\mathrm{message})$
Check $c' = c$
```

Classical vs Post Quantum Cryptography

Quantum supremacy

Quantum supremacy or "quantum advantage" is the potential ability of quantum computing devices to solve problems that classical computers practically cannot.

In computational complexity-theoretic terms, this generally means providing a superpolynomial speedup over the best known or possible classical algorithm.

So far, quantum supremacy has not been reached yet!

Google previously announced plans to demonstrate quantum supremacy before the end of 2017 by solving this problem with an array of 49 superconducting qubits.

In October 2017, IBM demonstrated the simulation of 56 qubits on a conventional supercomputer, increasing the number of qubits needed for quantum supremacy.

Then, on march 2018, Google announced Bristlecone, a new 72 qubits quantum processor, but it is still trying to make it work…

LET US IMAGINE THAT TODAY QUANTUM COMPUTERS EXIST WITH "GOOD" CAPACITY TO BREAK CRYPTOGRAPHY.
THERE REALLY IS NO REASON TO WORRY ABOUT QUANTUM COMPUTERS BREAKING EVERYTHING.
BUT I'M SCARED...
imgflip.com

Quantum computers VS classical crypto

- If a practical quantum computer would exist, it would break

all classical factorization and discrete log based public key crypto

(because of **Shor’s algorithm**)

and

would force doubling the key sizes of symmetric key cryptography.

(because of **Grover’s algorithm**)

- Not many other quantum algorithms are known… yet!
- So… what solutions should we adopt?

Two alternatives

- The physicists say:

  “Use quantum technologies to fight quantum technology!”

  QUANTUM CRYPTOGRAPHY

- The mathematicians say:

  “Just base your crypto on math that quantum computers can’t break.”

  QUANTUM RESISTANT or POST QUANTUM CRYPTOGRAPHY

Quantum cryptography in practice

