Cryptography

1. Goals and Resources of an attacker

Contact : kevin.carrier@cyu.fr

Plan

1. Fundamental principle of modern cryptography
2. Why use cryptography ?
3. The attacker model
4. Side Channel Attack
5. The Dolev-Yao model
6. What can compute an attacker ?
7. Quantum computing: a new paradigm

Fundamental principle of modern cryptography

Fundamental principle of modern cryptography

*"A cryptosystem should be secure even if **everything** about the system, except the secret key, **is public knowledge**."*

[August Kerckhoffs (1883)]

The three items to keep in mind:

1. **Computational or Semantical security:** The system must be practically, if not mathematically, indecipherable;
2. **Transparency:** It should not require secrecy, and it should not be a problem if it falls into enemy hands;
3. **Portability :** It must be possible to communicate and remember the key without using written notes, and correspondents must be able to change or modify it at will;

Why use cryptography ?

Why use cryptography ?

- **Confidentiality:** Protection against disclosure of unauthorized information. Only authorized persons have access to the content of the message.

Why use cryptography ?

- **Confidentiality:** Protection against disclosure of unauthorized information. Only authorized persons have access to the content of the message.

Forward secrecy

For instance, in France, the confidentiality of a classified document must be guarantee for 50 years.

Why use cryptography ?

- **Integrity:** The message cannot be changed without noticing.

Why use cryptography ?

- **Authentication:** It is a process allowing an entity to be sure of the identity of a second entity based on corroborating evidence.

Why use cryptography ?

- **Authentication:** It is a process allowing an entity to be sure of the identity of a second entity based on corroborating evidence.

Strong Authentication

The authentication must lean on at least 2 secret elements that only the entity to authenticate has. This can be something:

- he knows (a password);
- he holds (a smartcard);
- he is (biometrics).

Why use cryptography ?

- **Non-repudiation:**
  - **non-repudiation of origin:** The sender cannot deny having written the message and he can prove that he did not do so if this is indeed the case.
  - **non-repudiation of receipt:** The receiver cannot deny having received the message and he can prove that he did not reuse it if this is indeed the case.
  - **non-repudiation of transmission:** The sender cannot deny having sent the message and can prove that he did not do so if this is indeed the case.

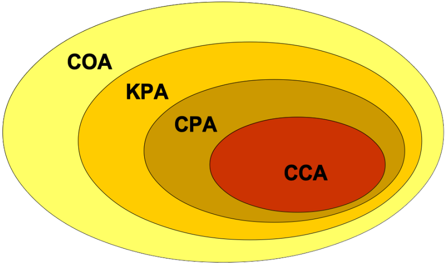
The attacker model

The attacker model

In cryptography, we define several models of attackers. Here are the most used:

- **Ciphertext-Only Attack (COA):** The attacker has only one or more ciphertexts that he wishes to decrypt.
- **Known Plaintext Attack (KPA):** The attacker not only has access to several ciphertexts but also to the corresponding plaintexts.
- **Chosen Plaintext Attack (CPA):** The attacker owns an encryption machine. So the attacker can encrypt all the messages he wants.
- **Chosen Ciphertext Attack (CCA):** The attacker owns a decryption machine. So the attacker can choose any ciphertext then decipher it to get its associated plaintext.

The attacker model



What can be the goal of an attacker?

- Find the secret/decryption key.
- More modestly, decrypt a particular ciphertext without necessarily discovering the key.

The attacker model

An additional requirement: **indistinguishability**

- IND-CPA, IND-CCA
- Given two ciphertexts and one plaintext, an attacker should not be able to know which ciphertext is associated with the plaintext.

The attacker model

What does he have access to?

- **black-box cryptography**: Computations are performed remotely (the attacker does not have access to the encryption or decryption machine).
- **white-box cryptography**: The attacker knows each step of the execution of the encryption/decryption algorithm (reverse engineering, debugger)
- **grey-box cryptography**: Side channel attack. The attacker have access to some partial information that leaks during the execution of the encryption/decryption algorithm.

