XIPHERA

PEACE OF MIND IN A DANGEROUS WORLD

Wednesday, June 8, 2022
16:00 EET

The Role of Elliptic Curve Cryptography in the Post-Quantum Era

Webinar series

Cryptography under the hood

Speaker

Kimmo Järvinen

CTO & Co-founder,
Xiphera

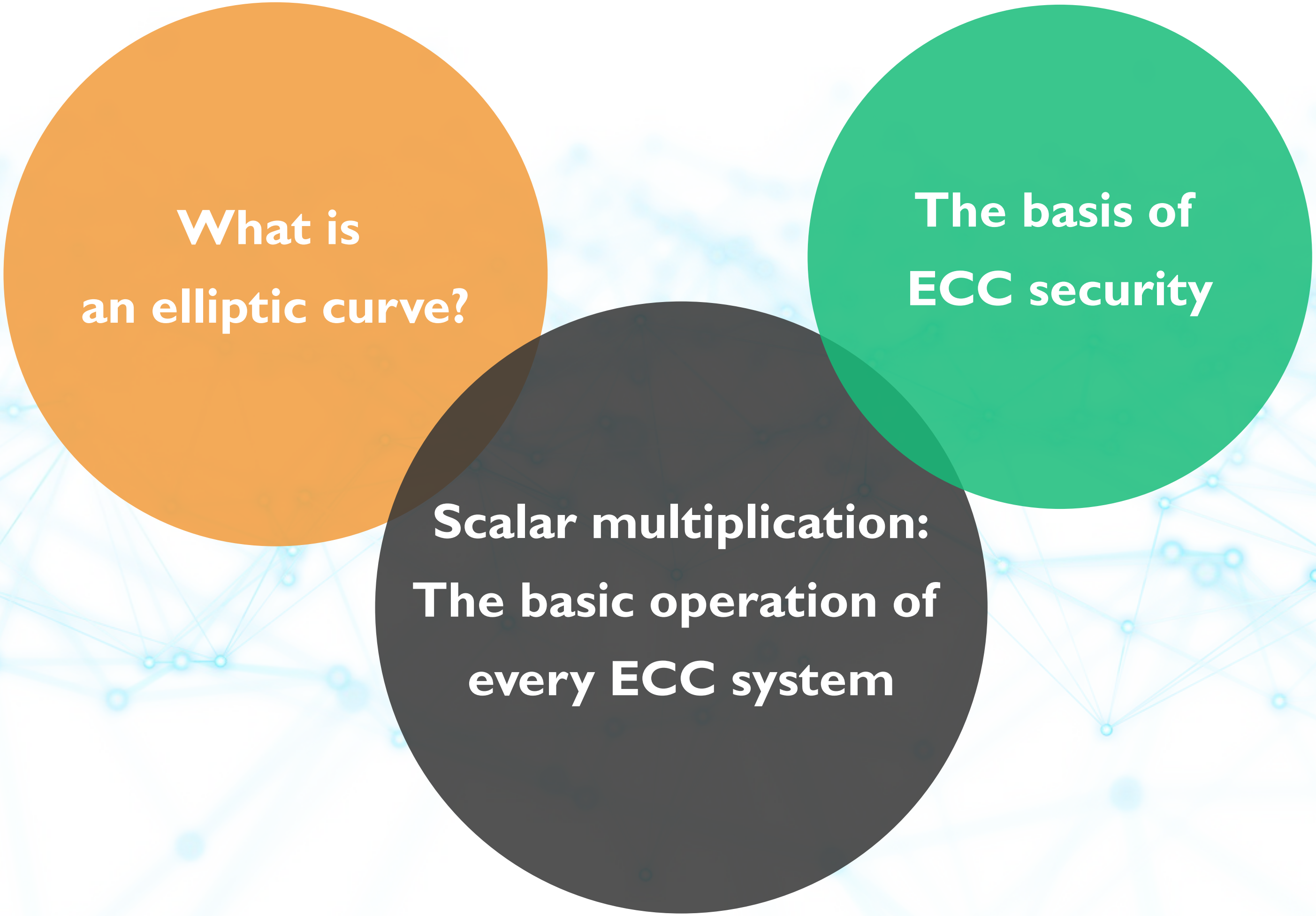
Agenda

I. Introduction to ECC

II. Implementation pitfalls

III. Secure ECC implementations

IV. ECC in the PQ era



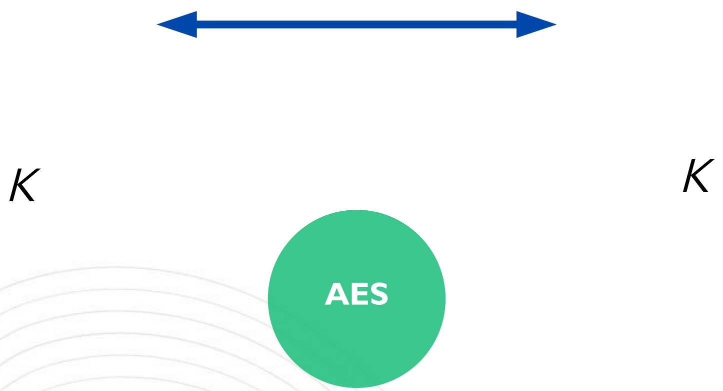
What is an elliptic curve?

The basis of ECC security

Scalar multiplication: The basic operation of every ECC system

Symmetric vs. Asymmetric

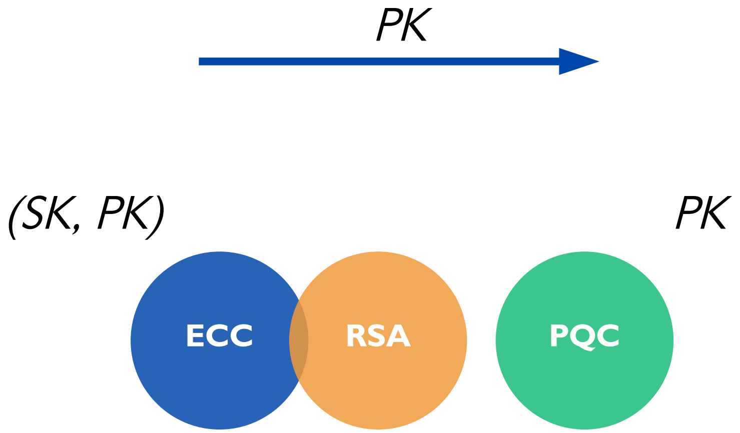
Symmetric



Shared key *K*

- Must be secret

Asymmetric

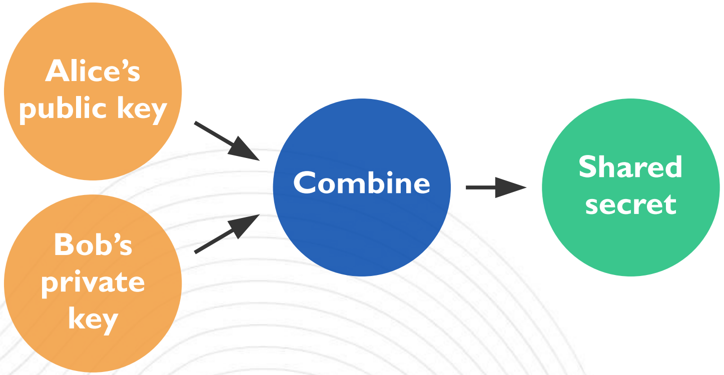


Key-pair

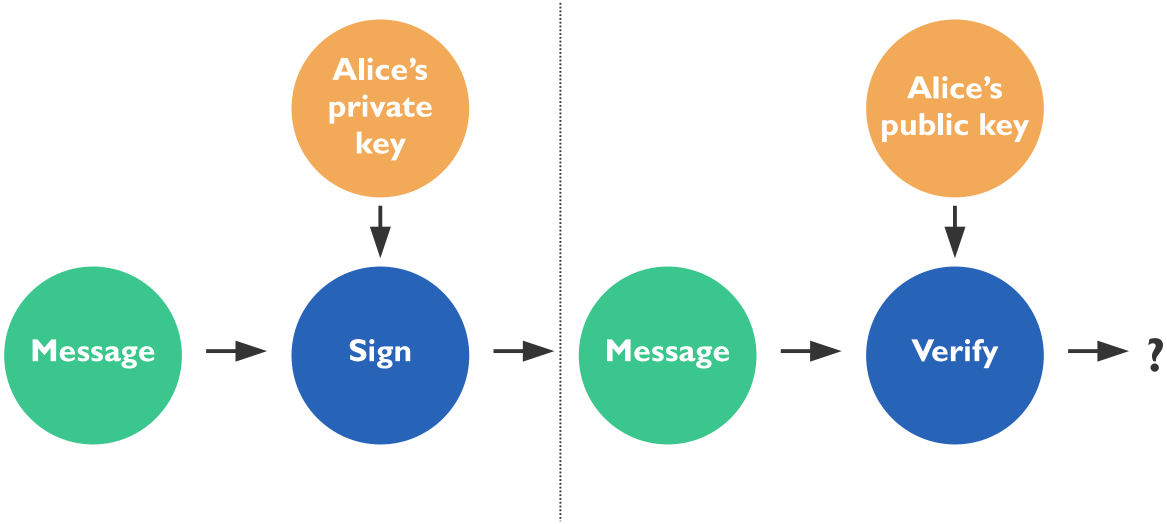
- Private key (SK) → Public key (PK)
- Public key (SK) ↛ Private key (PK)

Asymmetric Cryptography

Key exchange

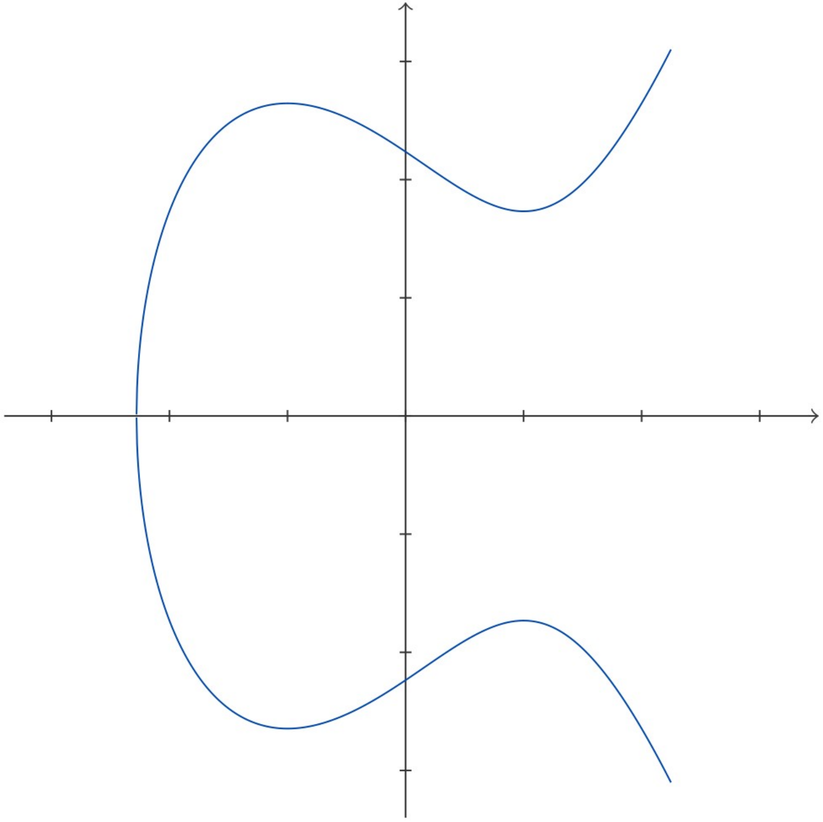


Digital signatures



Elliptic Curve Cryptography

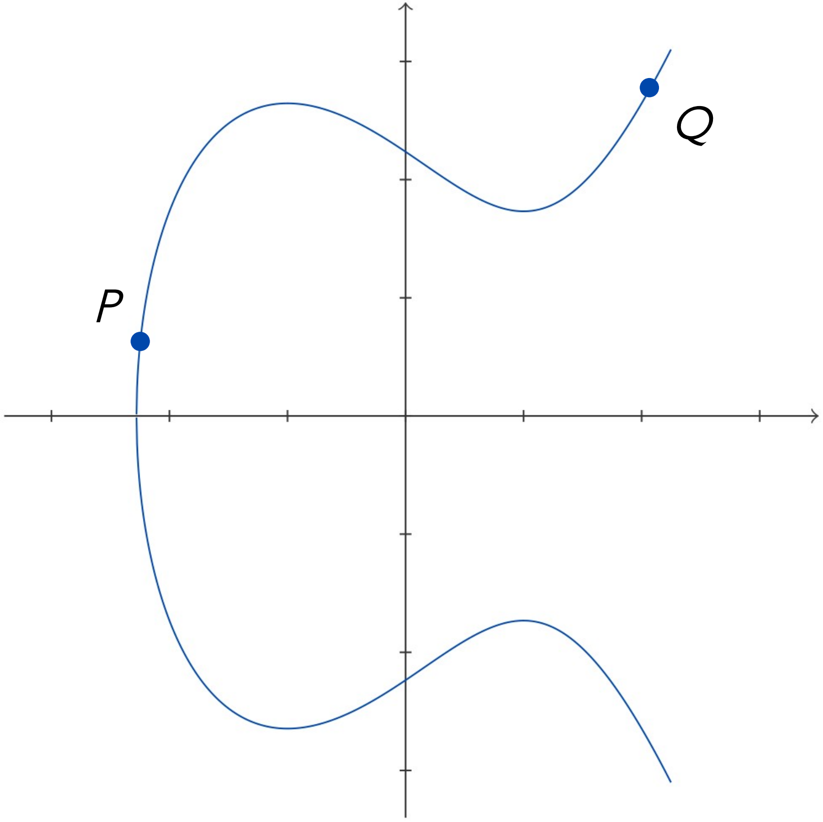
- **Since the mid-1980s:** Miller and Koblitz
- **Elliptic Curve Cryptosystems:** The most widely used asymmetric cryptography algorithms in today's systems
- **Key Exchange and Digital Signatures:** ECDH(E), ECDSA (X25519, EdDSA)



