

# Threshold Cryptography with Python

---

2020.09.05 @ PyCon Taiwan 2020



# Hello!

I am **changwu** (陳昶吾)

Adjunct Assistant Professor in the Department of Computer Science at NCCU

**[changwu@nccu.edu.tw](mailto:changwu@nccu.edu.tw)**



# Abstract

- Introduction to Threshold Cryptography
- Shamir's Secret Sharing
- Threshold ECDSA Signature
- SageMath



from <https://thespaces.com/inside-the-ned-5-of-its-best-hidden-spaces/>



# Threshold

- (m, n)-threshold scheme
- (2, 2)-threshold
- 3-of-5 threshold



# Status

- [Threshold Schemes for Cryptographic Primitives \(NIST\)](#)
- Keywords
  - threshold schemes, secure implementations, cryptographic primitives, threshold cryptography, secure multi- party computation, intrusion tolerance, distributed systems, resistance to side-channel attacks, standards and validation
- From Theory To Practice

# Motivation



COINTELEGRAPH



# Key loss and vanish

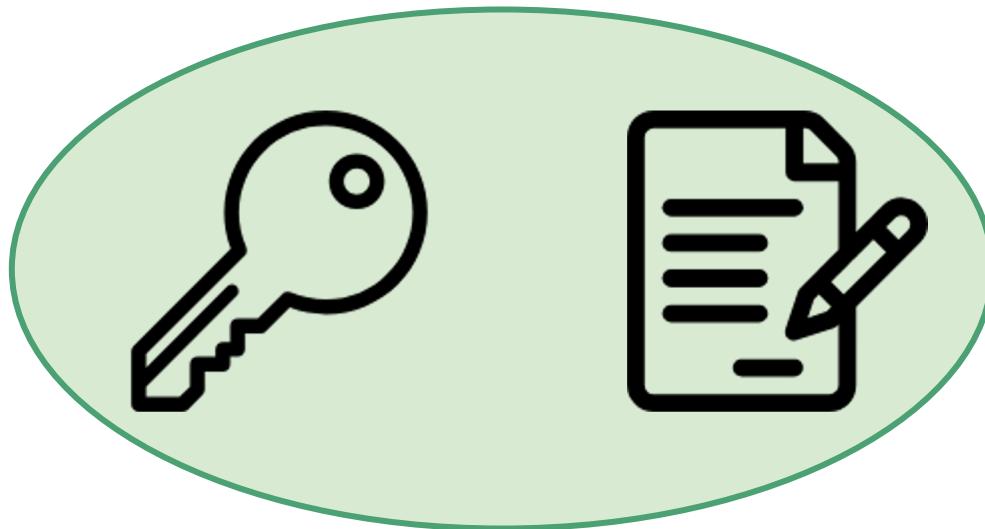
- Controversial QuadrigaCX cryptocurrency exchange placed in bankruptcy
  - \$190 million of funds got lost because the founder suddenly died in his trip. (April 8)
- Exit Scam? Dublin-Based Exchange Bitsane Vanishes With Users' Funds
  - Ireland-based cryptocurrency exchange Bitsane has apparently vanished, taking as many as 246,000 users' crypto deposits with it. The news was reported by Forbes on June 27.

# Hacked

- [Singapore-Based Crypto Exchange DragonEx Has Been Hacked](#) (Mar 26)
- [Crypto Exchange Bithumb Hacked for \\$13 Million in Suspected Insider Job](#) (Mar 30)

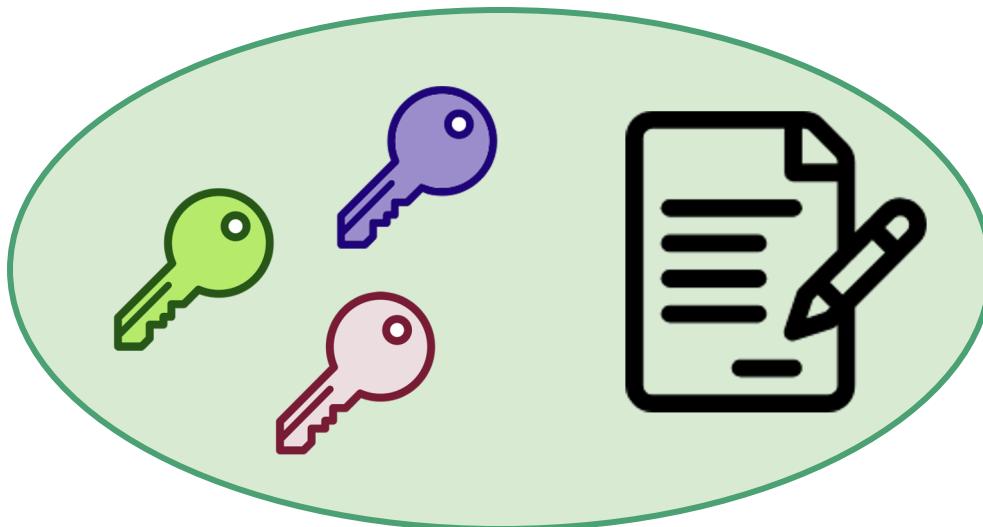


# What is a **valid** transaction?



Raw data + signature (Signed by private key)

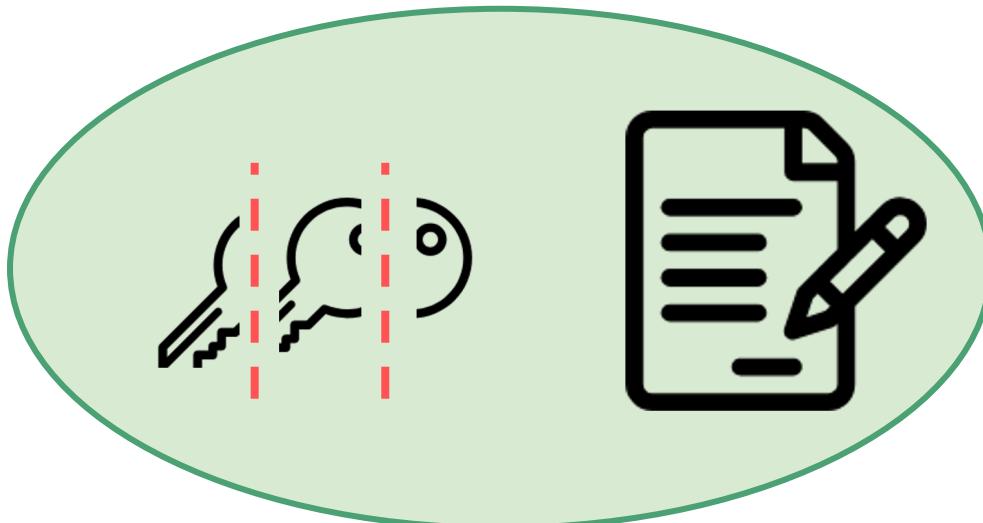
# Multisig transaction (m-of-n)



Raw data + multi-signature (Signed by multiple private keys)