- mainly limited to *quantum key distribution*,
- provides no authentication (apart from PUF technologies),
- requires direct fiber-optical connection or line of sight,
- has a problem with large distances,
- needs new infrastructure and new technology,
- does not work for mobile phones, sensor networks, cars, ...
- does not scale well,
- based on *laws of physics and information-theoretic security*, rather than **math and computational complexity**

Quantum Resistant Cryptography (or PQC)

Quantum computer do not solve **all hard** problems…

PQC consists of classical algorithms that

- run efficiently on classical computers in terms of time, memory, and communication
- are hard to break both by classical and quantum algorithms,
- rely on **different mathematical problems** than integer factorization or discrete logarithms.

NIST Post-quantum competition

- Feb. 2016 Announcement at PQCrypto 2016
- April 2016 NIST releases NISTIR 8105 - Report on Post-Quantum Cryptography
- Dec. 2016 Formal Call for Proposals
- Nov. 2017 Deadline for submissions
- Early 2018 Workshop — Submitter’s Presentations
- 3-5 years Analysis Phase — NIST will report findings

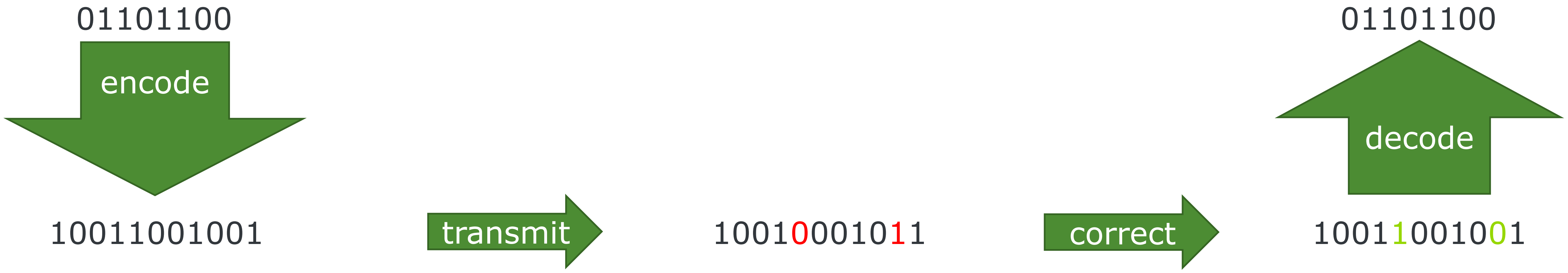
  1-2 workshops during this phase
- 2 years later Draft Standards ready

NIST candidates

Family	**Signatures**	**KEM/PKE**	**SUM**
Lattice	5	23	28
Codes	3	17	20
Multivariate	7	3	10
Hash	2	0	2
“others”	3	6	9
TOTAL	**20**	**49**	**69**

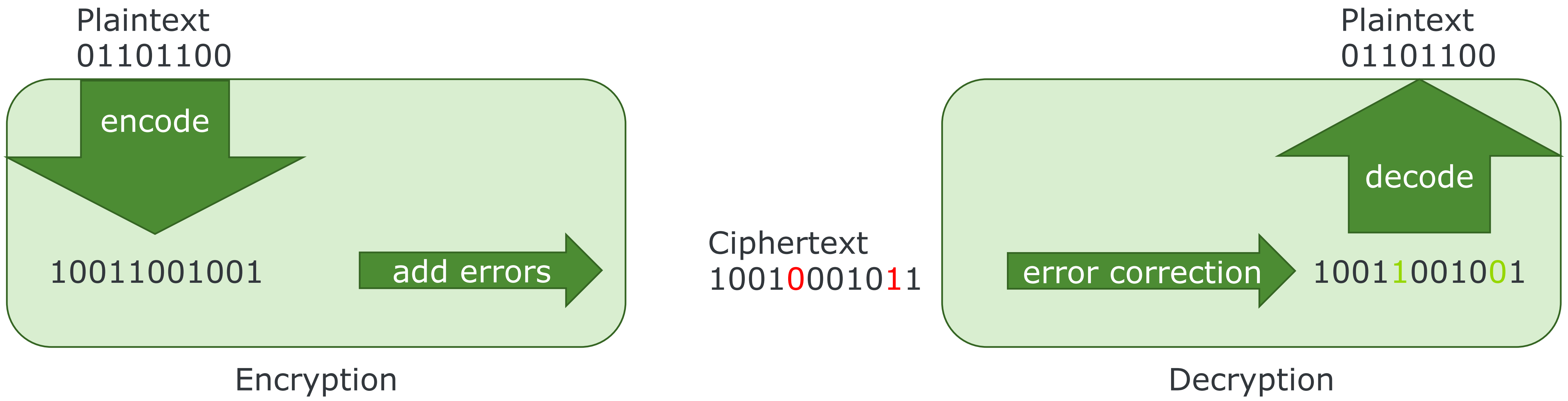
Code-Based Cryptography

Error correction



Error correction to encryption scheme

McEliece cryptosystem



Hamming metric

- It is possible to define a metric among vectors, by defining the

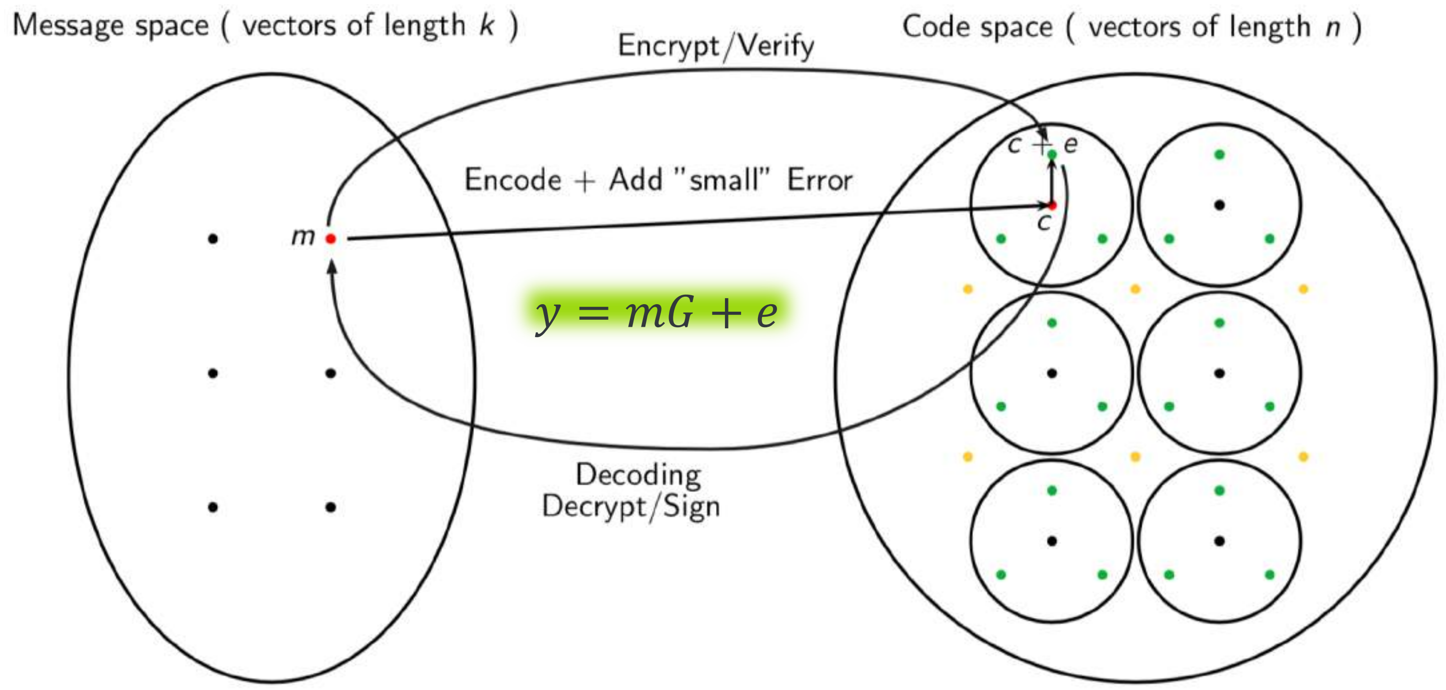
  Hamming distance

  of two such vectors as the number of coordinates the two vectors differ.