$E : y^2 = x^3 - 3x + 5$

Elliptic Curve Cryptography

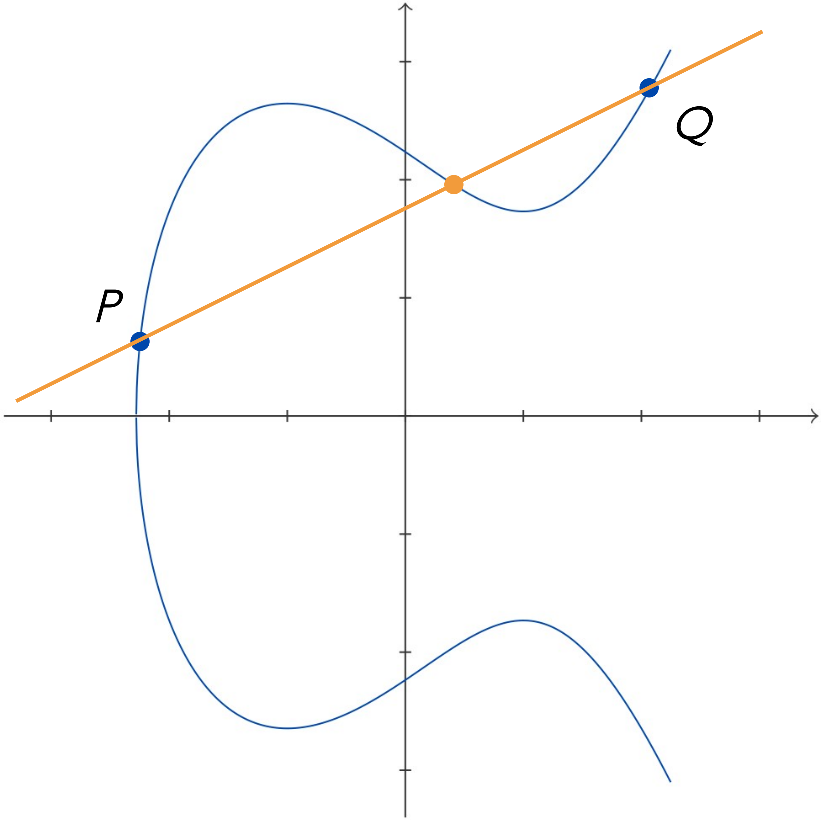
- **Since the mid-1980s:** Miller and Koblitz
- **Elliptic Curve Cryptosystems:** The most widely used asymmetric cryptography algorithms in today's systems
- **Key Exchange and Digital Signatures:** ECDH(E), ECDSA (X25519, EdDSA)
- **Point addition:** Add two points on a curve by drawing a line that intersects the points and a third point. Reflect the third point over the x-axis to get the result.



$E : y^2 = x^3 - 3x + 5$

Elliptic Curve Cryptography

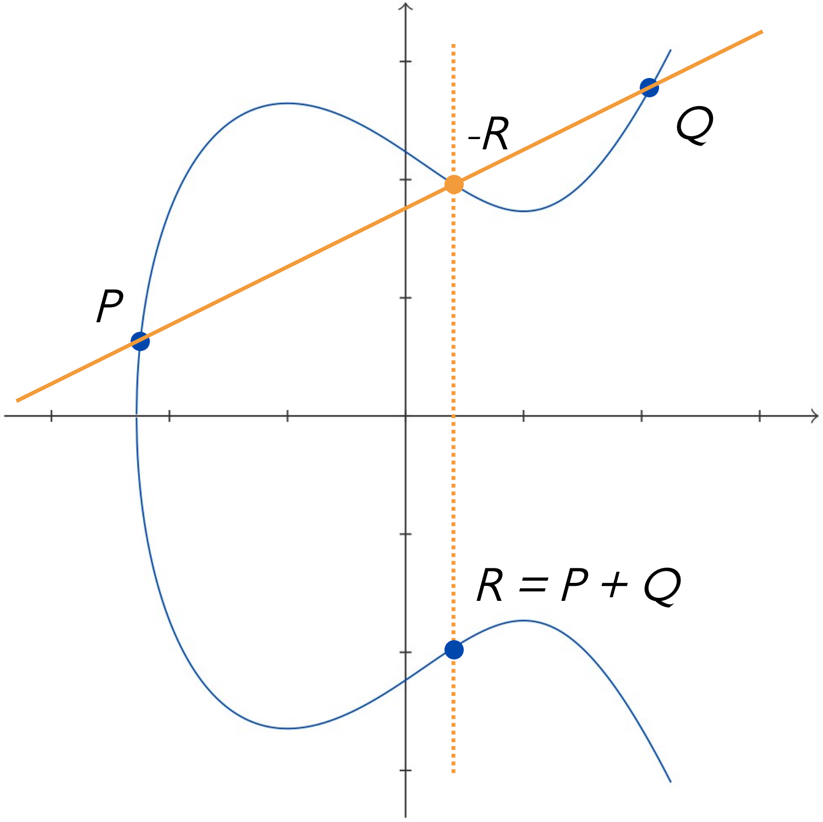
- **Since the mid-1980s:** Miller and Koblitz
- **Elliptic Curve Cryptosystems:** The most widely used asymmetric cryptography algorithms in today's systems
- **Key Exchange and Digital Signatures:** ECDH(E), ECDSA (X25519, EdDSA)
- **Point addition:** Add two points on a curve by drawing a line that intersects the points and a third point. Reflect the third point over the x-axis to get the result.



$E : y^2 = x^3 - 3x + 5$

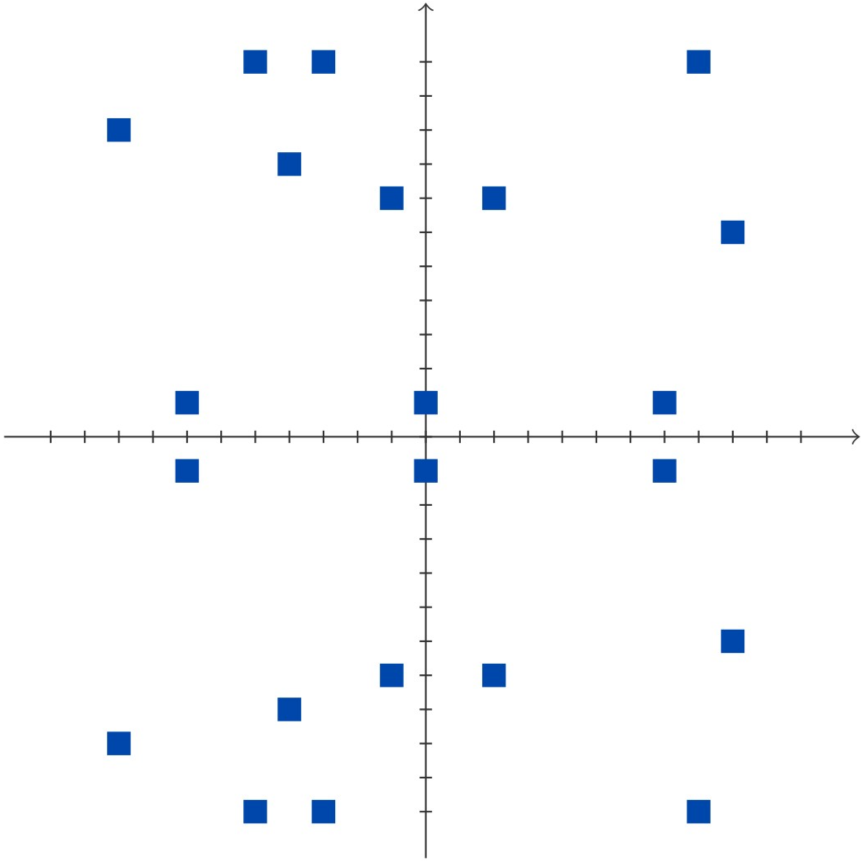
Elliptic Curve Cryptography

- **Since the mid-1980s:** Miller and Koblitz
- **Elliptic Curve Cryptosystems:** The most widely used asymmetric cryptography algorithms in today's systems
- **Key Exchange and Digital Signatures:** ECDH(E), ECDSA (X25519, EdDSA)
- **Point addition:** Add two points on a curve by drawing a line that intersects the points and a third point. Reflect the third point over the x-axis to get the result.



$E : y^2 = x^3 - 3x + 5$

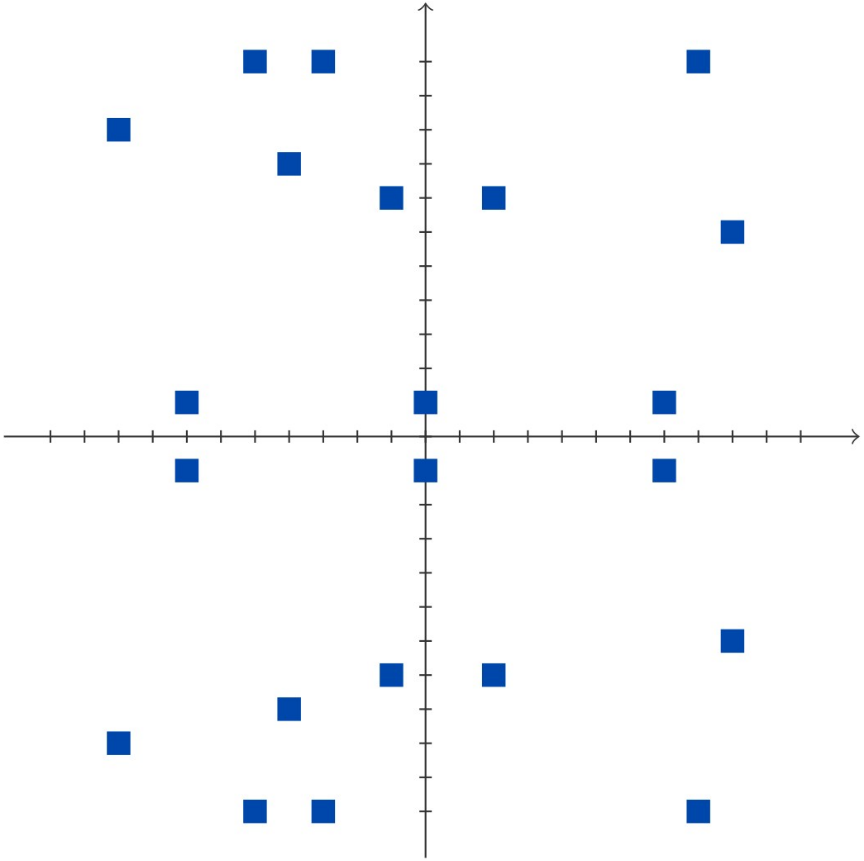
Elliptic Curve



$E : y^2 = x^3 - 3x + 1 \pmod{23}$

Elliptic Curve

This toy example:
23 points
incl. point at infinity
(5-bit prime)

