A Brief Introduction to Post-Quantum Cryptography

Fernando Virdia

NOVA.ID.FCT
Universidade NOVA de Lisboa

Outline

- Motivation: Cryptography and Quantum Computing
- Foundations: New Hardness Assumptions
- Standards: The US NIST process
- Deployment: Some of the challenges

Slides @ https://fundamental.domains

"Pre-Post-Quantum" Cryptography

Usually cryptography is presented as consisting of two components:

“Pre-Post-Quantum” Cryptography

Usually cryptography is presented as consisting of two components:

- Symmetric cryptography, that deals with secure communications between parties sharing a secret key or a password

"Pre-Post-Quantum" Cryptography

Usually cryptography is presented as consisting of two components:

- Symmetric cryptography, that deals with secure communications between parties sharing a secret key or a password
- Asymmetric cryptography (or PKC), that deals with allowing distant parties to agree on such a shared secret over an unsecured channel

"Pre-Post-Quantum" Cryptography

Usually cryptography is presented as consisting of two components:

- Symmetric cryptography, that deals with secure communications between parties sharing a secret key or a password
- Asymmetric cryptography (or PKC), that deals with allowing distant parties to agree on such a shared secret over an unsecured channel

Together, they enable the large-scale deployments of cryptography that we see today on the Internet, and in payment systems.

- Both kinds of primitives are constructed using varying degrees of mathematical structure.
- The structure should imply that an adversary trying to break the primitive, needs to solve some hard mathematical problem.

- Both kinds of primitives are constructed using varying degrees of mathematical structure.
- The structure should imply that an adversary trying to break the primitive, needs to solve some hard mathematical problem.
- We formalise these problems into concise "hardness assumptions".
- Part of the job of cryptographers is identifying hardness assumptions, trying to break them, and constructing primitives from them.

Today's PKC is mostly based on hardness assumptions related to two mathematical problems:

Today's PKC is mostly based on hardness assumptions related to two mathematical problems:

Factoring

Let p, q be different random primes of similar size, $\log p \approx \log q$.

$$\text{Given only } N = p \cdot q \text{, find } p \text{ and } q.$$

Today's PKC is mostly based on hardness assumptions related to two mathematical problems:

Factoring

Let p, q be different random primes of similar size, $\log p \approx \log q$.

$$\text{Given only } N = p \cdot q, \text{ find } p \text{ and } q.$$

Discrete logarithms (DLOG)

Let G be a finite group, and $g \in G$ be an element generating a large subgroup.
Let x be a random integer in $\{1, \ldots, |\langle g \rangle| - 1\}$.

$$\text{Given only } g^x, \text{ find } x.$$

Today's PKC is mostly based on hardness assumptions related to two mathematical problems:

Factoring

Let p, q be different random primes of similar size, $\log p \approx \log q$.

$$\text{Given only } N = p \cdot q \text{, find } p \text{ and } q.$$

Discrete logarithms (DLOG)

Let G be a finite group, and $g \in G$ be an element generating a large subgroup.
Let x be a random integer in $\{1, \ldots, |\langle g \rangle| - 1\}$.

$$\text{Given only } g^x \text{, find } x.$$

These problems received a lot of study, and are used everywhere in software and hardware.

What do we mean by “factoring is hard”?

What do we mean by "factoring is hard"?

- Intuitively, solving a random instance should require lots of resources (calculations, memory, energy, money. . .)

What do we mean by “factoring is hard”?

- Intuitively, solving a random instance should require lots of resources (calculations, memory, energy, money. . .)
- To decide if this is the case, we researach algorithms to solve the problem (cryptanalysis), and find a formula for the cost as a function of the problem’s parameters (eg., if $N = p \cdot q$, $\log N$).

What do we mean by “factoring is hard”?

- Intuitively, solving a random instance should require lots of resources (calculations, memory, energy, money. . .)
- To decide if this is the case, we researach algorithms to solve the problem (cryptanalysis), and find a formula for the cost as a function of the problem’s parameters (eg., if $N = p \cdot q$, $\log N$).
- For all the attacks known, we use these formulas to choose parameters so that the cost is “high enough” (Eg., so that it requires $\geq 2^{128}$ CPU cycles to solve)

What do we mean by “factoring is hard”?

- Intuitively, solving a random instance should require lots of resources (calculations, memory, energy, money. . .)
- To decide if this is the case, we research algorithms to solve the problem (cryptanalysis), and find a formula for the cost as a function of the problem’s parameters (eg., if $N = p \cdot q$, $\log N$).
- For all the attacks known, we use these formulas to choose parameters so that the cost is “high enough” (Eg., so that it requires $\geq 2^{128}$ CPU cycles to solve)
- We also research mathematical relations of the problem with similar ones (or itself)

What do we mean by "factoring is hard"?

- Intuitively, solving a random instance should require lots of resources (calculations, memory, energy, money. . .)
- To decide if this is the case, we research algorithms to solve the problem (cryptanalysis), and find a formula for the cost as a function of the problem's parameters (eg., if $N = p \cdot q$, $\log N$).
- For all the attacks known, we use these formulas to choose parameters so that the cost is "high enough" (Eg., so that it requires $\geq 2^{128}$ CPU cycles to solve)
- We also research mathematical relations of the problem with similar ones (or itself)

NOTE: We cannot have absolute certainty that the problem is hard. (Eg., maybe $P = NP$)

Example: the hardness of factoring

Example: the hardness of factoring

- Let $N = p \cdot q$ for random p, q such that $\log p \approx \log q$.

