Threshold ECDSA in Three Rounds

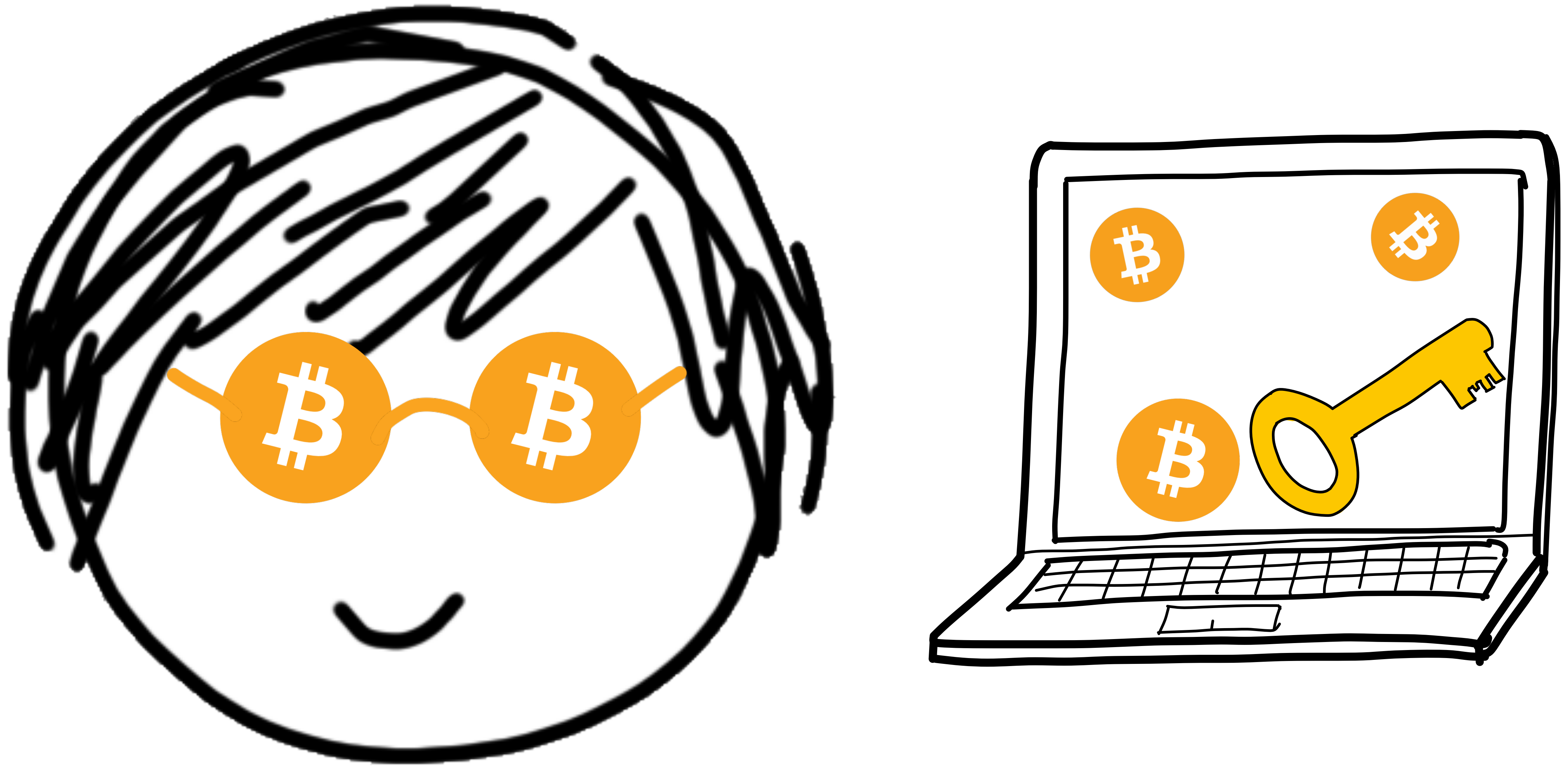
Jack Doerner   Yashvanth Kondi   Eysa Lee   abhi shelat