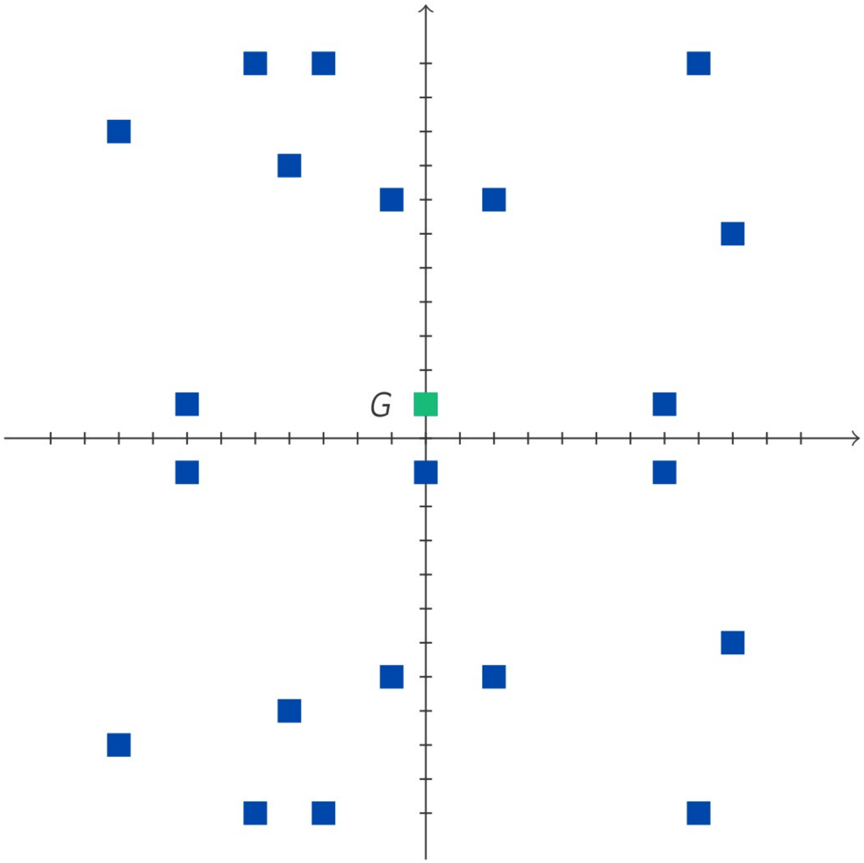


$E : y^2 = x^3 - 3x + 1 \pmod{23}$

In practice:
NIST P-384 has
394020061963944…
921227904010014…
613805079739270…
654466679469052…
962765939911326…
569398956308152…
949135544336539…
2643 points
incl. point at infinity
(384-bit prime)

Elliptic Curve

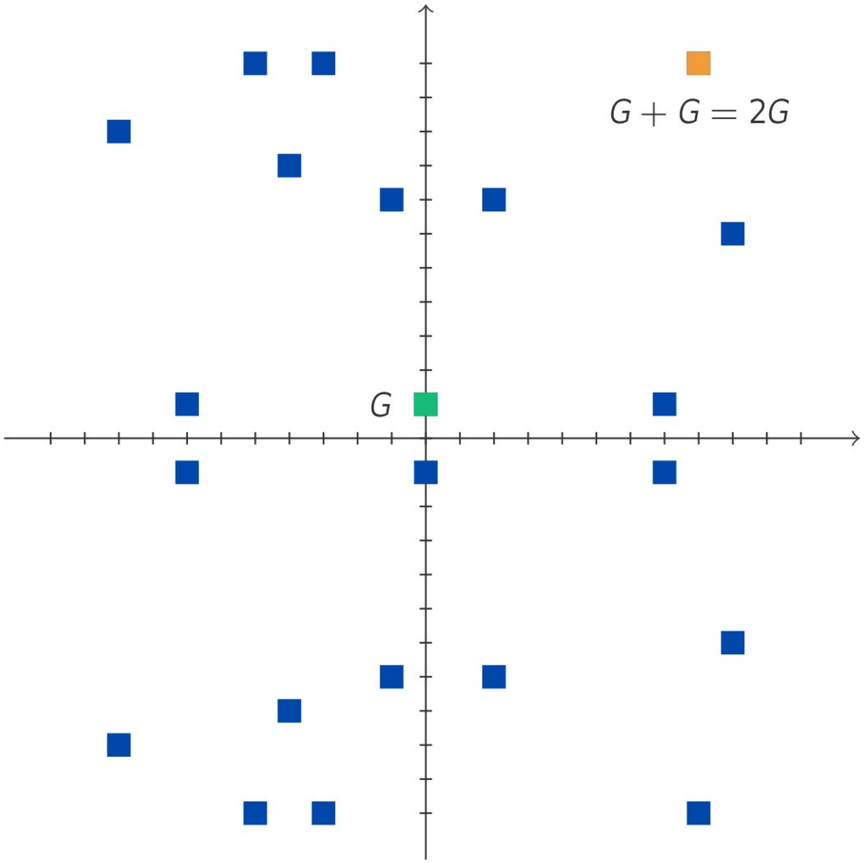
This toy example:
G = (0,1)



$E : y^2 = x^3 - 3x + 1 \pmod{23}$

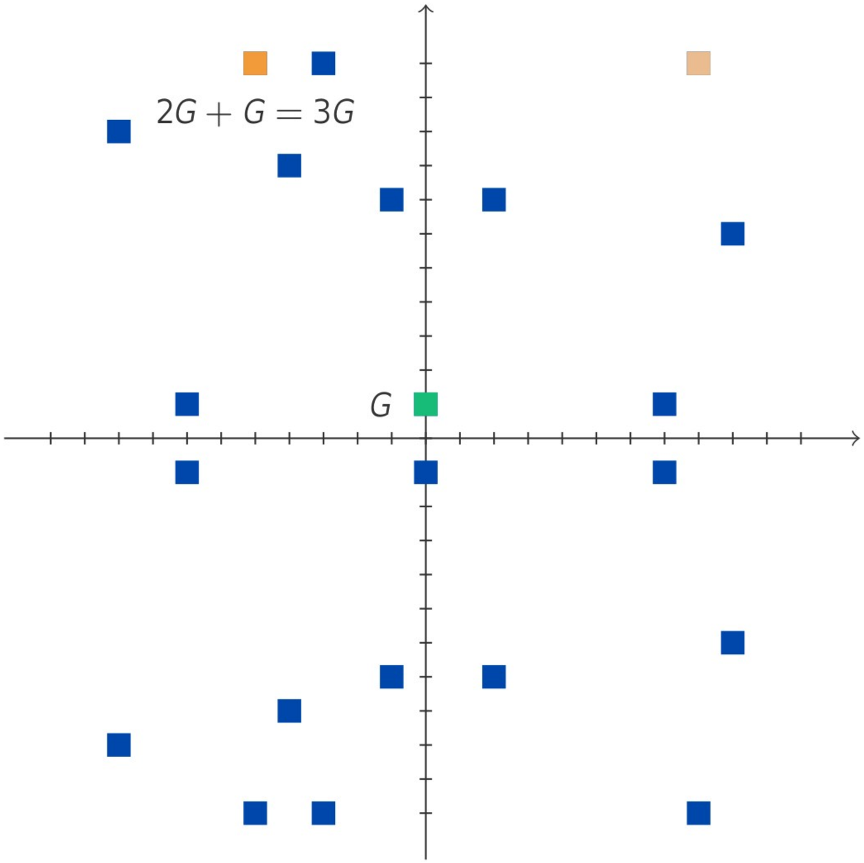
In practice:
NIST P-384 base point:
*G = (262470350957996892686…
2315674456698189185292…
3491109213387815615900…
9255188547380500890223…
8805397571978665087247…
6732087,
8325710961489029985546…
7512895201081792878530…
4886131559470920590248…
0503199884419224438643…
7603929473330780865116…
27871)*