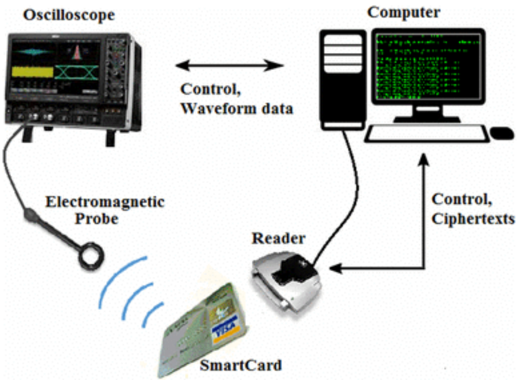
Side Channel Attack

Side Channel Attack

Initiated by Paul Kocker in the 90s.

- **Time attack**: The attacker can measure the running time of the encryption/decryption algorithm.
- **Power attack**: The attacker can measure the power consumption during the execution of the algorithm.
- **Electromagnetic attack**: The attacker can measure the electromagnetic radiation during the execution of the algorithm.
- **Micro-Ultrasound attack**: The attacker can measure the Micro-Ultrasound during the execution of the algorithm.

Side Channel Attack



Time Attack: Exercise

Let suppose a SmartCard used to authenticate. The user sent a PIN to the SmartCard which returns `ACCEPT` if the PIN is the same as the one saved in the card and `REJECT` otherwise. Here is the C implementation of the authentication protocol:

SmartCard.c

```c
#include <stdio.h>
#include <string.h>

char *PIN = [redacted]

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s <text_to_display>\n", argv[0]);
        return 1;
    }
    char *PIN_tmp = argv[1];
    if (strlen(PIN_tmp) == strlen(PIN)){
        for (int i = 0; i < strlen(PIN); ++i){
            if (PIN_tmp[i] != PIN[i]){
                printf("REJECT\n");
                return 1;
            }
        }
        printf("ACCEPT\n");
        return 0;
    } else {
        printf("REJECT\n");
        return 1;
    }
}
```

Assume we can simulate the use of the card by executing the `SmartCard.exe` binary. However, a device allows us to measure the execution time of the algorithm.

Propose an attack to find the size of the PIN then the PIN itself. Then propose a countermeasure.

The Dolev-Yao model

The Dolev-Yao model

An attacker model often consider in **Network security** is the **Dolev-Yao model**. Here it is supposed the attacker:

- can get all the messages circulating in the network ;
- can initiate a conversation with any member of the network ;
- can send a message to any member of the network pretending to be any member of the network ;
- **cannot** guess a integer which has been chosen uniformly at random ;
- **cannot** guess a private key associated to a public key.

What can compute an attacker ?

What can compute an attacker ?

Two notions of “security”:

1) The designer wants to achieve **unconditional security**: He can proof his cryptosystem is secure without prejudging the computing power of the attacker that can even be infinite !
$\rightarrow$ In particular, if a pair (plaintext,ciphertext) gives no information about the key, then we say the cryptosystem is **perfectly secure**. We will see later that is a notion that is difficult to achieve in practice.

What can compute an attacker ?

Two notions of “security”:

1) The designer wants to achieve **unconditional security**: He can proof his cryptosystem is secure without prejudging the computing power of the attacker that can even be infinite !
$\rightarrow$ In particular, if a pair (plaintext,ciphertext) gives no information about the key, then we say the cryptosystem is **perfectly secure**. We will see later that is a notion that is difficult to achieve in practice.

2) The **computational security** is based on the impossibility to decrypt a message or recover the secret key in a reasonable time, considering the computing power of a potential attacker. This notion of security depend on the state-of-the-art at a given moment.
$\rightarrow$ To be sure to be computationally secure, we need **proof of security** or **security reduction**. It consists in reducing the fact of “breaking a cryptosystem” to “solving a hard mathematical problem”

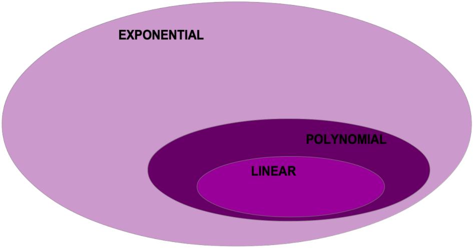
What is a hard problem ?