TECHNION Israel Institute of Technology

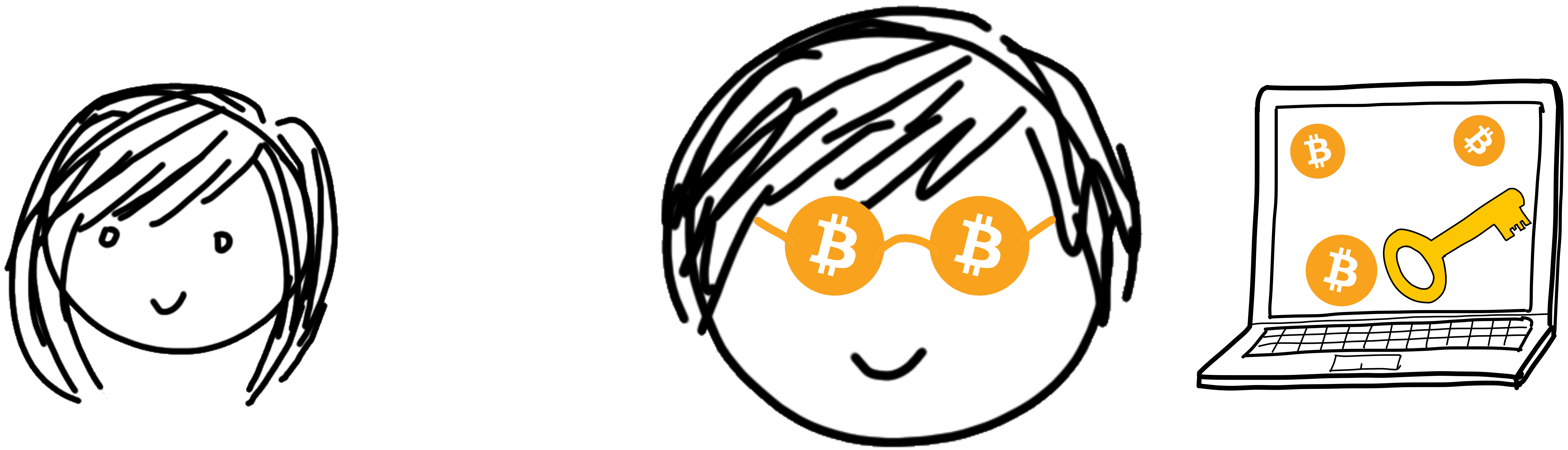
AARHUS UNIVERSITY

Northeastern University

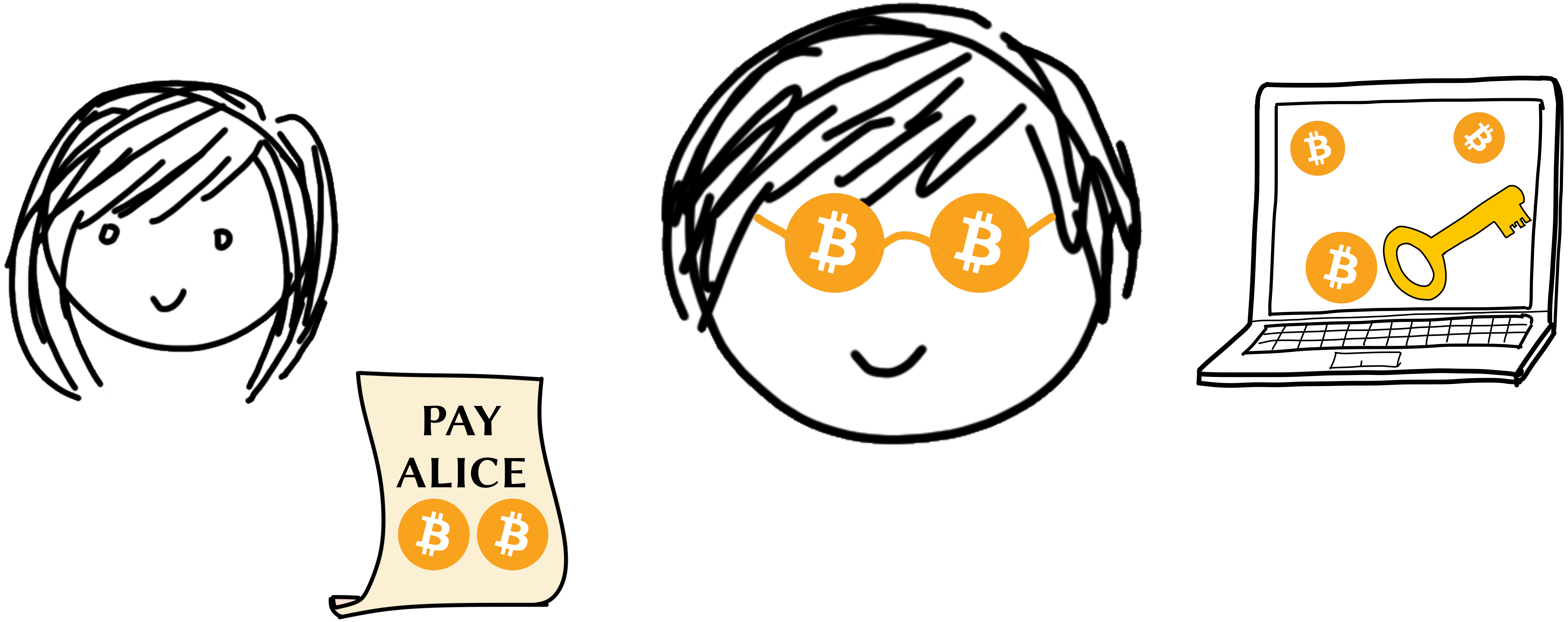
Ballad of ₿itcoin Bob



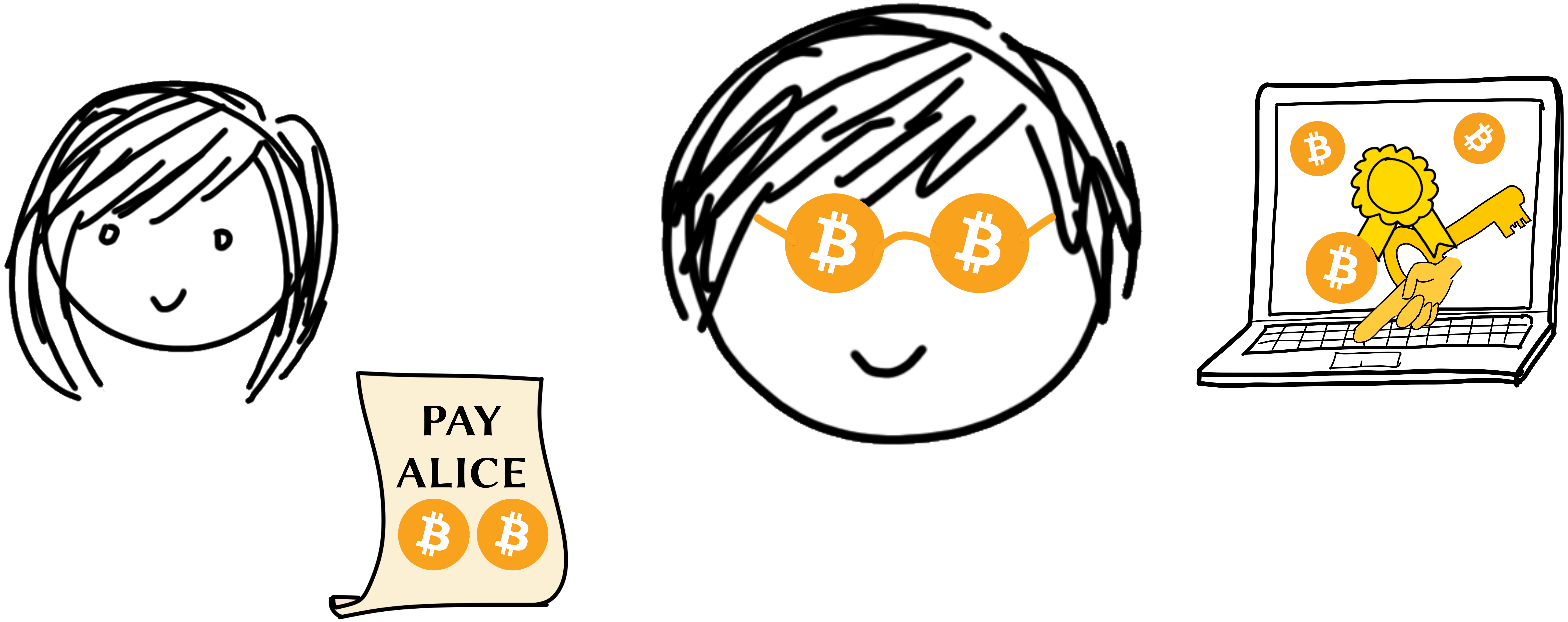
Ballad of ₿itcoin Bob



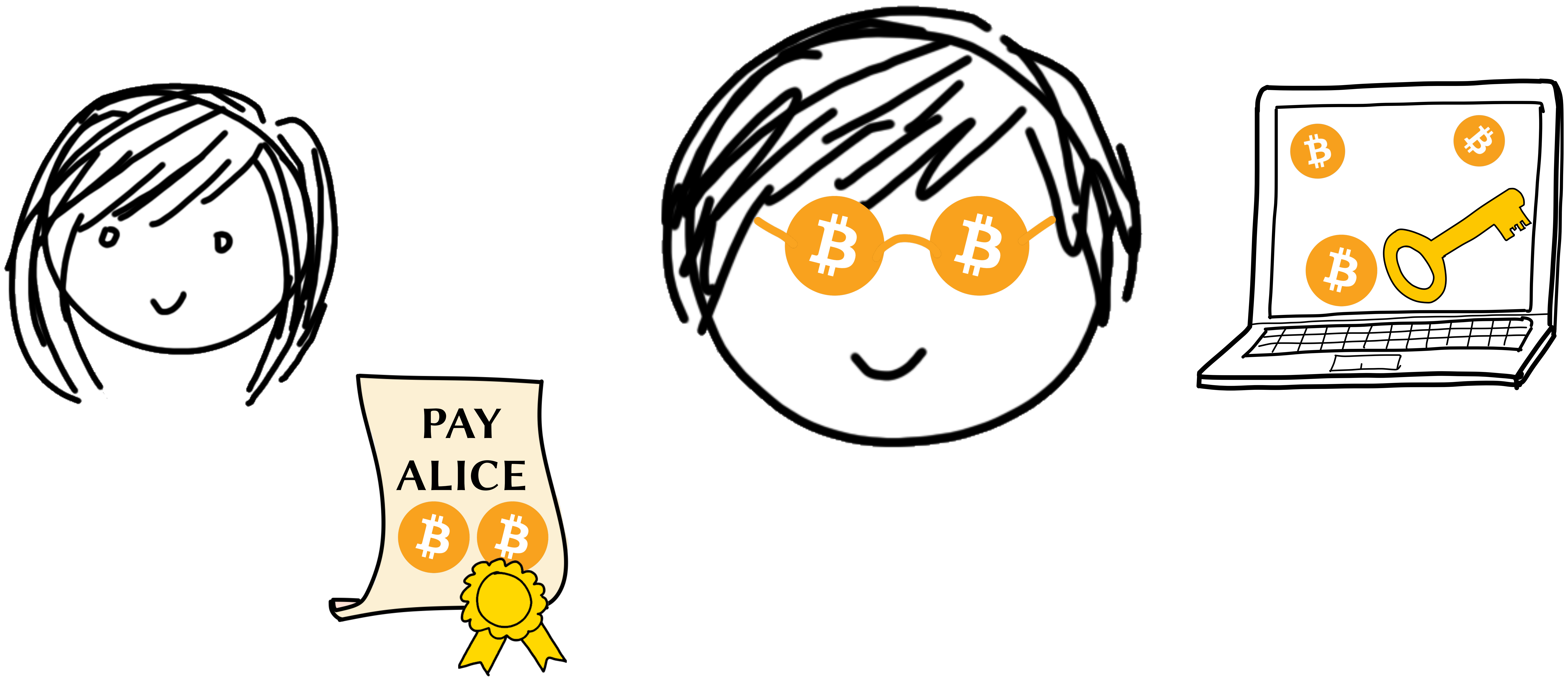
Ballad of ₿itcoin Bob



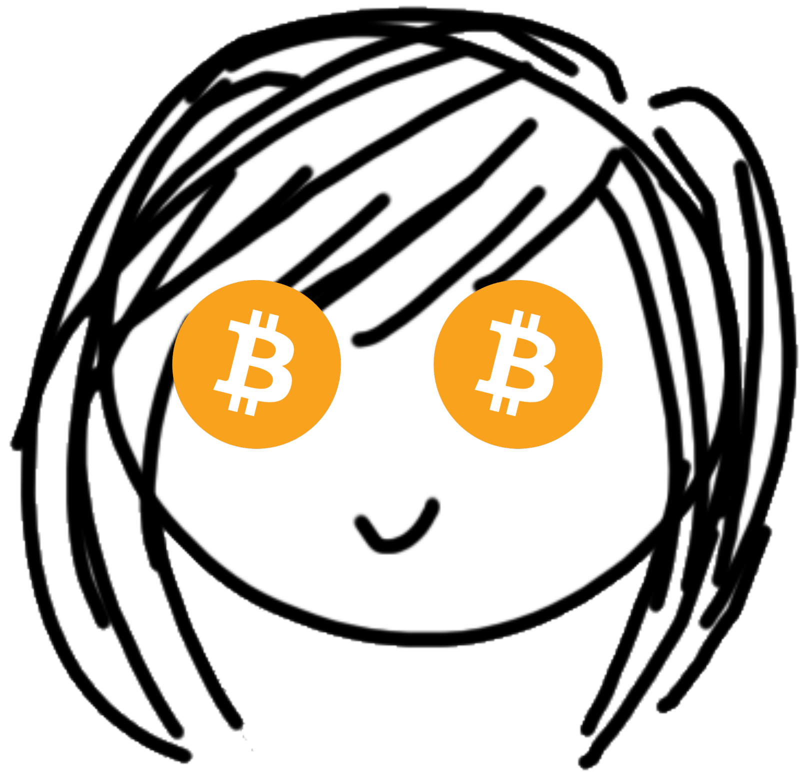
Ballad of Ƀitcoin Bob

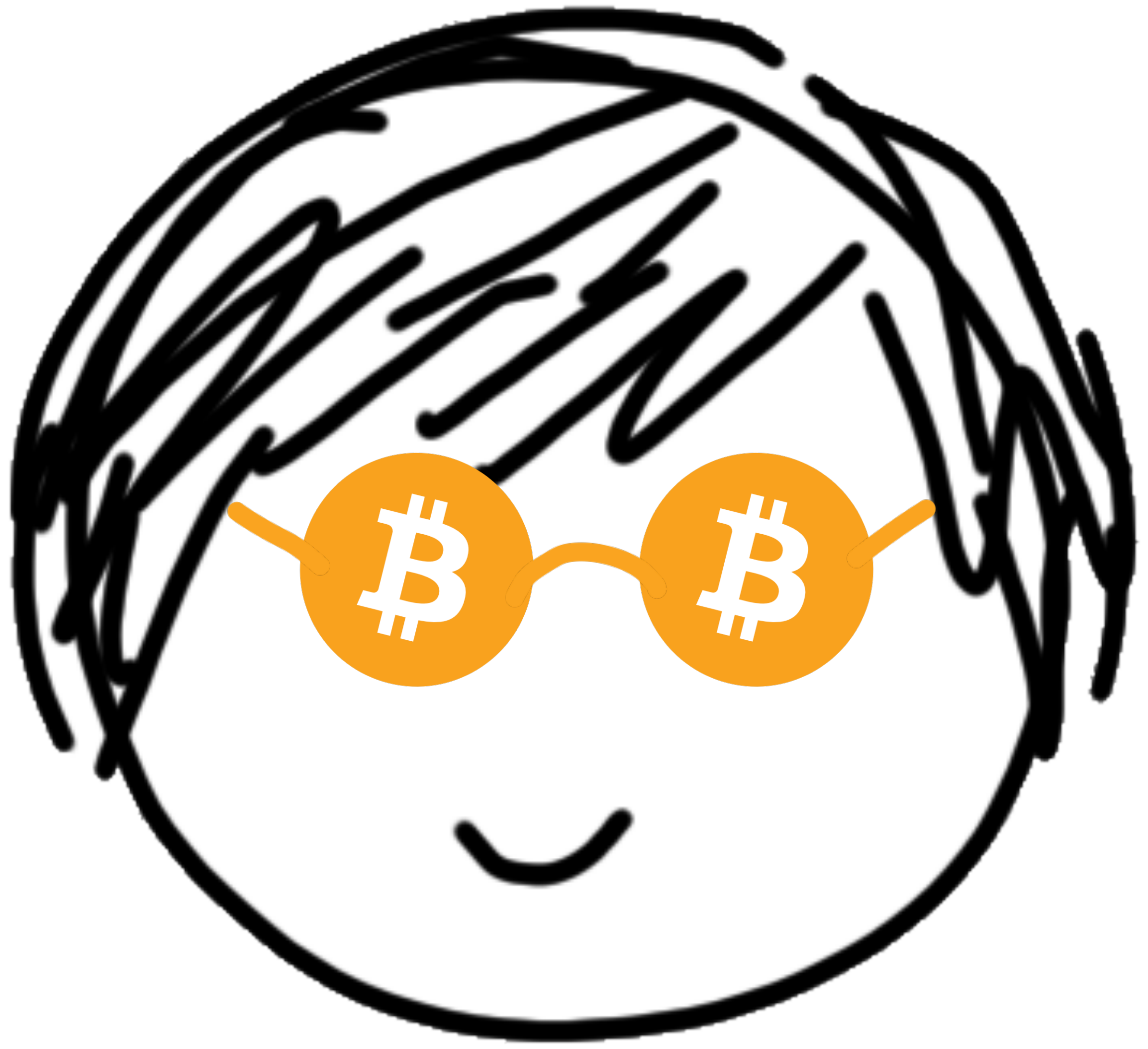


Ballad of ₿itcoin Bob

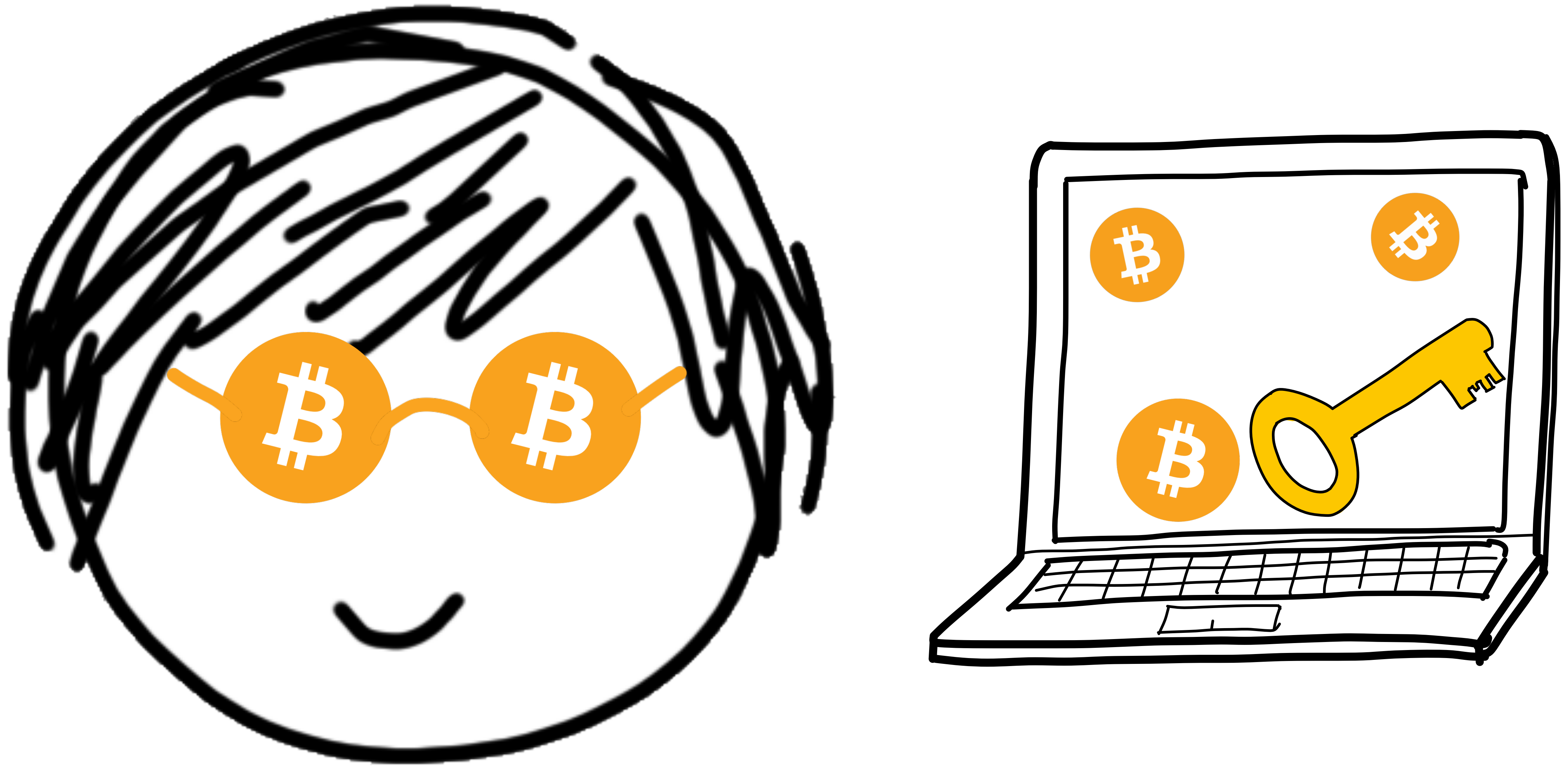


Ballad of ₿itcoin Bob

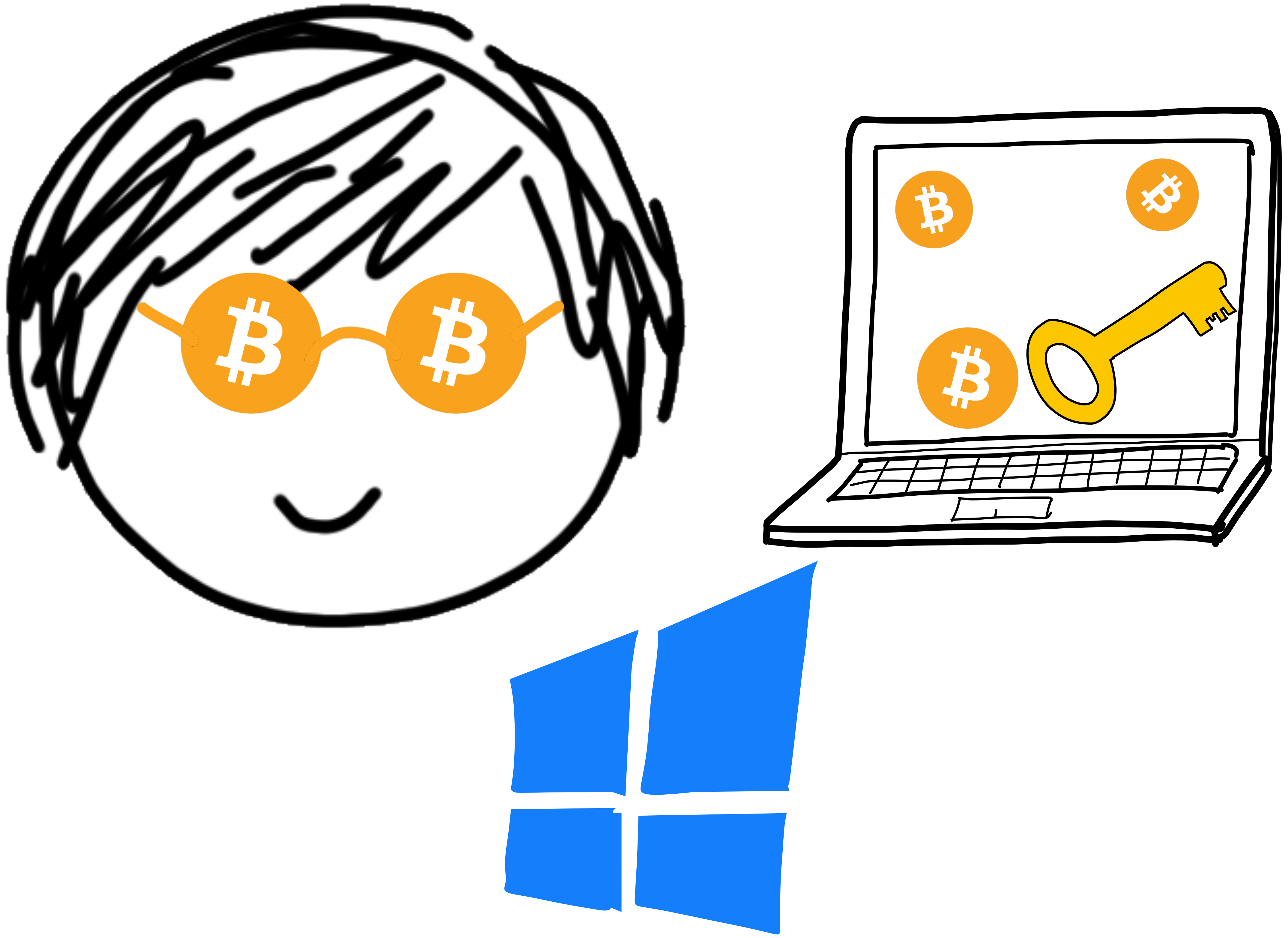




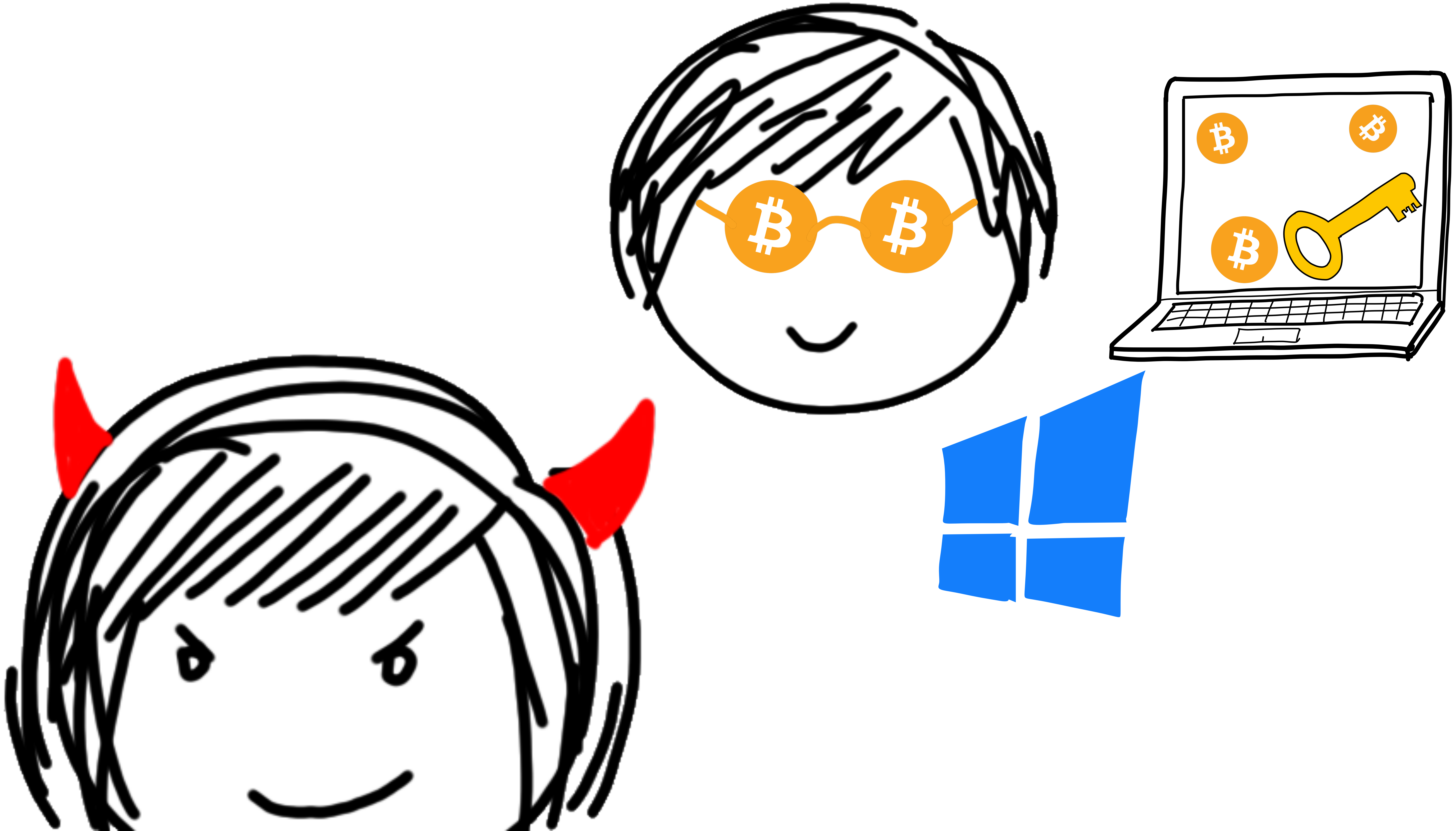
Ballad of ₿itcoin Bob