Elliptic Curve



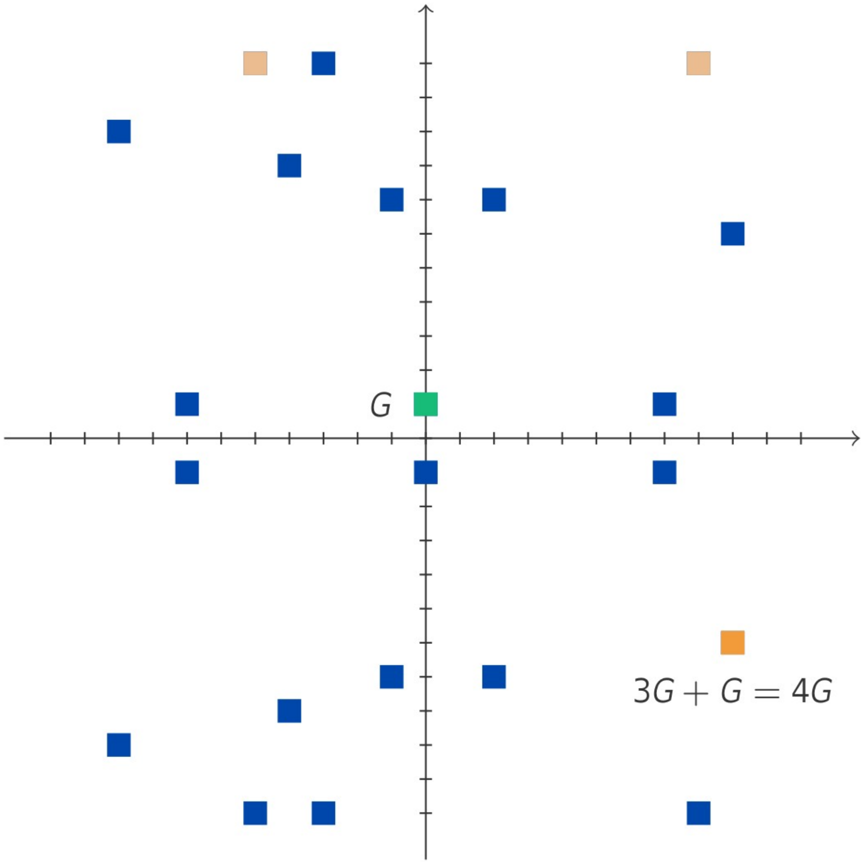
$E : y^2 = x^3 - 3x + 1 \pmod{23}$

Elliptic Curve



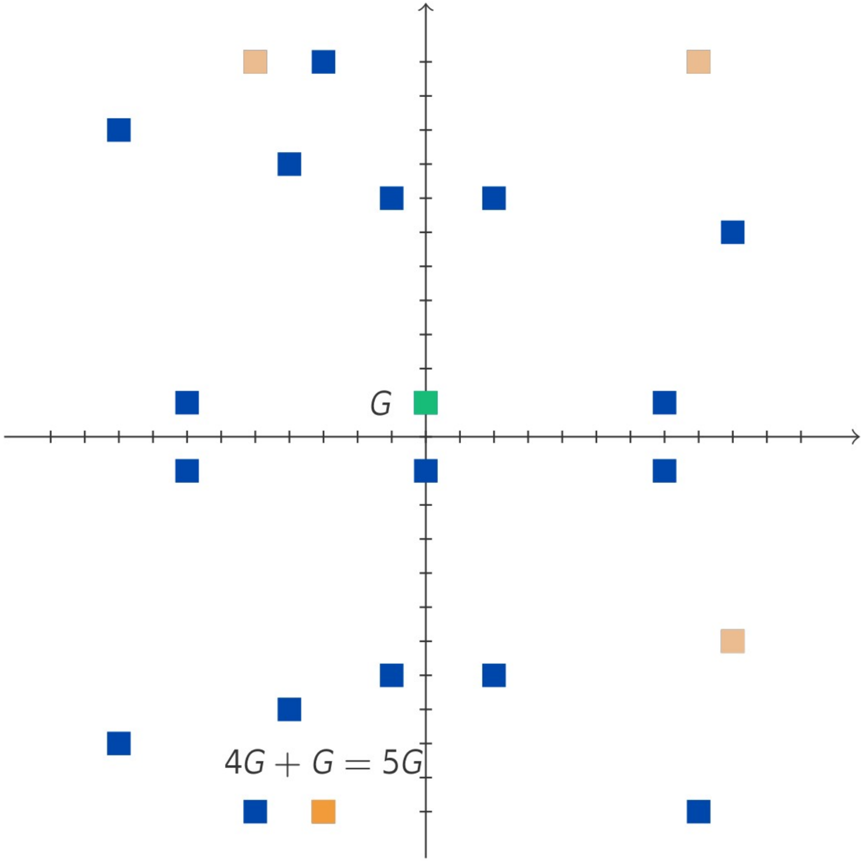
$$E : y^2 = x^3 - 3x + 1 \pmod{23}$$

Elliptic Curve



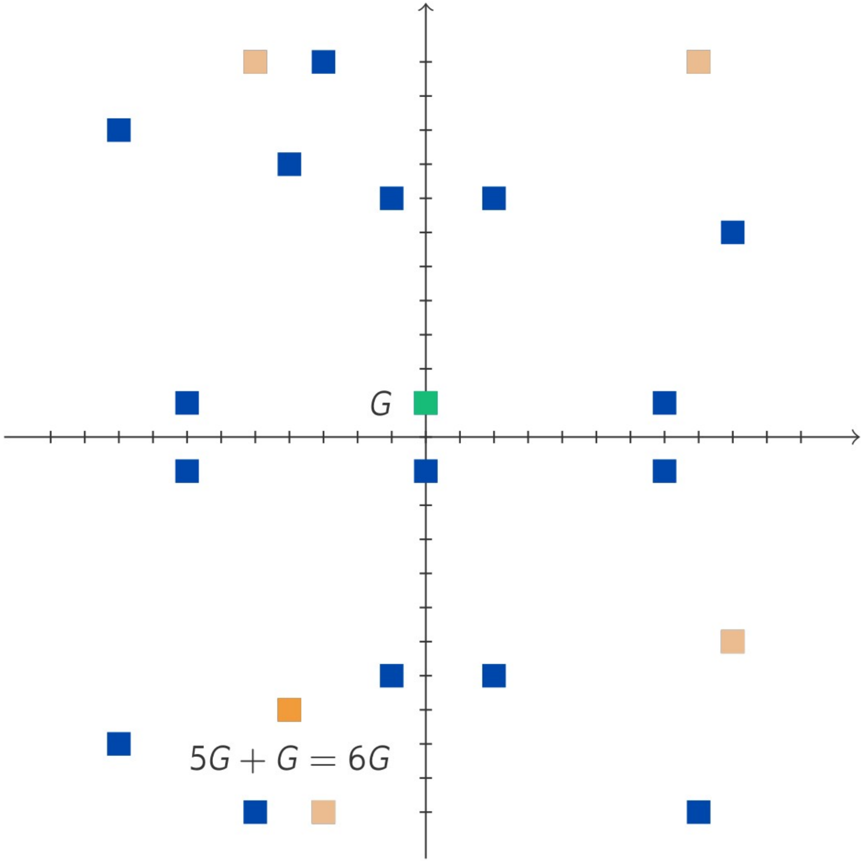
$$E : y^2 = x^3 - 3x + 1 \pmod{23}$$

Elliptic Curve



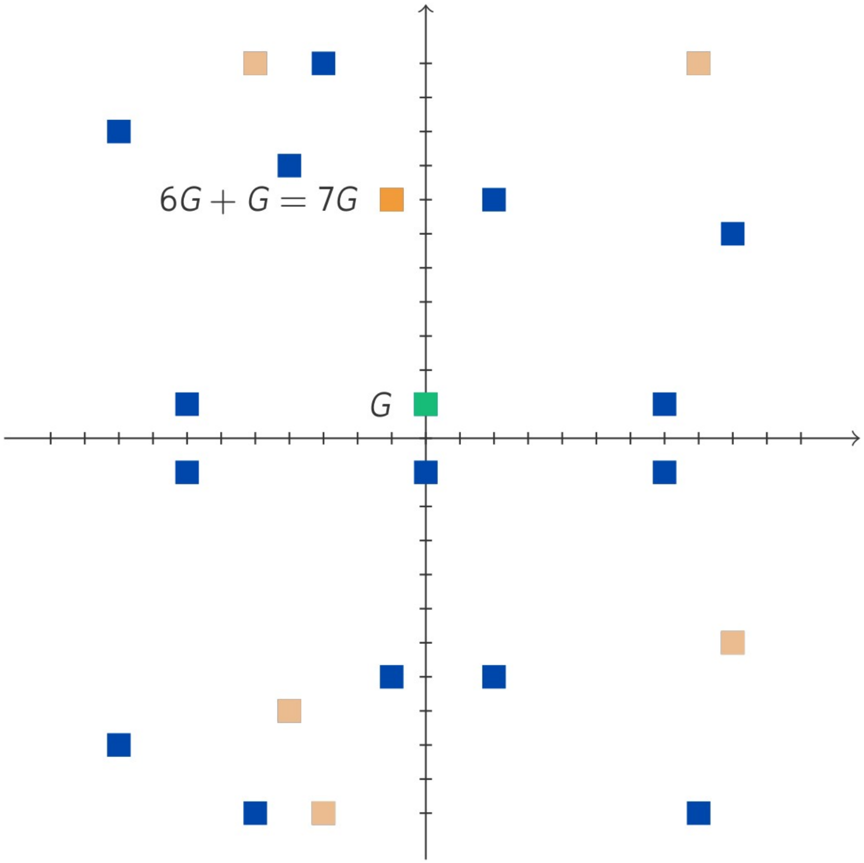
$E : y^2 = x^3 - 3x + 1 \pmod{23}$

Elliptic Curve



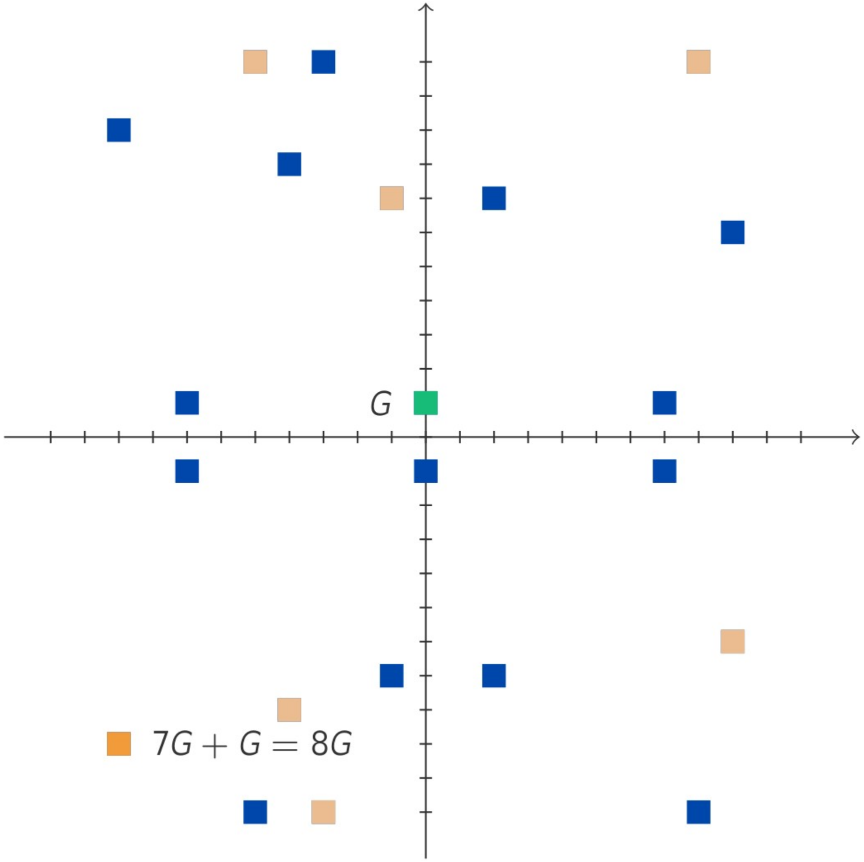
$$E : y^2 = x^3 - 3x + 1 \pmod{23}$$

Elliptic Curve



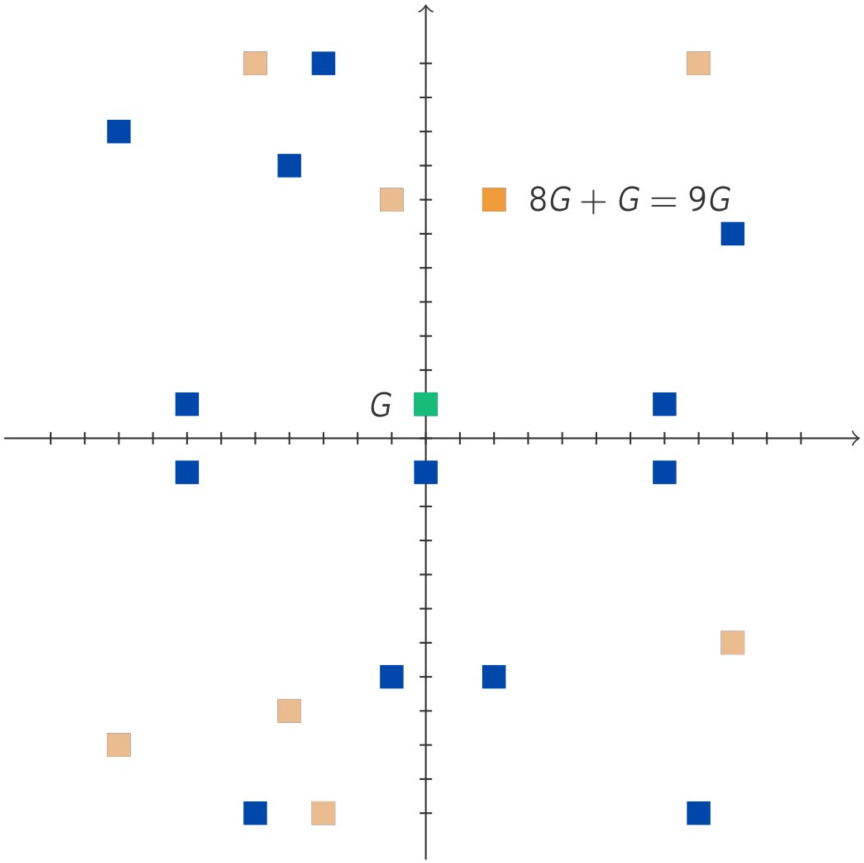
$E : y^2 = x^3 - 3x + 1 \pmod{23}$

Elliptic Curve



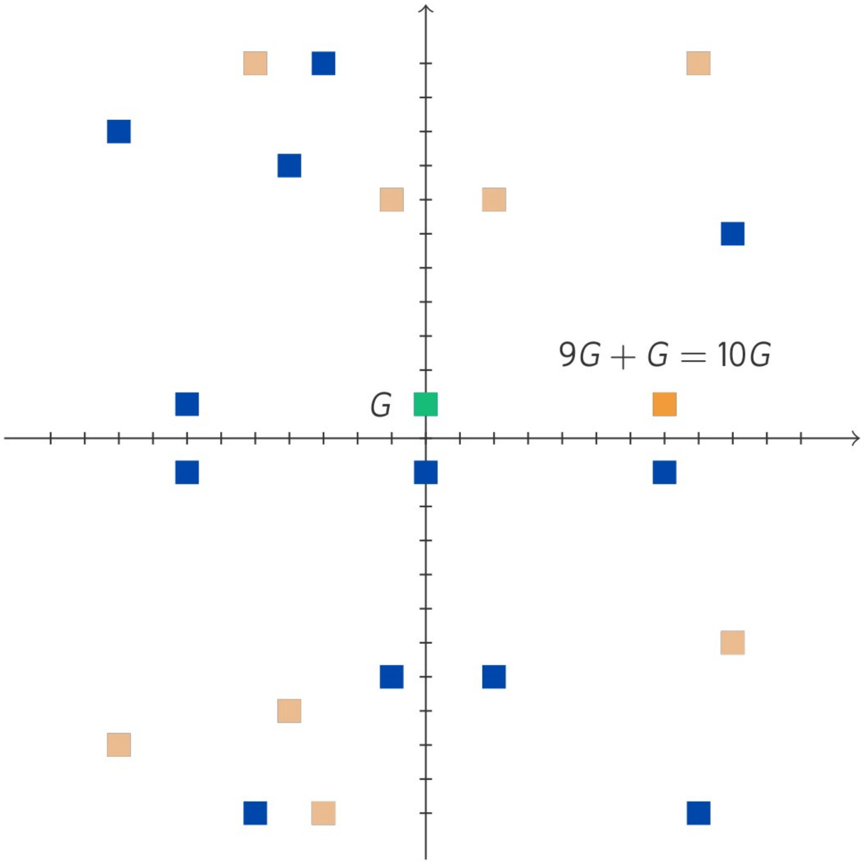
$$E : y^2 = x^3 - 3x + 1 \pmod{23}$$

Elliptic Curve



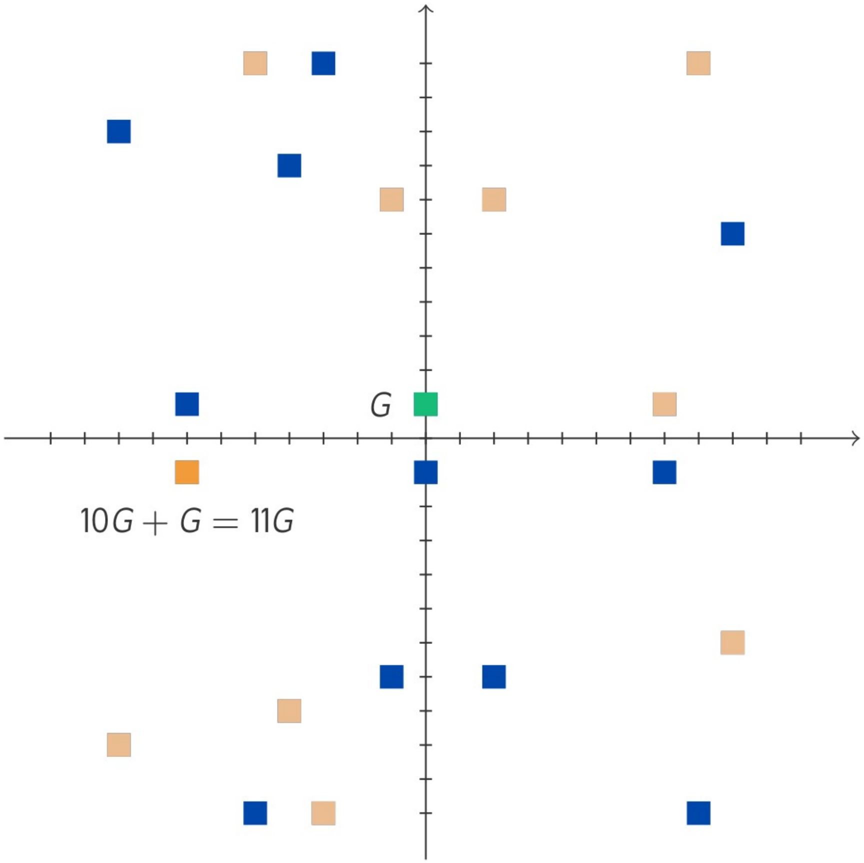
$E : y^2 = x^3 - 3x + 1 \pmod{23}$

Elliptic Curve