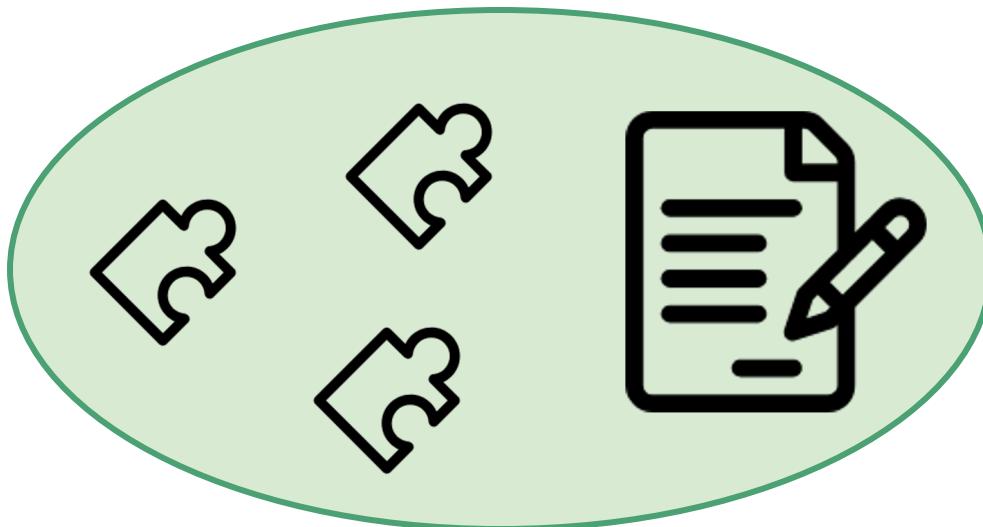
# Threshold key sharing



Key splitting



# Threshold signature (m-of-n)



Ephemeral puzzle



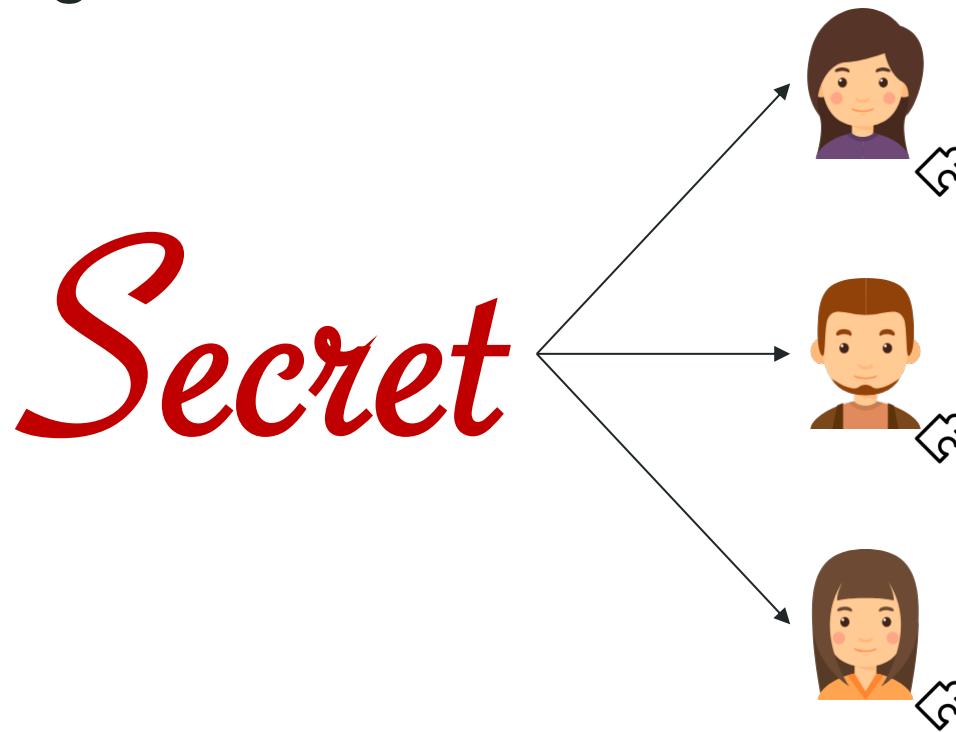
Private key will not be reconstructed.