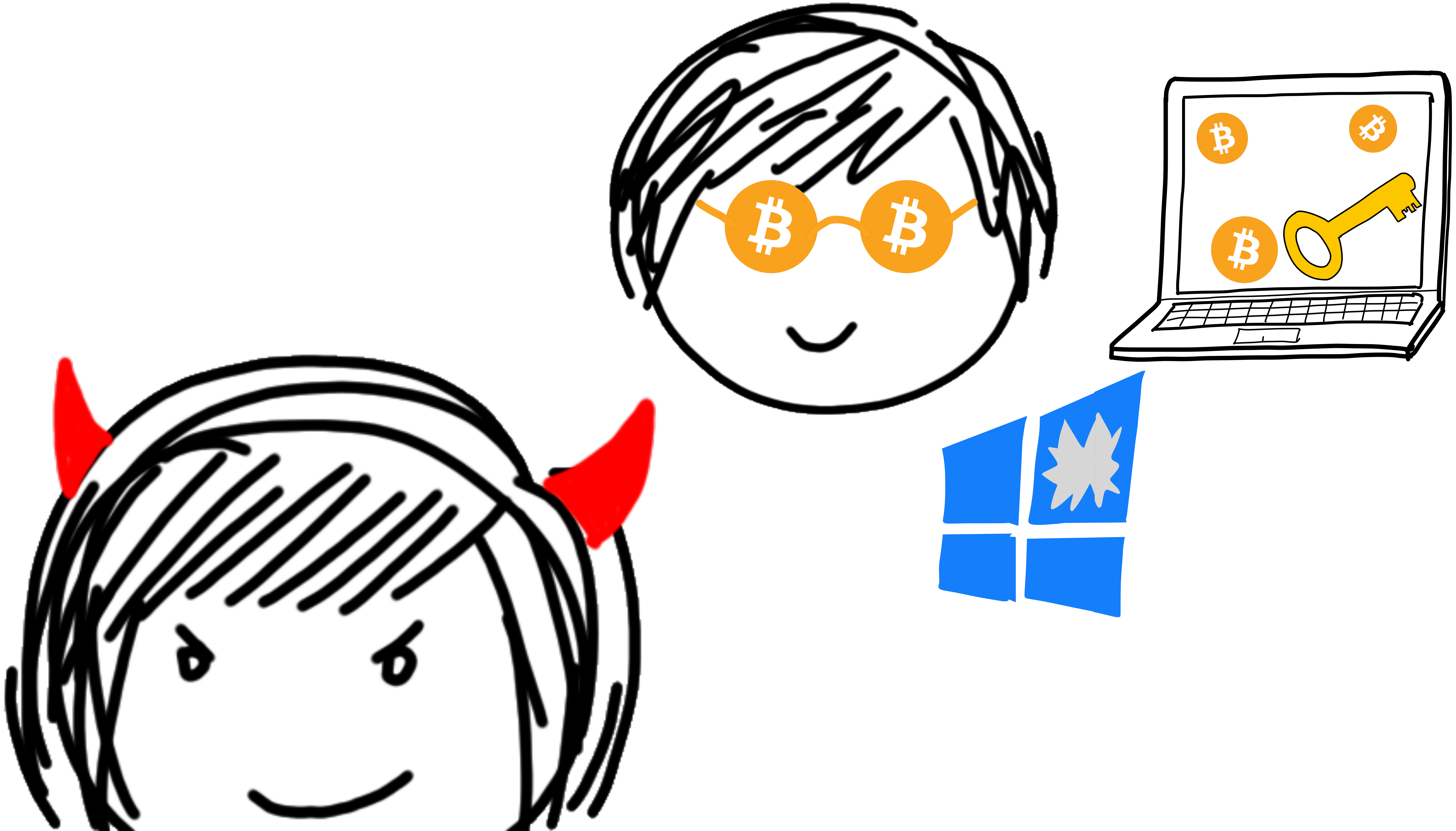
Ballad of ₿itcoin Bob



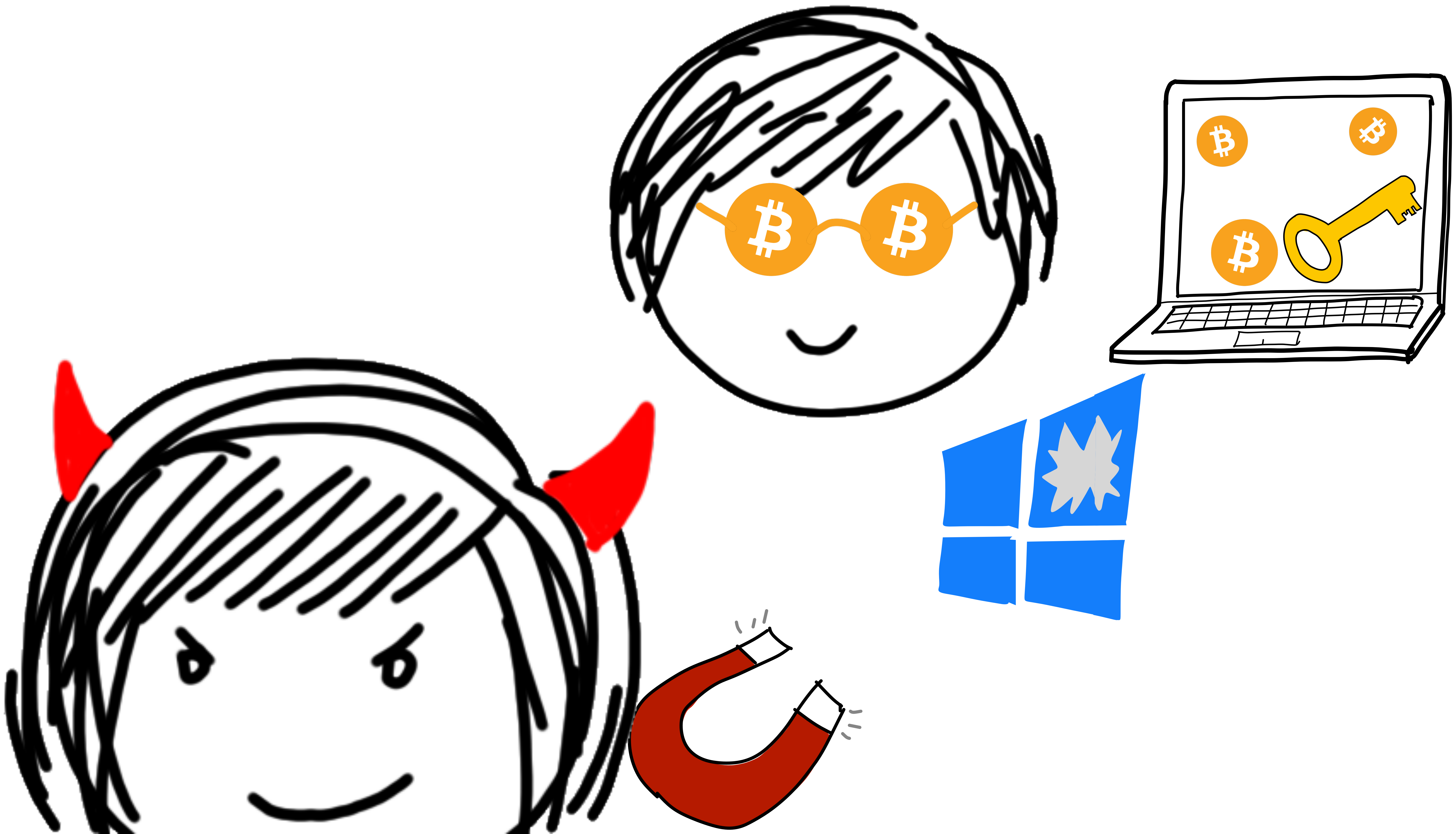
Ballad of ₿itcoin Bob



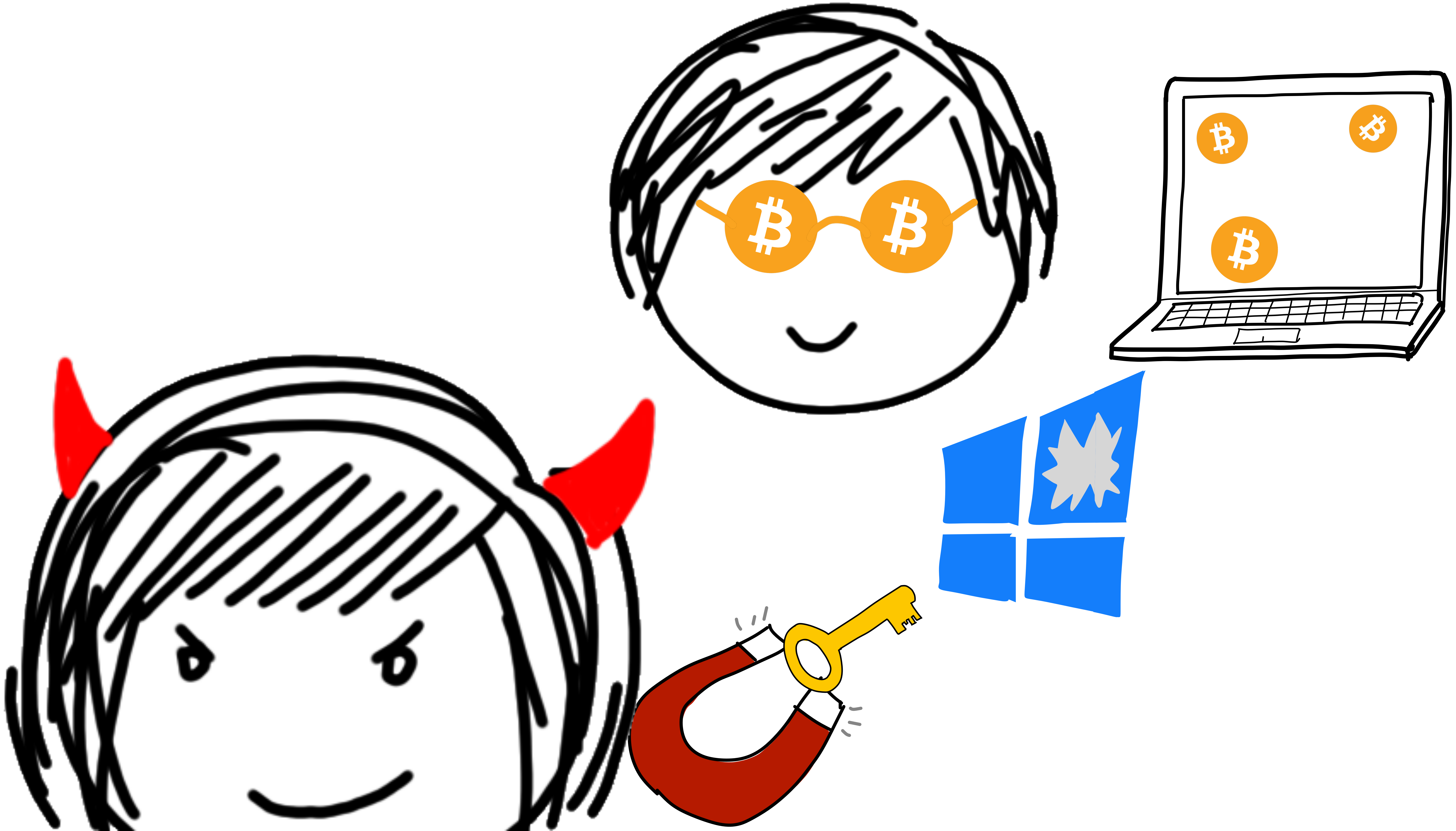
Ballad of Ƀitcoin Bob



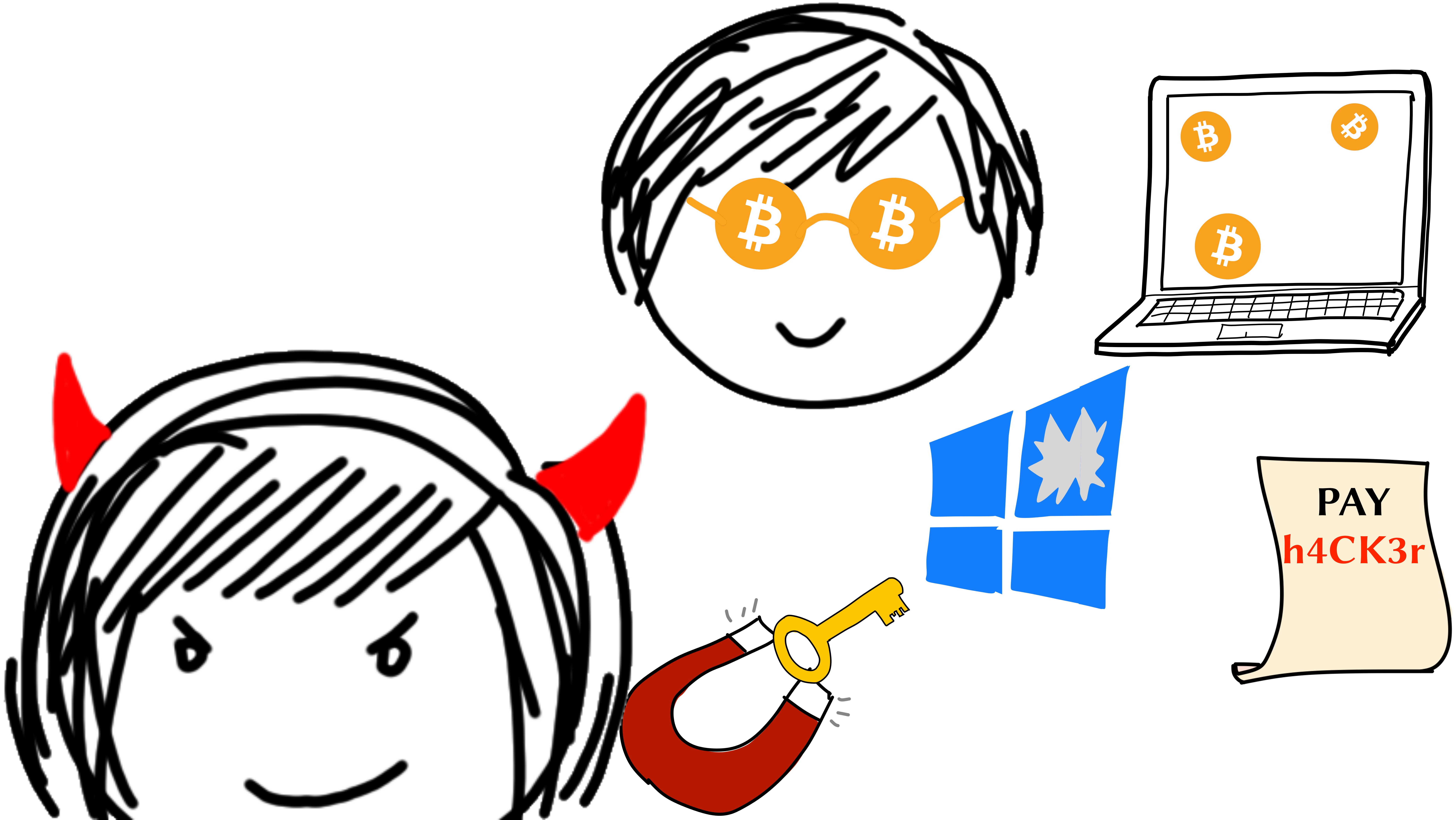
Ballad of Ƀitcoin Bob



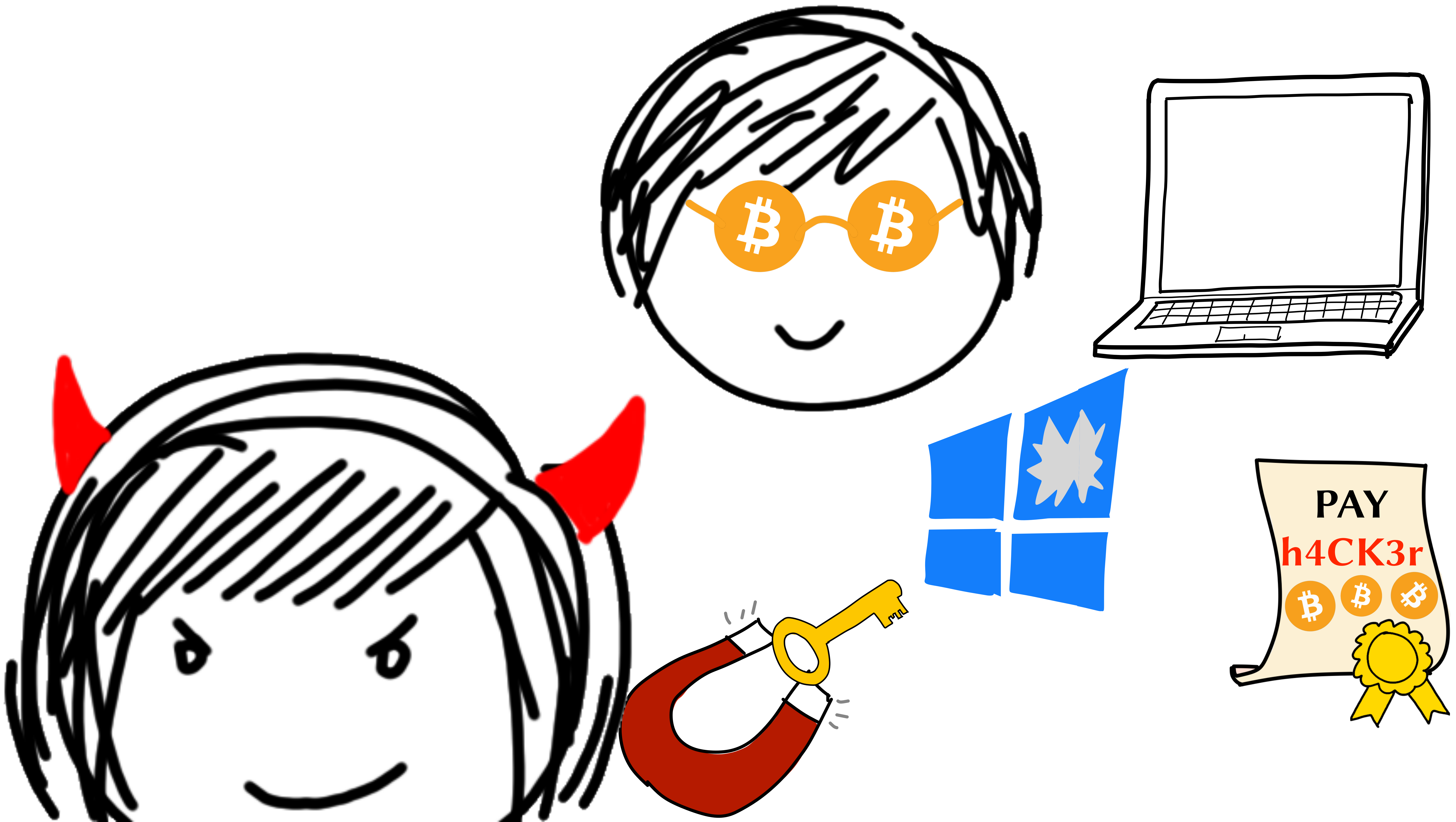
Ballad of ₿itcoin Bob



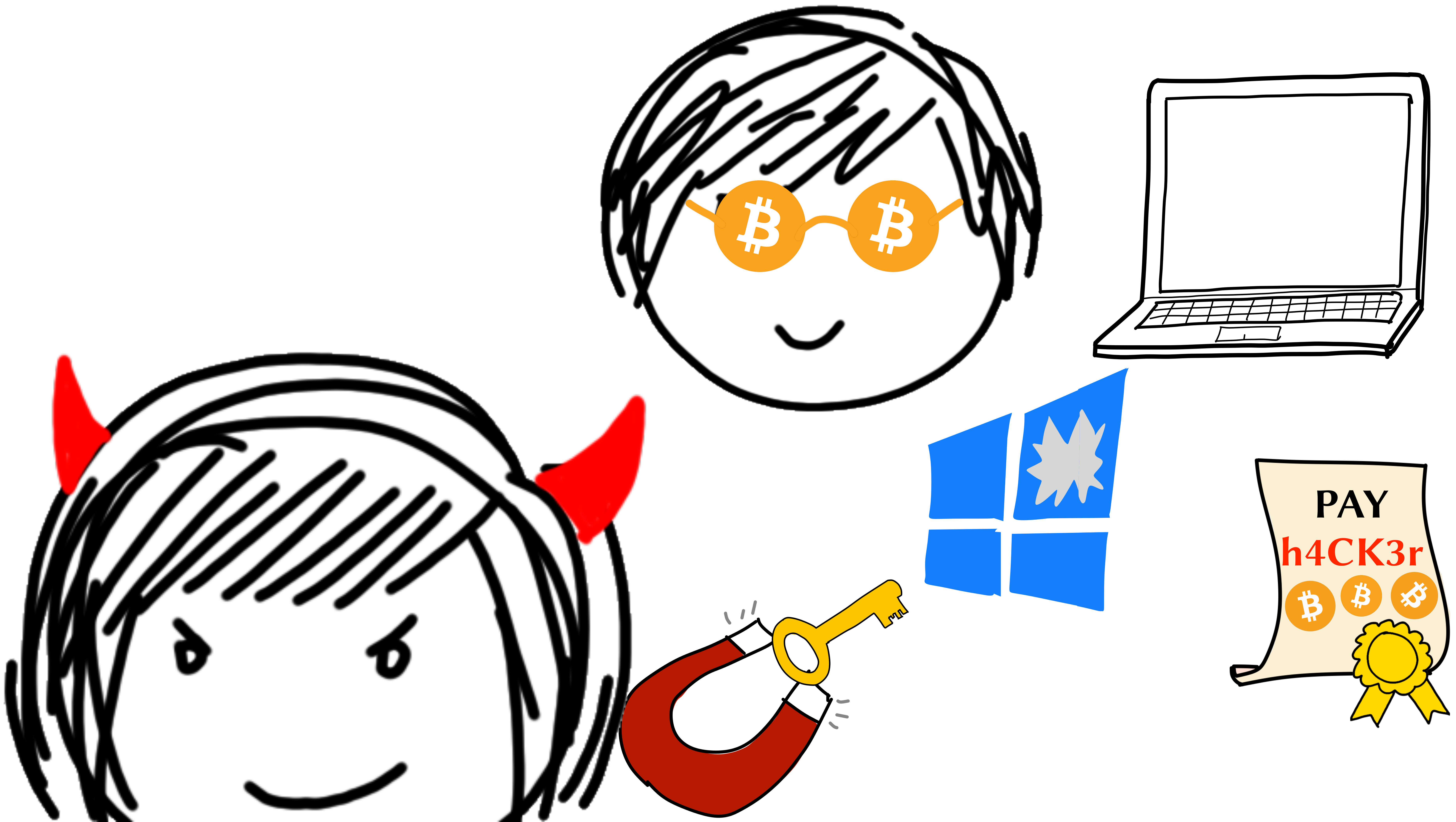
Ballad of ₿itcoin Bob



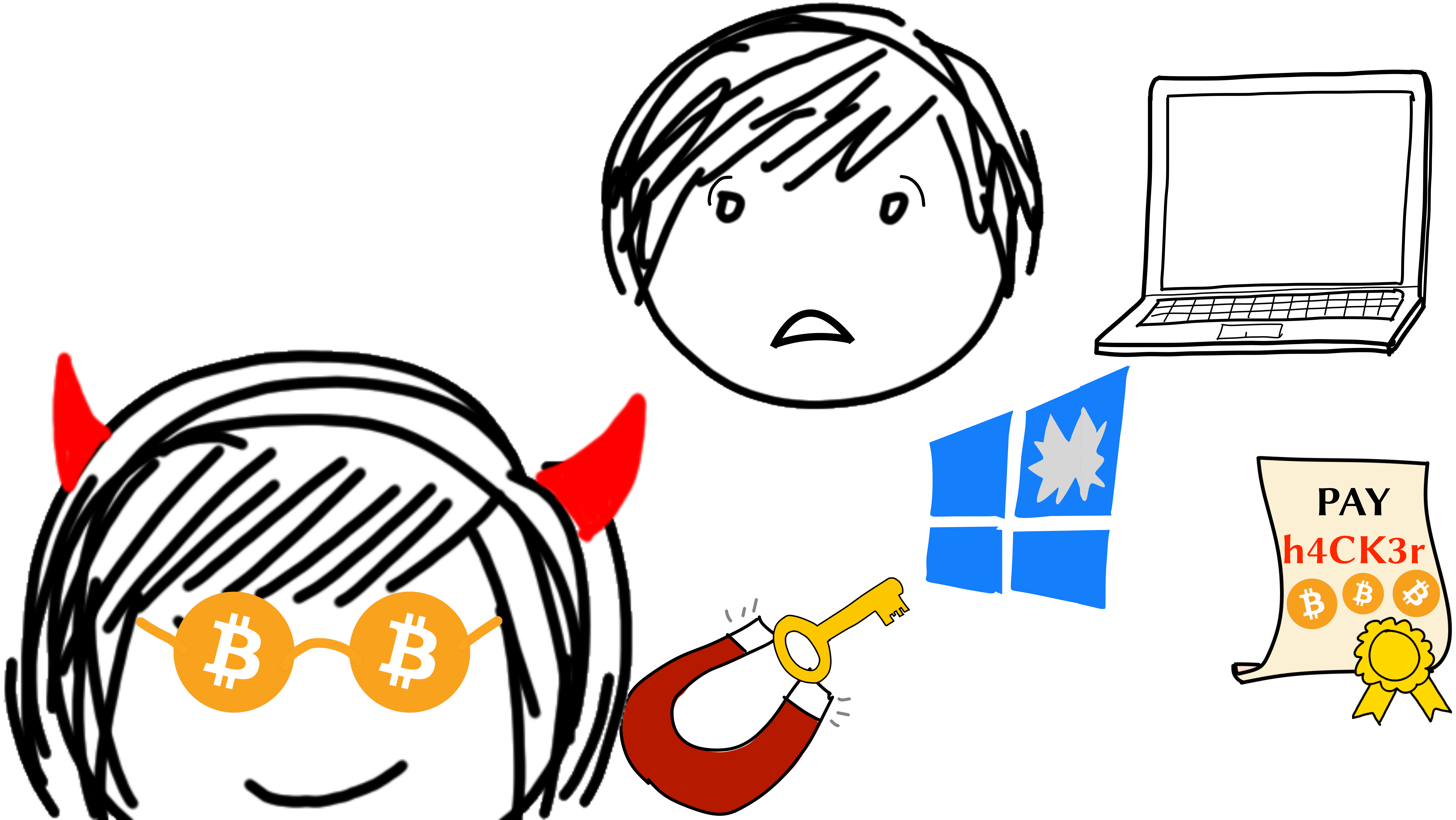
Ballad of ₿itcoin Bob



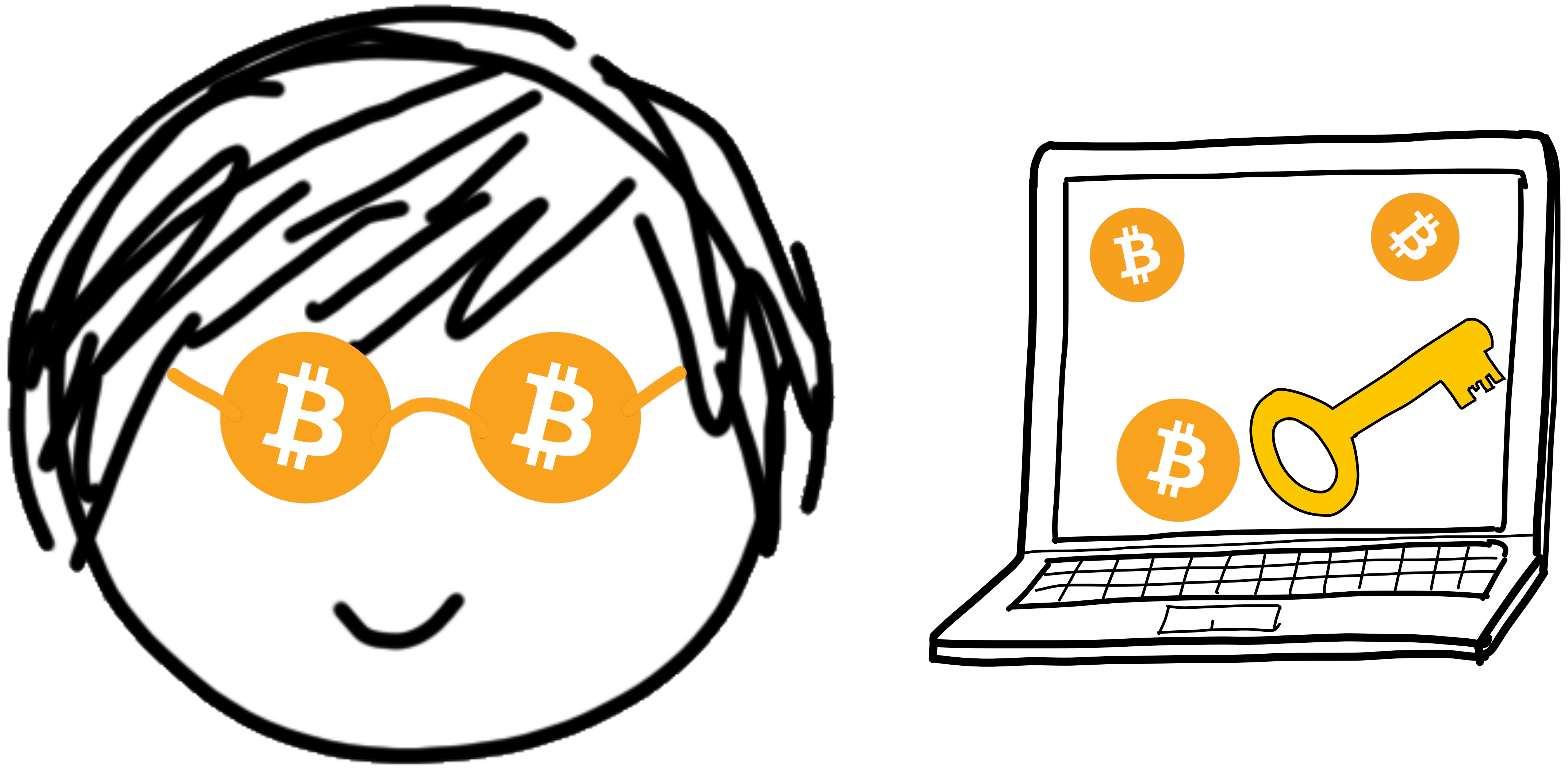
Ballad of Ƀitcoin Bob



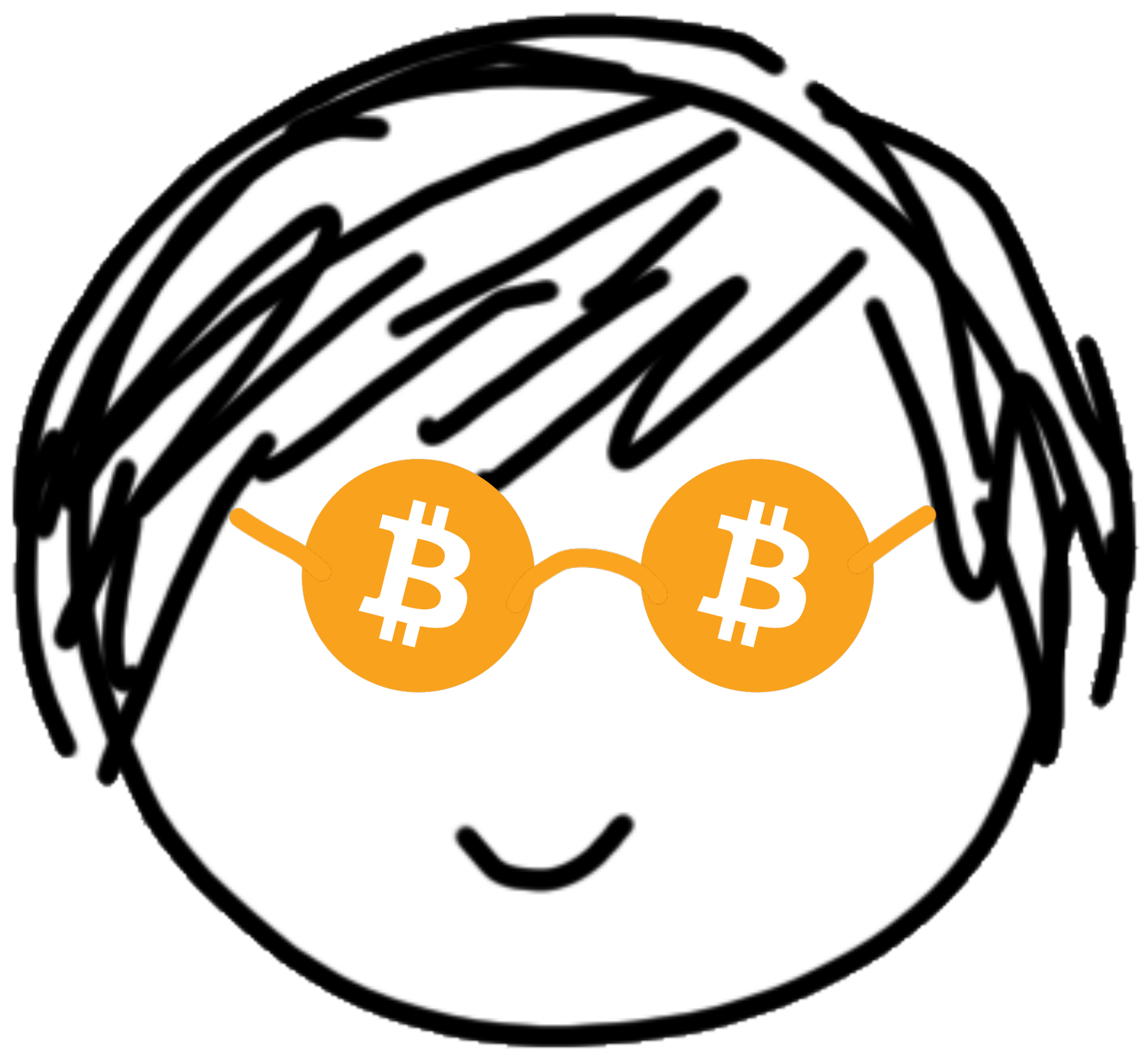
Ballad of ₿itcoin Bob