- The **Hamming weight** of a vector is its distance from the zero vector.

- **Example:**

  (1,1,0,1) has Hamming distance 2 from (1,1,1,0), they both have weight 3.
- Vectors can be represented as points.

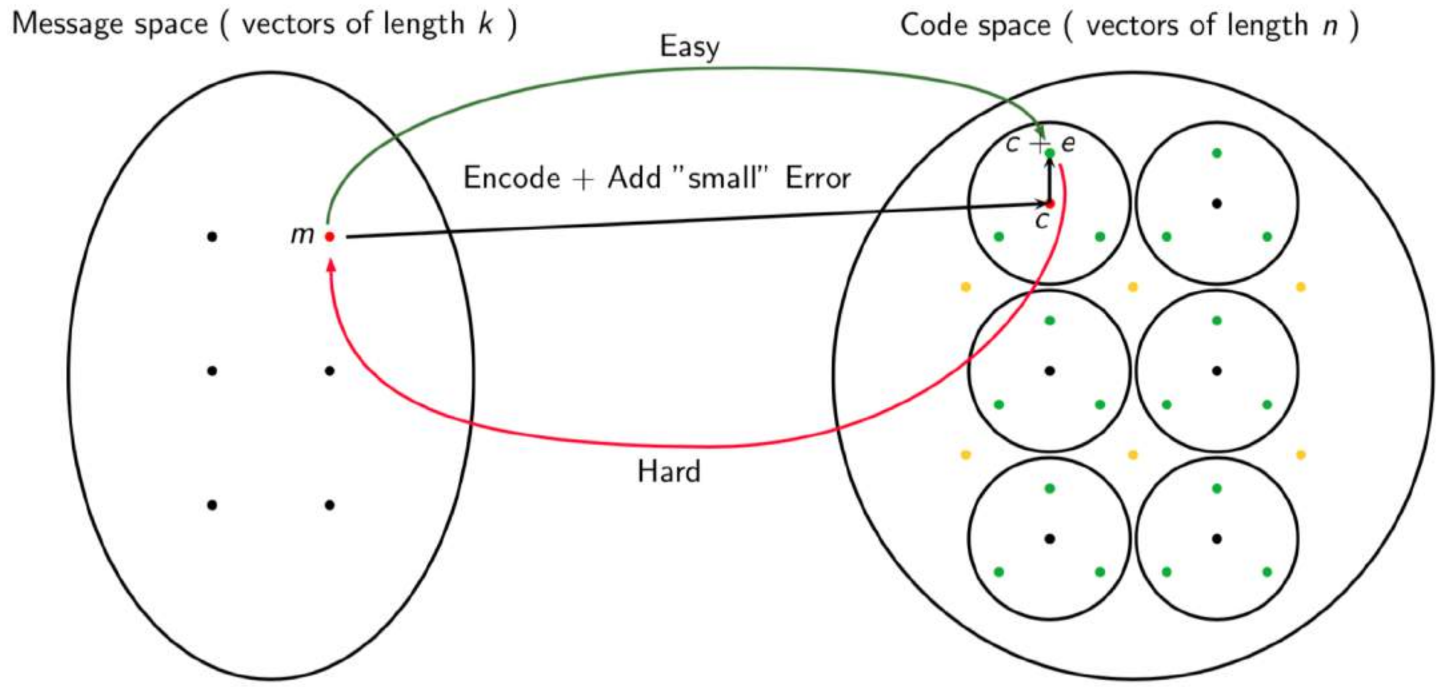
One-way functions in codes



Message space (vectors of length k)
Encrypt/Verify
Code space (vectors of length n)
$c + e$
Encode + Add "small" Error
m
c
$y = mG + e$
Decoding
Decrypt/Sign

- $k < n$, green dots are decodable vectors, orange dots are non-decodable vectors (i.e. *invalid ciphertext*).