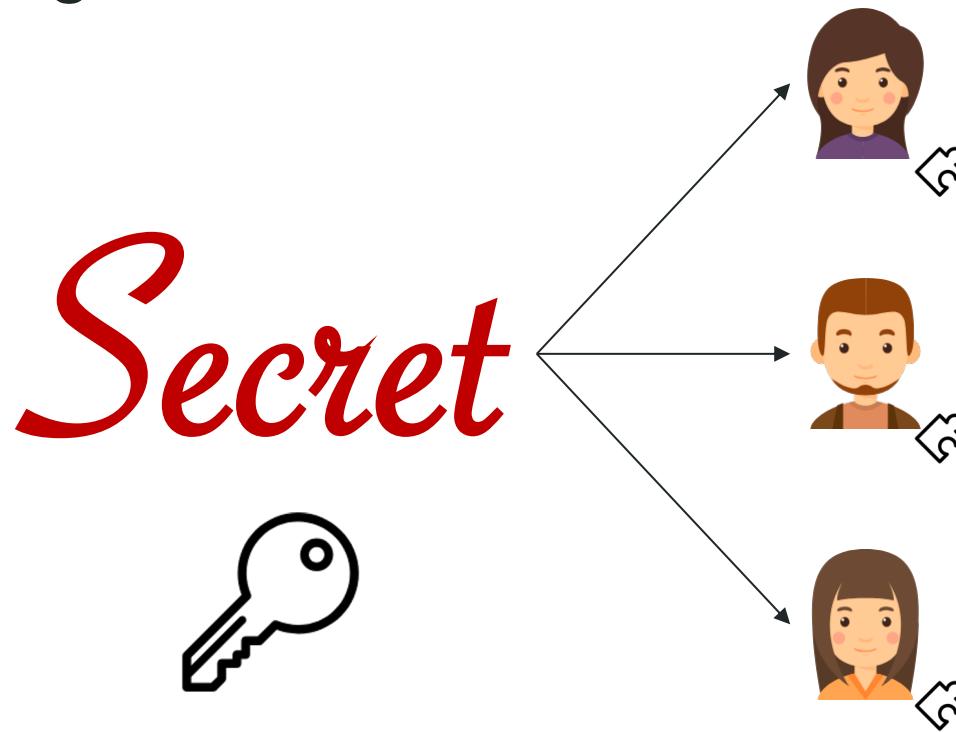
# Case 1

Secret sharing

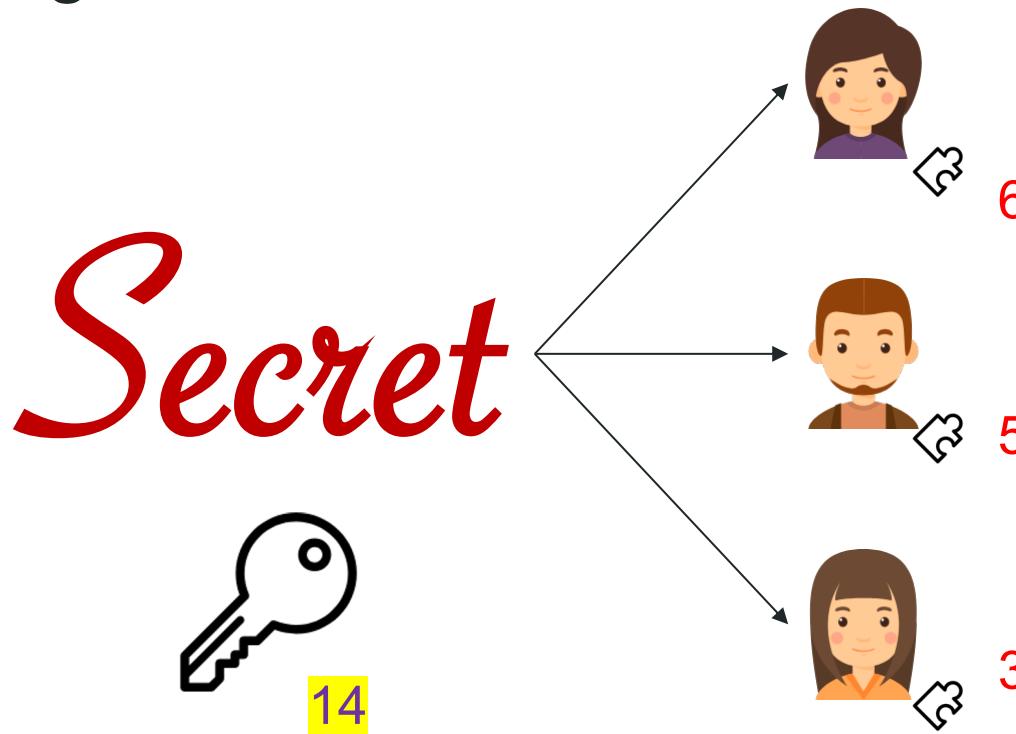
# Secret sharing



# Secret sharing



# Secret sharing



# Shamir's Secret Sharing

## Example

✓(3,5) secret sharing

✓K=11, p=17

# Shamir's Secret Sharing

## Example

✓(3,5) secret sharing

✓ $K=11$ ,  $p=17$

✓Construct a degree 2 random polynomial

$$F(x) = K + a_1x + a_2x^2 \bmod p$$

# Shamir's Secret Sharing

## Example

✓(3,5) secret sharing

✓ $K=11$ ,  $p=17$

✓Construct a degree 2 random polynomial

$$F(x) = K + a_1x + a_2x^2 \bmod p$$

✓For a random choice  $a_1=8$ ,  $a_2=7$

$$F(x) = 11 + 8x + 7x^2 \bmod 17$$

# Shamir's Secret Sharing

## Example

✓(3,5) secret sharing

✓ $K=11$ ,  $p=17$

✓Construct a degree 2 random polynomial

$$F(x) = K + a_1x + a_2x^2 \bmod p$$

✓For a random choice  $a_1=8$ ,  $a_2=7$

$$F(x) = 11 + 8x + 7x^2 \bmod 17$$

✓Share distribution

$$K_1 = F(1) = 7 \times 1^2 + 8 \times 1 + 11 \equiv 9 \pmod{17}$$

$$K_2 = F(2) = 7 \times 2^2 + 8 \times 2 + 11 \equiv 4 \pmod{17}$$

$$K_3 = F(3) = 7 \times 3^2 + 8 \times 3 + 11 \equiv 13 \pmod{17}$$

$$K_4 = F(4) = 7 \times 4^2 + 8 \times 4 + 11 \equiv 2 \pmod{17}$$

$$K_5 = F(5) = 7 \times 5^2 + 8 \times 5 + 11 \equiv 5 \pmod{17}$$

$K_1, K_2, K_3, K_4, K_5$  : shares given to  $(P_1, \dots, P_5)$

# Shamir's Secret Sharing

## Example

✓ (3,5) secret sharing

✓  $K=11$ ,  $p=17$

✓ Construct a degree 2 random polynomial

$$F(x) = K + a_1x + a_2x^2 \pmod{p}$$

✓ For a random choice  $a_1=8$ ,  $a_2=7$

$$F(x) = 11 + 8x + 7x^2 \pmod{17}$$

✓ Share distribution

$$K_1 = F(1) = 7 \times 1^2 + 8 \times 1 + 11 \equiv 9 \pmod{17}$$

$$K_2 = F(2) = 7 \times 2^2 + 8 \times 2 + 11 \equiv 4 \pmod{17}$$

$$K_3 = F(3) = 7 \times 3^2 + 8 \times 3 + 11 \equiv 13 \pmod{17}$$

$$K_4 = F(4) = 7 \times 4^2 + 8 \times 4 + 11 \equiv 2 \pmod{17}$$

$$K_5 = F(5) = 7 \times 5^2 + 8 \times 5 + 11 \equiv 5 \pmod{17}$$

$K_1, K_2, K_3, K_4, K_5$  : shares given to  $(P_1, \dots, P_5)$

*Lagrange polynomial:* Given  $t$  points, we can obtain the unique polynomial.