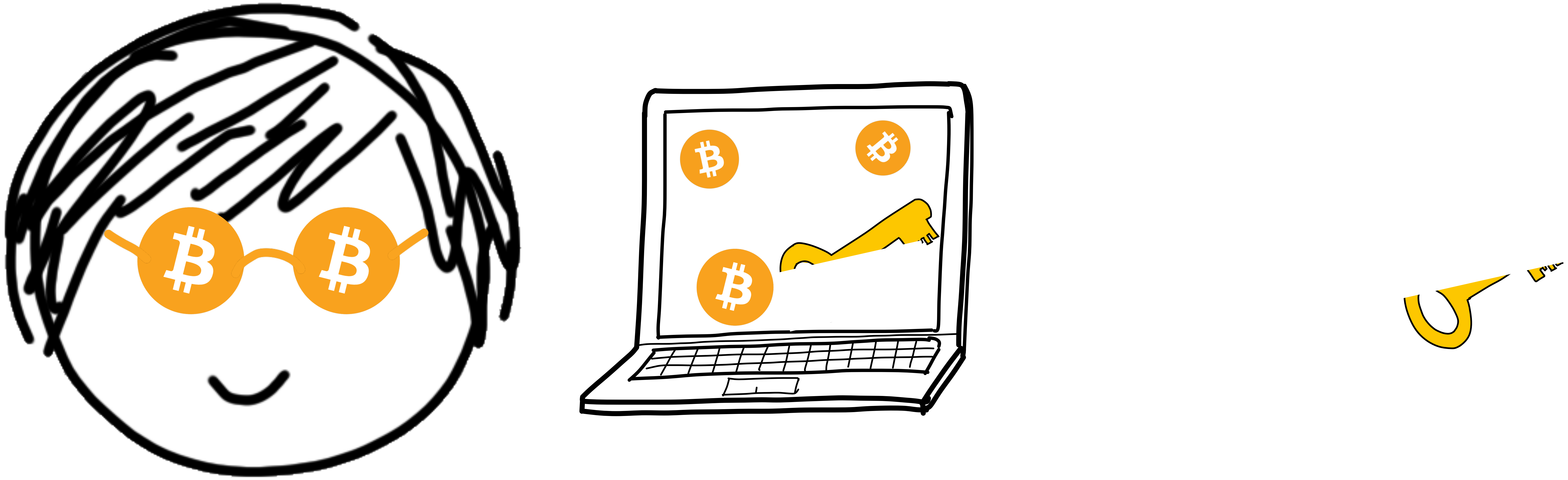
Ballad of ₿itcoin Bob



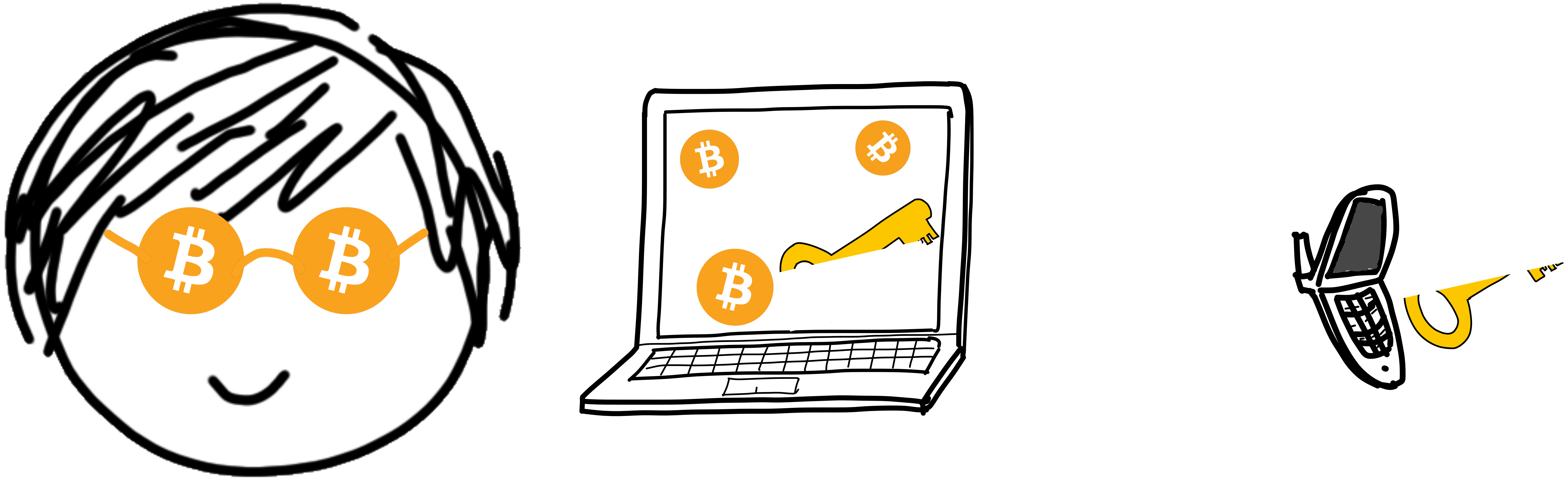
Ballad of Ƀitcoin Bob



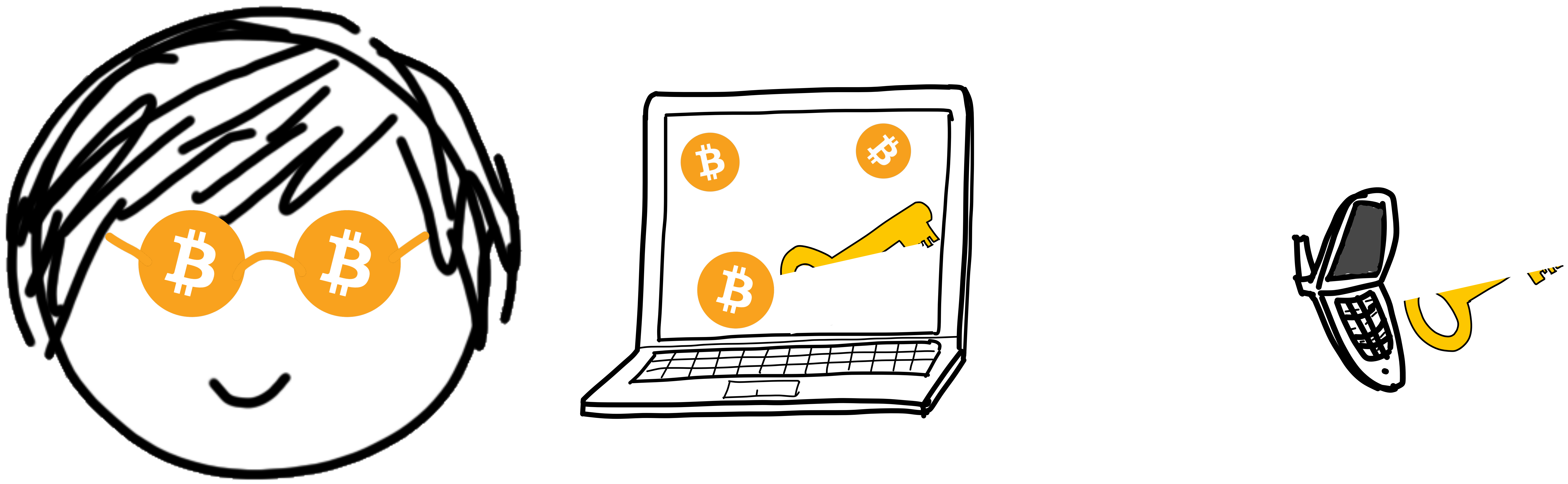
Ballad of ₿itcoin Bob



Ballad of Ƀitcoin Bob

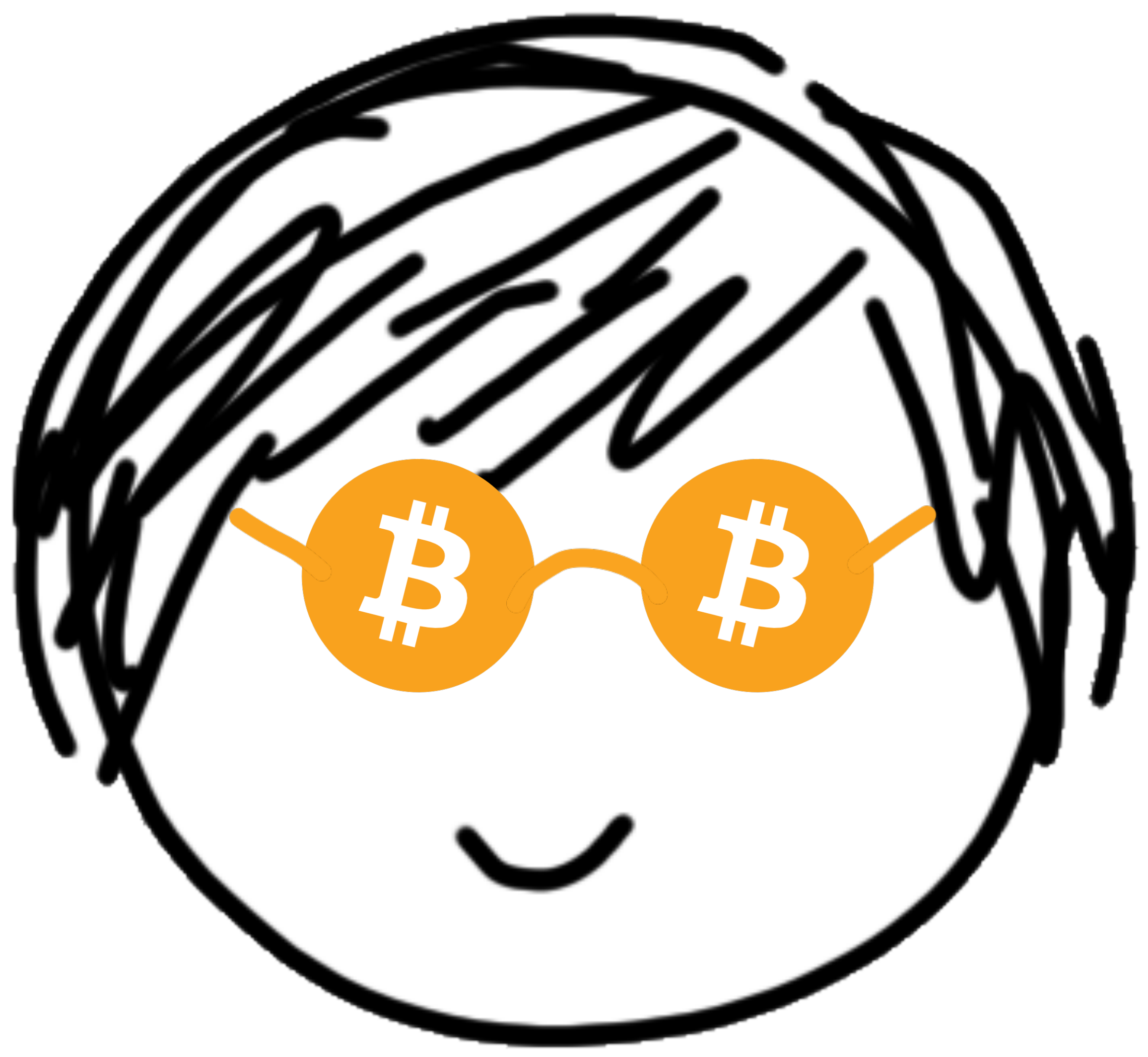


Ballad of Ƀitcoin Bob

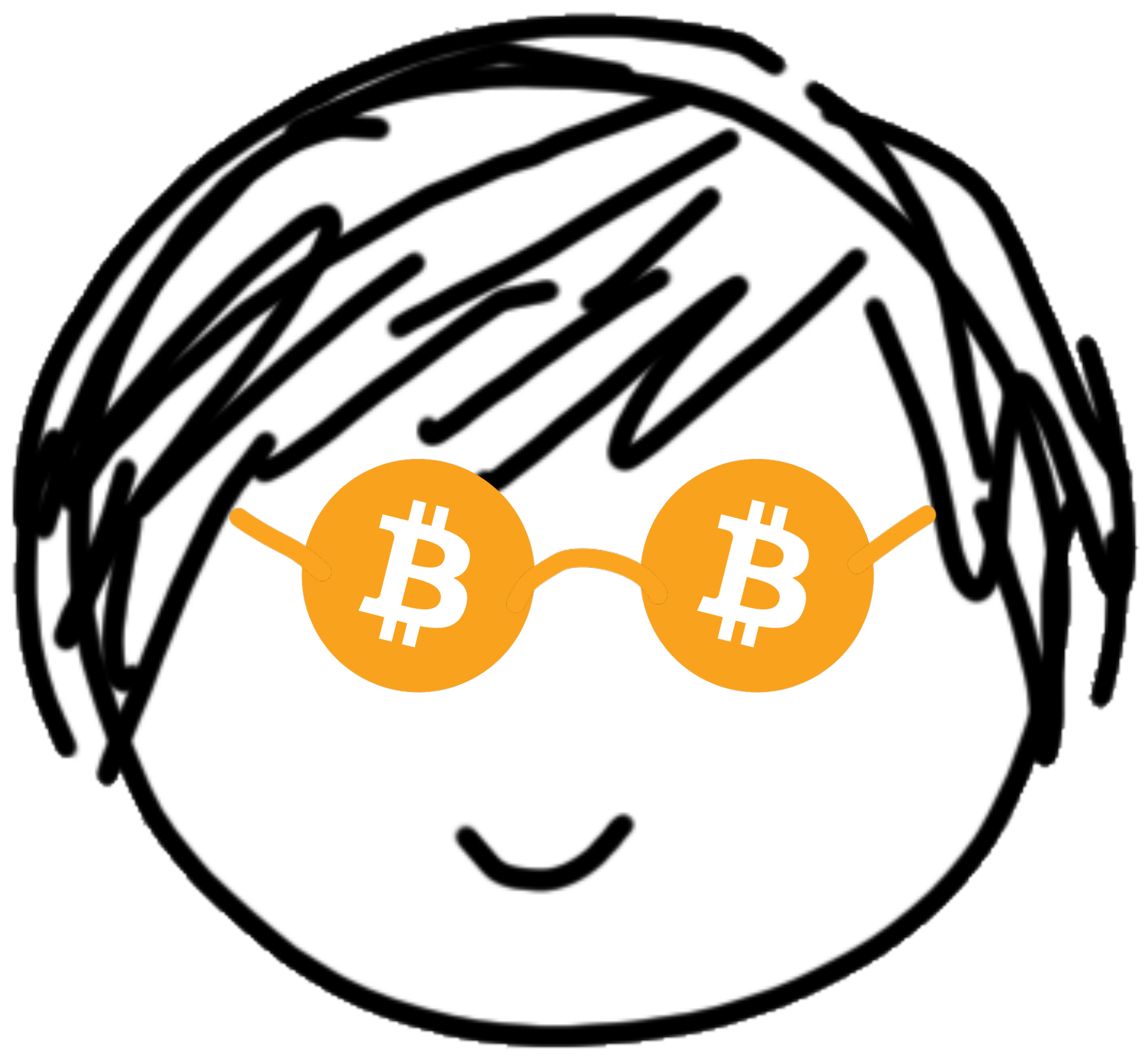


Ballad of Ƀitcoin Bob

Ballad of ₿itcoin Bob

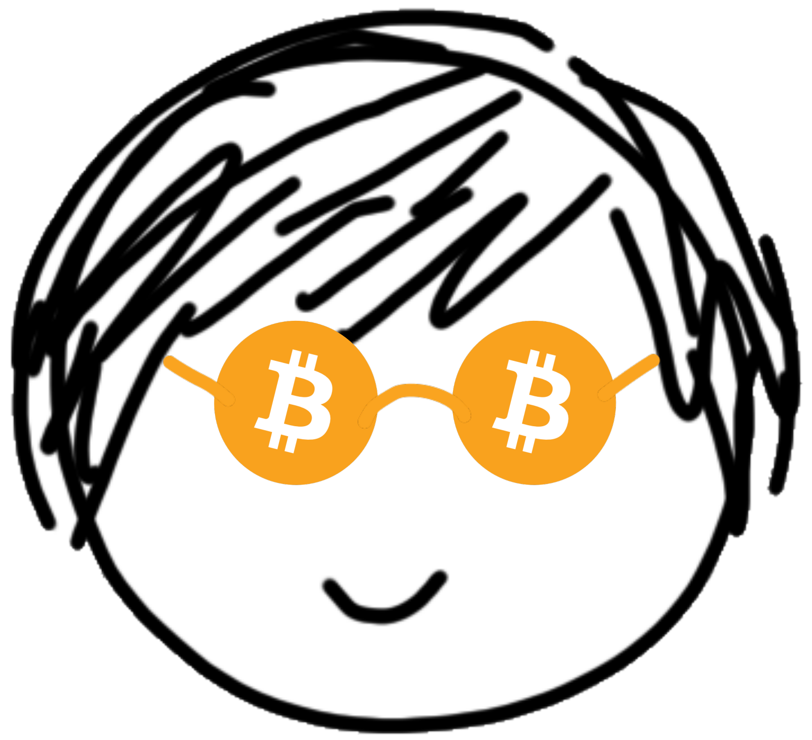


Ballad of ₿itcoin Bob



Ethereum Eysa

₿itcoin Bob

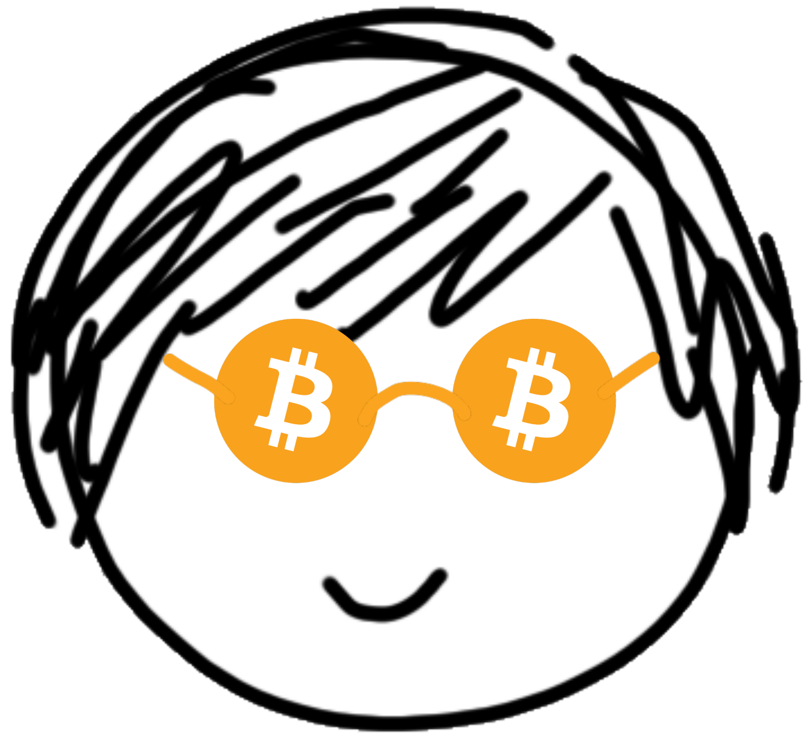


>$1T economy vulnerable to single points of failure in key management

Dogecoin Doerner

Ethereum Eysa

Ƀitcoin Bob



Large?