Example: the hardness of factoring

- Let $N = p \cdot q$ for random p, q such that $\log p \approx \log q$.
- It takes at most $2^{\frac{\log N}{2}} \approx 2^{\log p}$ division attempts (by trying to guess p) to factor.

Example: the hardness of factoring

- Let $N = p \cdot q$ for random p, q such that $\log p \approx \log q$.
- It takes at most $2^{\frac{\log N}{2}} \approx 2^{\log p}$ division attempts (by trying to guess p) to factor.
- But there exist much faster attacks, such as the (general number field sieve, GNFS) that takes

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln N)^{\frac{1}{3}}(\ln \ln N)^{\frac{2}{3}}\right) \text{ CPU operations.}$$

Example: the hardness of factoring

- Let $N = p \cdot q$ for random p, q such that $\log p \approx \log q$.
- It takes at most $2^{\frac{\log N}{2}} \approx 2^{\log p}$ division attempts (by trying to guess p) to factor.
- But there exist much faster attacks, such as the (general number field sieve, GNFS) that takes
$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln N)^{\frac{1}{3}}(\ln \ln N)^{\frac{2}{3}}\right) \text{ CPU operations.}$$
- Choosing $\ln N$ appropriately, we can make sure the GNFS is too costly to run.

Example: the hardness of factoring

- Let $N = p \cdot q$ for random p, q such that $\log p \approx \log q$.
- It takes at most $2^{\frac{\log N}{2}} \approx 2^{\log p}$ division attempts (by trying to guess p) to factor.
- But there exist much faster attacks, such as the (general number field sieve, GNFS) that takes
$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln N)^{\frac{1}{3}}(\ln \ln N)^{\frac{2}{3}}\right) \text{ CPU operations.}$$
- Choosing $\ln N$ appropriately, we can make sure the GNFS is too costly to run.

Do we know for sure no better attack exists? No! The only option is to make our best effort to study the problem and new possible attacks.

Questions so far?

Quantum Computing

- Factoring- and DLOG- related hardness assumptions worked well so far. What changed?

Quantum Computing

- Factoring- and DLOG- related hardness assumptions worked well so far. What changed?
- In the 1980s, some physicists started thinking of using quantum mechanical phenomena to perform computations.

Quantum Computing

- Factoring- and DLOG- related hardness assumptions worked well so far. What changed?
- In the 1980s, some physicists started thinking of using quantum mechanical phenomena to perform computations.
- For a long time, very small practical improvements.
- In the last decade, a lot of money from industry went into this technology [MQT18, MN18, AAB+19, Gib19, WFG21]

Quantum Computing

- Factoring- and DLOG- related hardness assumptions worked well so far. What changed?
- In the 1980s, some physicists started thinking of using quantum mechanical phenomena to perform computations.
- For a long time, very small practical improvements.
- In the last decade, a lot of money from industry went into this technology [MQT18, MN18, AAB+19, Gib19, WFG21]

Quantum computers would represent a new kind of "resource" in the hands of attackers.

What do these computers do?

- Say you have a classical computer, a function f and two inputs x and y.
- To learn $f(x)$ and $f(y)$ you need to compute f two times.

What do these computers do?

- Say you have a classical computer, a function f and two inputs x and y.
- To learn $f(x)$ and $f(y)$ you need to compute f two times.

In a quantum computer (heavily simplifying, including notation):

What do these computers do?

- Say you have a classical computer, a function f and two inputs x and y.
- To learn $f(x)$ and $f(y)$ you need to compute f two times.

In a quantum computer (heavily simplifying, including notation):

- You encode x and y in a single "superposed" register, as $\alpha |x\rangle + \beta |y\rangle$ for some $\alpha, \beta \in \mathbb{C}$.

What do these computers do?

- Say you have a classical computer, a function f and two inputs x and y.
- To learn $f(x)$ and $f(y)$ you need to compute f two times.

In a quantum computer (heavily simplifying, including notation):

- You encode x and y in a single "superposed" register, as $\alpha|x\rangle + \beta|y\rangle$ for some $\alpha, \beta \in \mathbb{C}$.
- You compute f once, and obtain $\alpha|f(x)\rangle + \beta|f(y)\rangle$.
- (You can also apply changes to α and β.)

What do these computers do?

- Say you have a classical computer, a function f and two inputs x and y.
- To learn $f(x)$ and $f(y)$ you need to compute f two times.

In a quantum computer (heavily simplifying, including notation):

- You encode x and y in a single "superposed" register, as $\alpha |x\rangle + \beta |y\rangle$ for some $\alpha, \beta \in \mathbb{C}$.
- You compute f once, and obtain $\alpha |f(x)\rangle + \beta |f(y)\rangle$.
- (You can also apply changes to α and β.)
- You then read ("measure") the register.

- If you could read *both* $f(x)$ and $f(y)$ from $\alpha \left| f(x) \right\rangle + \beta \left| f(y) \right\rangle$, it would be free parallel computation!

- If you could read *both* $f(x)$ and $f(y)$ from $\alpha\,|f(x)\rangle + \beta\,|f(y)\rangle$, it would be free parallel computation!
- Fortunately, not quite. You will read either $f(x)$ with probablity $|\alpha|^2$ or $f(y)$ with probability $|\beta|^2$.
- (This generalises to any finite number of inputs, not just two.)

- If you could read *both* $f(x)$ and $f(y)$ from $\alpha\,|f(x)\rangle + \beta\,|f(y)\rangle$, it would be free parallel computation!
- Fortunately, not quite. You will read either $f(x)$ with probablity $|\alpha|^2$ or $f(y)$ with probability $|\beta|^2$.
- (This generalises to any finite number of inputs, not just two.)