$$y(x) = \sum_{i=1}^t y(i)l_i(x)$$

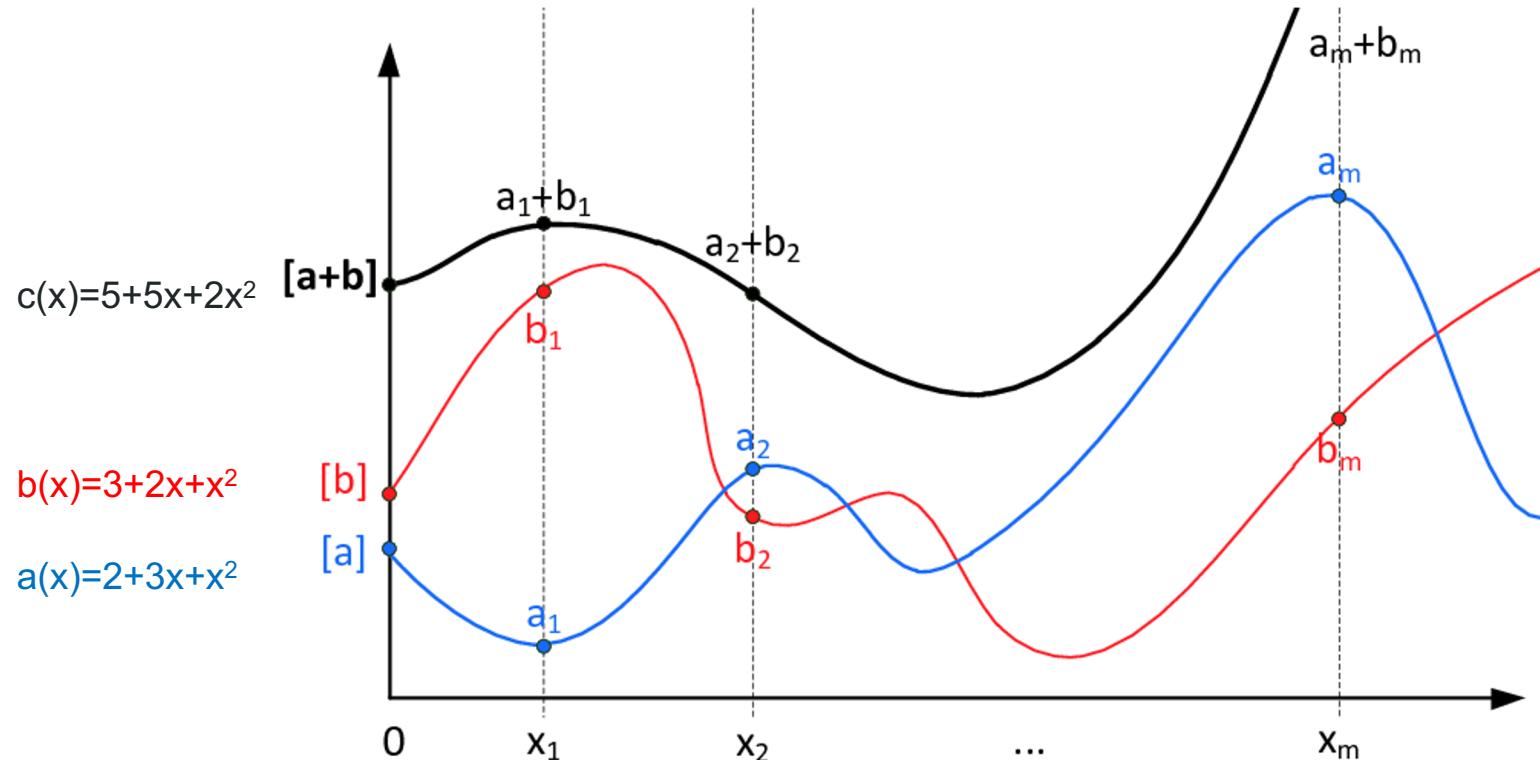
$$l_i(x) = \prod_{1 \leq m \leq t, m \neq i} \frac{(x - x_m)}{(x_i - x_m)} = \frac{(x - x_1)}{(x_i - x_1)} \cdots \frac{(x - x_t)}{(x_i - x_t)}$$

Using the Lagrange interpolation

For  $\Lambda = (K_1, K_2, K_3)$

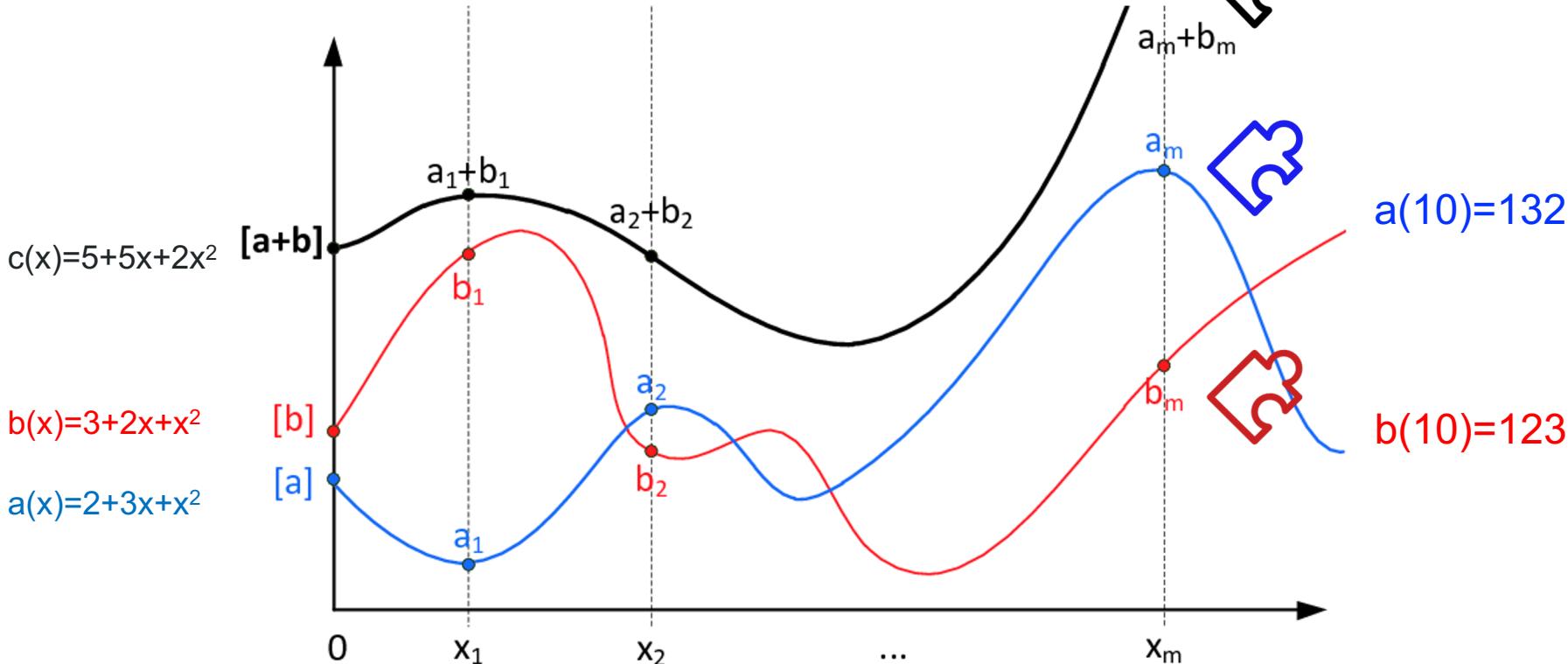
$$\begin{aligned} K &= K_1 \left( \frac{2}{2-1} \frac{3}{3-1} \right) + K_2 \left( \frac{1}{1-2} \frac{3}{3-2} \right) + K_3 \left( \frac{1}{1-3} \frac{2}{2-3} \right) \\ &= 9 \cdot 3 + 4 \cdot (-3) + 13 \cdot 1 \pmod{17} = 11 \end{aligned}$$