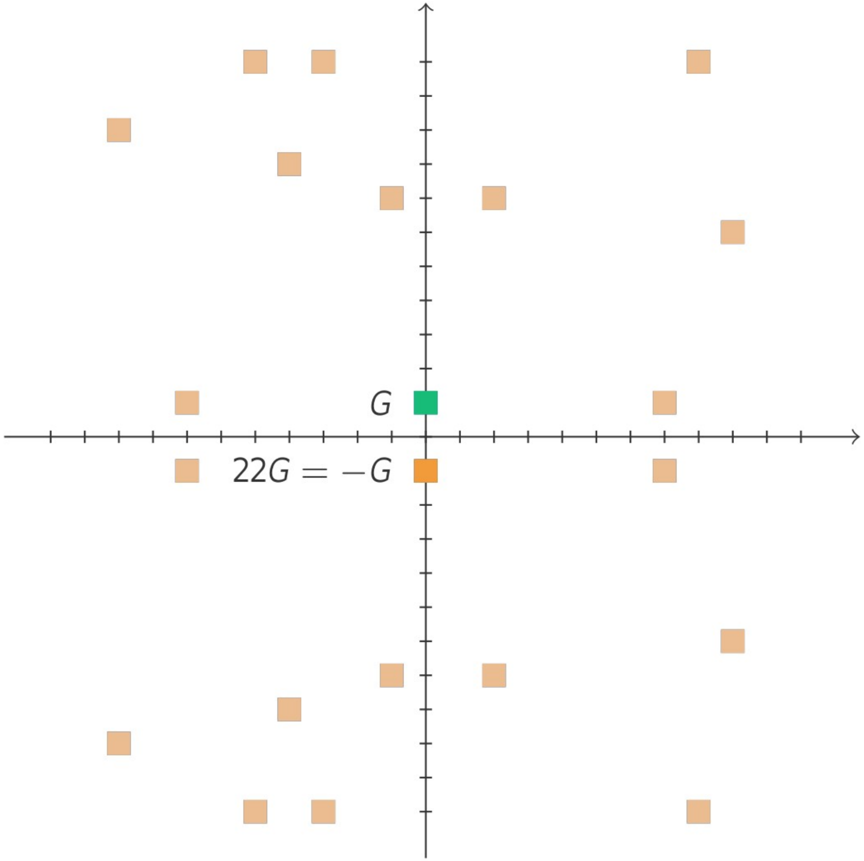
$$E : y^2 = x^3 - 3x + 1 \pmod{23}$$

Elliptic Curve



$$E : y^2 = x^3 - 3x + 1 \pmod{23}$$

Elliptic Curve



$E : y^2 = x^3 - 3x + 1 \pmod{23}$

Scalar Multiplication

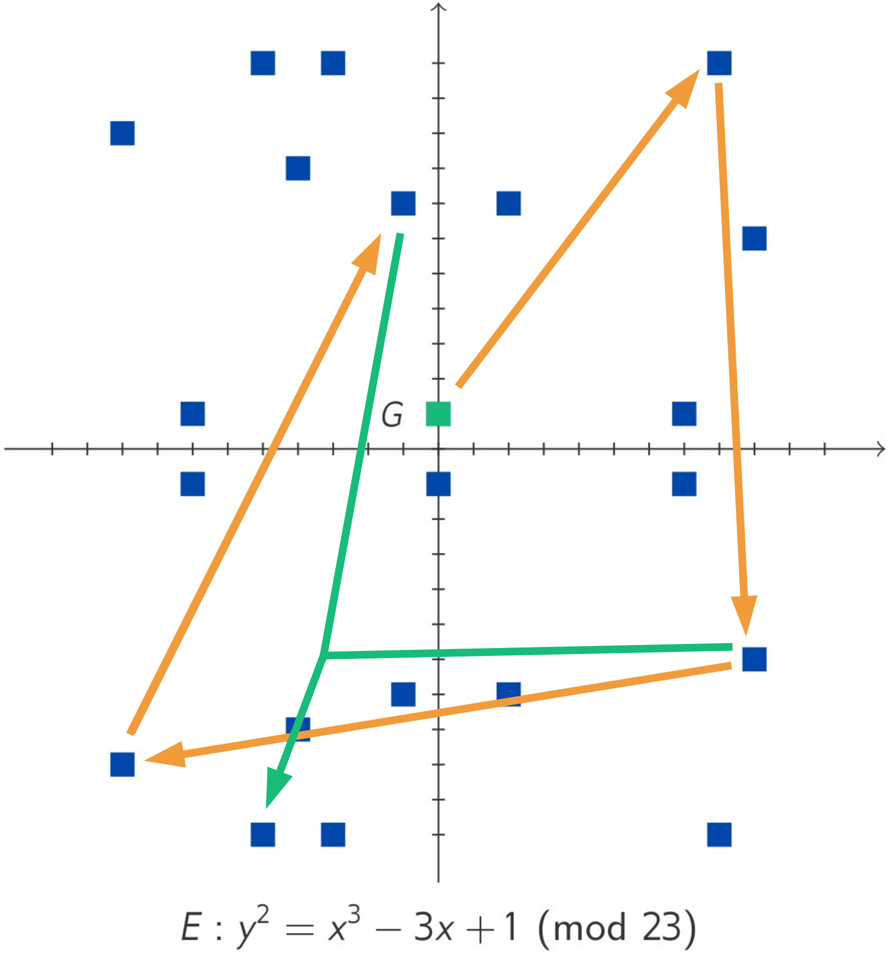
- The most important operation in ECC

$$Q = kP$$

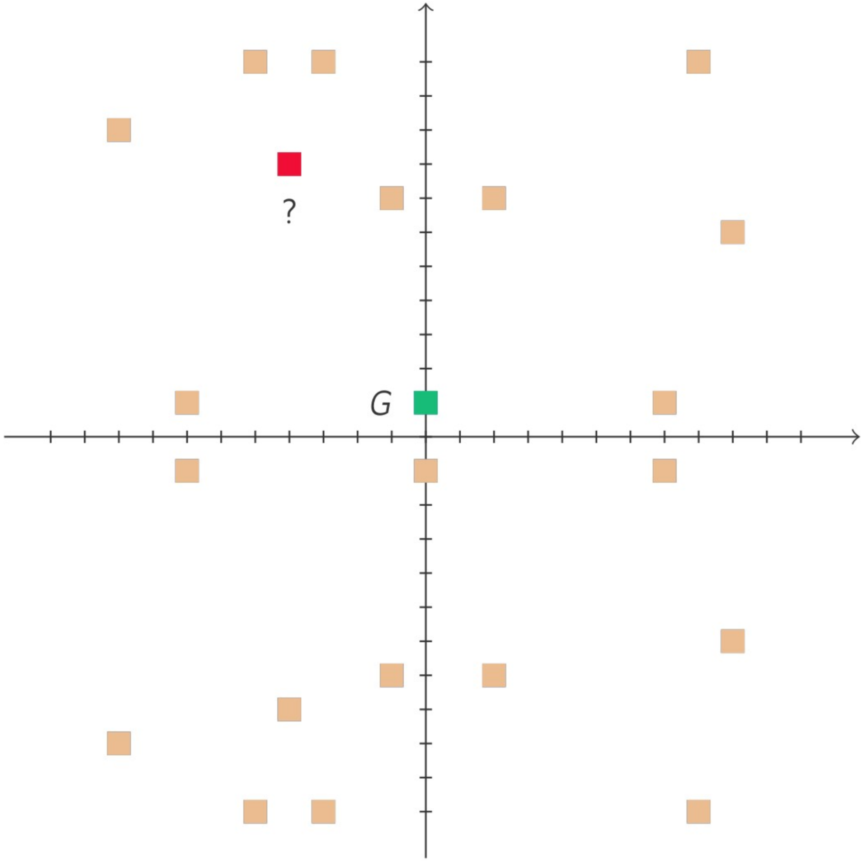
  - Q and P are points
  - k is an integer and must typically be secret (for example, the private key)

Scalar Multiplication

- The most important operation in ECC
  $$Q = kP$$
  - Q and P are points
  - k is an integer and must typically be secret (for example, the private key)
- Fast algorithms are available
  - **For example:** Compute $20G$ by computing $G+G$, $2G+2G$, $4G+4G$, $8G+8G$ and $16G+4G$ (only 5 additions)