Definition (Decision Problem)

A **decision problem** is a type of computational problem where the answer is either **yes** or **no** for a (binary) input of size n.

A decision problem has time complexity $f(n)$ if the number of state transitions (steps) required by a deterministic Turing machine on an input of (binary) size n to output the answer **yes** or **no**.

- **Linear**: $f(n) = an + b = O(n)$
- **Polynomial**: $f(n) = a_0 + a_1 n + a_2 n^2 + \cdots + a_d n^d = O(n^d)$
- **Exponential**: $f(n) = O(2^{\alpha n})$



What is a hard problem ?

Definition (Class of P problems)

A decision problem is in the complexity class P if its time complexity is polynomial.

What is a hard problem ?

Definition (Non-Deterministic Turing Machine)

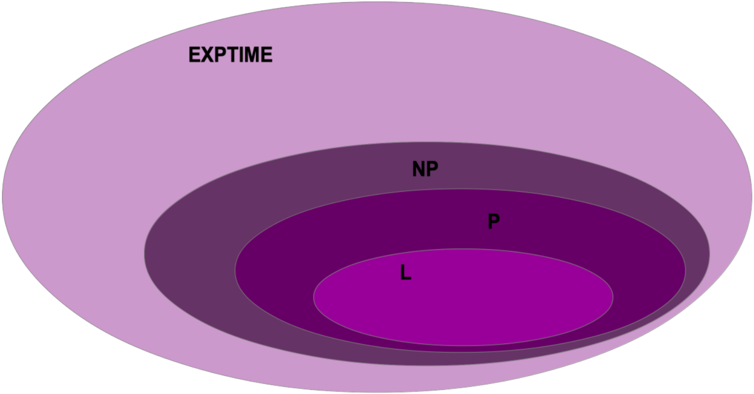
A variant of deterministic Turing machine is **non-deterministic Turing machine**. For every input at a state, there can be multiple paths/actions performed by the Turing machine. So, the transitions are not deterministic.

Definition (Class of NP problems)

A decision problem is in the complexity class NP if the number of state transitions required by a **non-deterministic Turing machine** on an input of size n to output the answer **yes** or **no** is **polynomial**.

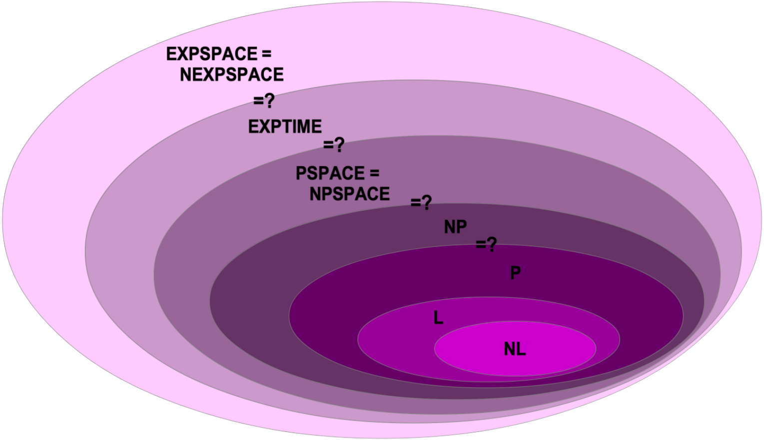
Essentially, a decision problem for which we can “verify” that a solution is right in a polynomial time is in the NP class.

What is a hard problem ?



What is a hard problem ?

We can consider **space** instead of **time** to define some other classes of problems: PSPACE, NPSPACE...



What is a hard problem ?

$\rightarrow$ **One million dollar:** $P = NP$**?**

What is a hard problem ?

$\rightarrow$ **One million dollar:** $P = NP$**?** Probably not...