# Shamir's Secret Sharing



# Shamir's Secret Sharing

$$c(10)=255$$



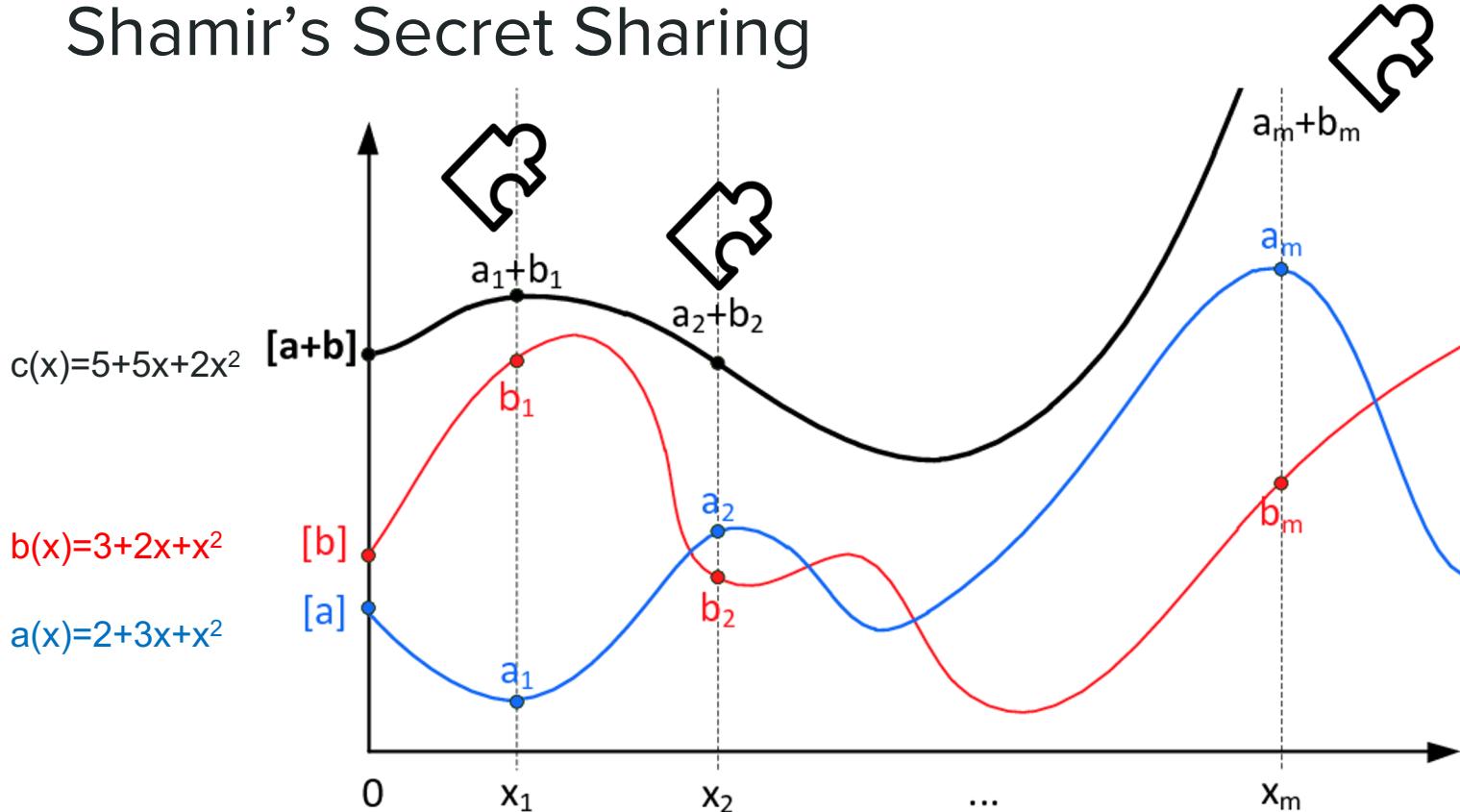
$$a(10)=132$$



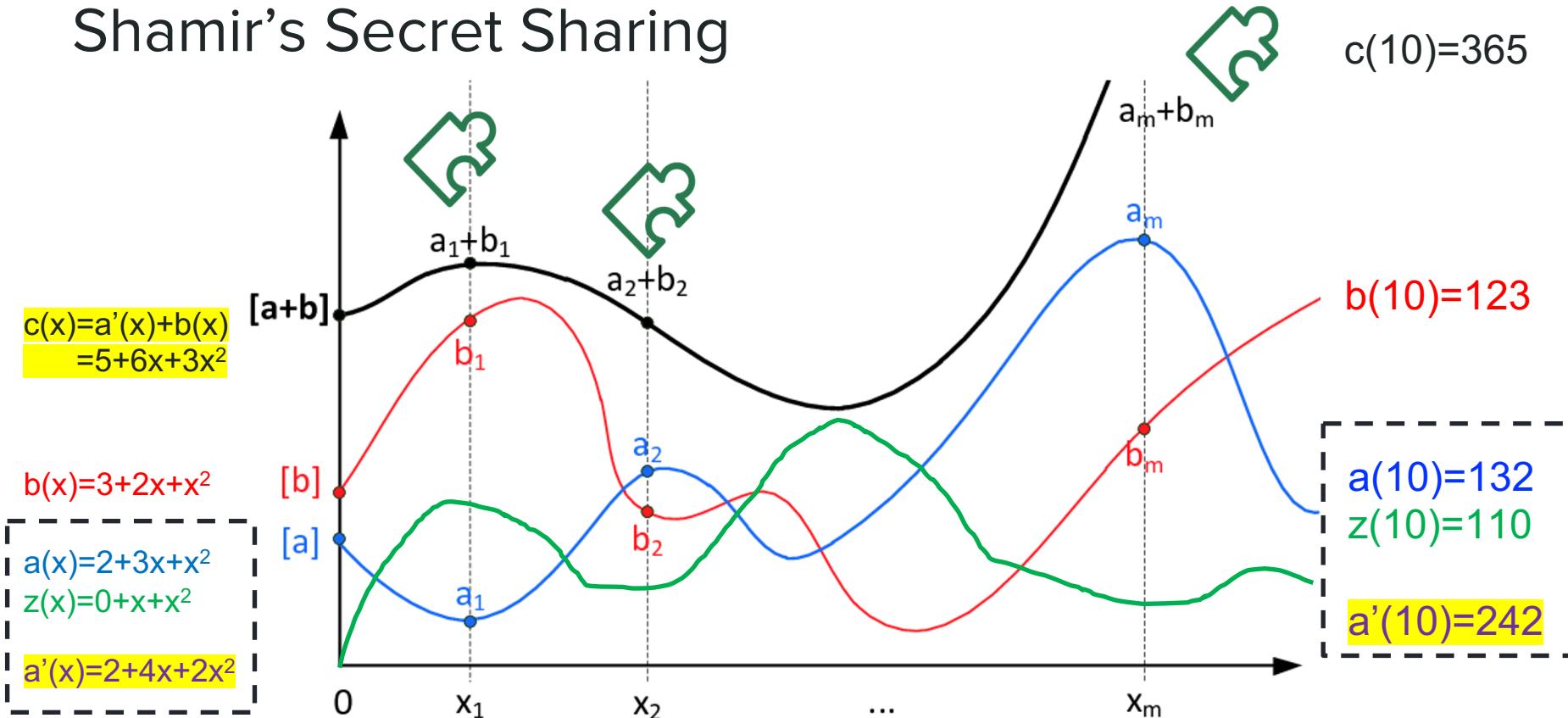
$$b(10)=123$$



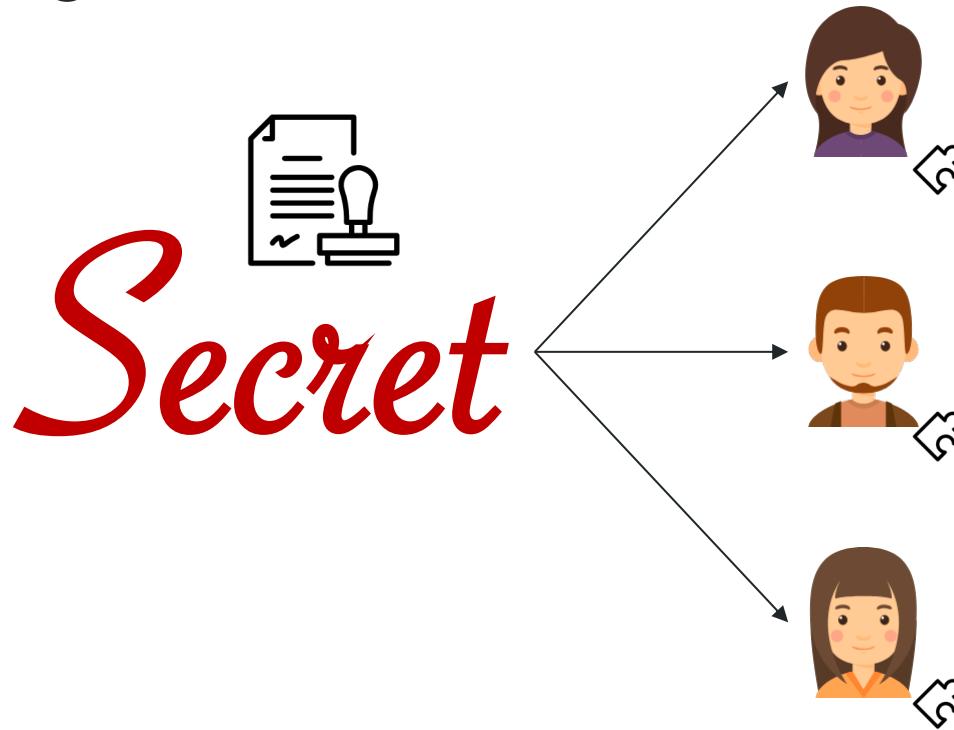