- *Encoding* is usually done by multiplying a vector by the *generator matrix* G of the code, while *decoding* is hard in general
- G is not unique, contains a base of the code space

One-way trapdoor function in codes: McEliece-like cryptosystems


Message space (vectors of length k)
Code space (vectors of length n)
Easy
Encode + Add "small" Error
Hard
m
$c + e$
c

If code not random linear code, but has special algebraic structure

- the encoding can be performed by $c = m \cdot (S \cdot G \cdot P)$, where G is secret and the public $S \cdot G \cdot P$ looks as the generator matrix of a random linear code, for random invertible S, P.
- The decoding of $c + e$ can be done efficiently only knowing G, S, P.

Comparison between RSA and McEliece

	RSA	McEliece
SK	p, q, d	S, G, P
PK	N, e, where $N = pq$	G', where $G' = SGP$
Encryption function	$y = x^e \bmod N$	$y = xG' + e$
Semantic security	No (deterministic enc)	Yes (probabilistic enc)
Trapdoor	Decomposition of $N = pq$, p, q primes	Decomposition of $G' = SGP$, G structured
Decryption function	$x = y^d \bmod N$ (with CRT)	$\mathrm{Decode}(yP^{-1})S^{-1}$
Decryption failure	Yes, as likely as randomly guessing SK	No (for valid ciphertext) => no reaction attacks
Invalid ciphertext	Yes, as likely as randomly guessing SK	Yes

Pro VS Cons of McEliece

- **Advantages**
  - Efficient
  - Very secure

    (Original McEliece has 40 years and more than 30 analysis papers)
- **Disadvantages**
  - Large key sizes

Alternatives to McEliece

- Use codes with a more compact representation, and less structure e.g.
  - Double circulant codes
  - low density parity check codes
- Use a different metric
  - rank metric codes
  - lattices

With some of these codes there is decoding failure!

What about code-based signatures?

Two approaches:

- **RSA/Trapdoor approach:**
  - The one-way function $f(m, e) = mG + e$ can be used just as the RSA one-way function $f(x) = x^e \mathrm{mod}\, N$ to build a cryptographic signature:

    add a hash and use trapdoor to sign and one-way function evaluation to verify
  - Leads to short signatures but very large keys (because hash could be a non-decodable word!)
- **Schnorr/Trapdoorless approach:**
  - Use the one-way function $f(m, e) = mG + e$ in the Schorr identification protocol instead of $f(x) = g^x$ and convert it to signature with Fiat-Shamir transform, which
    - is not known how secure this is against quantum adversaries,
    - does not have a quantum analogue
  - Leads to very short public keys and very large signatures

What about rank metric?

- A vector over a finite field extension is seen as a matrix over the base field (i.e. every coordinate, being a polynomial, can be seen as a column of the matrix).
- The weight of the vector is the **rank of the matrix**.
- The complexity of the best known attacks for solving the underlying mathematical problem (Rank Syndrom Decoding problem) have a **quadratic exponential complexity** in the parameters of the system, while for Hamming metric it is linear!
- As a consequence, notably **reduced key sizes** to achieve same security level than a Hamming distance-based cryptosystem.
- On the other side there is a **small loss on performance**.

dRANKula and fRANKenstein

dRANKula

- Is an IND-CCA2 PKE secure version of a McEliece-like rank metric based cryptosystem published by Loidreau at PQC 2017, with some improvement in the scheme parameters choice and in the decoding.
- Described in
  - *dRANKula: a McEliece-like rank metric based cryptosystem implementation*
    SECRYPT 2018, Porto.
  - *An IND-CCA transformation of a McEliece like cryptosystem over rank metric*
    to appear somewhere…

fRANKenstein

- Is a Schnorr-like signature scheme based on double circulant codes in the rank metric, with very small keys and medium size signatures.
  There exist only a couple of quantum resistant, not broken, efficient signature schemes based on codes.
- Described in
  - *Code-based signature schemes from identification protocols in the rank metric*
    CANS 2018, Napoli.