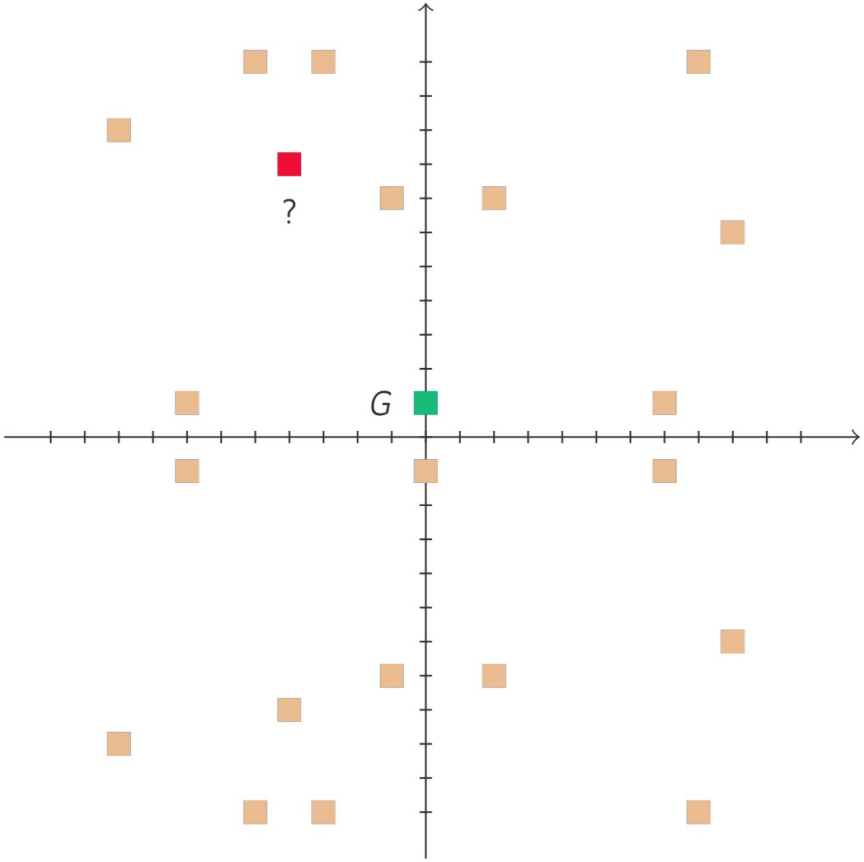
Discrete Logarithm Problem



$E : y^2 = x^3 - 3x + 1 \pmod{23}$

Discrete Logarithm Problem

This toy example:
The answer is *17 G*

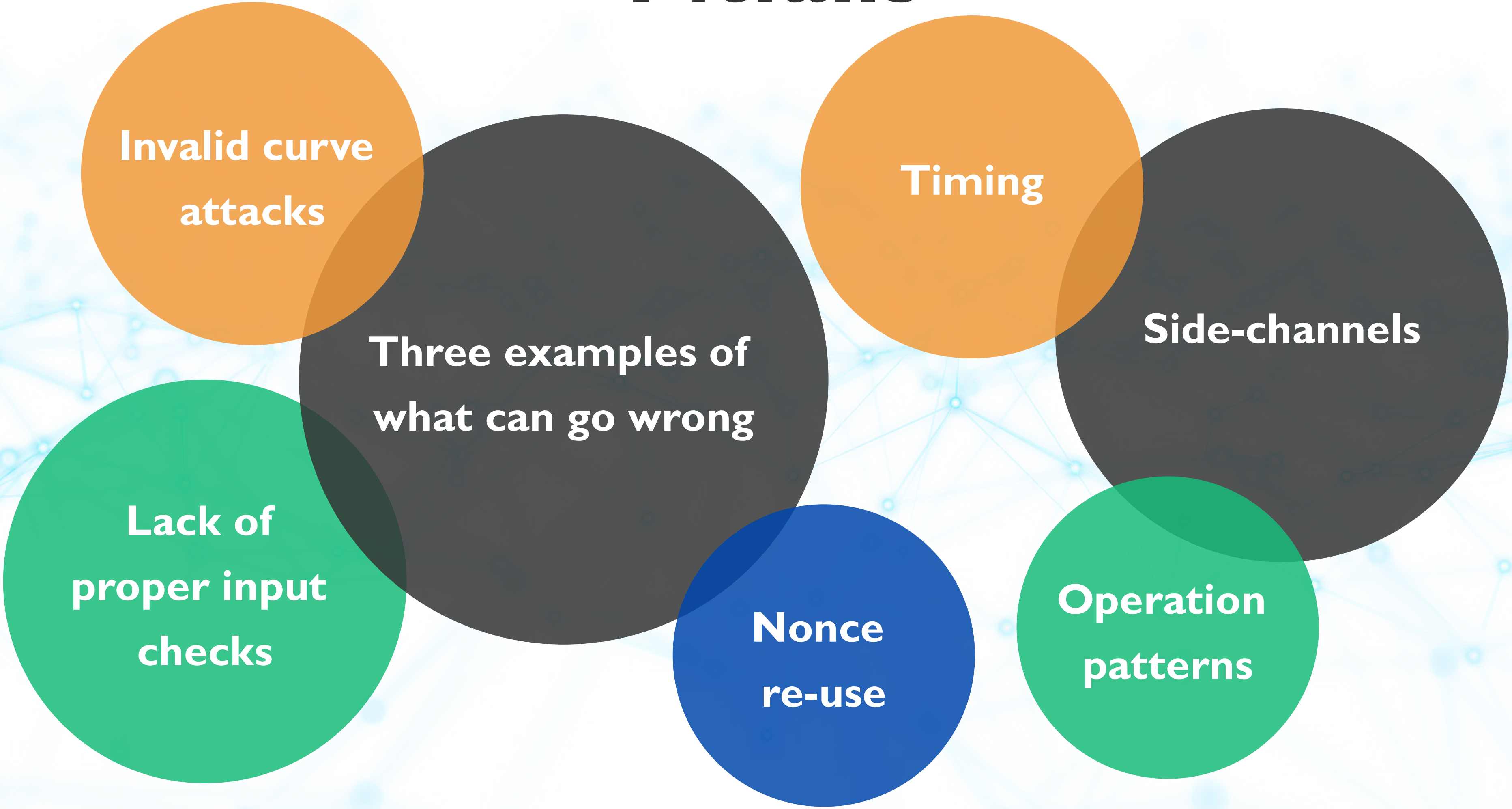


$E : y^2 = x^3 - 3x + 1 \pmod{23}$

In practice:
Impossible to solve

Security levels:
P-256 : 128-bit
P-384 : 192-bit
P-521 : 256-bit
Curve25519 : 128-bit
Curve448 : 224-bit

Pitfalls



Three examples of what can go wrong

Invalid curve attacks

Lack of proper input checks

Nonce re-use

Timing

Side-channels

Operation patterns

ECDSA: Nonce Re-use

- The hash *h* of a message is signed with signing key *d* as follows:
  - The nonce *k* is a cryptographically random integer in interval *[1,n-1]*

$$r \leftarrow [kG]_x$$

$$s \leftarrow \frac{h + r \cdot d}{k} \bmod n$$

ECDSA: Nonce Re-use

- The hash *h* of a message is signed with signing key *d* as follows:
  - The nonce *k* is a cryptographically random integer in interval *[1,n-1]*

$$r \leftarrow [kG]_x$$

$$s \leftarrow \frac{h + r \cdot d}{k} \bmod n$$

$$s_1 \leftarrow \frac{h_1 + r \cdot d}{k} \bmod n$$

$$s_2 \leftarrow \frac{h_2 + r \cdot d}{k} \bmod n$$

ECDSA: Nonce Re-use

- The hash *h* of a message is signed with signing key *d* as follows:
  - The nonce *k* is a cryptographically random integer in interval *[1,n-1]*

$$r \leftarrow [kG]_x$$

$$s \leftarrow \frac{h + r \cdot d}{k} \bmod n$$

$$s_1 \leftarrow \frac{h_1 + r \cdot d}{\cancel{k}} \bmod n$$

$$s_2 \leftarrow \frac{h_2 + r \cdot d}{\cancel{k}} \bmod n$$

$$\boxed{d = \frac{s_2 \cdot h_1 - s_1 \cdot h_2}{(s_1 - s_2)r} \bmod n}$$

ECDSA: Nonce Re-use

- The hash *h* of a message is signed with signing key *d* as follows:
  - The nonce *k* is a cryptographically random integer in interval *[1,n-1]*

$$r \leftarrow [kG]_x$$

$$s \leftarrow \frac{h + r \cdot d}{k} \bmod n$$

$$s_1 \leftarrow \frac{h_1 + r \cdot d}{\cancel{k}} \bmod n$$

$$s_2 \leftarrow \frac{h_2 + r \cdot d}{\cancel{k}} \bmod n$$

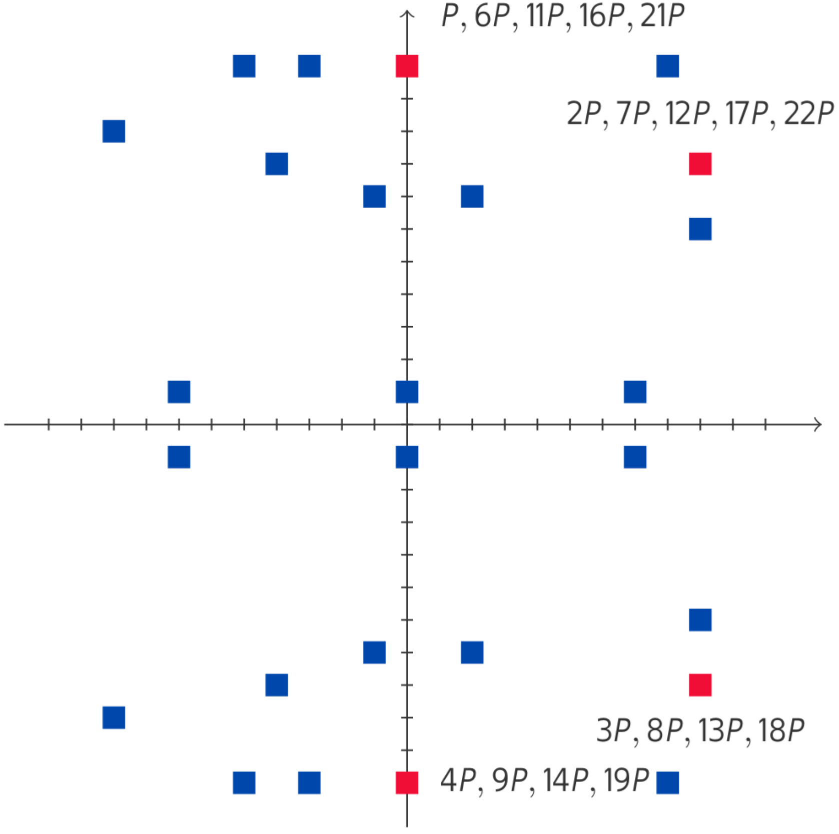
$$d = \frac{s_2 \cdot h_1 - s_1 \cdot h_2}{(s_1 - s_2)r} \bmod n$$

“Surely nobody does anything like that…”

- Sony PS3 was broken in 2010 (fixed *k*)
- Bitcoin hack in 2012 (bad RNG in Java SecureRandom in Android)

ECDH: Invalid Curve Attacks

- Often the other party selects the input point *P*
- An attacker sends *P* that is a point on *a weak curve instead of the correct curve* and gets information about the victim's private key
- **CVE-2015-2613:** Static private key of TLS-ECDH in a Bouncy Castle server after 3000 handshakes
- Check that the point is on the curve!



$E: y^2 = x^3 - 3x + 6 \pmod{23}$

ECDSA: Proper Input Checks

- ECDSA signature verification:
  1) Check that *r* and *s* are integers in the interval *[1,n-1]*
  2) Using signer's public key *P* compute

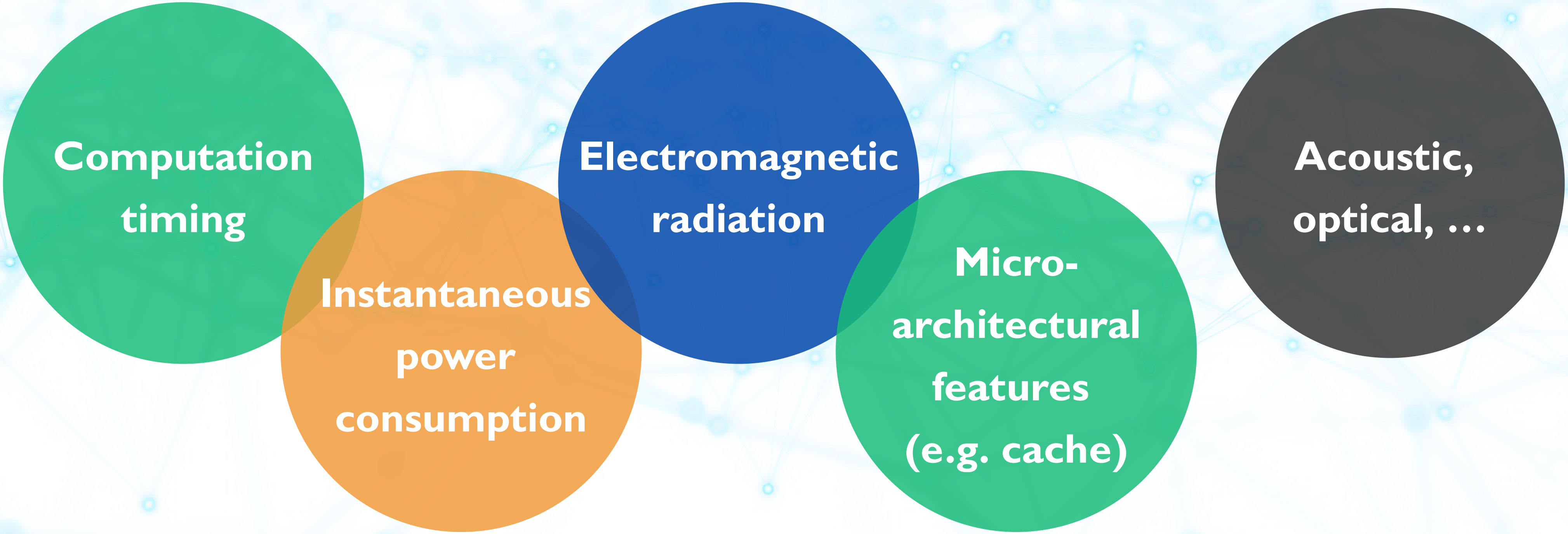
$$R \leftarrow \left(\frac{h}{s}\right) G + \left(\frac{r}{s}\right) P$$

  3) Accept signature if and only if $r = [R]_x$

- **CVE-2022-21449 (Apr. 19, 2022):**
  - Java ECDSA skips Step 1) and accepts *(r,s) = (0,0)* on any message

Side Channels

Side channel attacks use information *leaked by the implementation* of a cryptosystem to break the security.



Computation timing

Instantaneous power consumption

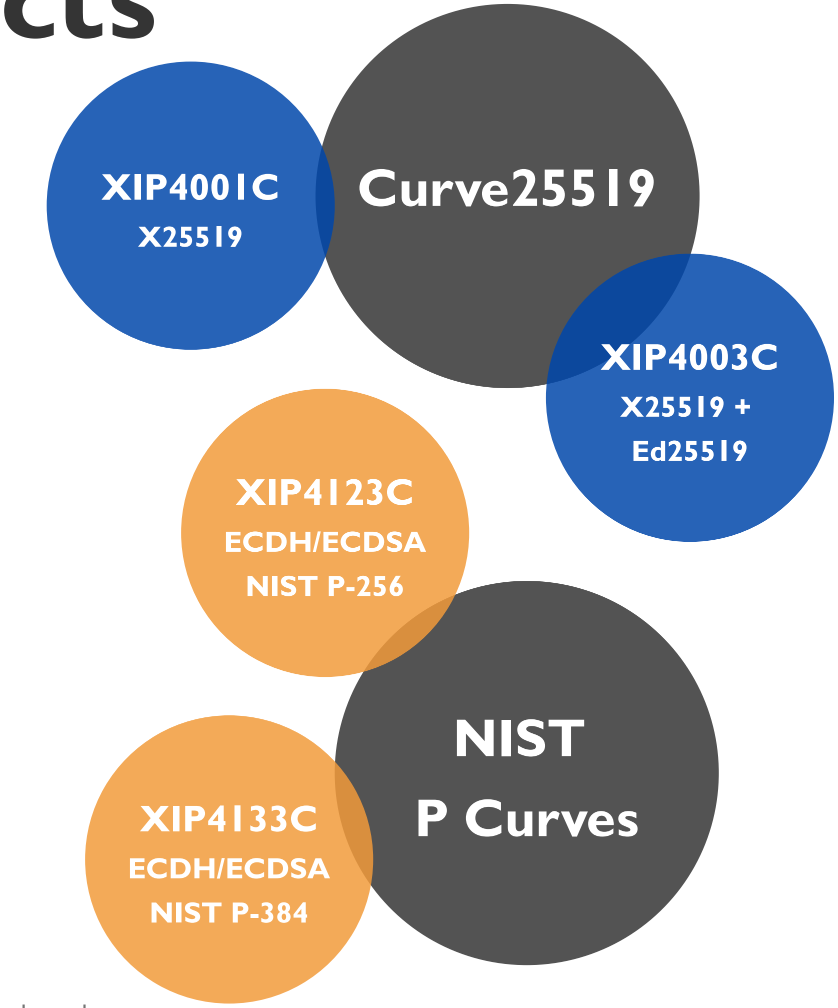
Electromagnetic radiation

Micro-architectural features (e.g. cache)