Is this really that powerful then? How does it threaten cryptography?

- If you could read *both* $f(x)$ and $f(y)$ from $\alpha\,|f(x)\rangle + \beta\,|f(y)\rangle$, it would be free parallel computation!
- Fortunately, not quite. You will read either $f(x)$ with probablity $|\alpha|^2$ or $f(y)$ with probability $|\beta|^2$.
- (This generalises to any finite number of inputs, not just two.)

Is this really that powerful then? How does it threaten cryptography?

Unfortunately, yes. So far, in the form of two algorithms: Grover's and Shor's.

Questions so far?

Grover's algorithm

- Let L be a list of N different elements, randomly shuffled.

Grover's algorithm

- Let L be a list of N different elements, randomly shuffled.
- Say you know $x \in L$, but you need to find its index.

Grover's algorithm

- Let L be a list of N different elements, randomly shuffled.
- Say you know $x \in L$, but you need to find its index.
- Classically, this takes $O(N)$ comparisons.

Grover's algorithm

- Let L be a list of N different elements, randomly shuffled.
- Say you know $x \in L$, but you need to find its index.
- Classically, this takes $O(N)$ comparisons.
- Grover's algorithm lets you find x in $O(\sqrt{N})$ superposed comparisons.

Grover's algorithm

How does it affect cryptography?

Grover's algorithm

How does it affect cryptography?

- Say you have a cipher with 2^{128} possible secret keys.
- Classically, finding the right one takes $\approx 2^{128}$ attempts.

Grover's algorithm

How does it affect cryptography?

- Say you have a cipher with 2^{128} possible secret keys.
- Classically, finding the right one takes $\approx 2^{128}$ attempts.
- Quantumly, it may take $\approx \sqrt{2^{128}} = 2^{64}$ attempts.

Grover's algorithm

How does it affect cryptography?

- Say you have a cipher with 2^{128} possible secret keys.
- Classically, finding the right one takes $\approx 2^{128}$ attempts.
- Quantumly, it may take $\approx \sqrt{2^{128}} = 2^{64}$ attempts.

Every cipher is automatically weaker! Need keys twice as long!

(See my talk on Friday about why this may not be so clear in practice.)

Shor's algorithm

- Recall the runtime of the best factoring algorithm, GNFS:

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}}\right) \text{ CPU.}$$

Shor's algorithm

- Recall the runtime of the best factoring algorithm, GNFS:

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln N)^{\frac{1}{3}}(\ln \ln N)^{\frac{2}{3}}\right) \text{ CPU.}$$

- In 1994 Peter Shor develops a quantum algorithm running in

$$O\left((\log N)^2(\log \log N)(\log \log \log N)\right) \text{ quantum operations.}$$

Shor's algorithm

- Recall the runtime of the best factoring algorithm, GNFS:

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}}\right)\text{ CPU.}$$

- In 1994 Peter Shor develops a quantum algorithm running in

$$O\left((\log N)^2(\log\log N)(\log\log\log N)\right)\text{ quantum operations.}$$

- From subexponential in $\log N$ (hard!) to poly-logaritmic (easy!)

Shor's algorithm

- Recall the runtime of the best factoring algorithm, GNFS:

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}}\right) \text{ CPU.}$$

- In 1994 Peter Shor develops a quantum algorithm running in

$$O\left((\log N)^2(\log\log N)(\log\log\log N)\right) \text{ quantum operations.}$$

- From subexponential in $\log N$ (hard!) to poly-logaritmic (easy!)
- Worse news: it does not only affect factoring, but also DLOG!

Questions so far?

- In the span of one algorithm, we lost two families of hardness assumptions.
- In particular, the two that most PKC used commercially is based on.

- In the span of one algorithm, we lost two families of hardness assumptions.
- In particular, the two that most PKC used commercially is based on.
- This means that if/once a quantum computer capable of running Shor's algorithm, future encrypted communications will be at risk.

- In the span of one algorithm, we lost two families of hardness assumptions.
- In particular, the two that most PKC used commercially is based on.
- This means that if/once a quantum computer capable of running Shor's algorithm, future encrypted communications will be at risk.
- It also means that any such encrypted messages shared until then and stored, will also be at risk of decryption, even if today they are secure.

- In the span of one algorithm, we lost two families of hardness assumptions.
- In particular, the two that most PKC used commercially is based on.
- This means that if/once a quantum computer capable of running Shor's algorithm, future encrypted communications will be at risk.
- It also means that any such encrypted messages shared until then and stored, will also be at risk of decryption, even if today they are secure.

We need new hardness assumptions, that can't be solved with quantum computers.
We need "post-quantum" cryptography (PQC).

Towards Post-Quantum Cryptography

What would this upgrade entail? There are many steps.

Towards Post-Quantum Cryptography

What would this upgrade entail? There are many steps.

- Identify new hardness assumptions that resist quantum computing

Towards Post-Quantum Cryptography

What would this upgrade entail? There are many steps.

- Identify new hardness assumptions that resist quantum computing
- Design cryptographic primitives based on these, use them to upgrade more complex protocols

Towards Post-Quantum Cryptography

What would this upgrade entail? There are many steps.

- Identify new hardness assumptions that resist quantum computing
- Design cryptographic primitives based on these, use them to upgrade more complex protocols
- Produce secure implementations and legal standards

Towards Post-Quantum Cryptography

What would this upgrade entail? There are many steps.