  - *Veron identification and signature schemes in the rank metric*
    to appear hopefully at PQC 2019.

Why DH KEX fails in codes (…and lattices)?

Cyclic group

Public parameter: generator g

$$\begin{array}{ccc} a & & b \\ & \text{-- } g^a \text{ -->} & \\ (g^b)^a & \text{<-- } g^b \text{ --} & (g^a)^b \end{array}$$

Common secret:

$$g^{ab}$$

Error correcting code

Public parameter: generator matrix G

$$\begin{array}{ccc} m, e & & m', e' \\ & \text{-- } mG + e \text{ -->} & \\ ??? & \text{<-- } m'G + e' \text{ --} & ??? \end{array}$$

Common secret:

$$???$$

The image of $f(x) = g^x$ is still in $\langle g \rangle$

The image of $f(m) = mG + e$ is not a matrix!!!

Homomorphic properties of one-way functions

- $f(x) = g^x$:

$$g^x g^y = g^{x+y}$$
$$f(x)f(y) = f(x+y)$$

- $f(x) = x^e \bmod N$:

$$x^e y^e \bmod N = xy^e \bmod N$$
$$f(x)f(y) = f(xy)$$

- $f(x,e) = xG + e$:

$$(xG + e) + (yG + f) = (x+y)G + (e+f)$$
$$f(x,e) + f(y,f) = f(x+y, e+f)$$

The new error might be not decodable!!

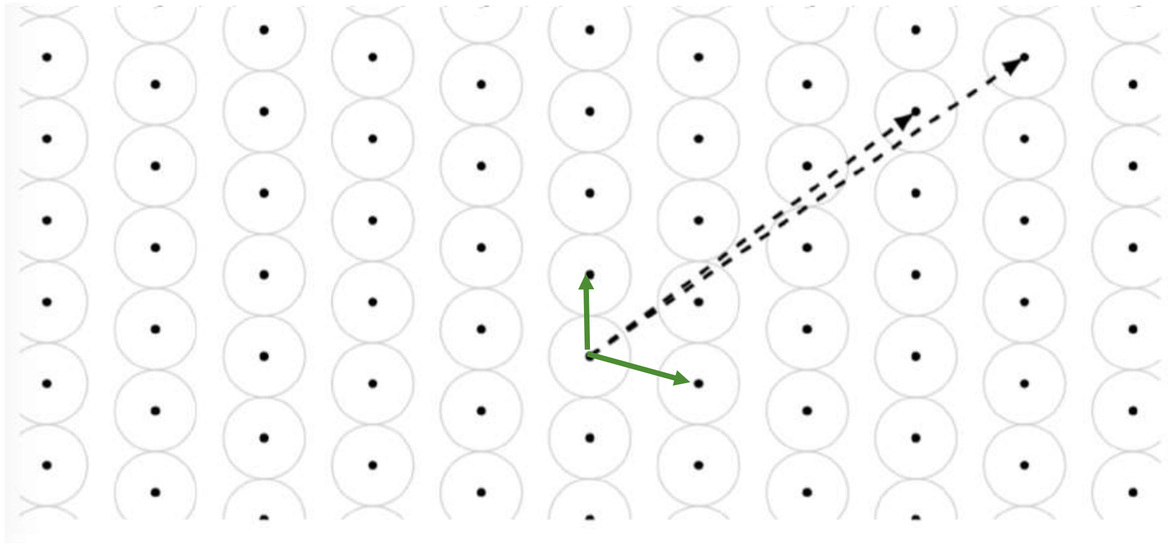
- Codes allow ElGamal constructions (randomized PKE scheme with DH one-way function)
- What about $f(x,e)f(y,f)$?