# Shamir's Secret Sharing



# Shamir's Secret Sharing



# Secret sharing

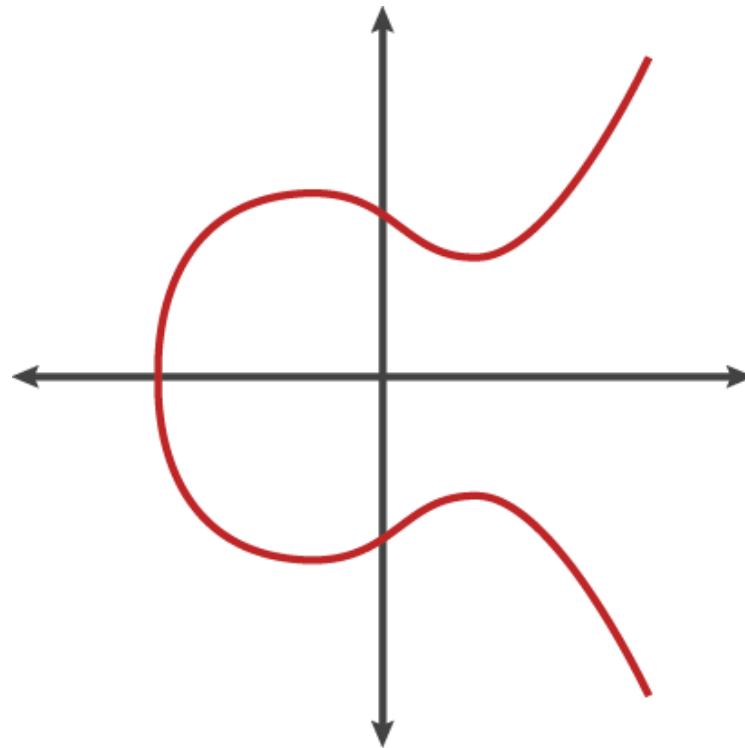


What if we can just generate a valid signature without the private key?