- Identify new hardness assumptions that resist quantum computing
- Design cryptographic primitives based on these, use them to upgrade more complex protocols
- Produce secure implementations and legal standards
- Deploy in real-world systems

Let's start with the assumptions. There's many kinds, some newer, some older.

A variety of mathematical structures are used. A *very* non-exhaustive list includes:

Let's start with the assumptions. There's many kinds, some newer, some older.

A variety of mathematical structures are used. A *very* non-exhaustive list includes:

- Error-correcting codes (ECC)
- Polynomial rings and algebraic lattices
- Multivariate quadratic equation systems (MQ)

Let's start with the assumptions. There's many kinds, some newer, some older.

A variety of mathematical structures are used. A *very* non-exhaustive list includes:

- Error-correcting codes (ECC)
- Polynomial rings and algebraic lattices
- Multivariate quadratic equation systems (MQ)

Lattice-based and isogeny-based cryptography will be explained at AS Crypto on Tuesday!

Mathematical refresher: polynomials

Mathematical refresher: polynomials

- Given a ring like the integers $\mathbb{Z}$ or the integers modulo q, $\mathbb{Z}_q$,

Mathematical refresher: polynomials

- Given a ring like the integers $\mathbb{Z}$ or the integers modulo q, $\mathbb{Z}_q$,
- and an unknown variable x,

Mathematical refresher: polynomials

- Given a ring like the integers $\mathbb{Z}$ or the integers modulo q, $\mathbb{Z}_q$,
- and an unknown variable x,
- one can define the ring of polynomials $\mathbb{Z}[x]$ with elements $p(x)$ such that:

$$p(x) = p_0 + p_1 \cdot x + p_2 \cdot x^2 + \cdots + p_n \cdot x^n,$$

Mathematical refresher: polynomials

- Given a ring like the integers $\mathbb{Z}$ or the integers modulo q, $\mathbb{Z}_q$,
- and an unknown variable x,
- one can define the ring of polynomials $\mathbb{Z}[x]$ with elements $p(x)$ such that:
$$p(x) = p_0 + p_1 \cdot x + p_2 \cdot x^2 + \cdots + p_n \cdot x^n,$$
where $p_0, \ldots, p_n \in \mathbb{Z}$. We say n is the *degree* of p.

Mathematical refresher: polynomials

- Given a ring like the integers $\mathbb{Z}$ or the integers modulo q, $\mathbb{Z}_q$,
- and an unknown variable x,
- one can define the ring of polynomials $\mathbb{Z}[x]$ with elements $p(x)$ such that:
$$p(x) = p_0 + p_1 \cdot x + p_2 \cdot x^2 + \cdots + p_n \cdot x^n,$$
where $p_0, \ldots, p_n \in \mathbb{Z}$. We say n is the *degree* of p.
- We can add, multiply, and divide polynomials.

PQC from error-correcting codes

Originally ECC are used for communication over noisy channels, they allow to recover a clean signal from a damaged one

PQC from error-correcting codes

Originally ECC are used for communication over noisy channels, they allow to recover a clean signal from a damaged one

A famous cryptographic construction is from McEliece:

- Let G be a matrix generating a binary ECC of dimension k, correcting t errors
- Let S be a $k \times k$ random invertible matrix, and P a $n \times n$ permutation matrix
- Given a message $\boldsymbol{m} \in \{0,1\}^k$, encode it and perturb it on t indices, using $\boldsymbol{z} \in \{0,1\}^n$ of Hamming weight t.

PQC from error-correcting codes

Originally ECC are used for communication over noisy channels, they allow to recover a clean signal from a damaged one

A famous cryptographic construction is from McEliece:

- Let G be a matrix generating a binary ECC of dimension k, correcting t errors
- Let S be a $k \times k$ random invertible matrix, and P a $n \times n$ permutation matrix
- Given a message $\boldsymbol{m} \in \{0,1\}^k$, encode it and perturb it on t indices, using $\boldsymbol{z} \in \{0,1\}^n$ of Hamming weight t.

The hardness assumption [McE78]

Dados t, $G^{\text{pub}} := SGP$ y $\boldsymbol{c} := \boldsymbol{m}G^{\text{pub}} \oplus \boldsymbol{z}$, recuperar $\boldsymbol{m}$

PQC using polynomial rings

Let

- $n, q \in \mathbb{Z}$ and $\phi \in \mathbb{Z}[x]$ be a monic irreducible polynomial of degree n,
- $\mathcal{R}_q := \mathbb{Z}_q[x]/(\phi)$,
- $f \in \mathcal{R}_q^\times$ and $g \in \mathcal{R}_q$ be polynomials with small coefficients (eg. in $\{-1, 0, 1\}$).

PQC using polynomial rings

Let

- $n, q \in \mathbb{Z}$ and $\phi \in \mathbb{Z}[x]$ be a monic irreducible polynomial of degree n,
- $\mathcal{R}_q := \mathbb{Z}_q[x]/(\phi)$,
- $f \in \mathcal{R}_q^\times$ and $g \in \mathcal{R}_q$ be polynomials with small coefficients (eg. in $\{-1, 0, 1\}$).

NTRU [HPS98]

Given $h := g/f \mod q$, recover g or f.

More PQC using polynomial rings

Let

- $k, q \in \mathbb{Z}$, $n := 2^k$ y $\phi = x^n + 1$,
- $\mathcal{R}_q := \mathbb{Z}_q[x]/(\phi)$,
- $a \leftarrow U(\mathcal{R}_q)$, and $s, e \in \mathcal{R}_q$ sampled such that the coefficients follow a Gaussian distribution rounded to the nearest integer in $[-q/2, q/2)$.