Lattice-Based Cryptography

Lattices and codes

- Ideas in lattice-based crypto are very similar to the ones in code-based crypto

  - A lattice is a subgroup of the additive group $\mathbb{R}^n$ which is isomorphic to the additive group $\mathbb{Z}^n$… …in other words, a set of points with n coordinates, a sum, a scalar product and the Euclidean metric…
  - There exists "good" and "bad" bases
  - It is possible to draw the largest "ball" such that when centered in each lattice points the balls do not intersect
  - A message is a point of the lattice
  - To encrypt we add a "small" (inside the circle) error to the initial message
  - Given only the encryption and a bad basis (the **public key**) it is difficult to find a combination of the bad basis that reaches the initial message
  - But given a good basis (the **secret key**)…
  - It is easy to recover the initial message, as the closest point to the encryption point



Linear systems

Solving linear systems is easy

(use Gaussian elimination, polynomial time)

- Given

$$\begin{aligned} 1s_1 + 2s_2 + 5s_3 + 2s_4 &= 9 \, mod \, 13 \\ 12s_1 + 1s_2 + 1s_3 + 6s_4 &= 7 \, mod \, 13 \\ 6s_1 + 10s_2 + 3s_3 + 6s_4 &= 1 \, mod \, 13 \\ 10s_1 + 4s_2 + 12s_3 + 8s_4 &= 0 \, mod \, 13 \, . \end{aligned}$$

- Find s_1, s_2, s_3, s_4

Linear systems with errors

Solving linear systems with errors is hard

- Given

$$1s_1 + 2s_2 + 5s_3 + 2s_4 \approx 9 \; mod \; 13$$
$$12s_1 + 1s_2 + 1s_3 + 6s_4 \approx 7 \; mod \; 13$$
$$6s_1 + 10s_2 + 3s_3 + 6s_4 \approx 1 \; mod \; 13$$
$$10s_1 + 4s_2 + 12s_3 + 8s_4 \approx 0 \; mod \; 13 \, .$$

- Find s_1, s_2, s_3, s_4 , knowing that the solution is incorrect by ± 1 …
- The problem is called Learning With Errors (LWE)
- The associated one-way function is

$$f(s, e) = sA + e$$

Where $s = (s_1, \dots, s_4)$, A is the coefficient matrix, e is a vector of small errors

Lattice peculiarity: security proofs

- There are security proofs and worst-case to average-case reductions
- Security proofs are not *tight*: security parameters are chosen based on *best-known* attacks, not based on security proofs

Problems

- Attack-complexity not yet deeply understood
- Attacks are improved frequently

Multivariate-Based Cryptography

Underlying problem

Solving a system of m multivariate polynomial equations in n variables over $\mathbb{F}_q$.

This is called the

MP Problem

the MP problem is an *NP-Complete* problem even for multivariate *quadratic* system and $q = 2$

Example with $\boldsymbol{m = 3, n = 3}$:

$$5x_1^3x_2x_3^2 + 17x_2^4x_3 + 23x_1^2x_2^4 + 13\,x_1 + 12x_2 + 5 = 0$$

$$12x_1^3x_2^3x_3 + 15x_1x_3^3 + 25x_2x_3^3 + 5\,x_1 + 6x_3 + 12 = 0$$

$$28x_1x_2x_3^4 + 14x_2^3x_3^2 + 16x_1x_3 + 32\,x_2 + 7x_3 + 10 = 0$$

Basic idea

- System parameters $m, n \in \mathbb{N}$
- **Key generation**: choose “random” $f \in \mathcal{MQ}(\mathbb{F}_2^n, \mathbb{F}_2^m)$, such that f^{-1} is secretly known.
- **Public key**: f
- **Secret key**: f^{-1}
- **Encryption**: to encrypt a message $m \in \mathbb{F}_2^n$, compute $c = f(m)$
- **Decryption**: $m = f^{-1}(c)$

Problem: how do you find f and f^{-1} such that f is a hard instance of $\mathcal{MQ}$?

One-way trapdoor function for MP problem

Usually, f is constructed as a sequence of invertible functions, e.g.

$$f = r \circ s \circ t$$

With r, t multivariate linear and s quadratic with a easy to invert structure.

Unfortunately, this often does NOT result in a hard instance of $\mathcal{MQ}$!