What is a hard problem ?

$\rightarrow$ **One million dollar:** $P = NP$**?** Probably not...

The **NP-complete** problems is a sub-class of NP. A problem is NP-complete if all the NP problems are at least as hard as it. In other words, a problem A is NP-complete if for any NP problem B, there is a polynomial reduction of B to A.

$\Rightarrow$ They are the hardest problems of the NP class.

What is a hard problem ?

$\rightarrow$ **One million dollar:** $P = NP$**?** Probably not...

The **NP-complete** problems is a sub-class of NP. A problem is NP-complete if all the NP problems are at least as hard as it. In other words, a problem A is NP-complete if for any NP problem B, there is a polynomial reduction of B to A.
$\Rightarrow$ They are the hardest problems of the NP class.

$\Rightarrow$ For proving $P = NP$, one only has to choose one NP-complete problem and prove it is P.

What is a hard problem ?

$\rightarrow$ **One million dollar:** $P = NP$**?** Probably not...

The **NP-complete** problems is a sub-class of NP. A problem is NP-complete if all the NP problems are at least as hard as it. In other words, a problem A is NP-complete if for any NP problem B, there is a polynomial reduction of B to A.
$\Rightarrow$ They are the hardest problems of the NP class.

$\Rightarrow$ For proving $P = NP$, one only has to choose one NP-complete problem and prove it is P.

In conclusion, a NP-complete problem is believed to be hard and so we want to reduce our cryptosystems to such problems.

What is a hard problem ?

Boolean satisfiability problem (SAT) is NP-complete

Determine if there exists an interpretation (Boolean inputs) that satisfies a given Boolean formula.

3-SAT is NP-complete

SAT with Boolean formula in conjonctive normal form.

Travelling Salesman Problem (TSP) is NP-complete

Given a length L, the task is to decide whether a graph has a tour whose length is at most L.

The decoding problem is NP-complete [McEliece, 78]

Decide if there is a solution to a linear system with an Hamming weight constraint on this solution.

What is a hard problem ?

Contrary to popular belief, the **factorization** problem and the **discreet logarithm** problem (which are the main mathematical problems on which is based the modern cryptography) are NP-hard but **not NP-complete**!

Computational security

Definition (bits of security)

When we say a cryptosystem has x bits of security, that means an attacker needs $O(2^x)$ elementary operations to break it.

- $O(2^{30})$: reasonable limit of what a powerful computer can do.
- $\geq 2^{80}$: we consider it is secure.
- $\geq 2^{128}$ or $\geq 2^{256}$: what standardization organisms ask.

Computational security

Definition (bits of security)

When we say a cryptosystem has x bits of security, that means an attacker needs $O(2^x)$ elementary operations to break it.

- $O(2^{30})$: reasonable limit of what a powerful computer can do.
- $\geq 2^{80}$: we consider it is secure.
- $\geq 2^{128}$ or $\geq 2^{256}$: what standardization organisms ask.

Theorem (bits of security and key size)

If the key we want to recover is encoded with n bits, then the cryptosystem has ***at most*** *n bits of security.*

Proof. *Brute force attack*

Computational security

Definition (bits of security)

When we say a cryptosystem has x bits of security, that means an attacker needs $O(2^x)$ elementary operations to break it.

- $O(2^{30})$: reasonable limit of what a powerful computer can do.
- $\geq 2^{80}$: we consider it is secure.
- $\geq 2^{128}$ or $\geq 2^{256}$: what standardization organisms ask.

Theorem (bits of security and key size)

*If the key we want to recover is encoded with n bits, then the cryptosystem has **at most** n bits of security.*

Proof. *Brute force attack*

⚠ **The converse is false!**

Quantum computing: a new paradigm

Quantum computing: a new paradigm

The quantum attacker model

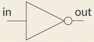
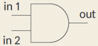
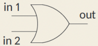
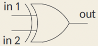
An attacker which has access to a quantum computer defined a new attacker model.