More PQC using polynomial rings

Let

- $k, q \in \mathbb{Z}$, $n := 2^k$ y $\phi = x^n + 1$,
- $\mathcal{R}_q := \mathbb{Z}_q[x]/(\phi)$,
- $a \leftarrow U(\mathcal{R}_q)$, and $s, e \in \mathcal{R}_q$ sampled such that the coefficients follow a Gaussian distribution rounded to the nearest integer in $[-q/2, q/2)$.

Search Ring Learning With Errors (RLWE) [Reg05, SSTX09, LPR10]

Given $(a, b := a \cdot s + e \mod q) \in \mathcal{R}_q \times \mathcal{R}_q$, recover s.

More PQC using polynomial rings

Let

- $k, q \in \mathbb{Z}$, $n := 2^k$ y $\phi = x^n + 1$,
- $\mathcal{R}_q := \mathbb{Z}_q[x]/(\phi)$,
- $a \leftarrow U(\mathcal{R}_q)$, and $s, e \in \mathcal{R}_q$ sampled such that the coefficients follow a Gaussian distribution rounded to the nearest integer in $[-q/2, q/2)$.

Search Ring Learning With Errors (RLWE) [Reg05, SSTX09, LPR10]

Given $(a, b := a \cdot s + e \mod q) \in \mathcal{R}_q \times \mathcal{R}_q$, recover s.

Decision Ring Learning With Errors (RLWE) [Reg05, SSTX09, LPR10]

Given $(a, b) \in \mathcal{R}_q \times \mathcal{R}_q$, guess whether $b \sim U(\mathcal{R}_q)$ o si $b = a \cdot s + e \mod q$.

PQC from multivariate quadratic equation systems

Let

- $q, n, m \in \mathbb{Z}$ be integers and $\mathbb{F}_q$ be the finite field of q elements,
- $p_1(\boldsymbol{x}), \ldots, p_m(\boldsymbol{x})$ be quadratic polynomials over n variables $\boldsymbol{x} = (x_1, \ldots, x_n)$.

PQC from multivariate quadratic equation systems

Let

- $q, n, m \in \mathbb{Z}$ be integers and $\mathbb{F}_q$ be the finite field of q elements,
- $p_1(\boldsymbol{x}), \ldots, p_m(\boldsymbol{x})$ be quadratic polynomials over n variables $\boldsymbol{x} = (x_1, \ldots, x_n)$.

Multivariate Quadratic (MQ)

Given the $p_1, \ldots, p_m$ polynomials, find a solution $\boldsymbol{y}$ to the system of equations $p_1(\boldsymbol{y}) = \cdots = p_m(\boldsymbol{y}) = \boldsymbol{0} \mod q$, if it exists.

Questions so far?

- Given new assumptions, one needs new designs.

- Given new assumptions, one needs new designs.
- Sometimes similarities between “pre-quantum” and “post-quantum” assuptions means the designs can be similar.

- Given new assumptions, one needs new designs.
- Sometimes similarities between “pre-quantum” and “post-quantum” assuptions means the designs can be similar.
- Even in those cases subtle difference may be introduced.

For example, there are some similarities between LWE variants and DLOG:

For example, there are some similarities between LWE variants and DLOG:

Similarity between RLWE and DLOG

“given $(a, a \cdot s + e)$, recover s” $\sim$ “given (g, g^x), recover x”

For example, there are some similarities between LWE variants and DLOG:

Similarity between RLWE and DLOG

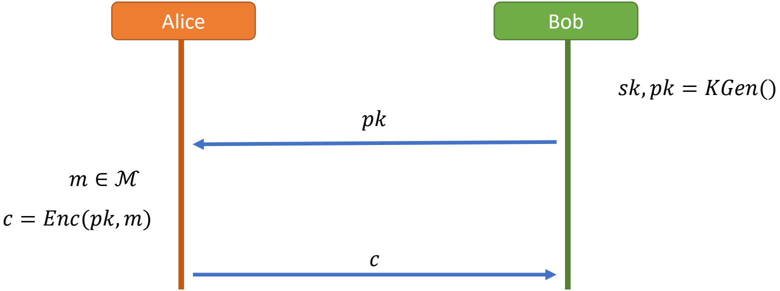
“given $(a, a \cdot s + e)$, recover s” $\sim$ “given (g, g^x), recover x”

Let’s try using this to port a DLOG primitive to RLWE.

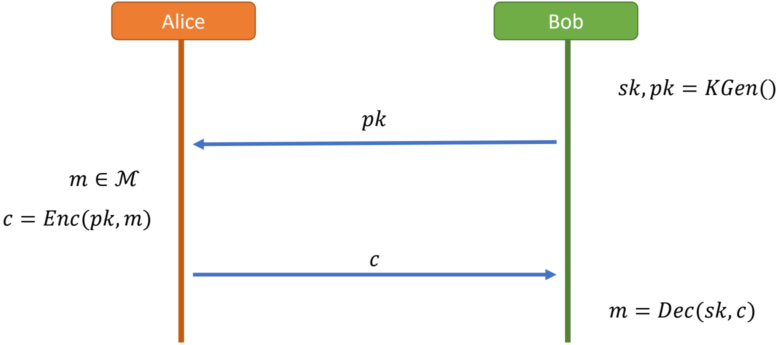
- I will be presenting passively-secure ElGamal encryption
- This is a classic public-key encryption scheme, very close to Diffie-Hellman key exchange

- I will be presenting passively-secure ElGamal encryption
- This is a classic public-key encryption scheme, very close to Diffie-Hellman key exchange

- I will be presenting passively-secure ElGamal encryption
- This is a classic public-key encryption scheme, very close to Diffie-Hellman key exchange



- I will be presenting passively-secure ElGamal encryption
- This is a classic public-key encryption scheme, very close to Diffie-Hellman key exchange



Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^{\times}$.