# Case 2

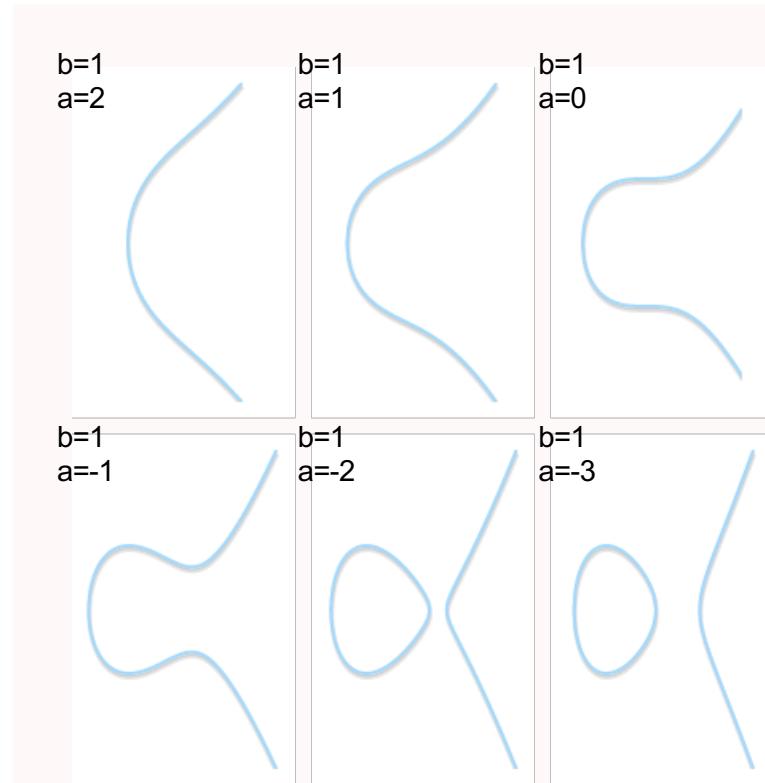
ECDSA signature

# Elliptic Curves

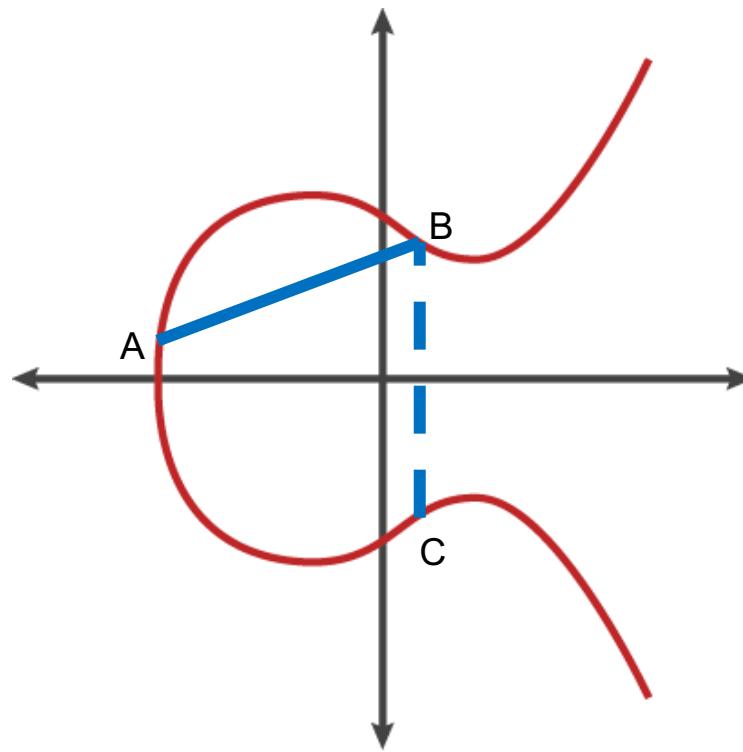


$$y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0$$

# Elliptic Curves

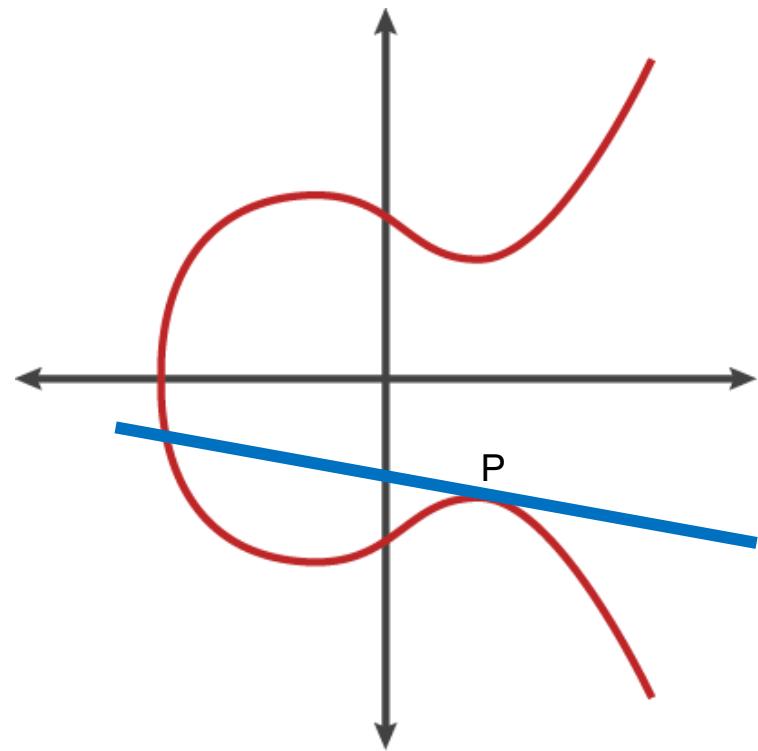


# Point addition in elliptic curves



# Trap-door function

- Discrete Logarithm Problem (DLP)
- $P + P + P + P + \dots + P = k * P$



# Why ECC?

- A 256 bit key in ECC offers about the same security as 3072 bit key using RSA.

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512
NIST recommend key sizes		

1. <https://csrc.nist.gov/publications/detail/sp/800-131a/rev-1/final>
2. <https://www.keylength.com/en/compare/>

# ECDSA

- Private key: **d**
- Public key:  **$d^*G$**

```
p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
a = 0
b = 7
Gx = 0x79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
Gy = 0x483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8
n = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
```

## Sign

- Message: e
- Secret number: **k**
- $r_x = k^*G$
- $s = k^{-1} * (e + d^*r_x)$
- sig:  $(r_x, s)$

## Verify

- Compute  $s^{-1}$
- $r_x = (e^*s^{-1}) * G + (r^*s^{-1}) * (d^*G)$
- Compare  $r == r_x$

secp256k1's elliptic curve:  $y^2 = x^3 + 7$

# ECDSA

- Private key: **d**
- Public key:  **$d \cdot G$**

$p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F$

$a = 0$

$b = 7$

$G_x = 0x79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798$

$G_y = 0x483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8$

$n = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141$

## Sign

- Message:  $e$
- Secret number: **k**
- $r_x = k \cdot G$
- $s = k^{-1} \cdot (e + d \cdot r_x)$
- sig:  $(r_x, s)$

## Verify

- Compute  $s^{-1}$
- $r_x = (e \cdot s^{-1}) \cdot G + (r \cdot s^{-1}) \cdot (d \cdot G)$
- Compare  $r == r_x$

# Generation of k

Two different signatures but with the same  $r_x$

- Private key:  $d$
- Public key:  $d^*G$

$(r_x, s_1)$   
 $(r_x, s_2)$

## Sign

- Message:  $e$
- Secret number:  $k$
- $r_x = k^*G$
- $s = k^{-1} * (e + d^*r_x)$
- sig:  $(r_x, s)$

Be cautious to have an unique random number  $k$   
Ref: [RFC 6979](#)

$$\begin{array}{rcl} s_1 & = & k^{-1} * (e_1 + d^*r) \\ \hline \hline s_2 & = & k^{-1} * (e_2 + d^*r) \end{array}$$

# Generation of k

Two different signatures but with the same  $r_x$

- Private key:  $d$
- Public key:  $d^*G$

$(r_x, s_1)$   
 $(r_x, s_2)$

## Sign

- Message:  $e$
- Secret number:  $k$
- $r_x = k^*G$
- $s = k^{-1} * (e + d^*r_x)$
- sig:  $(r_x, s)$

Be cautious to have an unique random number  $k$   
Ref: [RFC 6979](#)

$$\begin{array}{rcl} s_1 & = & \cancel{k^{-1} * (e_1 + d^*r)} \\ \hline \hline s_2 & = & \cancel{k^{-1} * (e_2 + d^*r)} \end{array}$$

# Generation of k

Two different signatures but with the same  $r_x$

- Private key:  $d$
- Public key:  $d^*G$

$(r_x, s_1)$   
 $(r_x, s_2)$

## Sign

- Message:  $e$
- Secret number:  $k$
- $r_x = k^*G$
- $s = k^{-1} * (e + d^*r_x)$
- sig:  $(r_x, s)$

Be cautious to have an unique random number  $k$   
Ref: [RFC 6979](#)

$$\begin{array}{rcl} s_1 & = & (e_1 + d^*r) \\ \hline s_2 & = & (e_2 + d^*r) \end{array} \quad (1)$$

$$s_1^*e_2 + s_1^*d^*r = s_2^*e_1 + s_2^*d^*r \quad (2)$$

$$d^*(s_1^*r - s_2^*r) = s_2^*e_1 - s_1^*e_2 \quad (3)$$

$$d = (s_2^*e_1 - s_1^*e_2) / (s_1^*r - s_2^*r) \quad (4)$$

# ECDSA

- Private key:  $d$
- Public key:  $d^*G$

```
p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
a = 0
b = 7
Gx = 0x79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
Gy = 0x483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8
n = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
```

## Sign

- Message:  $e$
- Secret number:  $k$
- $r_x = k^*G$
- $s = k^{-1} * (e + d^*r_x)$
- sig:  $(r_x, s)$

## Masking

- (0) Choose two random numbers  $k, a$
- (1)  $\mu = k^*a$
- (2)  $\beta = a^*G$
- (3)  $\mu^{-1}\beta = (k^*a)^{-1} * (a^*G) = k^{-1}G$

Multiplication to addition? (MtA process)

# Reference implementation

- <https://github.com/getamis/alice>



# Thanks!

Any questions?