Quantum computers immerse us in a new world where:

- The NP-completeness no longer makes sense
- The factorization and the discreet logarithm problem can be solved in a polynomial time with the **Shor algorithm** (1994)
- Searching a particular element in an unstructured set of 2^N elements has often a cost of order $\leq 2^{N/2}$ operations thanks to the **Grover algorithm** (1996)

Quantum computing: a new paradigm

How does a classical computer work?

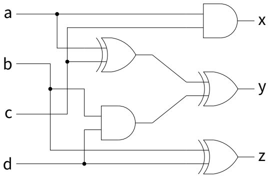
<table>
<tr><th>Porte logique</th><th>Symbole électronique</th><th>Table de vérité</th></tr>
<tr><td>NO</td><td></td><td><table><tr><th>in</th><th>out</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></td></tr>
<tr><td>AND</td><td></td><td><table><tr><th>in 1</th><th>in 2</th><th>out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>OR</td><td></td><td><table><tr><th>in 1</th><th>in 2</th><th>out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>XOR</td><td></td><td><table><tr><th>in 1</th><th>in 2</th><th>out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td></tr>
</table>

For instance:

1: electric current passes

0 : electric current does not passe

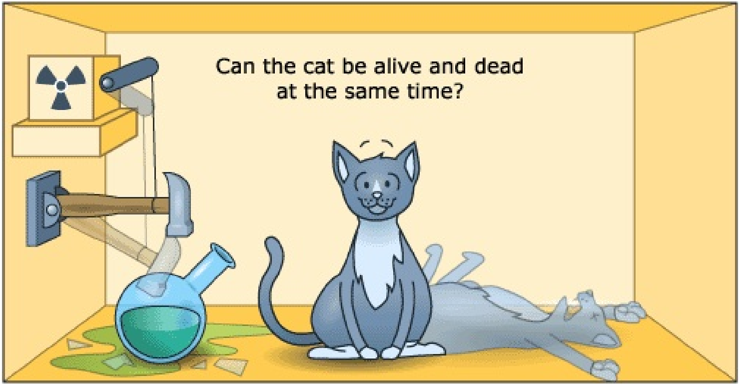
It forms a Turing-complete machine.
Example of an algorithm performing an addition in base 2:



		a	b
+		c	d
=	x	y	z

Quantum computing: a new paradigm

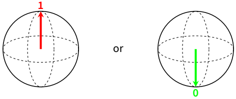
And with the quantum superposition?



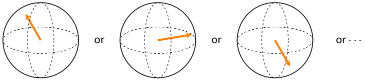
Quantum computing: a new paradigm

Bit vs Qubit (Bloch sphere)

A bit is **1 or 0** :



A **qubit** is both **1 and 0**...

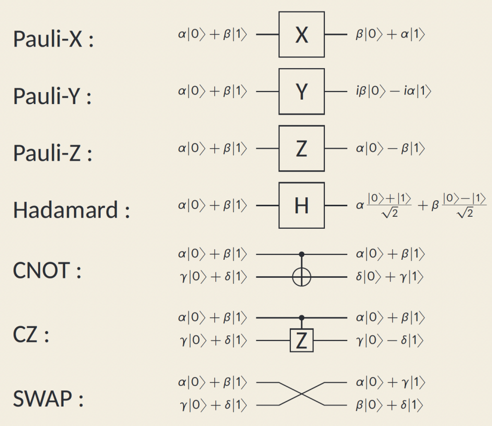


It is only when we **measure** that we can determine if the qubit is:

- in the north hemisphere: we then consider the value is **1**
- in the south hemisphere: we then consider the value is **0**

Quantum computing: a new paradigm

We consider new **quantum gates**:



... and so the algorithmic is different.

Quantum computing: a new paradigm

Let a function that takes a binary string of size n as input:

- a classical computer can test the 2^n inputs **one after another**;
- a **quantum computer** can test the 2^n inputs **in the same time** thanks to the principles of **state superposition** and **quantum entanglement**.