<u>KGen():</u>

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

<u>KGen():</u>

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

<u>Enc(pk, m):</u>

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x}$

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x}$

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $pk \leftarrow (g, h := g^x)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

<u>KGen():</u>

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$,

<u>Enc(pk, m):</u>

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

<u>Dec($sk, (c_1, c_2)$):</u>

- $m' \leftarrow c_2/c_1^x = h^y \cdot m/g^{y\,x} = (g^x)^y \cdot m/g^{y\,x} = m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$,
- $c_2 \leftarrow h^y \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$, $c_1 \leftarrow a \cdot r + f$,
- $c_2 \leftarrow h^y \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$, $c_1 \leftarrow a \cdot r + f$,
- $c_2 \leftarrow h^y \cdot m$, $c_2 \leftarrow b \cdot r + f' + \frac{q}{2} \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2/c_1^x = h^y \cdot m/g^{y\,x} = (g^x)^y \cdot m/g^{y\,x} = m$,

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$, $c_1 \leftarrow a \cdot r + f$,
- $c_2 \leftarrow h^y \cdot m$, $c_2 \leftarrow b \cdot r + f' + \frac{q}{2} \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,
  $m' \leftarrow \;\; c_2 - s \cdot c_1$

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$, $c_1 \leftarrow a \cdot r + f$,
- $c_2 \leftarrow h^y \cdot m$, $c_2 \leftarrow b \cdot r + f' + \frac{q}{2} \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,
  $m' \leftarrow c_2 - s \cdot c_1 = (b \cdot r + f' + \frac{q}{2} \cdot m) - s \cdot (a \cdot r + f)$

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$, $c_1 \leftarrow a \cdot r + f$,
- $c_2 \leftarrow h^y \cdot m$, $c_2 \leftarrow b \cdot r + f' + \frac{q}{2} \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{yx} = (g^x)^y \cdot m / g^{yx} = m$,
  $m' \leftarrow c_2 - s \cdot c_1 = (b \cdot r + f' + \frac{q}{2} \cdot m) - s \cdot (a \cdot r + f) = \frac{q}{2} \cdot m + e \cdot r - s \cdot f + f'$

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$, $c_1 \leftarrow a \cdot r + f$,
- $c_2 \leftarrow h^y \cdot m$, $c_2 \leftarrow b \cdot r + f' + \frac{q}{2} \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2/c_1^x = h^y \cdot m/g^{y\,x} = (g^x)^y \cdot m/g^{y\,x} = m$,
  $m' \leftarrow \lfloor c_2 - s \cdot c_1 \rceil = \lfloor (b \cdot r + f' + \frac{q}{2} \cdot m) - s \cdot (a \cdot r + f) \rceil = \lfloor \frac{q}{2} \cdot m + e \cdot r - s \cdot f + f' \rceil$

Let $\langle g \rangle$ be a large subgroup of $\mathbb{F}_q^\times$. Let $a \sim U(\mathbb{Z}_q[x]/\langle \phi \rangle)$, $\phi = x^n + 1$.

KGen():

- $sk \leftarrow x \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $sk \leftarrow (s, e) \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $pk \leftarrow (g, h := g^x)$, $pk \leftarrow (a, b := a \cdot s + e)$,

Enc(pk, m):

- $y \sim U(\mathbb{Z}_{|\langle g \rangle|})$, $(r, f, f') \sim U(\mathbb{Z}_2[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle) \times \chi(\mathbb{Z}_q[x]/\langle \phi \rangle)$,
- $c_1 \leftarrow g^y$, $c_1 \leftarrow a \cdot r + f$,
- $c_2 \leftarrow h^y \cdot m$, $c_2 \leftarrow b \cdot r + f' + \frac{q}{2} \cdot m$,

Dec($sk, (c_1, c_2)$):

- $m' \leftarrow c_2 / c_1^x = h^y \cdot m / g^{y\,x} = (g^x)^y \cdot m / g^{y\,x} = m$,
  $m' \leftarrow \lfloor c_2 - s \cdot c_1 \rceil = \lfloor (b \cdot r + f' + \frac{q}{2} \cdot m) - s \cdot (a \cdot r + f) \rceil = \lfloor \frac{q}{2} \cdot m + e \cdot r - s \cdot f + f' \rceil$
  $= \frac{q}{2} \cdot m$ *with high probability.*

Questions so far?

Secure implementations and legal standards

- Secure implementations is a giant field in cryptography
- It is not specific to post-quantum cryptography, so I will not be covering it

Secure implementations and legal standards

- Secure implementations is a giant field in cryptography
- It is not specific to post-quantum cryptography, so I will not be covering it
- However lots of post-quantum research is going on, as deployment gets closer