# Appendix

# Homomorphic cryptosystem (Paillier cryptosystem)

## Homomorphic properties [\[ edit \]](#)

A notable feature of the Paillier cryptosystem is its homomorphic properties along with its non-deterministic encryption (see Electronic voting in Applications for usage). As the encryption function is additively homomorphic, the following identities can be described:

- **Homomorphic addition of plaintexts**

The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts,

$$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n.$$

The product of a ciphertext with a plaintext raising  $g$  will decrypt to the sum of the corresponding plaintexts,

$$D(E(m_1, r_1) \cdot g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n.$$

- **Homomorphic multiplication of plaintexts**

An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,

$$D(E(m_1, r_1)^{m_2} \bmod n^2) = m_1 m_2 \bmod n,$$

$$D(E(m_2, r_2)^{m_1} \bmod n^2) = m_1 m_2 \bmod n.$$

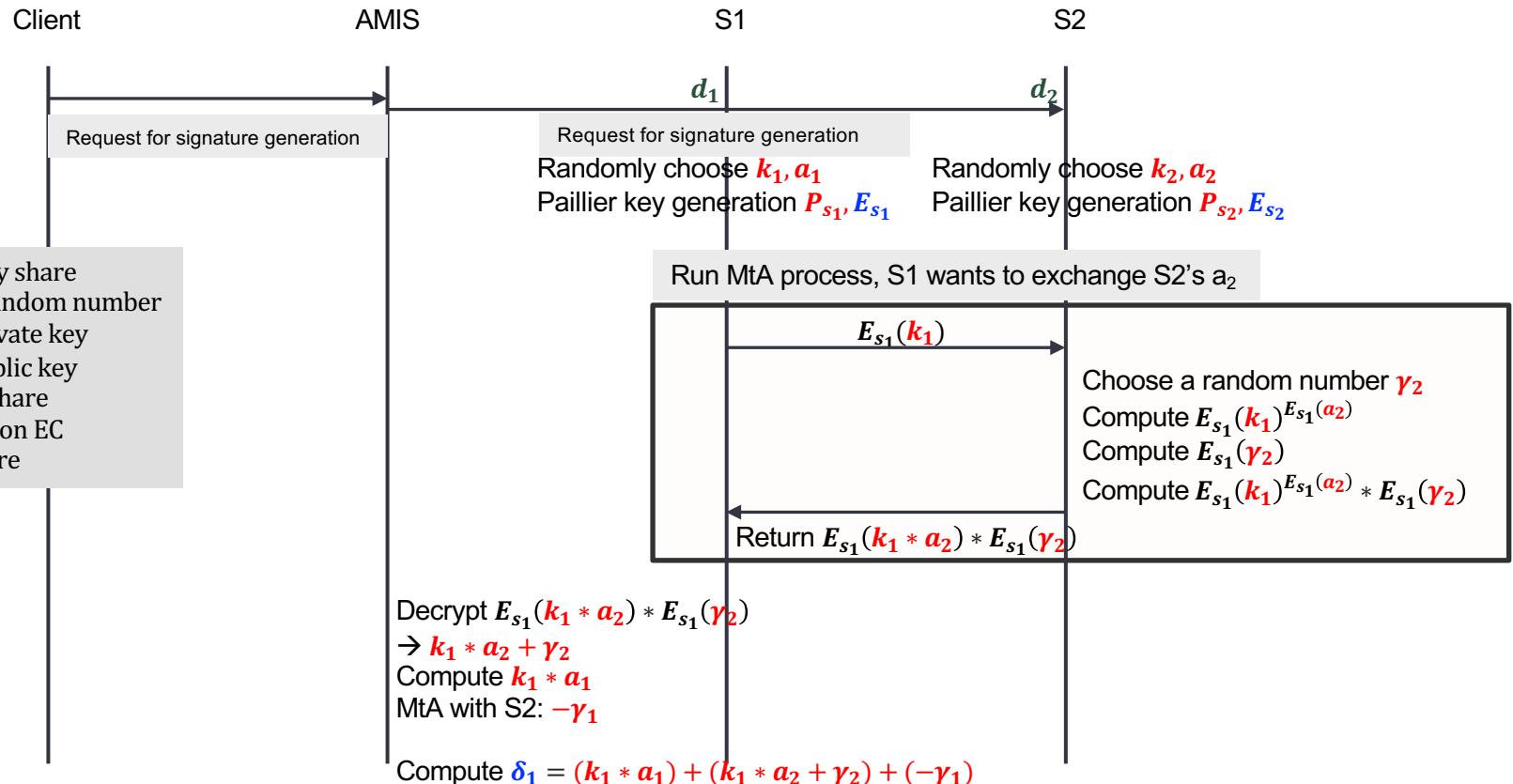
More generally, an encrypted plaintext raised to a constant  $k$  will decrypt to the product of the plaintext and the constant,

$$D(E(m_1, r_1)^k \bmod n^2) = k m_1 \bmod n.$$

However, given the Paillier encryptions of two messages there is no known way to compute an encryption of the product of these messages without knowing the private key.

# MtA process (1/2)

$$\begin{aligned}
 k^*a &= (k_1 + k_2) * (a_1 + a_2) \\
 &= (k_1 * a_1 + k_1 * a_2 + k_2 * a_1 + k_2 * a_2)
 \end{aligned}$$



# MtA process (2/2)

Client

$d_i$ : private key share  
 $k_i, a_i, \gamma_i, \theta_i$ : random number  
 $P$ : Paillier private key  
 $E$ : Paillier public key  
 $\delta_i, \sigma_i$ : secret share  
 $G$ : base point on EC  
 $(r, s)$ : signature

AMIS

S1

S2

Randomly choose  $k_1, a_1$   
Paillier key generation  $P_{s1}, E_{s1}$

Randomly choose  $k_2, a_2$   
Paillier key generation  $P_{s2}, E_{s2}$

Run MtA process, S2 wants to exchange S1's  $a_1$

Choose a random number  $\gamma_1$   
Compute  $E_{s2}(k_2)^{E_{s2}(a_1)}$   
Compute  $E_{s2}(\gamma_1)$   
Compute  $E_{s2}(k_2)^{E_{s2}(a_1)} * E_{s2}(\gamma_1)$

$E_{s2}(k_2)$

Return  $E_{s2}(k_2 * a_1) * E_{s2}(\gamma_1)$

Decrypt  $E_{s2}(k_2 * a_1) * E_{s2}(\gamma_1)$   
 $\rightarrow k_2 * a_1 + \gamma_1$   
Compute  $k_2 * a_2$   
MtA with S1:  $-\gamma_2$

Return  $\delta_1$

Return  $\delta_2$

Compute  $\delta_2 = (k_2 * a_2) + (k_2 * a_1 + \gamma_1) + (-\gamma_2)$

[註]

$$\begin{aligned}\delta_1 &= (k_1 * a_1) + (k_1 * a_2 + \gamma_2) + (-\gamma_1) \\ \delta_2 &= (k_2 * a_2) + (k_2 * a_1 + \gamma_1) + (-\gamma_2)\end{aligned}$$

$$\delta_1 + \delta_2 = (k_1 + k_2) * (a_1 + a_2)$$

Compute  $\mu = \sum_i \delta_i \text{ mod } q$