~~>$1T~~ **economy** vulnerable to single points of failure in key management

Dogecoin Doerner

Intro

How to distribute ECDSA

Tradeoffs

Intro

How to distribute ECDSA

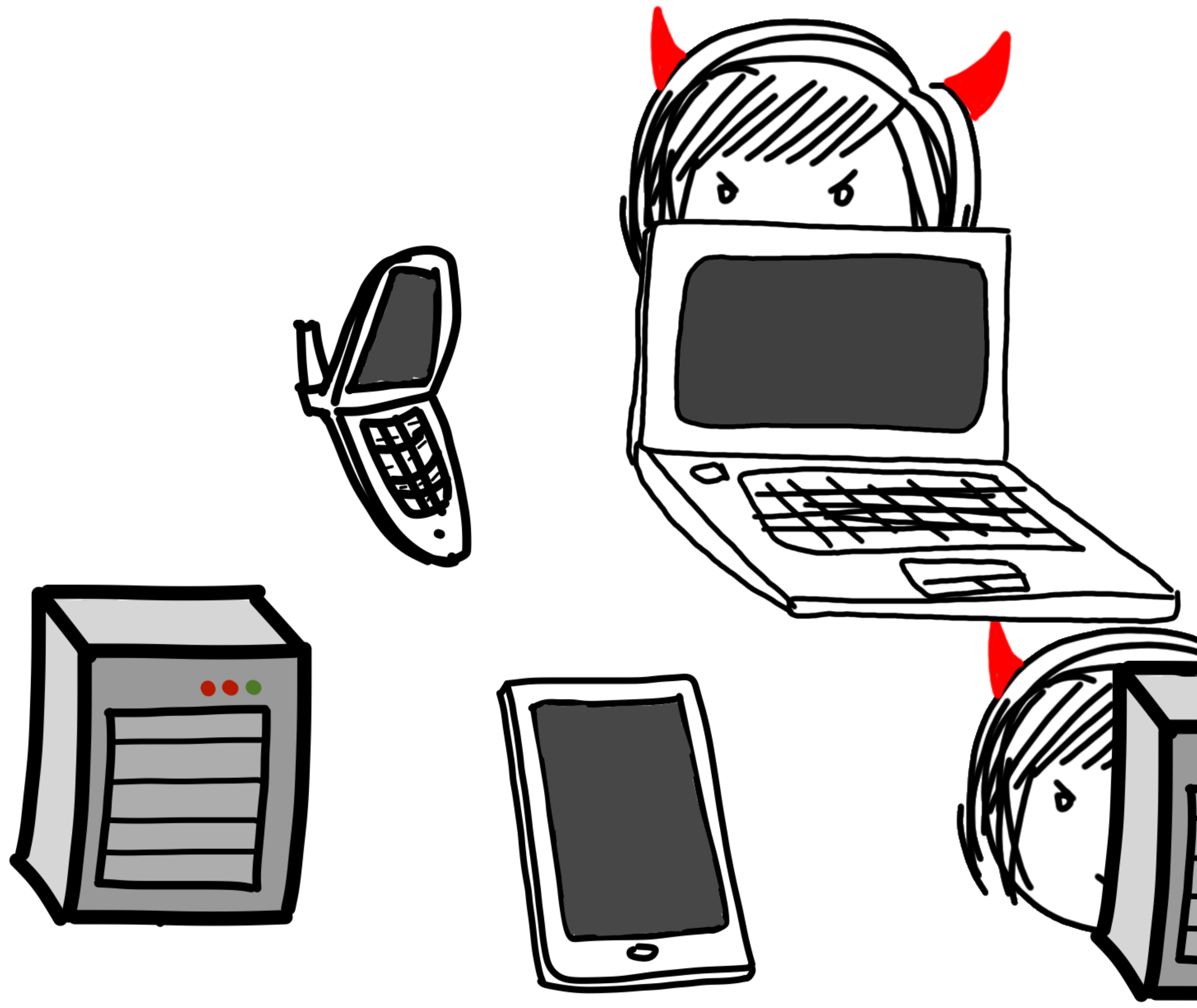
Tradeoffs

MPC for Schnorr is easy

but not ECDSA

Adversary Model

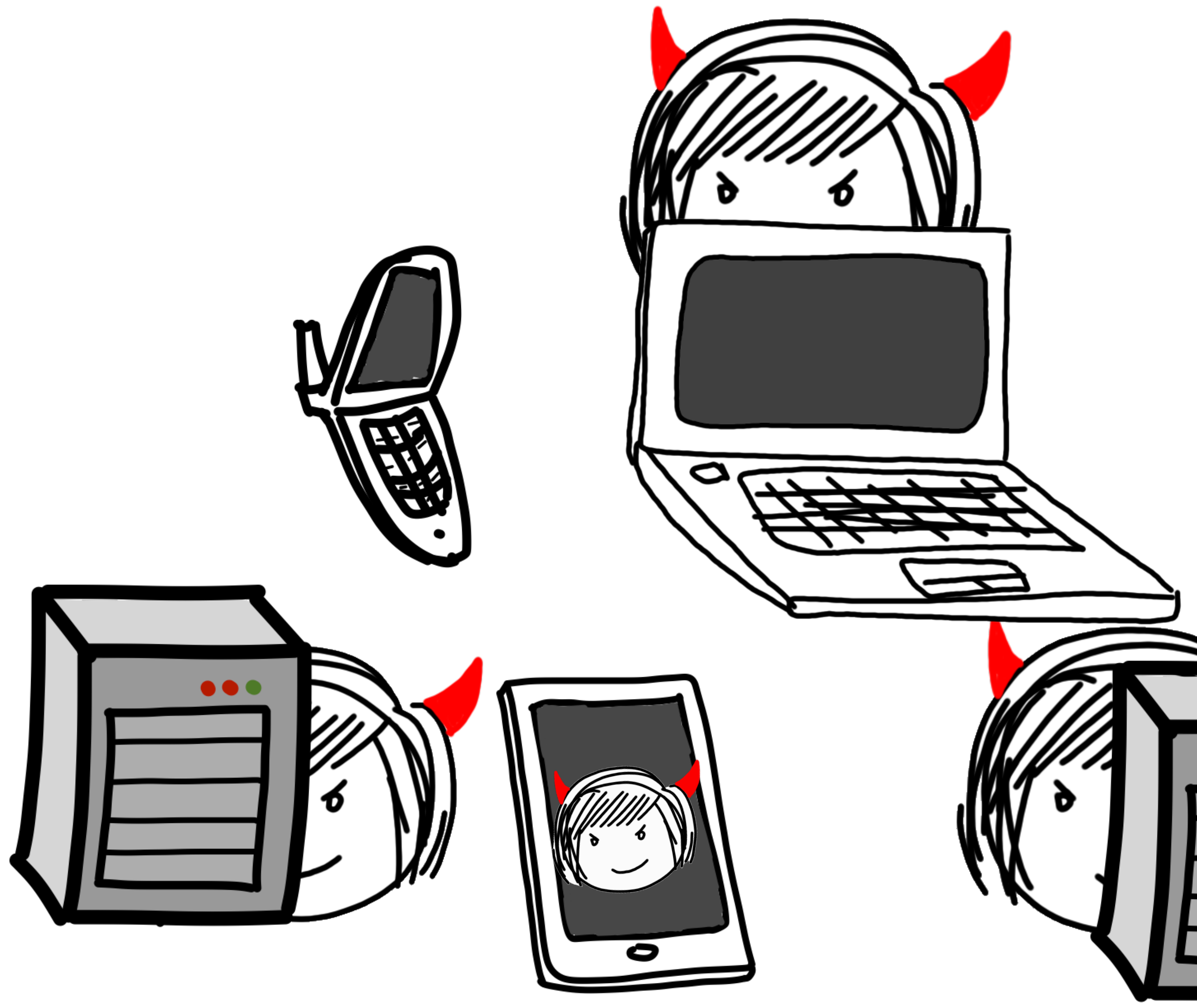
- Corruption threshold



Dishonest majority
(only one device uncompromised)

Adversary Model

- Corruption threshold



Dishonest majority
(only one device uncompromised)

Adversary Model

- Corruption threshold

Dishonest majority
(only one device uncompromised)

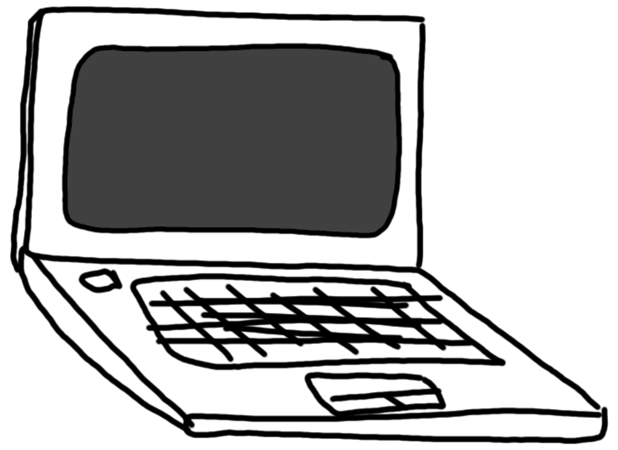
- Adversarial behaviour

Adversary Model

- Corruption threshold

Dishonest majority
(only one device uncompromised)

- Adversarial behaviour

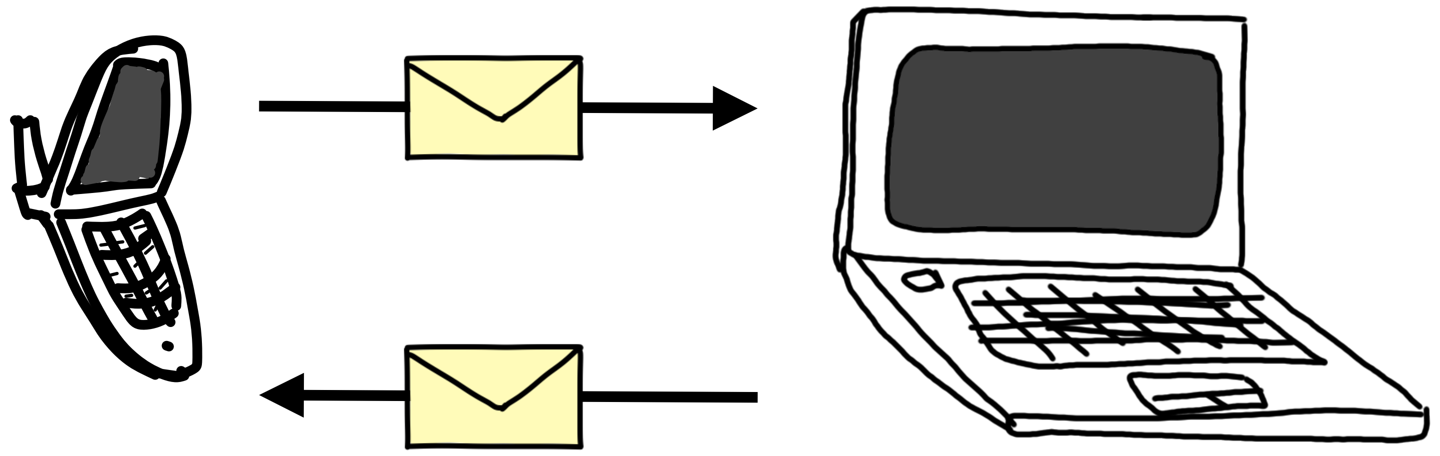


Adversary Model

- Corruption threshold

Dishonest majority
(only one device uncompromised)

- Adversarial behaviour

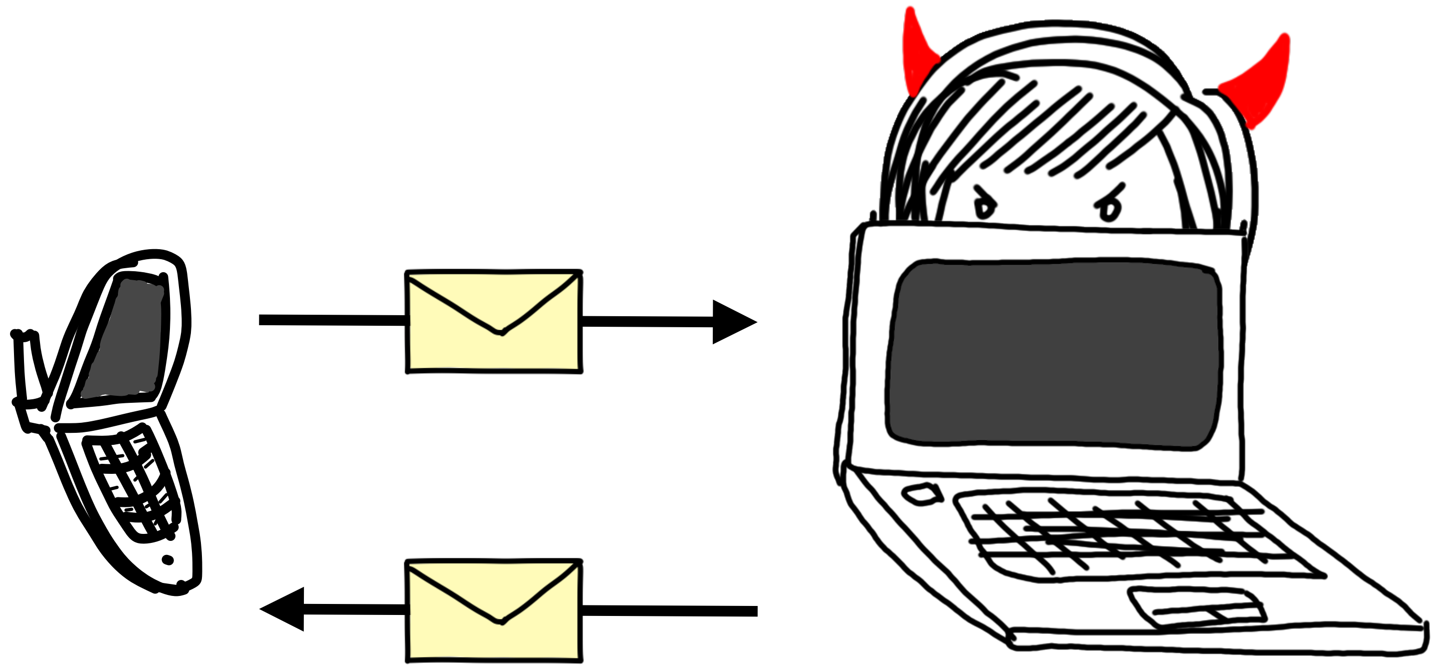


Adversary Model

- Corruption threshold

Dishonest majority
(only one device uncompromised)

- Adversarial behaviour

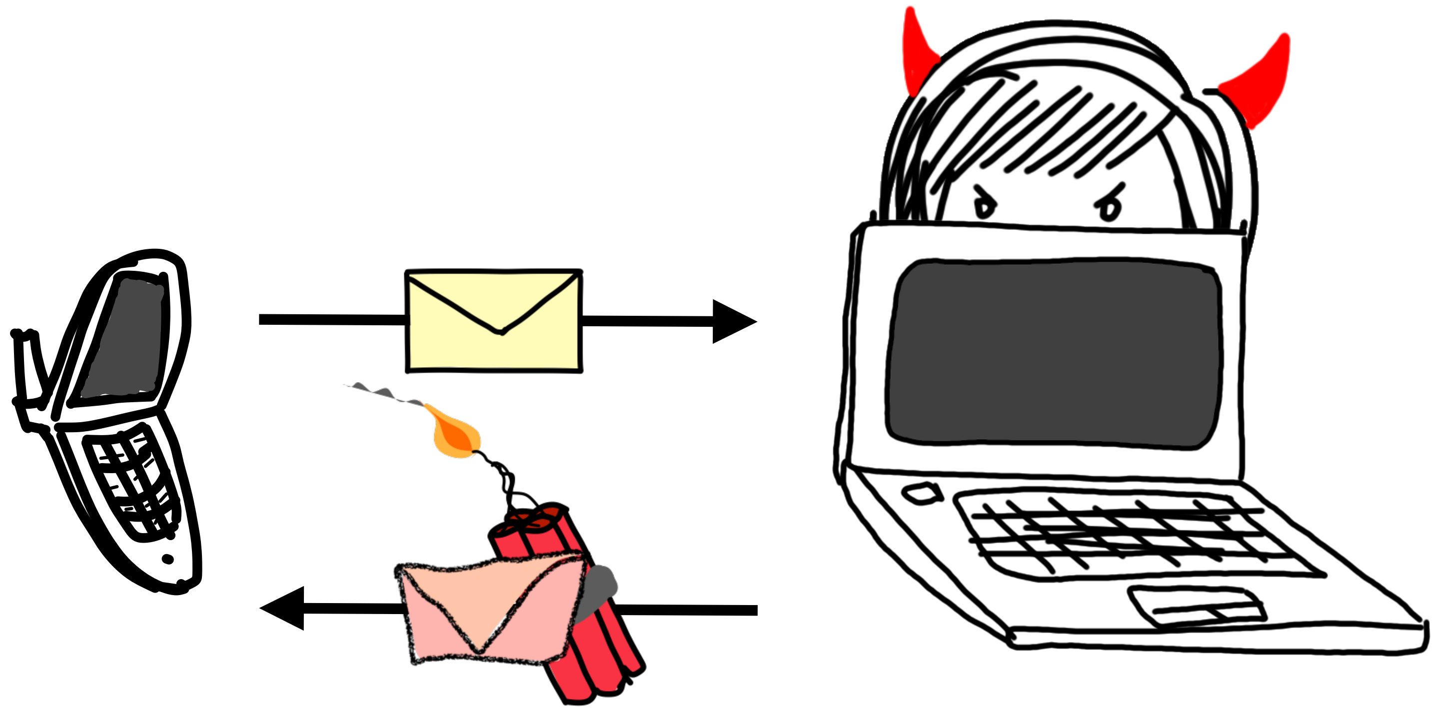


Adversary Model

- Corruption threshold

Dishonest majority
(only one device uncompromised)

- Adversarial behaviour



Malicious
(arbitrary deviations from protocol)

Concrete Example: Schnorr Signatures

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

    Group elements

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:

  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$

  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

$$(x + y) \cdot G = x \cdot G + y \cdot G$$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]
- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$$

Discrete Logarithm Problem: Given random $X \in \mathbb{G}$, find its discrete logarithm

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order $\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$$

Discrete Logarithm Problem: Given random $X \in \mathbb{G}$, find its discrete logarithm

For certain elliptic curves, best known algorithms for DLP run in time $\Theta\left(\sqrt{q}\right)$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Very informally:

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$$

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order $\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$$

Very informally: $x \to X$ ***EASY***

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Generator

(Large prime) order

Addition law

Points on an Elliptic Curve

$\approx 2^{256}$

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$$

Very informally:

$x \rightarrow X$ ***EASY***

$X \rightarrow x$ **HARD**

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Generator

(Large prime) order $\approx 2^{256}$

Addition law

Points on an Elliptic Curve

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$$

Very informally:

$x \rightarrow X$	***EASY***	$30\mu s$
$X \rightarrow x$	**HARD**	

Concrete Example: Schnorr Signatures

- Elegant signature scheme based on the Discrete Logarithm problem [Schnorr 89]

- Tools:
  - Hash function - $H : \{0,1\}^* \to \mathbb{Z}_q$
  - Group - $(\mathbb{G}, G, q, +)$

Group elements

Points on an Elliptic Curve

Generator

(Large prime) order

$\approx 2^{256}$

Addition law

Sixty Seconds on Cyclic Groups

If $X, Y \in \mathbb{G}$ then $X + Y = Z \in \mathbb{G}$

Any $X \in \mathbb{G}$ can be written as $x \cdot G$

$x \in \mathbb{Z}_q$ is the ***discrete logarithm*** of X

Integer addition mod q Group addition

$(x \boxed{+} y) \cdot G = x \cdot G \boxed{+} y \cdot G$

Very informally:

$x \to X$	*EASY*	$30\mu s$
$X \to x$	HARD	Many billion billions of years

Schnorr Key Generation

$$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$$
$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$
$$\mathsf{PK} = \mathsf{sk} \cdot G$$
$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

<u>s</u>ecret <u>k</u>ey: kept private

<u>P</u>ublic <u>K</u>ey: exposed to the outside world

Schnorr Signing

$\mathsf{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\mathsf{SchnorrSign}(\mathsf{sk}, m):$

Schnorr Signing

$\mathsf{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\mathsf{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

Schnorr Signing

$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\text{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

Schnorr Signing

$\mathsf{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\mathsf{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$\boxed{R} = k \cdot G$$

NONCE
One-time use value

Schnorr Signing

$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\mathrm{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$\boxed{R} = k \cdot G$$

NONCE
One-time use
value

$$e = H(R \| m)$$

Schnorr Signing

$$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$$
$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$
$$\mathsf{PK} = \mathsf{sk} \cdot G$$
$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$$\mathrm{SchnorrSign}(\mathsf{sk}, m):$$
$$k \leftarrow \mathbb{Z}_q$$
$$\boxed{R} = k \cdot G$$
$$e = H(R \| m)$$
$$s = k - \mathsf{sk} \cdot e \pmod q$$

NONCE
One-time use value

Schnorr Signing

$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\text{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$\boxed{R} = k \cdot G$$

NONCE
One-time use value

$$e = H(R \| m)$$

$$s = k - \mathsf{sk} \cdot e \pmod{q}$$

$$\sigma = (s, R)$$

$$\text{output } \sigma$$

Schnorr Signing

$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\text{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$\boxed{R} = k \cdot G$$

NONCE
One-time use value

$$e = H(R \| m)$$

$$\boxed{s = k - \mathsf{sk} \cdot e} \pmod q$$

$$\sigma = (s, R)$$

$$\text{output } \sigma$$

Verifying a signature: $\boxed{s \cdot G \stackrel{?}{=} R - e \cdot \mathsf{PK}}$

Schnorr Signing

$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\mathsf{sk} \leftarrow \mathbb{Z}_q$

$\mathsf{PK} = \mathsf{sk} \cdot G$

$\text{output } (\mathsf{sk}, \mathsf{PK})$

$\text{SchnorrSign}(\mathsf{sk}, m):$

$k \leftarrow \mathbb{Z}_q$

$\boxed{R} = k \cdot G$

NONCE
One-time use value

$e = H(R \| m)$

$\boxed{s = k - \mathsf{sk} \cdot e} \pmod q$

$\sigma = (s, R)$

$\text{output } \sigma$

Verifying a signature: $s \cdot G \stackrel{?}{=} \underbrace{\boxed{R}}_{k \cdot G} - e \cdot \underbrace{\boxed{\mathsf{PK}}}_{\mathsf{sk} \cdot G}$

Schnorr Signing

$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

$\mathrm{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

NONCE
One-time use value

$$e = H(R \| m)$$

$$s = k - \mathsf{sk} \cdot e \pmod{q}$$

$$\sigma = (s, R)$$

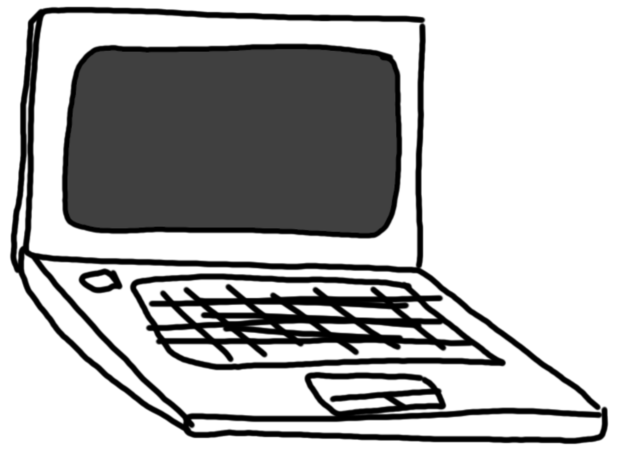
$$\text{output } \sigma$$

Verifying a signature: $s \cdot G \stackrel{?}{=} \boxed{R} - e \cdot \boxed{\mathsf{PK}}$

$$\boxed{k \cdot G} \qquad \boxed{\mathsf{sk} \cdot G}$$

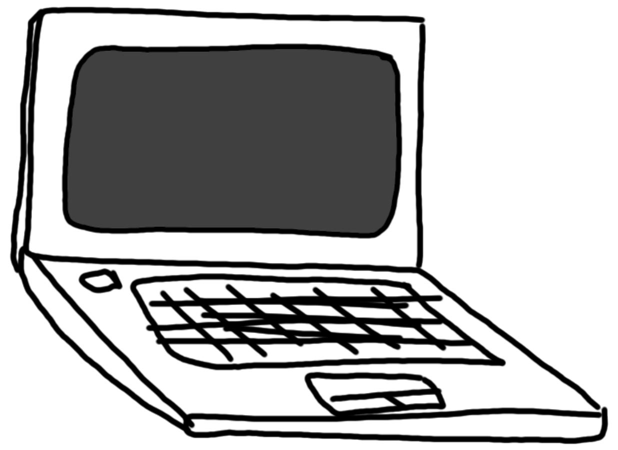
$$s \cdot G \stackrel{?}{=} (k - e \cdot \mathsf{sk}) \cdot G$$