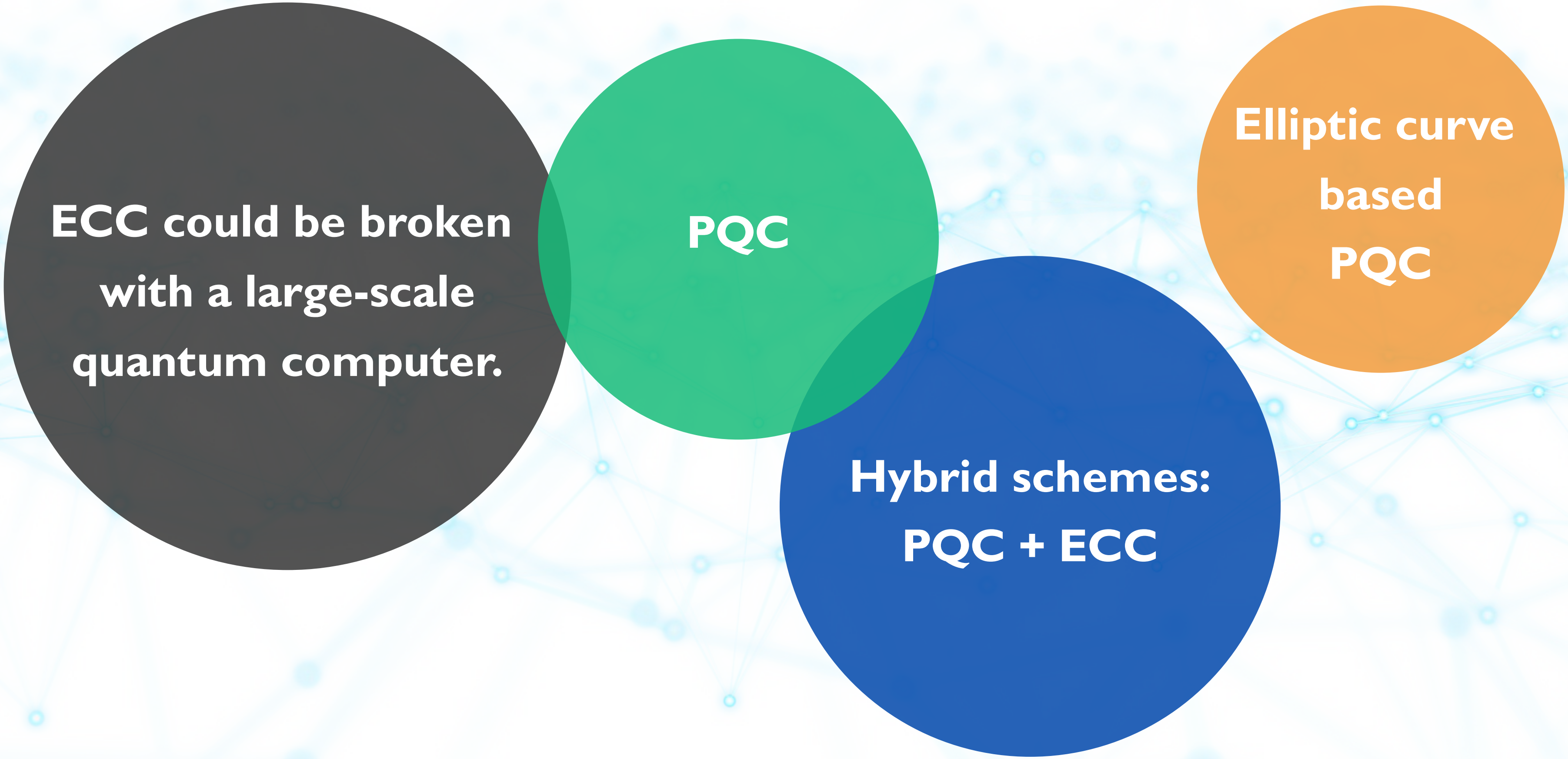
Acoustic, optical, …

Our Products

- Xiphera's ECC portfolio
  - Compact IP cores
  - **X25519/Ed25519**
  - **New IP cores:**
    - ECDH/ECDSA on P-256/384
- Secure designs
  - All relevant checks
  - Constant time / operation patterns



The Future of ECC



ECC could be broken with a large-scale quantum computer.

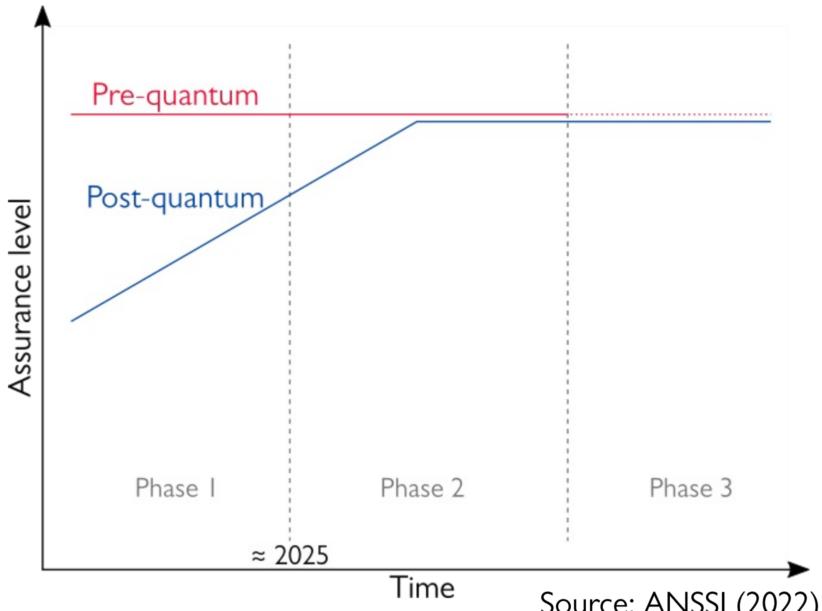
PQC

Hybrid schemes: PQC + ECC

Elliptic curve based PQC

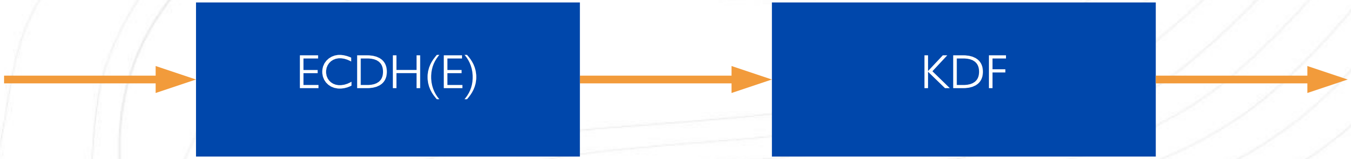
Why Hybrid Systems?

- We cannot fully trust that the new PQC schemes are secure
  - **Example:** NIST finalist Rainbow was broken!
- Many recommend using a hybrid system
  - ANSSI (France) recommends it at least until 2030
- ECC will not go away for a long time!

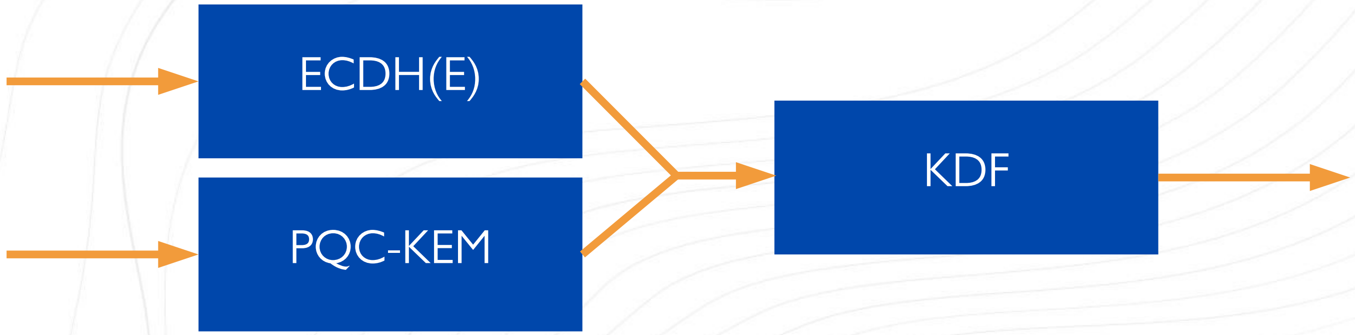


Source: ANSSI (2022)