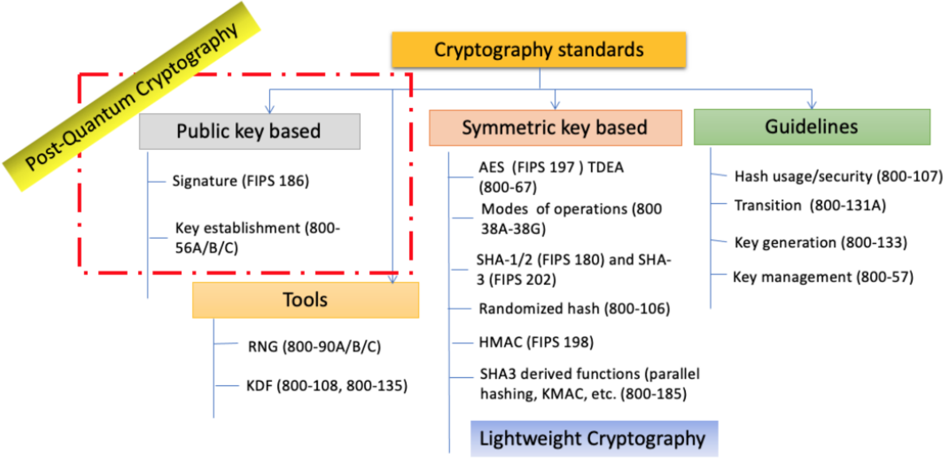
Secure implementations and legal standards

- Secure implementations is a giant field in cryptography
- It is not specific to post-quantum cryptography, so I will not be covering it
- However lots of post-quantum research is going on, as deployment gets closer
- Keep an eye on the CHES conference publications: https://tches.iacr.org

- In terms of standardisation, multiple processes are ongoing.

- In terms of standardisation, multiple processes are ongoing.
- The most prominent effort has been run by the US National Institute of Standards and Technology (NIST).

- In terms of standardisation, multiple processes are ongoing.
- The most prominent effort has been run by the US National Institute of Standards and Technology (NIST).



- In 2016 they made an open call for proposals to design post-quantum digital signatures (DSA) and key encapsulation mechanisms (KEM, think: PKE)

- In 2016 they made an open call for proposals to design post-quantum digital signatures (DSA) and key encapsulation mechanisms (KEM, think: PKE)
- In 2017 the 69 submissions were presented.

- In 2016 they made an open call for proposals to design post-quantum digital signatures (DSA) and key encapsulation mechanisms (KEM, think: PKE)
- In 2017 the 69 submissions were presented.
- After multiple review rounds, in 2023 the first draft standards have been posted for comment, `https://csrc.nist.gov/projects/post-quantum-cryptography`

Four algorithms are being standardised:

Four algorithms are being standardised:

- ML-KEM: a lattice-based KEM proposed with the name Kyber

Four algorithms are being standardised:

- ML-KEM: a lattice-based KEM proposed with the name Kyber
- ML-DSA and NT-DSA: two lattice-based signature schemes proposed as Dilithium and Falcon

Four algorithms are being standardised:

- ML-KEM: a lattice-based KEM proposed with the name Kyber
- ML-DSA and NT-DSA: two lattice-based signature schemes proposed as Dilithium and Falcon
- SLH-DSA: a hash-based signature scheme known as Sphincs+

- Meanwhile, some more KEM schemes are still in consideration as part of the original process
- NIST also started a second process exclusively for more digital signatures

- Meanwhile, some more KEM schemes are still in consideration as part of the original process
- NIST also started a second process exclusively for more digital signatures

Discussions about standardisation can be followed on
https://csrc.nist.gov/Projects/post-quantum-cryptography/Email-List

Ok, we have new hardness assumptions, primitives, and standards. We can deploy, right?

Ok, we have new hardness assumptions, primitives, and standards. We can deploy, right?

- Not so easy in practice!

Ok, we have new hardness assumptions, primitives, and standards. We can deploy, right?

- Not so easy in practice!
- PQC algorithms tend to have larger public keys and/or ciphertexts!

Ok, we have new hardness assumptions, primitives, and standards. We can deploy, right?

- Not so easy in practice!
- PQC algorithms tend to have larger public keys and/or ciphertexts!
  - RSA (128-bits security): $|pk| = |c| = 384$ B
  - EC-ElGamal (128-bit security): $|pk| = 32$ B, $|c| = 64$ B

Ok, we have new hardness assumptions, primitives, and standards. We can deploy, right?

- Not so easy in practice!
- PQC algorithms tend to have larger public keys and/or ciphertexts!
  - RSA (128-bits security): $|pk| = |c| = 384$ B
  - EC-ElGamal (128-bit security): $|pk| = 32$ B, $|c| = 64$ B
  - ML-KEM (128-bit security): $|pk| = 800$ B, $|c| = 768$ B

Ok, we have new hardness assumptions, primitives, and standards. We can deploy, right?

- Not so easy in practice!
- PQC algorithms tend to have larger public keys and/or ciphertexts!
  - RSA (128-bits security): $|pk| = |c| = 384$ B (signatures: $|\sigma| = 384$ B)
  - EC-ElGamal (128-bit security): $|pk| = 32$ B, $|c| = 64$ B (signatures: $|\sigma| = 65$ B)
  - ML-KEM (128-bit security): $|pk| = 800$ B, $|c| = 768$ B

Ok, we have new hardness assumptions, primitives, and standards. We can deploy, right?

- Not so easy in practice!
- PQC algorithms tend to have larger public keys and/or ciphertexts!
  - RSA (128-bits security): $|pk| = |c| = 384$ B (signatures: $|\sigma| = 384$ B)
  - EC-ElGamal (128-bit security): $|pk| = 32$ B, $|c| = 64$ B (signatures: $|\sigma| = 65$ B)
  - ML-KEM (128-bit security): $|pk| = 800$ B, $|c| = 768$ B
  - SLH-DSA (128-bit security): $|pk| = 32$ B, $|\sigma| = 7856$ B

Why is this a problem?

Why is this a problem?