Additive Secret Sharing



Additive Secret Sharing

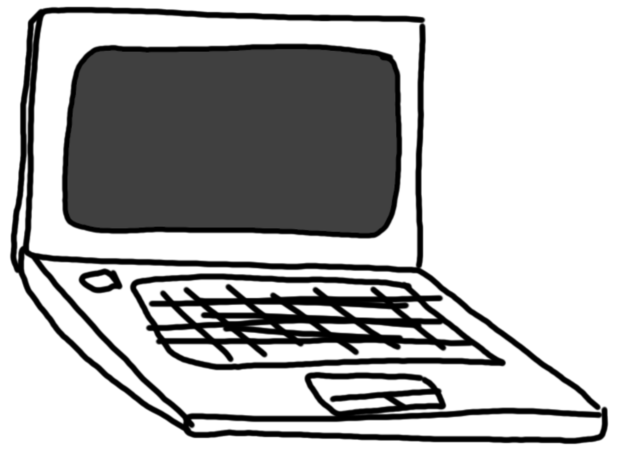
$$x \in \mathbb{Z}_q$$



Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

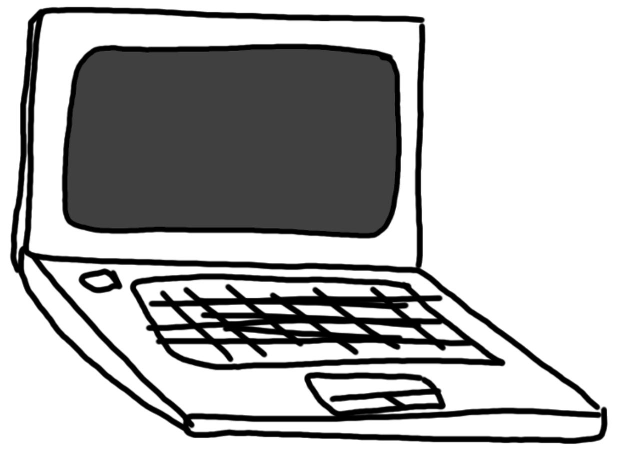
$$x_A + x_B = x$$



Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$



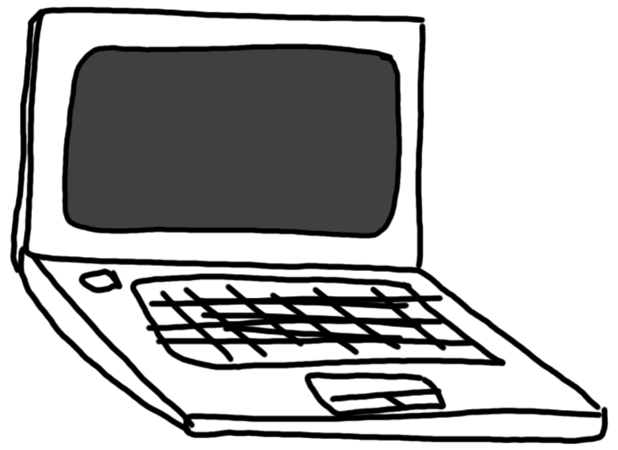
x_A

x_B

Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$

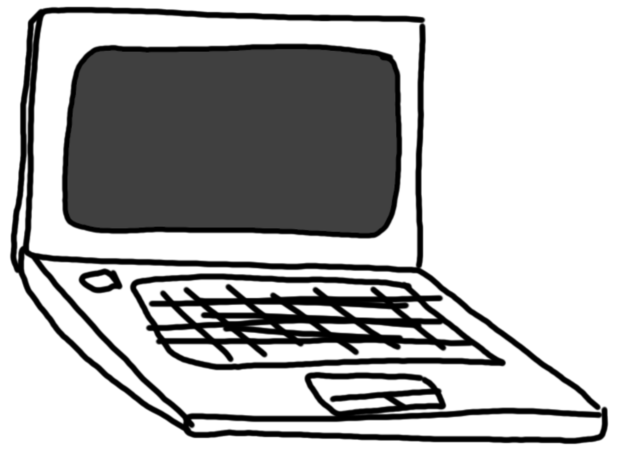


x_A $[x]$ x_B

Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$



x_A

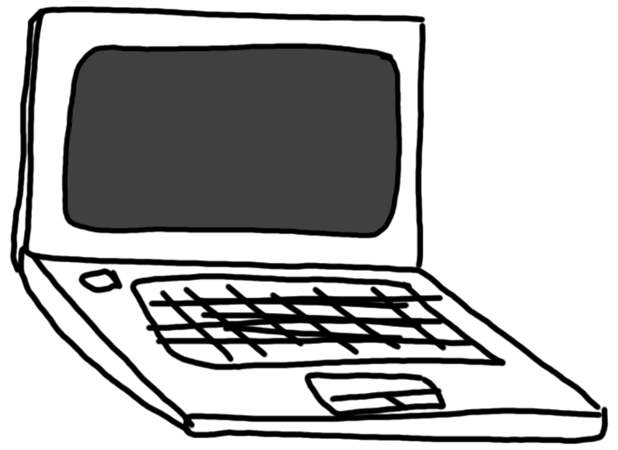
$[x]$

x_B

Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$

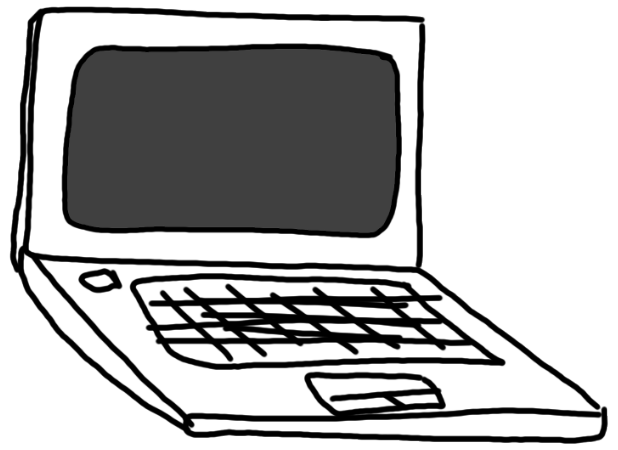


x_A $[x]$ x_B

Additive Secret Sharing

$$\boxed{?} \in \mathbb{Z}_q$$

$$x_A + \boxed{?} = \boxed{?}$$

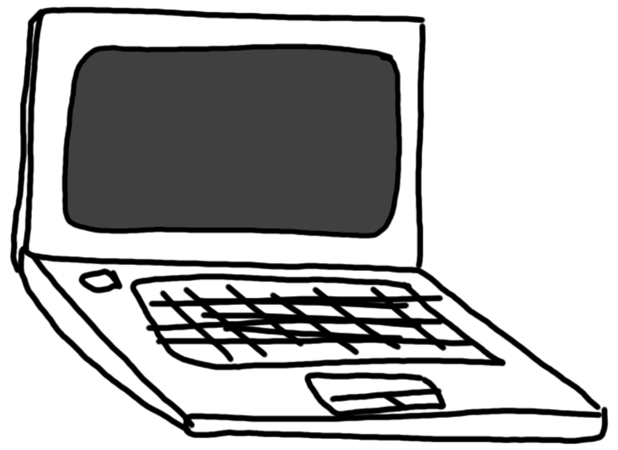


x_A $[\boxed{?}]$ $\boxed{?}$

Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$



x_A

$$[x]$$

x_B

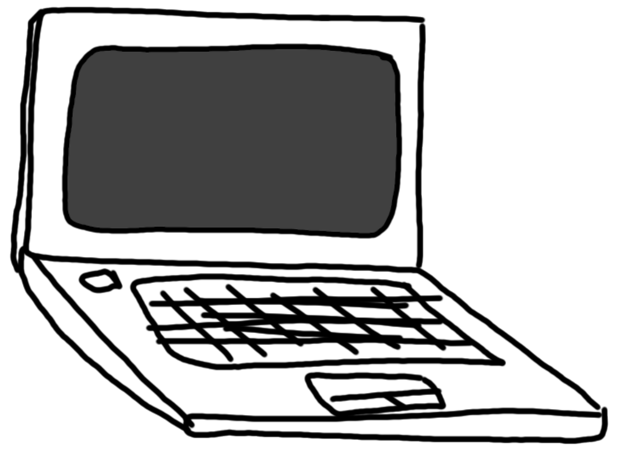
$$[y]$$

$$y_A + y_B = y$$

Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$



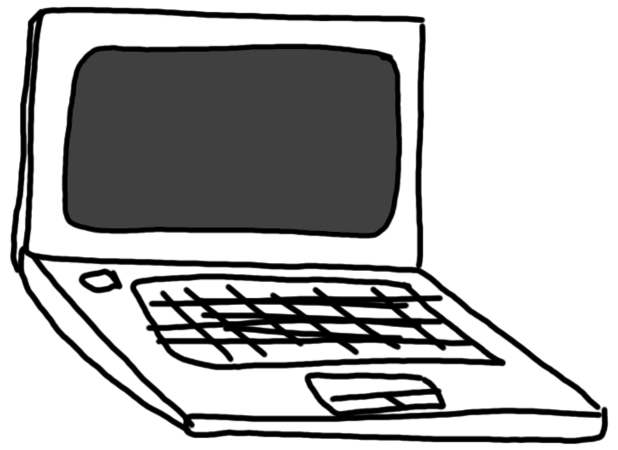
x_A $[x]$ x_B

y_A $[y]$ y_B

Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$



x_A $[x]$ x_B

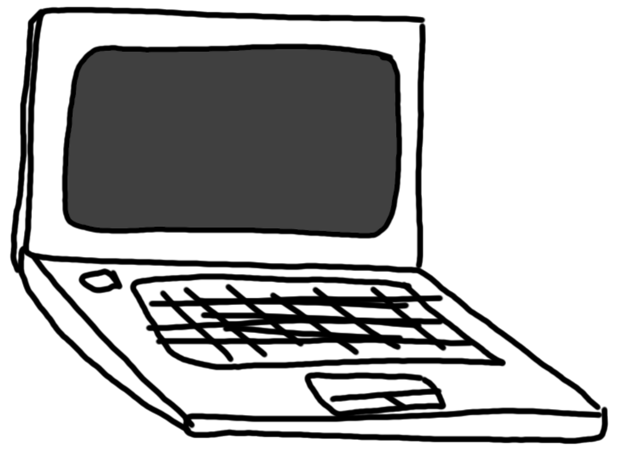
y_A $[y]$ y_B

$$[z = cx + y]$$

Additive Secret Sharing

$$x \in \mathbb{Z}_q$$

$$x_A + x_B = x$$



x_A	$[x]$	x_B
y_A	$[y]$	y_B
$z_A = cx_A + y_A$	$[z = cx + y]$	$z_B = cx_B + y_B$

Schnorr Key Generation

$$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$$

$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$

$$\mathsf{PK} = \mathsf{sk} \cdot G$$

$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

<u>s</u>ecret <u>k</u>ey: kept private

<u>P</u>ublic <u>K</u>ey: exposed to the outside world

Distributing Schnorr KeyGen

$$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$$
$$\mathsf{sk} \leftarrow \mathbb{Z}_q$$
$$\mathsf{PK} = \mathsf{sk} \cdot G$$
$$\text{output } (\mathsf{sk}, \mathsf{PK})$$

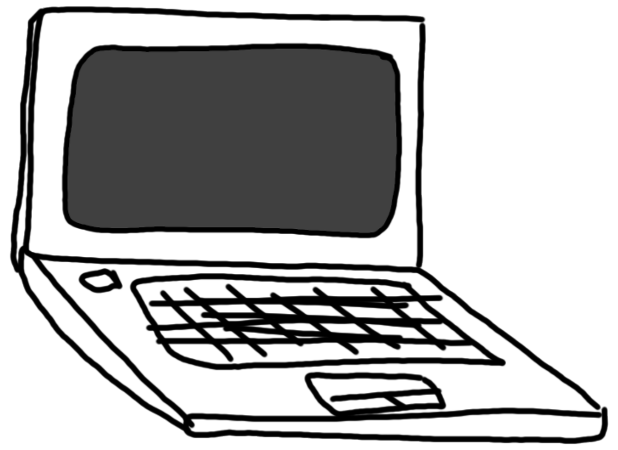
Distributing Schnorr KeyGen

$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$

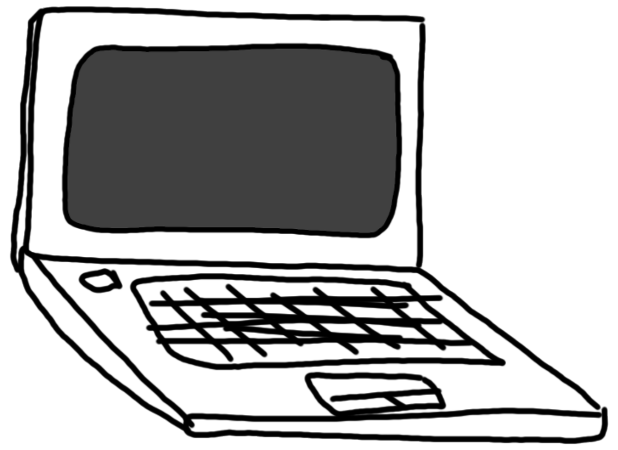
$\mathsf{sk} \leftarrow \mathbb{Z}_q$

$\mathsf{PK} = \mathsf{sk} \cdot G$

output $(\mathsf{sk}, \mathsf{PK})$



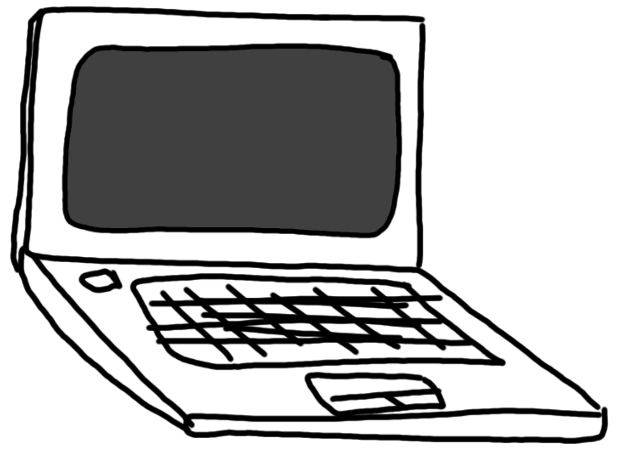
Distributing Schnorr KeyGen



$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\mathsf{sk} \leftarrow \mathbb{Z}_q$	$\mathsf{sk}_\mathsf{A} \leftarrow \mathbb{Z}_q$	$\mathsf{sk}_\mathsf{B} \leftarrow \mathbb{Z}_q$
$\mathsf{PK} = \mathsf{sk} \cdot G$		
$\text{output } (\mathsf{sk}, \mathsf{PK})$		

Distributing Schnorr KeyGen

		
$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$		
$\mathsf{sk} \leftarrow \mathbb{Z}_q$	$\mathsf{sk}_A \leftarrow \mathbb{Z}_q$	$\mathsf{sk}_B \leftarrow \mathbb{Z}_q$
$\mathsf{PK} = \mathsf{sk} \cdot G$	$\mathsf{PK}_A = \mathsf{sk}_A \cdot G$	$\mathsf{PK}_B = \mathsf{sk}_B \cdot G$
output $(\mathsf{sk}, \mathsf{PK})$		

Distributing Schnorr KeyGen

$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\mathsf{sk} \leftarrow \mathbb{Z}_q$

$\mathsf{PK} = \mathsf{sk} \cdot G$

output $(\mathsf{sk}, \mathsf{PK})$

$\mathsf{sk}_\mathsf{A} \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_\mathsf{A} = \mathsf{sk}_\mathsf{A} \cdot G$

output $(\mathsf{sk}_\mathsf{A}, \mathsf{PK}_\mathsf{A})$

$\mathsf{sk}_\mathsf{B} \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_\mathsf{B} = \mathsf{sk}_\mathsf{B} \cdot G$

output $(\mathsf{sk}_\mathsf{B}, \mathsf{PK}_\mathsf{B})$

Distributing Schnorr KeyGen

$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\mathsf{sk} \leftarrow \mathbb{Z}_q$

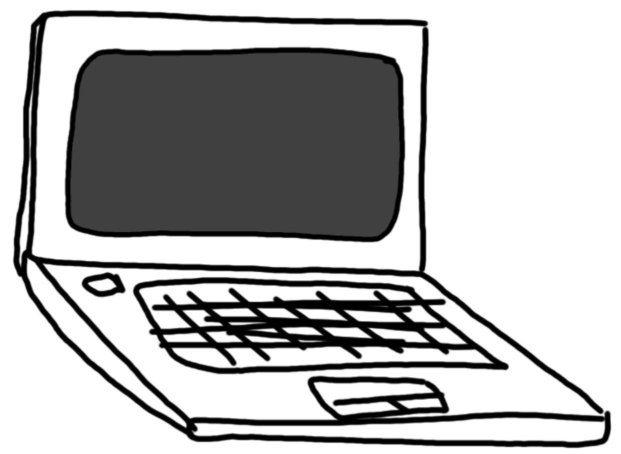
$\mathsf{PK} = \mathsf{sk} \cdot G$

output $(\mathsf{sk}, \mathsf{PK})$

$\mathsf{sk}_\mathsf{A} \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_\mathsf{A} = \mathsf{sk}_\mathsf{A} \cdot G$

output $(\mathsf{sk}_\mathsf{A}, \mathsf{PK}_\mathsf{A})$



$\mathsf{sk}_\mathsf{B} \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_\mathsf{B} = \mathsf{sk}_\mathsf{B} \cdot G$

output $(\mathsf{sk}_\mathsf{B}, \mathsf{PK}_\mathsf{B})$

How are the values related?

$$\mathsf{sk} = \mathsf{sk}_\mathsf{A} + \mathsf{sk}_\mathsf{B}$$

$$\mathsf{PK} = \mathsf{PK}_\mathsf{A} + \mathsf{PK}_\mathsf{B}$$

Distributing Schnorr KeyGen

$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\mathsf{sk} \leftarrow \mathbb{Z}_q$

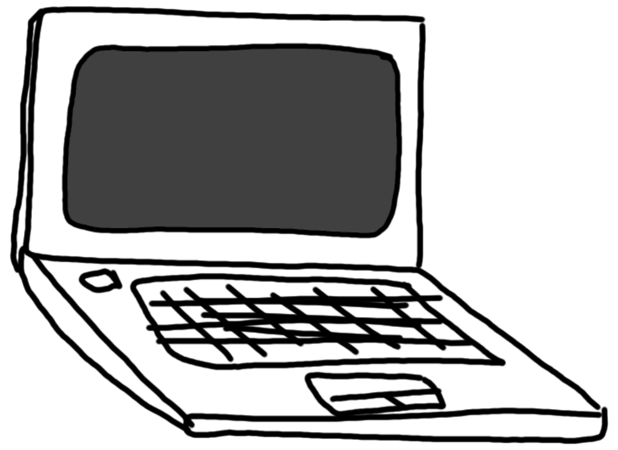
$\mathsf{PK} = \mathsf{sk} \cdot G$

$\text{output } (\mathsf{sk}, \mathsf{PK})$

$\mathsf{sk_A} \leftarrow \mathbb{Z}_q$

$\mathsf{PK_A} = \mathsf{sk_A} \cdot G$

$\text{output } (\mathsf{sk_A}, \mathsf{PK_A})$



$\mathsf{sk_B} \leftarrow \mathbb{Z}_q$

$\mathsf{PK_B} = \mathsf{sk_B} \cdot G$

$\text{output } (\mathsf{sk_B}, \mathsf{PK_B})$

How are the values related?

$\mathsf{sk} = \mathsf{sk_A} + \mathsf{sk_B}$

$\mathsf{PK} = \mathsf{PK_A} + \mathsf{PK_B}$

Distributing Schnorr KeyGen

$\text{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\mathsf{sk} \leftarrow \mathbb{Z}_q$

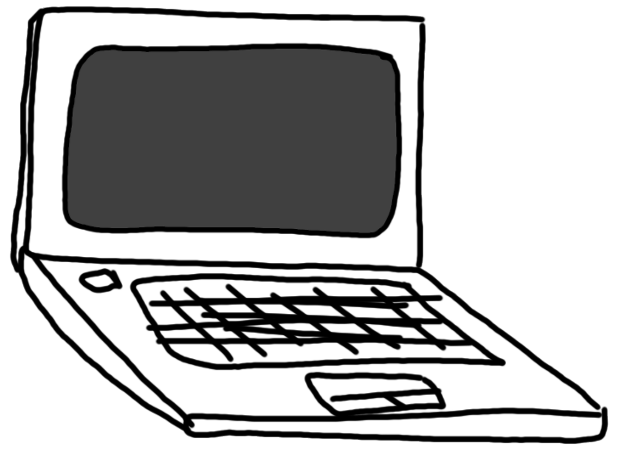
$\mathsf{PK} = \mathsf{sk} \cdot G$

output $(\mathsf{sk}, \mathsf{PK})$

$\mathsf{sk}_\mathsf{A} \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_\mathsf{A} = \mathsf{sk}_\mathsf{A} \cdot G$

output $(\mathsf{sk}_\mathsf{A}, \mathsf{PK}_\mathsf{A})$



$\mathsf{sk}_\mathsf{B} \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_\mathsf{B} = \mathsf{sk}_\mathsf{B} \cdot G$

output $(\mathsf{sk}_\mathsf{B}, \mathsf{PK}_\mathsf{B})$

How are the values related?

$\mathsf{sk} = \mathsf{sk}_\mathsf{A} + \mathsf{sk}_\mathsf{B}$

$\mathsf{PK} = \mathsf{PK}_\mathsf{A} + \mathsf{PK}_\mathsf{B}$

Distributing Schnorr KeyGen

$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\boxed{?} \leftarrow \mathbb{Z}_q$

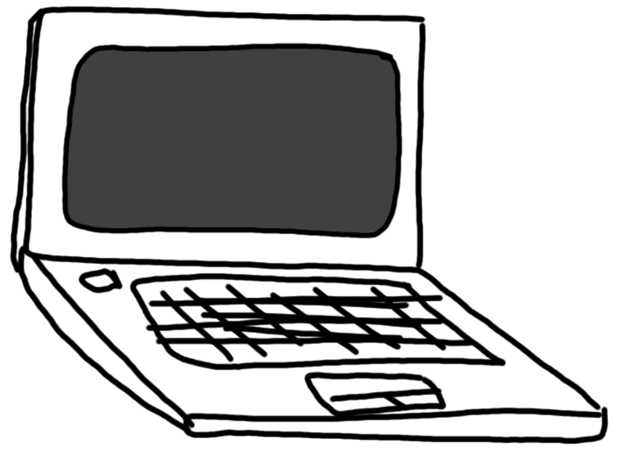
$\mathsf{PK} = \boxed{?} \cdot G$

output $(\boxed{?}, \mathsf{PK})$

$\mathsf{sk}_A \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_A = \mathsf{sk}_A \cdot G$

output $(\mathsf{sk}_A, \mathsf{PK}_A)$



$\boxed{?} \leftarrow \mathbb{Z}_q$

$\mathsf{PK}_B = \boxed{?} \cdot G$

output $(\boxed{?}, \mathsf{PK}_B)$

How are the values related?

$\boxed{?} = \mathsf{sk}_A + \boxed{?}$

$\mathsf{PK} = \mathsf{PK}_A + \mathsf{PK}_B$

Distributing Schnorr KeyGen

$\mathrm{SchnorrKeyGen}(\mathbb{G}, G, q):$

$\boxed{?} \leftarrow \mathbb{Z}_q$

$\mathsf{PK} = \boxed{?} \cdot G$

output $(\boxed{?}, \mathsf{PK})$

$\mathsf{sk_A} \leftarrow \mathbb{Z}_q$

$\mathsf{PK_A} = \mathsf{sk_A} \cdot G$

output $(\mathsf{sk_A}, \mathsf{PK_A})$

$\boxed{?} \leftarrow \mathbb{Z}_q$

$\mathsf{PK_B} = \boxed{?} \cdot G$

output $(\boxed{?}, \mathsf{PK_B})$

How are the values related?