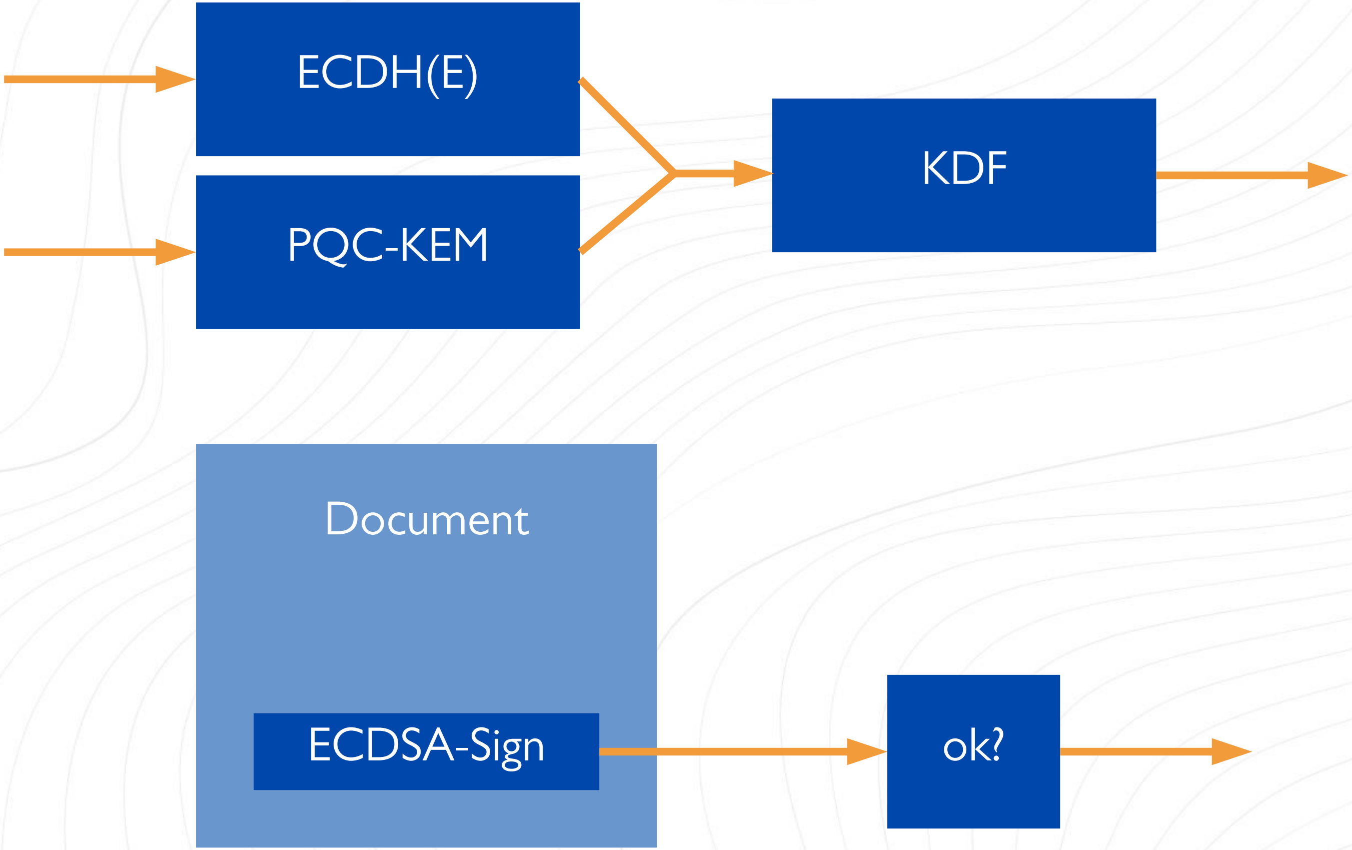
Hybrid Systems: PQC + ECC



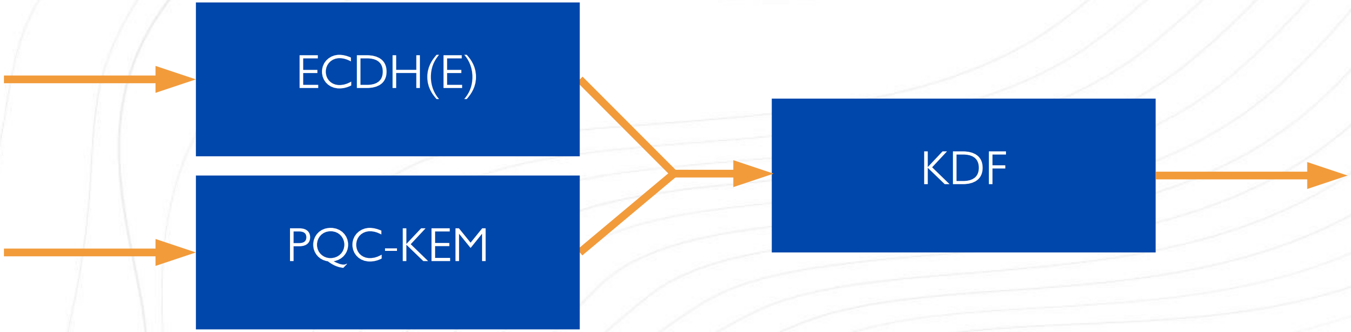
Hybrid Systems: PQC + ECC



Hybrid Systems: PQC + ECC



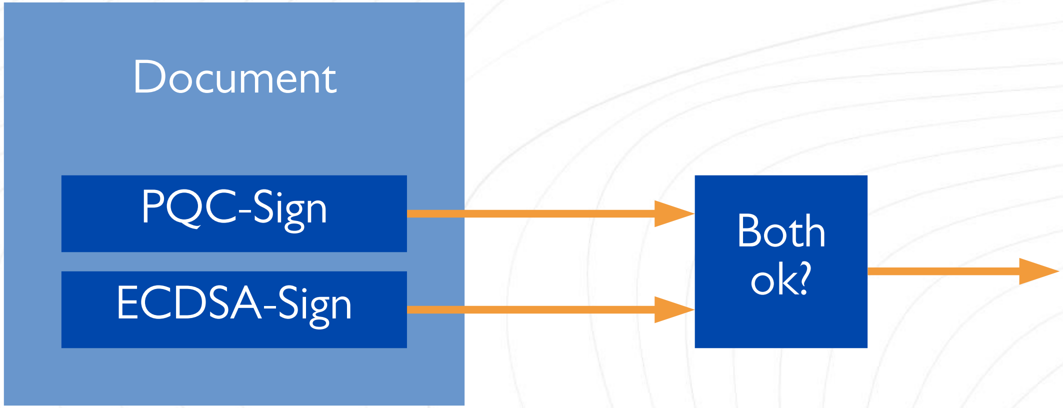
Hybrid Systems: PQC + ECC



ECDH(E)

PQC-KEM

KDF



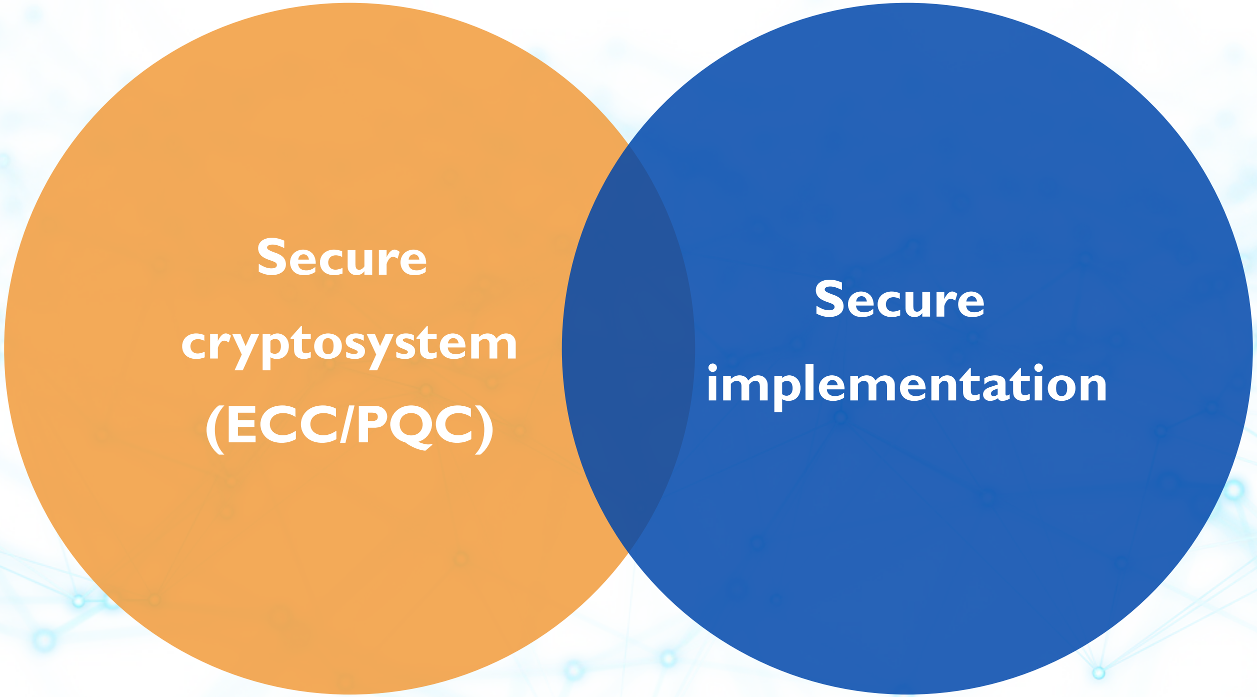
Document

PQC-Sign

ECDSA-Sign

Both ok?

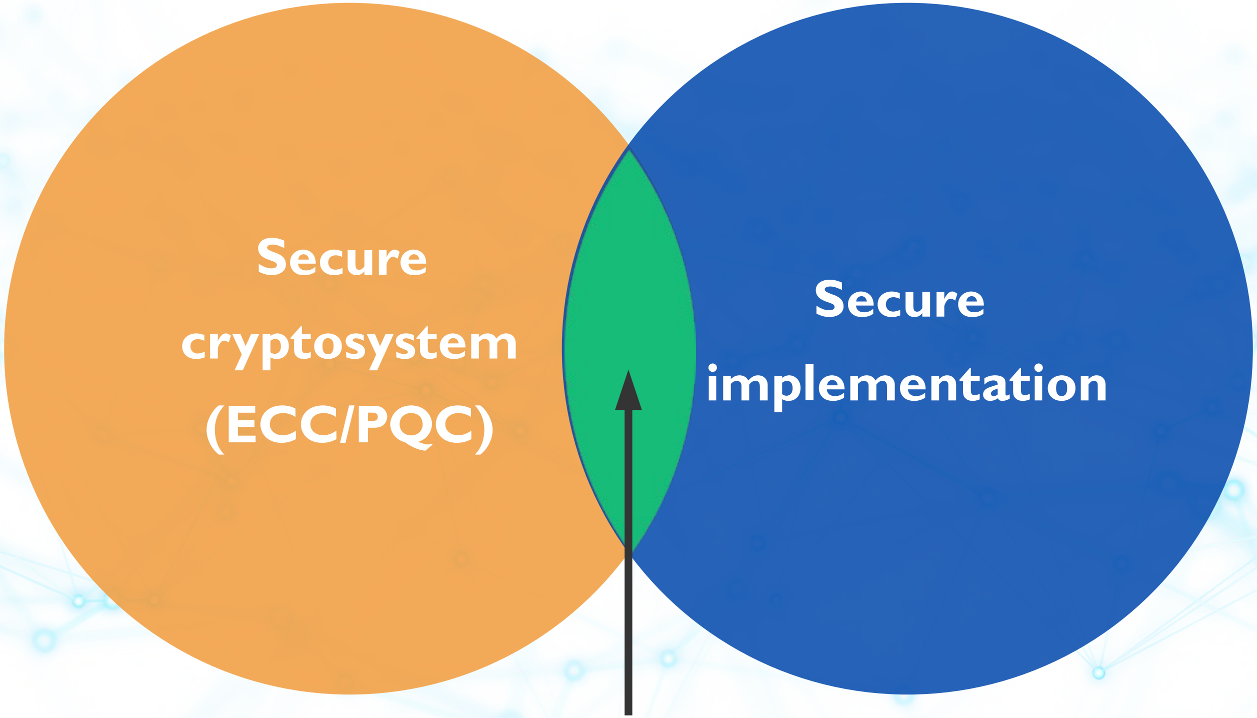
Key Take-Away



Secure cryptosystem (ECC/PQC)

Secure implementation

Key Take-Away

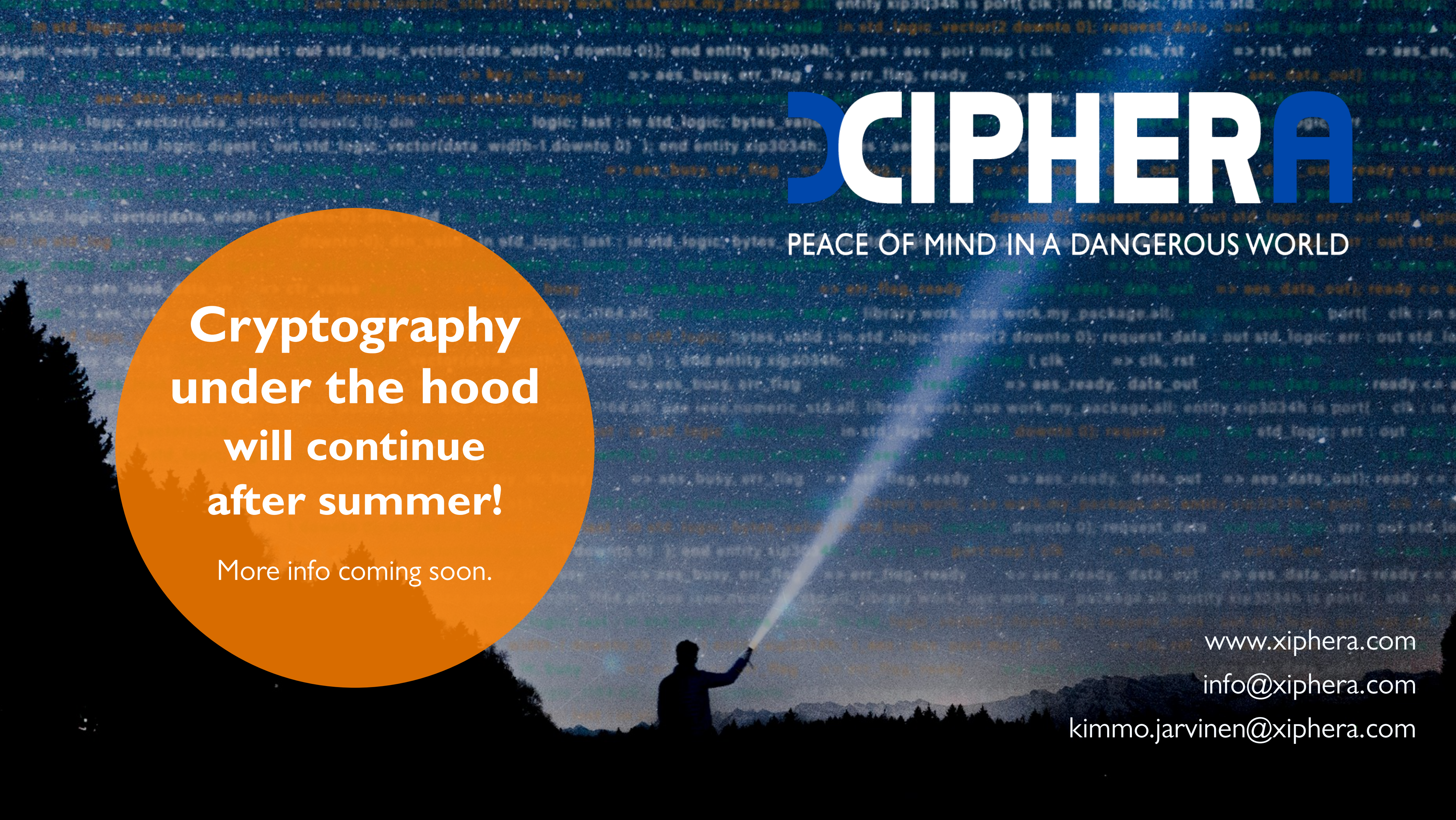


A system is secure *only* if it is here!

XIPHERA

Thank you!

www.xiphera.com
info@xiphera.com
kimmo.jarvinen@xiphera.com



XIPHERA

PEACE OF MIND IN A DANGEROUS WORLD

Cryptography under the hood will continue after summer!

More info coming soon.

www.xiphera.com

info@xiphera.com

kimmo.jarvinen@xiphera.com