- If your protocol sends a lot of keys, ciphertext or signatures, increased cost and delays

Why is this a problem?

- If your protocol sends a lot of keys, ciphertext or signatures, increased cost and delays
- Even worse: what if your protocol implementation *assumes* fixed sizes?

```
unsigned char ciphertext[64]
```

Why is this a problem?

- If your protocol sends a lot of keys, ciphertext or signatures, increased cost and delays
- Even worse: what if your protocol implementation *assumes* fixed sizes?

```
unsigned char ciphertext[64]
```

- A lot of *sensitive* code will need rewriting, with all the risks that follow! (Eg., CVE-2022-21449: Psychic Signatures in Java)

Not only issues with size.

Not only issues with size.

Some of these problems have not been studied as much. Could they break?

Not only issues with size.

Some of these problems have not been studied as much. Could they break?

- Even though RSA and DLOG existed since the 70s, and stadards like PKCS #1 v1.1 dates back to 1992, their cryptanalysis was not stable until the mid-90s [Len93].

Not only issues with size.

Some of these problems have not been studied as much. Could they break?

- Even though RSA and DLOG existed since the 70s, and stadards like PKCS #1 v1.1 dates back to 1992, their cryptanalysis was not stable until the mid-90s [Len93].
- In the same way, schemes like Rainbow (a NIST signature scheme finalist first defined in 2005) was fully broken by Beullens in 2022 [Beu22].
- And the SIKE scheme (a NIST KEM finalist, defined in 2011) was fully broken in 2022 [CD23]

Not only issues with size.

Some of these problems have not been studied as much. Could they break?

- Even though RSA and DLOG existed since the 70s, and stadards like PKCS #1 v1.1 dates back to 1992, their cryptanalysis was not stable until the mid-90s [Len93].
- In the same way, schemes like Rainbow (a NIST signature scheme finalist first defined in 2005) was fully broken by Beullens in 2022 [Beu22].
- And the SIKE scheme (a NIST KEM finalist, defined in 2011) was fully broken in 2022 [CD23]
- A lot of work in cryptanalysis left to do!

But we need PQC as soon as possible!

But we need PQC as soon as possible!

- Use hybrid schemes!

But we need PQC as soon as possible!

- Use hybrid schemes!
- For PKE: encrypt with EC-ElGamal, and encrypt the result with ML-KEM

But we need PQC as soon as possible!

- Use hybrid schemes!
- For PKE: encrypt with EC-ElGamal, and encrypt the result with ML-KEM
- For signatures: sign with (say) EC-DSA and ML-DSA, verify *both* signatures

Conclusions

- PQC has received a significant boost in research and industry effort

Conclusions

- PQC has received a significant boost in research and industry effort
- Regardless of whether QC ever happen, legal requirements mean that PQC will be deployed in the near term future

Conclusions

- PQC has received a significant boost in research and industry effort
- Regardless of whether QC ever happen, legal requirements mean that PQC will be deployed in the near term future
- Lots of research is currently happening: theoretical and practical issues remain open, and make for a good space to perform research

Conclusions

- PQC has received a significant boost in research and industry effort
- Regardless of whether QC ever happen, legal requirements mean that PQC will be deployed in the near term future
- Lots of research is currently happening: theoretical and practical issues remain open, and make for a good space to perform research

Thank you

Slides @ https://fundamental.domains

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, and William *et al.* Courtney.
Quantum supremacy using a programmable superconducting processor.
Nature, 574(7779):505–510, Oct 2019.

Ward Beullens.
Breaking rainbow takes a weekend on a laptop.
IACR Cryptol. ePrint Arch., page 214, 2022.

Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen.
On the hardness of module-lwe with binary secret.
In *Cryptographers' Track at the RSA Conference*, pages 503–526. Springer, 2021.

Wouter Castryck and Thomas Decru.
An efficient key recovery attack on sidh.
In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 423–447. Springer, 2023.

Elizabeth Gibney.
Quantum gold rush: the private funding pouring into quantum start-ups.
Nature, 574(7776):22–24, October 2019.

Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman.
NTRU: A ring-based public key cryptosystem.

In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*, pages 267–288. Springer, Heidelberg, June 1998.

Paul Kirchner and Pierre-Alain Fouque.
An improved bkw algorithm for lwe with applications to cryptography and lattices.
In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35*, pages 43–62. Springer, 2015.

H. W. Lenstra.
The number field sieve: An annotated bibliography.
In Arjen K. Lenstra and Hendrik W. Lenstra, editors, *The development of the number field sieve*, pages 1–3, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev.
On ideal lattices and learning with errors over rings.
In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

Robert J. McEliece.
A public-key cryptosystem based on algebraic coding theory.
The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology, January/February 1978.
https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF.

Daniele Micciancio.
On the hardness of learning with errors with binary secrets.

Theory of Computing, 14(1):1–17, 2018.

Samuel K. Moore and Amy Nordrum.
Intel's new path to quantum computing.
IEEE Spectrum, 2018.

Microsoft Quantum Team.
Developing a topological qubit.
Cloud Perspectives Blog, 2018.

Oded Regev.
On lattices, learning with errors, random linear codes, and cryptography.
In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa.
Efficient public key encryption based on ideal lattices.
In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, December 2009.

Chao Sun, Mehdi Tibouchi, and Masayuki Abe.
Revisiting the hardness of binary error lwe.
In *Australasian Conference on Information Security and Privacy*, pages 425–444. Springer, 2020.

Karl Wehden, Ismael Faro, and Jay Gambetta.
IBM's roadmap for building an open quantum software ecosystem.
IBM Research Blog, 2021.