$\boxed{?} = \mathsf{sk_A} + \boxed{?}$

$\mathsf{PK} = \mathsf{PK_A} + \mathsf{PK_B}$

Note:

Computing $\mathsf{sk_B}$ given $\mathsf{PK_B}$ is just as hard as computing sk given PK

Distributing Schnorr KeyGen

$$\begin{aligned}
&\text{SchnorrKeyGen}(\mathbb{G}, G, q) : \\
&\quad \boxed{[\mathsf{sk}]} \leftarrow \mathbb{Z}_q \\
&\quad \mathsf{PK} = \boxed{[\mathsf{sk}]} \cdot G \\
&\quad \text{output } (\mathsf{sk}, \mathsf{PK})
\end{aligned}$$

Schnorr Signing

$\mathsf{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

$$e = H(R \| m)$$

$$s = k - \mathsf{sk} \cdot e$$

$$\sigma = (s, R)$$

$$\text{output } \sigma$$

Schnorr Signing

$$\mathrm{SchnorrSign}(\mathsf{sk}, m):$$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

Identical to KeyGen
Same protocol applies

$$e = H(R \| m)$$

$$s = k - \mathsf{sk} \cdot e$$

$$\sigma = (s, R)$$

$$\text{output } \sigma$$

Schnorr Signing

$$\mathrm{SchnorrSign}([\mathsf{sk}], m):$$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(R \| m)$$

$$s = k - \mathsf{sk} \cdot e$$

$$\sigma = (s, R)$$

$$\text{output } \sigma$$

Identical to KeyGen
Same protocol applies

Schnorr Signing

$\mathsf{SchnorrSign}([\mathsf{sk}], m):$

$$\boxed{\begin{aligned}[k] &\leftarrow \mathbb{Z}_q \\ R &= [k] \cdot G\end{aligned}}$$

Identical to KeyGen
Same protocol applies

$$e = H(R \| m)$$

$$\boxed{[s] = [k] - [\mathsf{sk}] \cdot e}$$

Linear function of k, sk
Make use of linearity of secret sharing scheme

$$\sigma = (s, R)$$

output σ

3 Round Schnorr Signing

Folklore, [Lindell 22]

Input : $\mathsf{pk} = [\mathsf{sk}] \cdot G$, $[\mathsf{sk}]$, $[k]$

Establish $R = [k] \cdot G$

Round 1	Exchange Commit $\left(R_i = [k]_i \cdot G\right)$
Round 2	Release R_i, set $R = \Sigma_i\ R_i$
Round 3	Reveal $s = [\mathsf{sk}] \cdot H(m, R) + [k]$

Output (R, s)

(Threshold) Schnorr in Practice?

- Schnorr signatures are **old** (well-studied), **compact**, **fast** to generate and verify, and **easy to distribute with MPC (i.e. thresholdize)**
- However it was patented - major barrier for internet adoption
- Patent expired recently but the damage is done; adoption is increasing but much of the **internet infrastructure does not support Schnorr**
- Instead, **ECDSA** is widely deployed in its place—similar performance and security, and patent-free but MPC-unfriendly

ECDSA

- **E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm
- Devised by Scott Vanstone in 1992, standardised by NIST
- Differs from Schnorr enough so that patent doesn't apply
- Widespread adoption across the internet

Threshold ECDSA: Challenges

$\mathrm{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$
$$R = k \cdot G$$
$$e = H(R \| m)$$
$$s = k - \mathsf{sk} \cdot e$$
$$\sigma = (s, R)$$

output σ

$\mathrm{ECDSASign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$
$$R = k \cdot G$$
$$e = H(m)$$

Threshold ECDSA: Challenges

$\mathsf{SchnorrSign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

$$e = H(R \| m)$$

$$s = k - \mathsf{sk} \cdot e$$

$$\sigma = (s, R)$$

output σ

$\mathsf{ECDSASign}(\mathsf{sk}, m):$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

$$e = H(m)$$

$$s = \frac{e + \mathsf{sk} \cdot r_x}{k}$$

output $\sigma = (s, R)$

Threshold ECDSA: Challenges

$$\mathrm{ECDSASign}(\mathsf{sk}, m):$$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

$$e = H(m)$$

$$\boxed{s = \frac{e + \mathsf{sk} \cdot r_x}{k}}$$

$$\text{output } \sigma = (s, R)$$

Threshold ECDSA: Challenges

$$\mathrm{ECDSASign}(\mathsf{sk}, m):$$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

$$e = H(m)$$

$$s = \frac{e + \boxed{\mathsf{sk}} \cdot r_x}{\boxed{k}}$$

$$\text{output } \sigma = (s, R)$$

Multiplication of secret values

Threshold ECDSA: Challenges

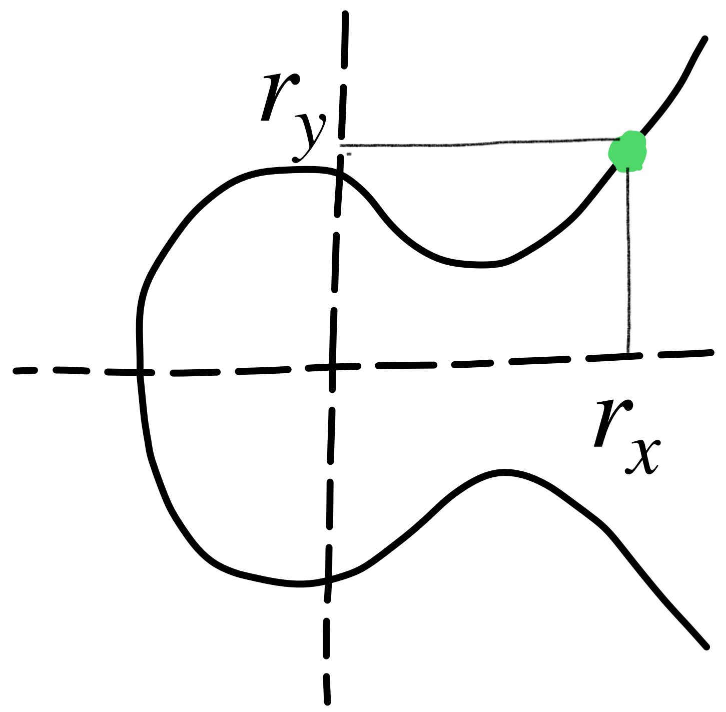
$$
\begin{aligned}
&\mathrm{ECDSASign}(\mathsf{sk}, m):\\
&k \leftarrow \mathbb{Z}_q\\
&R = k \cdot G\\
&e = H(m)\\
&s = \frac{e + \mathsf{sk} \cdot r_x}{\boxed{k}}\\
&\text{output } \sigma = (s, R)
\end{aligned}
$$

Multiplication of secret values

Division (Modular inverse)

Threshold ECDSA: Challenges

$R = (r_x, r_y)$



$$\mathsf{ECDSASign}(\mathsf{sk}, m):$$

$$k \leftarrow \mathbb{Z}_q$$

$$R = k \cdot G$$

$$e = H(m)$$

$$s = \frac{e + \mathsf{sk} \cdot r_x}{\boxed{k}}$$

$$\text{output } \sigma = (s, R)$$

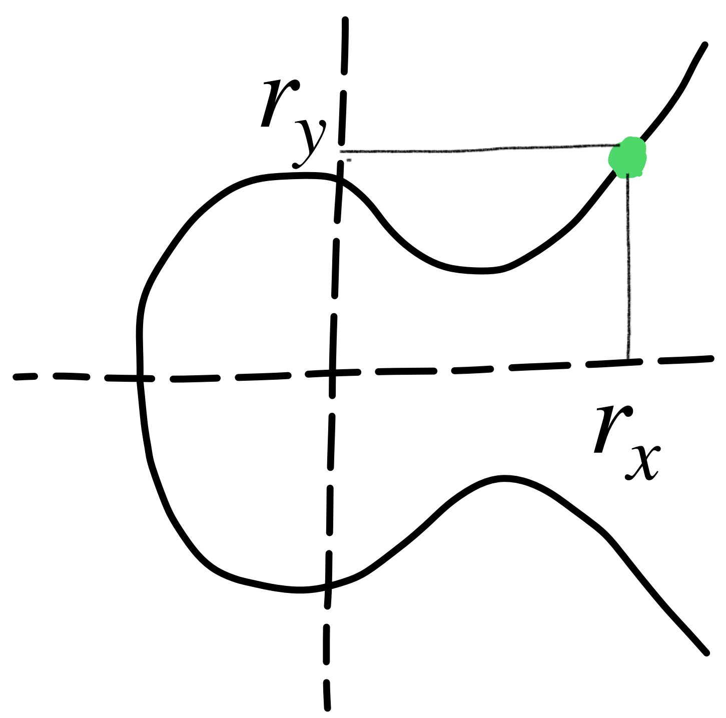
Multiplication of secret values

x-coordinate of ***R*** (not secret)

Division (Modular inverse)

Threshold ECDSA: Challenges

$R = (r_x, r_y)$



$$\mathsf{ECDSASign}(\mathsf{sk}, m):$$
$$k \leftarrow \mathbb{Z}_q$$
$$R = k \cdot G$$
$$e = H(m)$$
$$s = \frac{e + \mathsf{sk} \cdot r_x}{\boxed{k}}$$
$$\text{output } \sigma = (s, R)$$

There is no one-size-fits-all solution

Multiplication of secret values

x-coordinate of $\boldsymbol{R}$ (not secret)

Division (Modular inverse)

Threshold ECDSA: State of the Art

- Rough costs with 256-bit curve, for each additional party (computation aggregated across [Gavenda 21, XAXYC 21, BMP 22]):

Protocol	Tool	Rounds	Bandwidth (KB)	Computation (ms)
[DKLs 19]	OT	$\log(t) + 6$	90	<10
[HLNR 18/23]	OT+	7	40	50—100
[CGGMP 20]	Paillier	4	15	Hundreds
[GG 18]		8	7	Hundreds
[CCLST20, YCX21]	Class Groups	4	4	> 1000

Threshold ECDSA: Goal

- Rough costs with 256-bit curve, for each additional party (computation aggregated across [Gavenda 21, XAXYC 21, BMP 22]):

Protocol	Tool	Bandwidth (KB)	Computation (ms)
[DKLs 19]	OT	90	<10
[HLNR 18/23]	OT+	40	50—100
[CGGMP 20]	Paillier	15	Hundreds
[GG 18]		7	Hundreds
[CCLST20, YCX21]	Class Groups	4	> 1000

Threshold ECDSA: Goal

- Rough costs with 256-bit curve, for each additional party (computation aggregated across [Gavenda 21, XAXYC 21, BMP 22]):

Protocol	Tool	Bandwidth (KB)	Computation (ms)
[DKLs 19]	OT	90	<10
[HLNR 18/23]	OT+	40	50—100
[CGGMP 20]	Paillier	15	Hundreds
[GG 18]		7	Hundreds
[CCLST20, YCX21]	Class Groups	4	> 1000

<u>This work:</u>
3 Round Signing
from
2 round MULT

Threshold ECDSA: Goal

- Rough costs with 256-bit curve, for each additional party (computation aggregated across [Gavenda 21, XAXYC 21, BMP 22]):

Protocol	Tool	Bandwidth (KB)	Computation (ms)
[DKLs 19]	OT	90	<10
[HLNR 18/23]	OT+	40	50—100
[CGGMP 20]	Paillier	15	Hundreds
[GG 18]	Paillier	7	Hundreds
[CCLST20, YCX21]	Class Groups	4	> 1000

<u>This work:</u>
3 Round Signing
from
2 round MULT

mild/no overhead

Threshold ECDSA: Goal

- Rough costs with 256-bit curve, for each additional party (computation aggregated across [Gavenda 21, XAXYC 21, BMP 22]):

Protocol	Tool	Bandwidth (KB)	Computation (ms)
[DKLs 19]	OT	90	<10
[HLNR 18/23]	OT+	40	50—100
[CGGMP 20]	Paillier	15	Hundreds
[GG 18]	Paillier	7	Hundreds
[CCLST20, YCX21]	Class Groups	4	> 1000

<u>This work:</u>
3 Round Signing from 2 round MULT

mild/no overhead

<u>Insight</u>: well-chosen correlation+simple consistency check

Threshold ECDSA: Goal

- Rough costs with 256-bit curve, for each additional party (computation aggregated across [Gavenda 21, XAXYC 21, BMP 22]):

Protocol	Tool	Bandwidth (KB)	Computation (ms)
[DKLs 23]	OT	60	<10
[HLNR 18/23]	OT+	40	50—100
[CGGMP 20]	Paillier	15	Hundreds
[GG 18]	Paillier	7	Hundreds
[CCLST20, YCX21]	Class Groups	4	> 1000

This work:
3 Round Signing from 2 round MULT

mild/no overhead

Insight:
well-chosen correlation+simple consistency check

Intro		How to distribute ECDSA			Tradeoffs
MPC for Schnorr is easy	but not ECDSA	Rewriting ECDSA	ECDSA Tuples	2P-MUL + Consistency	

MPC for ECDSA

- In principle: can use generic MPC to compute $[s] = (e + [\mathsf{sk}] \cdot r_x) \cdot [k^{-1}]$
- However, computing $[k^{-1}]$ given $[k]$ naively is prohibitively expensive
- Rewrite ECDSA signing equation to an “MPC-friendly” equivalent
  i.e. only additions and multiplications of secret values
- Bar-Ilan Beaver 89: Inversion outside MPC

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$$\text{ECDSASign}(\mathsf{sk}, m):$$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$s = \frac{e + [\mathsf{sk}] \cdot r_x}{[k]}$$

$$\text{output } \sigma = (s, R)$$

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$$\mathrm{ECDSASign}(\mathsf{sk}, m):$$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$s = \frac{e + [\mathsf{sk}] \cdot r_x}{[k]} \cdot \frac{[\phi]}{[\phi]}$$

$$\text{output } \sigma = (s, R)$$

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$$\mathrm{ECDSASign}(\mathsf{sk}, m):$$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$\alpha = (e + [\mathsf{sk}] \cdot r_x)\,[\phi]$$

$$\beta = [k][\phi]$$

$$s =$$

$$\text{output } \sigma = (s, R)$$

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$\mathsf{ECDSASign}(\mathsf{sk}, m):$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$\alpha = (e + [\mathsf{sk}] \cdot r_x)\,[\phi]$$

$$\beta = [k][\phi]$$

$$s = \frac{\alpha}{\beta}$$

output $\sigma = (s, R)$

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$\mathsf{ECDSASign}(\mathsf{sk}, m):$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$[\phi] \leftarrow \mathbb{Z}_q$$

$$\alpha = (e + [\mathsf{sk}] \cdot r_x)\,[\phi]$$

$$\beta = [k][\phi]$$

$$s = \frac{\alpha}{\beta}$$

output $\sigma = (s, R)$

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$$\mathrm{ECDSASign}(\mathsf{sk}, m):$$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$[\phi] \leftarrow \mathbb{Z}_q$$

Public values

$$\alpha = (e + [\mathsf{sk}] \cdot r_x)\,[\phi]$$

$$\beta = [k][\phi]$$

$$s = \frac{\alpha}{\beta}$$

$$\text{output } \sigma = (s, R)$$

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$\mathrm{ECDSASign}(\mathsf{sk}, m):$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$[\phi] \leftarrow \mathbb{Z}_q$$

$$\alpha = (e + [\mathsf{sk}] \cdot r_x)\,[\phi]$$

$$\beta = [k][\phi]$$

Public values

Safe to reveal
β: because ϕ is OTP
α: because fixed by β, s

$$s = \frac{\alpha}{\beta}$$

output $\sigma = (s, R)$

Rewriting ECDSA for MPC

[Lindell Nof Ranellucci 18] [Abram Nof Orlandi Scholl Shlomovits 22]

$$\mathrm{ECDSASign}(\mathsf{sk}, m):$$

$$[k] \leftarrow \mathbb{Z}_q$$

$$R = [k] \cdot G$$

$$e = H(m)$$

$$[\phi] \leftarrow \mathbb{Z}_q$$

$$\alpha = (e + [\mathsf{sk}] \cdot r_x)\,[\phi]$$

$$\beta = [k][\phi]$$

$$s = \frac{\alpha}{\beta}$$

$$\text{output } \sigma = (s, R)$$

Public values

Safe to reveal
β: because ϕ is OTP
α: because fixed by β, s

Secure mult: Only (nonlinear) combination of secret values

Signing from ECDSA Tuples

[Abram Nof Orlandi Scholl Shlomovits 22]

$[\mathsf{sk}]$ $[k]$ $[\phi]$ $[\phi k]$ $[\phi\mathsf{sk}]$

Signing from ECDSA Tuples

[Abram Nof Orlandi Scholl Shlomovits 22]

$[\mathsf{sk}]$ $[k]$ $[\phi]$ $[\phi k]$ $[\phi \mathsf{sk}]$

Round 1

Round 2

Establish $R = [k] \cdot G$

Signing from ECDSA Tuples

[Abram Nof Orlandi Scholl Shlomovits 22]

$[\mathsf{sk}]$ $[k]$ $[\phi]$ $[\phi k]$ $[\phi \mathsf{sk}]$

Round 1 Round 2	Establish $R = [k] \cdot G$
Round 3	Reveal $\alpha = e + r_x[\phi \mathsf{sk}]$ and $\beta = [\phi k]$

Output $(R, s = \alpha/\beta)$

ECDSA Tuple Generation

$[\mathsf{sk}][k]$

$[\phi]$

$[\phi k]$

$[\phi \mathsf{sk}]$

ECDSA Tuple Generation

Input : $[\mathsf{sk}][k]$

Sample : $[\phi]$

$[\phi k]$

$[\phi \mathsf{sk}]$

ECDSA Tuple Generation

Local

Input : $[\mathsf{sk}][k]$

Sample : $[\phi]$

$[\phi k]$

$[\phi \mathsf{sk}]$

ECDSA Tuple Generation

Local

$$\text{Input}: [\mathsf{sk}][k]$$
$$\text{Sample}: [\phi]$$

$$[\phi k] = \mathsf{MULT}([\phi], [k])$$
$$[\phi \mathsf{sk}] = \mathsf{MULT}([\phi], [\mathsf{sk}])$$

ECDSA Tuple Generation

Local

$$\text{Input}: [\mathsf{sk}][k]$$
$$\text{Sample}: [\phi]$$

Consistency: straightforward

$$[\phi k] = \mathsf{MULT}([\phi],[k])$$
$$[\phi \mathsf{sk}] = \mathsf{MULT}([\phi],[\mathsf{sk}])$$

ECDSA Tuple Generation

Local

$$\text{Input}: [\mathsf{sk}][k]$$
$$\text{Sample}: [\phi]$$

Consistency: straightforward

$$[\phi k] = \mathsf{MULT}([\phi], [k])$$
$$[\phi \mathsf{sk}] = \mathsf{MULT}([\phi], [\mathsf{sk}])$$

Verify:
$$[k] \cdot G = R$$
$$[\mathsf{sk}] \cdot G = \mathsf{pk}$$

ECDSA Tuple Generation

Local

$$\text{Input}: [\mathsf{sk}][k]$$
$$\text{Sample}: [\phi]$$

Consistency: straightforward

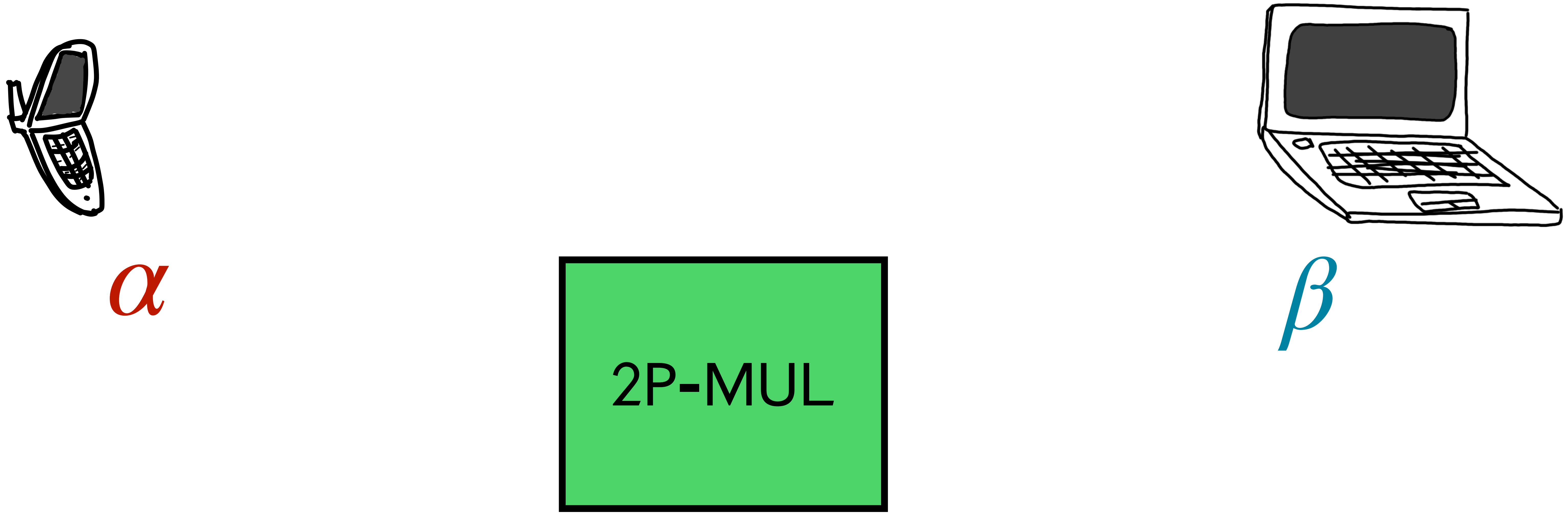
$$[\phi k] = \mathsf{MULT}([\phi], [k])$$
$$[\phi \mathsf{sk}] = \mathsf{MULT}([\phi], [\mathsf{sk}])$$