Concern about MQ schemes

- Few recent secure examples (not yet broken?) are
  - Rainbow signature scheme
  - Quartz of HFEv-signature scheme
  - PMI+public key encryption scheme
- Many public-key encryption schemes have been broken
- Efficient (sparse) $\mathcal{MQ}$ instances have problems with randomness

Isogeny-Based Cryptography

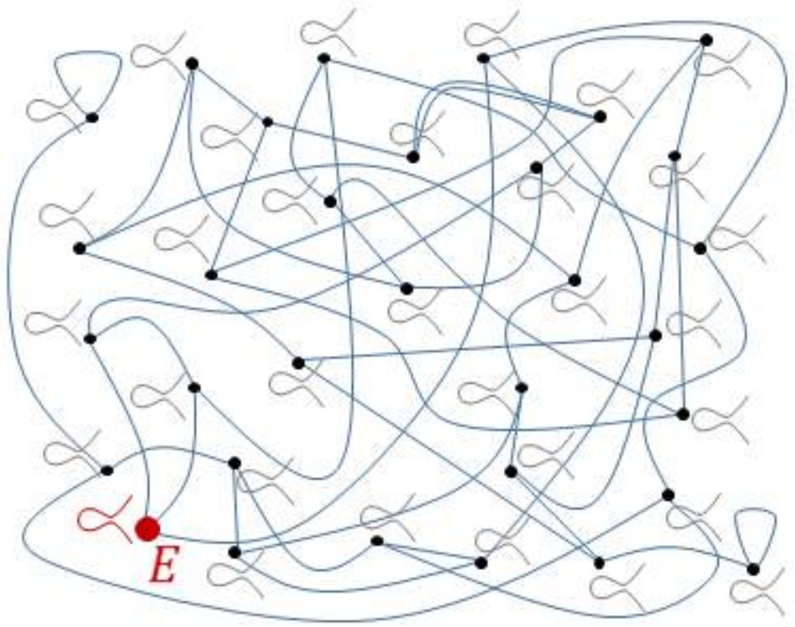
Classic ECC vs Isogeny based crypto

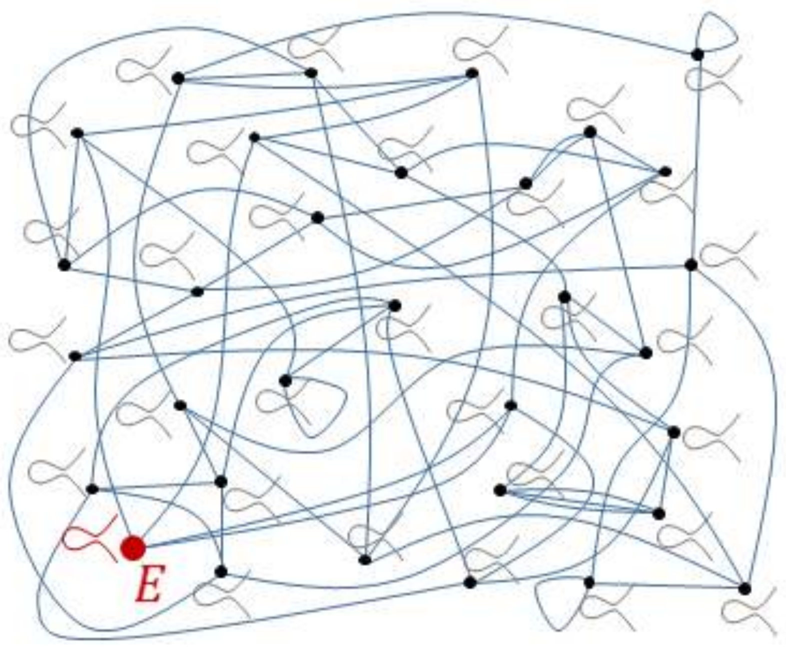
- **Classic ECC**: discover hidden relation between two points *P, P′* on a given elliptic curve *E*
- **Isogeny-based crypto**: find connection between *two elliptic curves* E, E′ inside a certain large pool

	Classic DH	**Classic ECC**	**Isogeny-based Crypto**
Public parameter	Group generator g	A set of points (elliptic curve) + Base Point P	A set of curves + Base Curve E
Secret key	Scalar a (sends a group element to another)	Scalar a (sends a point to another)	Isogeny φ (sends a curve to another)
Public key	The group element g^a	The point aP	The curve $\varphi(E)$ (plus some additional info)

Two and three isogeny maps

- **Two-isogenies maps**: each curve is connected to 3 other curves
- **Three-isogenies maps**: each curve is connected to 4 other





Underlying problem

Problem:

Given an elliptic curve *E′* obtained by applying a random sequence of two- or three-isogenies starting from a base curve *E*, find how to get from *E* to *E*′ using the same map.

This problem is hard only for some special family of curves, the so called

isogeny class of supersingular elliptic curves.

Supersingular isogeny class

Definition:

The *supersingular isogeny class* is the set of all *supersingular* elliptic curves over a finite field with p^2 elements, where p is a prime number of a rather specific form, chosen such that all two- and three-isogenies can be described in a convenient way.

Supersingular-Isogeny Diffie-Hellman

- Alice and Bob agree on a pool of curves and a starting curve E.
- Alice applies a secret random sequence of two-isogenies, resulting in an elliptic curve E_A.
- Bob secretly walks a random trail of three-isogenies in order to end up at a curve E_B.
- Alice and bob exchange E_A and E_B publicly,
- Alice reapplies her sequence of two-isogenies *starting from Bob's curve* E_B,
- Bob does the same starting from Alice's curve E_A.