Quantum computing: a new paradigm

Let a function that takes a binary string of size n as input:

- a classical computer can test the 2^n inputs **one after another**;
- a **quantum computer** can test the 2^n inputs **in the same time** thanks to the principles of **state superposition** and **quantum entanglement**.

Attention

The outputs are also in a state of quantum superposition. When we *measure* the output, we get one output among all the possible outputs without knowing what is the corresponding input...

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **classical computer**, at each intersection, you can go right **or** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.

Quantum computing: a new paradigm

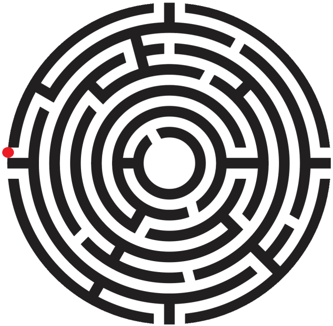
Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.

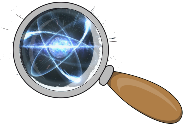
Quantum computing: a new paradigm

Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.



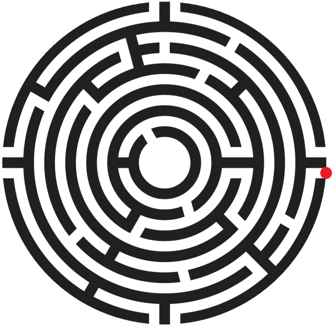


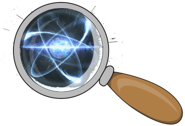
Quantum computing: a new paradigm

Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.

Quantum computing: a new paradigm

Labyrinth example: with a **quantum computer**, at each intersection, you can go right **and** go left.





Quantum computing: a new paradigm

About solving the labyrinth problem with a quantum computer:

1. If there is several exits, then the measure only allows to get one of them drawn randomly from all the possible exits.
2. The quantum computer find one exit of the labyrinth without holding back the path.

Quantum computing: a new paradigm

Some famous quantum algorithms:

1. **Grover 1996** : find a particular element in an unstructured set of 2^N elements with $O\left(\sqrt{2^N}\right) = O\left(2^{N/2}\right)$ operations
2. **Shor 1994** : factorize an integer N with only $O\left(\log(N)^3\right)$ operations. The Shor algorithm can also be used to solve the Discrete Logarithm problem over $\mathbb{Z}_n$ or on an Elliptic Curve.

Quantum computing: a new paradigm

The problem of decoherence

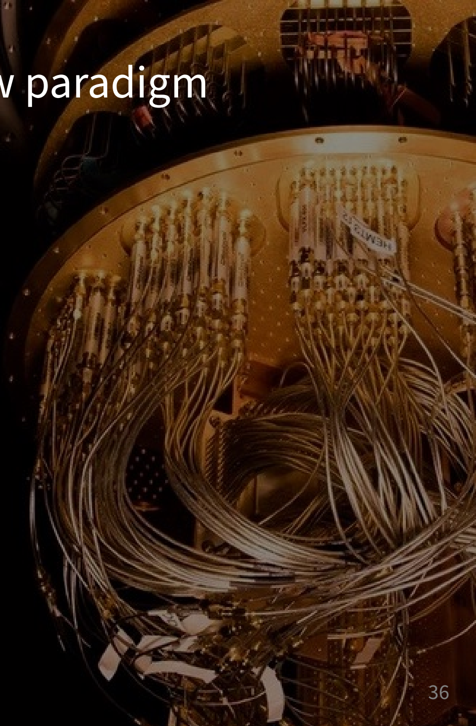
The larger the number of qubits, the more difficult it is to maintain them in a state of entanglement and quantum superposition for long enough.

- protect the computing environment: cool to **absolute zero**, ...
- use **quantum error correcting codes**: the *surface codes* require a large number of physical qubits per logical qubit.