- It turns out Alice and Bob eventually end up with the *same* elliptic curve E_{AB}, which is the common secret, only known by Alice and Bob

Pro vs Cons

- **Pro**
  - Small communication overhead.
  - Small keys.
  - Only setting where Non-interactive Key Exchange (as DH) are possible (CSIDH)!
- **Cons**
  - High computational cost.
  - Very recent proposal; security not yet well understood.
  - First proposal with *ordinary* curves broken by quantum computers.
  - New proposal using *supersingular* curves under examination.

Hash-Based Cryptography

Main idea

- Use a hash chain to prove knowledge of a secret
  - Alice computes $p = Hash\left(Hash\left(Hash(Hash(k))\right)\right)$ and sends it to Bob
  - Every time Alice wants to authenticate shows to Bob the previous intermediate value of the chain, i.e $Hash\left(Hash(Hash(k))\right)$, then $Hash(Hash(k))$, then $Hash(k)$, then k.
  - This way Alice can authenticate to Bob a finite number of times
- The same principle can be used to build **stateful hash-based signatures**, such as XMSS, or LMSS.
- By creating a large enough state to cover the number of possible usages, the signature can be treated as **stateless**. The two stateless signature schemes presented at NIST competition are called SPHINCS+ and GRAVITHY-SPHINCS (very similar).
- Hash-based signatures allow many trade offs, but in general have very slow key generation and verification.
- Hash-based signatures only rely their security on the properties of the hash function.
- It is not known how to build efficient PKE or KEM using only hash…

Conclusions

PQC – Public key encryption and signatures schemes

1 Codes

- Most tested
- Biggest key sizes
- Fast

2 Lattices

- Relatively “new”
- Small-ish key sizes
- Fastest

3 Isogenies

- Graphs and elliptic curve related
- Smallest key sizes
- Slow

4 Multivariate

- Proposals look fragile
- Small-ish key sizes (for signatures)
- Average speed

PQC – Hash-based signatures schemes

PROS

- Easy implementation
- Fast signing
- Small key sizes

CONS

- Slow key generation
- Somehow slow validation
- Stateful – Stateless
- Only signatures

Issues with PQC

- Some have slower computation (e.g isogenies)
- Some have larger keys, signatures, ciphertexts (e.g. codes)
- Some have security less well understood (e.g. multivariate, lattices, isogenies)
- Some have design issues (e.g., stateful hash-based signing)

All this drawbacks should be taken into consideration when designing a modern secure communication system, in order to choose the proper solution, or at least the "less worse".

Rough performance comparison

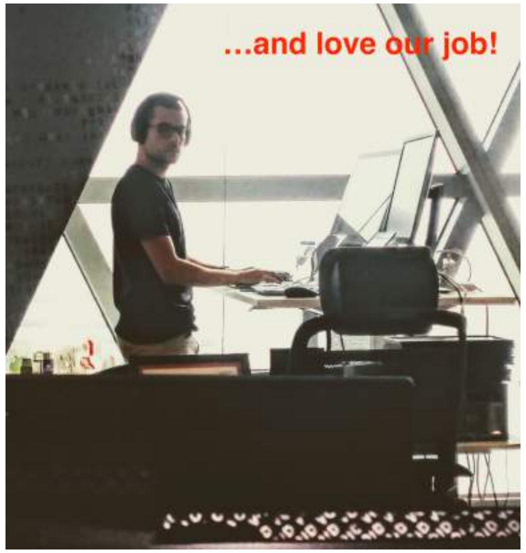
Take it with a grain of salt…

Family	Key generation	Public key encryption/ verification	Private key decryption/ signing
Code-based	slow	**fast**	medium
Hash-based	slow	**fast**	slow
Lattice-based	**fast**	**fast**	**fast**
Multivariate-based	slow	**fast**	medium
Isogeny-based	slow	slow	slow
ECC-256	fast	medium	fast
RSA-3072	slow	fast	slow

To conclude, should we be worried?

- Post-quantum schemes already exists for powerful devices but are very challenging for constrained devices such as microprocessors, HSMs, etc.
- RSA is very old and easy to be implemented unsecurely.
- ECC is very elegant and efficient but somehow limited to very few use cases.
- Hybrid encryption could be a good temporary solution.
- Designing new protocols to better fit post quantum schemes limitation is certainly a must.
- …and to answer the title’s question… yes, we should!

We are hiring!



Thanks for your attentions!