Quantum computing: a new paradigm

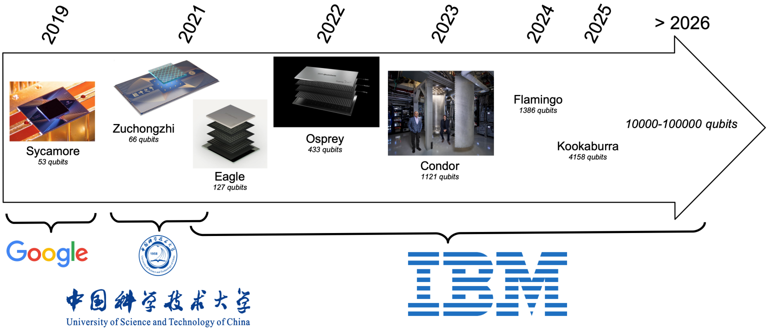
What technical solutions?

- The superconducting qubit
- The silicon qubit
- The trapped ion qubit
- The photonic qubit



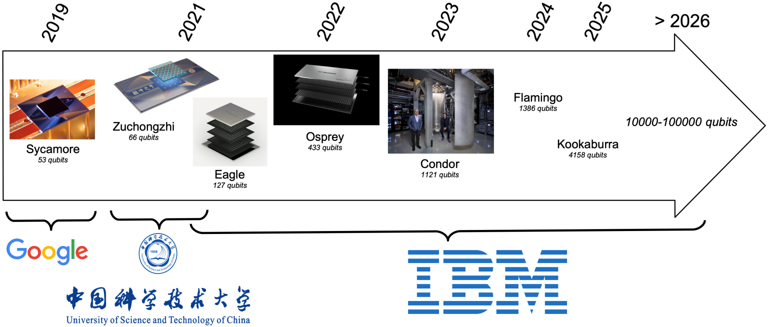
Quantum computing: a new paradigm

Where we are?



Quantum computing: a new paradigm

Where we are?

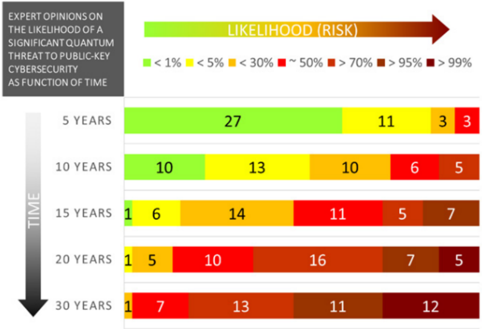


logial qubits / physical qubits

In the figure above, we count **physical qubits**. It is necessary to combine many physical (error-prone) qubits to obtain a logical (error-free) qubit. We are talking about quantum corrector codes.

Quantum computing: a new paradigm

Where we are?



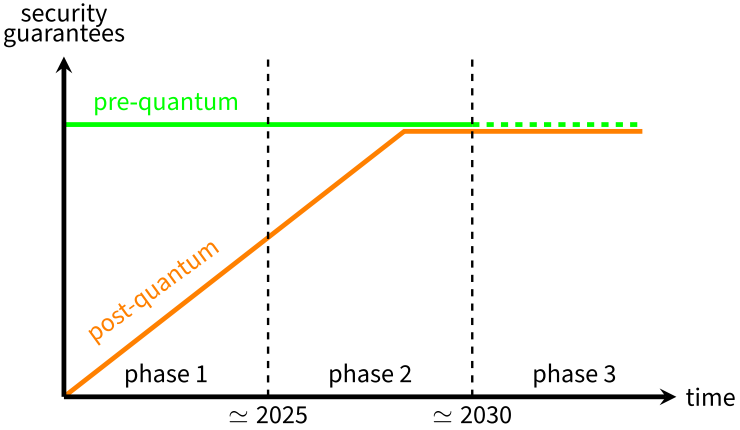
Numbers reflect how many experts (out of 44) assigned a certain probability range.

Forward secrecy

Even if the number of logical qubits is insufficient today to break RSA, we must protect ourselves now to guarantee the **forward secrecy**.

Quantum computing: a new paradigm

The ANSSI recommendations



Quantum transition plan of the ANSSI presented at PQCrypto 2021.

Quantum computing: a new paradigm

The NIST competitions

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

https://csrc.nist.gov/projects/post-quantum-cryptography