Previous works: ZK proofs, MACs, etc.
<u>This work</u>: Simple pairwise check

Verify:
$$[k] \cdot G = R$$
$$[\mathsf{sk}] \cdot G = \mathsf{pk}$$

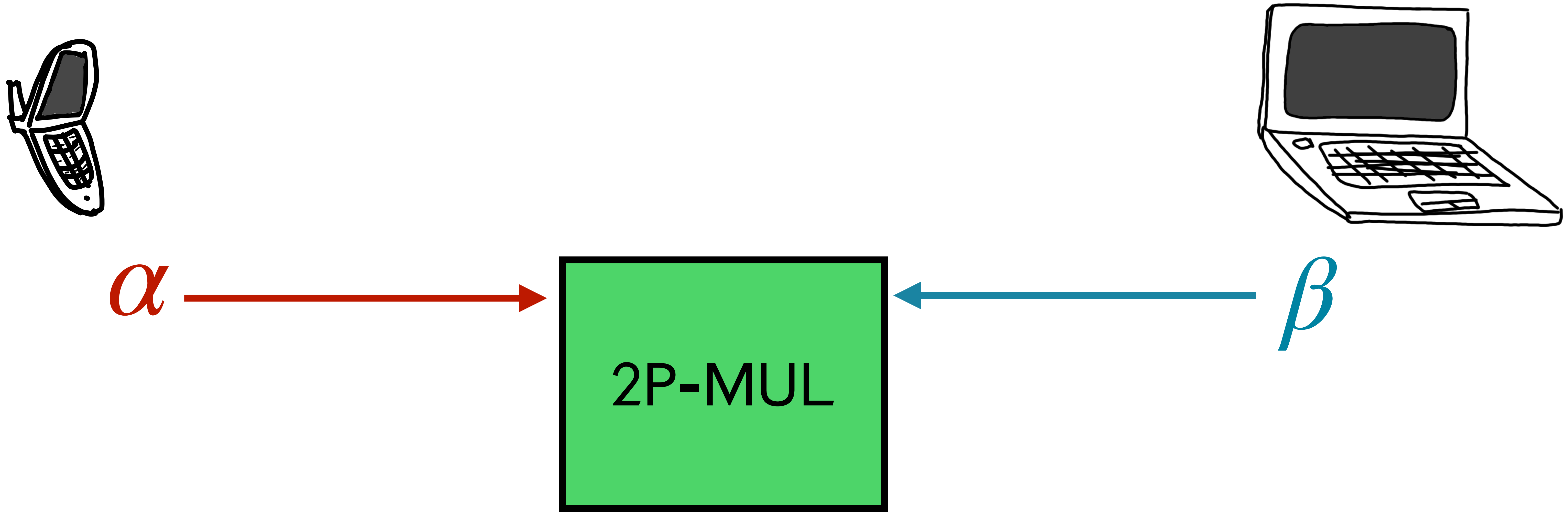
Secure Two-Party Multiplication

a.k.a. OLE, Mult2Add



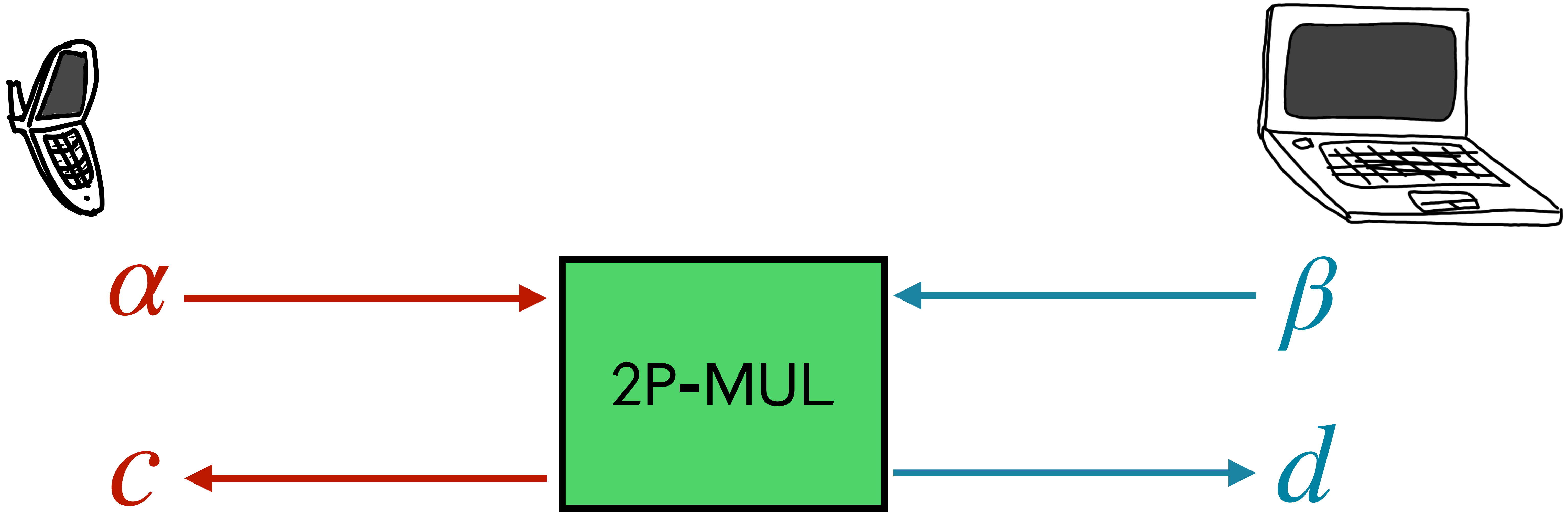
Secure Two-Party Multiplication

a.k.a. OLE, Mult2Add



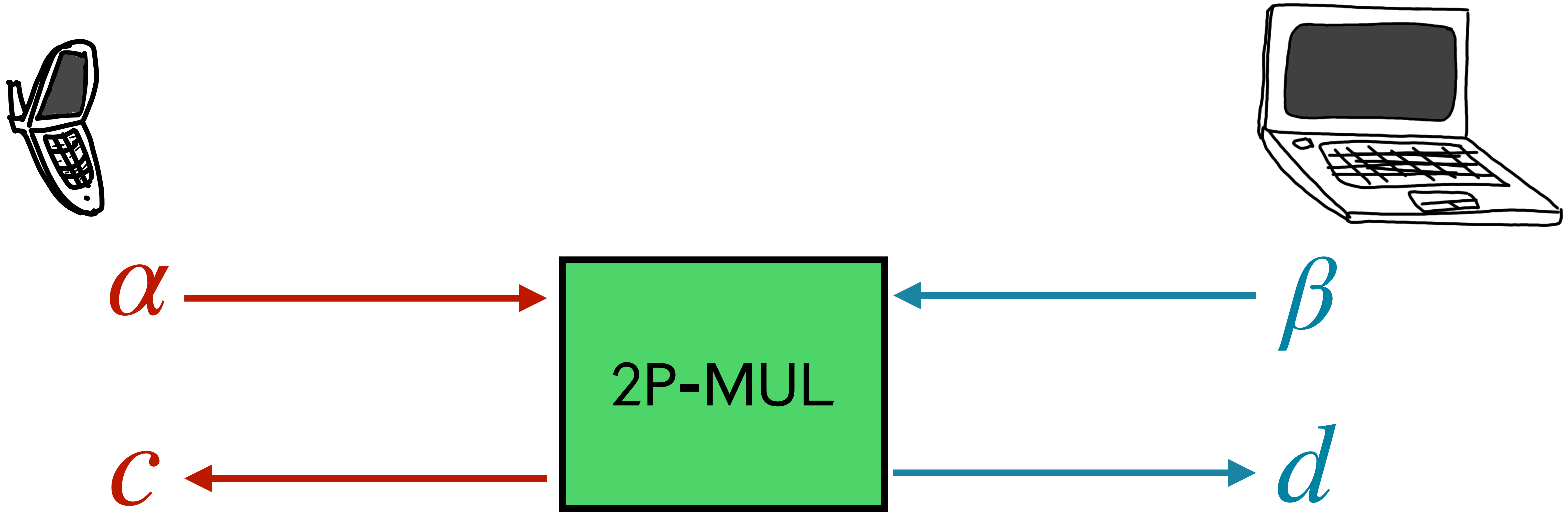
Secure Two-Party Multiplication

a.k.a. OLE, Mult2Add



Secure Two-Party Multiplication

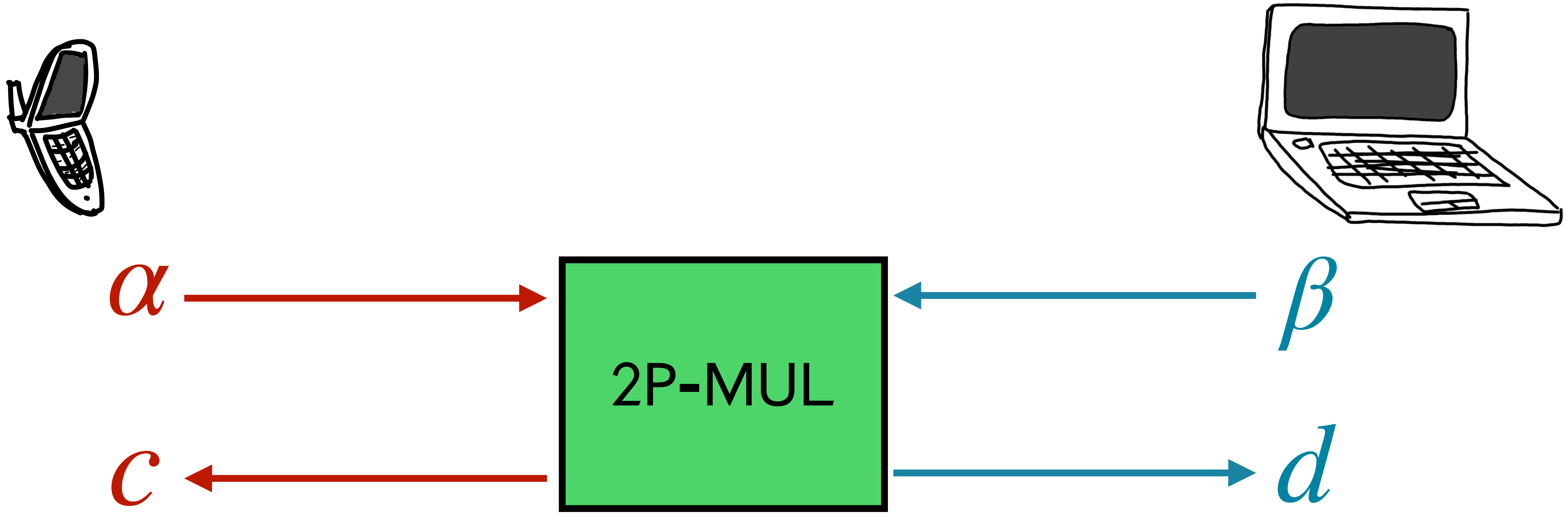
a.k.a. OLE, Mult2Add



$$c + d = \alpha \cdot \beta$$

Secure Two-Party Multiplication

a.k.a. OLE, Mult2Add

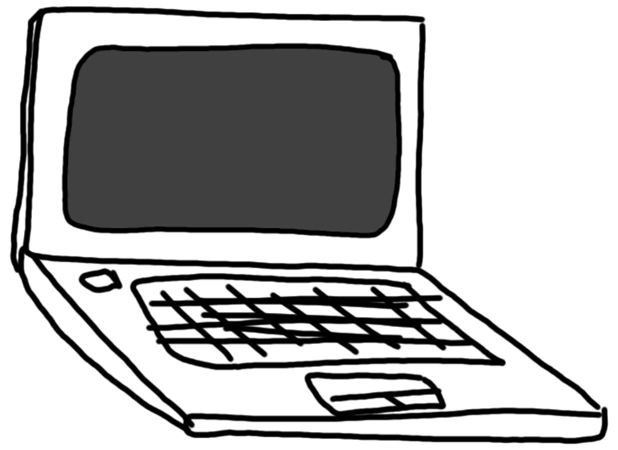


$$c + d = \alpha \cdot \beta$$

Gadget to split a product of secret inputs $\alpha\beta$ into additive secrets c, d

Instantiable efficiently from: OT, Paillier, Class Groups

Two-Round 2P-MUL



$$e = \mathsf{Enc}_1(\beta)$$

$\longleftarrow$

$\longrightarrow$

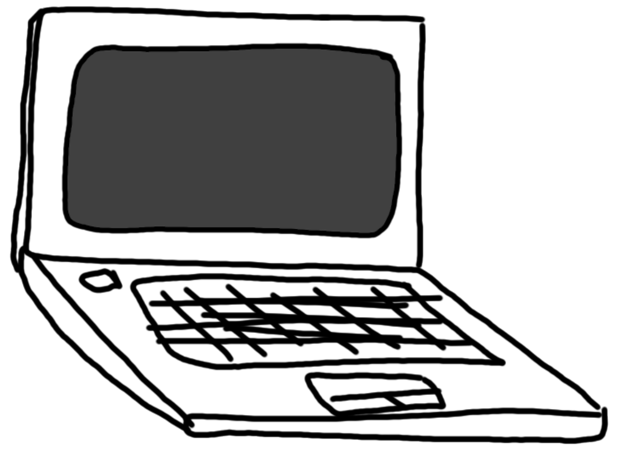
$$\mathsf{Enc}_2(e, \alpha)$$

c

d

$$c + d = \alpha \cdot \beta$$

Two-Round 2P-MUL



$e = \mathsf{Enc}_1(\beta)$

$\longleftarrow$

$\longrightarrow$

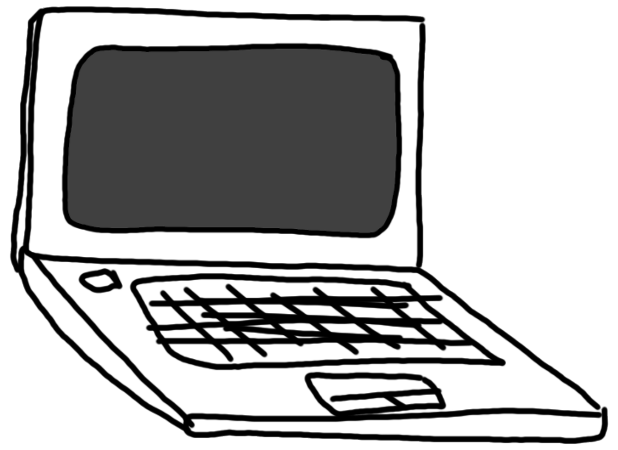
$\mathsf{Enc}_2(e, \alpha)$
$\mathsf{Enc}_2(e, \hat{\alpha})$

c

d

$$c + d = \alpha \cdot \beta$$
$$\hat{c} + \hat{d} = \hat{\alpha} \cdot \beta$$

Two-Round 2P-MUL



$$e = \mathsf{Enc}_1(\beta)$$

$$\mathsf{Enc}_2(e, \alpha)$$
$$\mathsf{Enc}_2(e, \hat{\alpha})$$

c

d

$$c + d = \alpha \cdot \beta$$
$$\hat{c} + \hat{d} = \hat{\alpha} \cdot \beta$$

Consistency
“for free”

ECDSA Tuple Generation

$$\text{Input} : [\mathsf{sk}][k]$$
$$\text{Sample} : [\phi]$$

Consistency: straightforward ✓

$$[\phi k] = \mathsf{MULT}([\phi], [k])$$
$$[\phi \mathsf{sk}] = \mathsf{MULT}([\phi], [\mathsf{sk}])$$

ECDSA Tuple Generation

$\text{Input} : [\mathsf{sk}][k]$
$\text{Sample} : [\phi]$

Consistency: straightforward ✓

$$[\phi k] = \mathsf{MULT}([\phi], [k])$$
$$[\phi \mathsf{sk}] = \mathsf{MULT}([\phi], [\mathsf{sk}])$$

Verify: $[k] \cdot G = R$
$[\mathsf{sk}] \cdot G = \mathsf{pk}$

Verifying Consistency w.r.t. $\mathbb{G}$

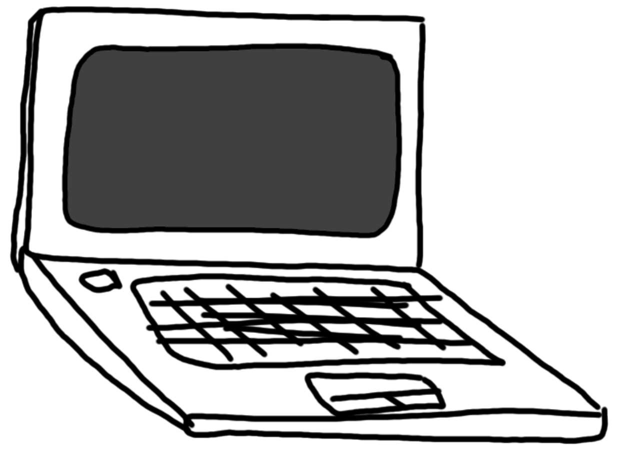
$\mathsf{MULT}([\phi], [k])$

Simplified :

ϕ

2P-MUL

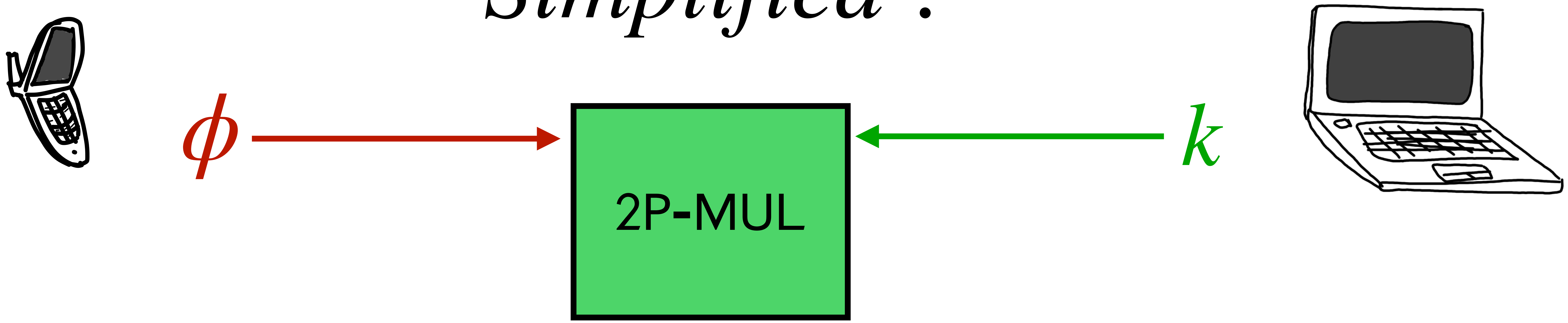
k



Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

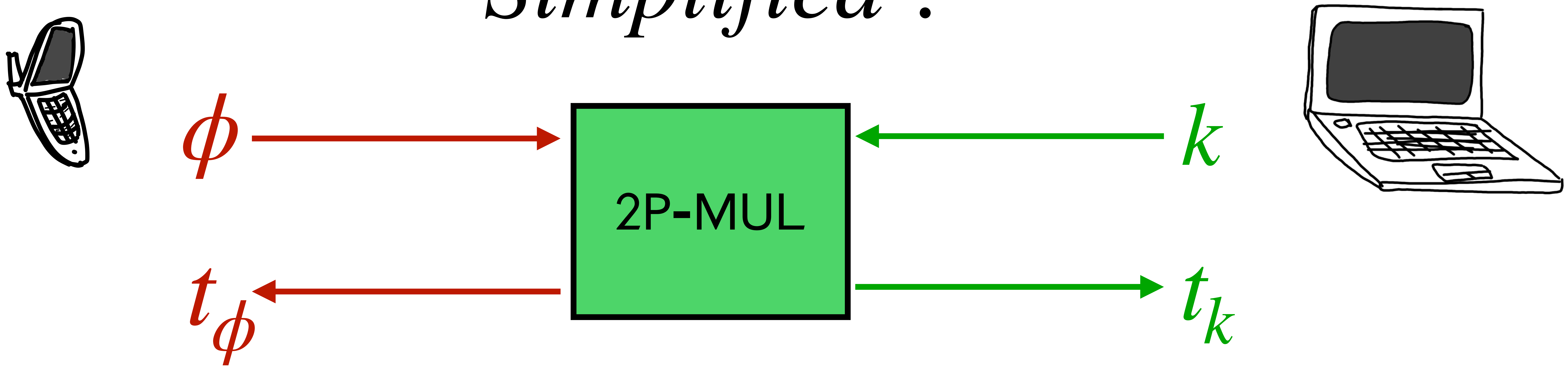
Simplified :



Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

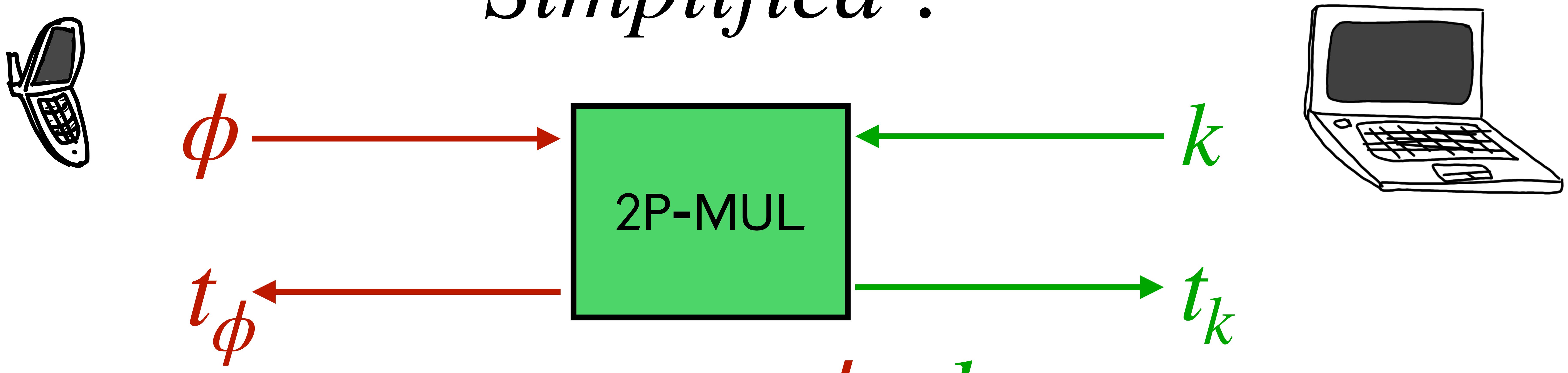
Simplified :



Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi],[k])$

Simplified :

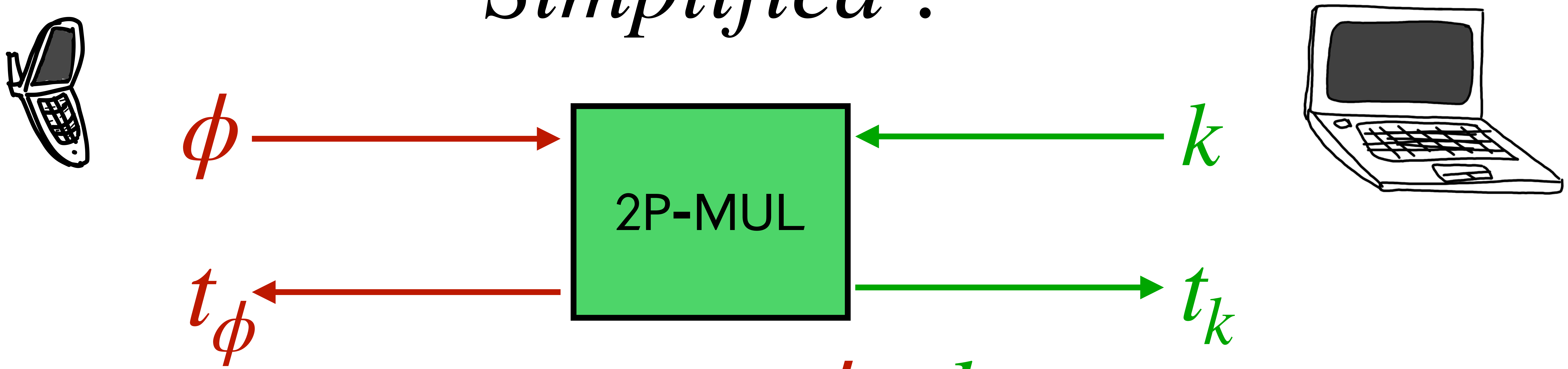


$$t_\phi + t_k = \phi \cdot k$$

Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

Simplified :



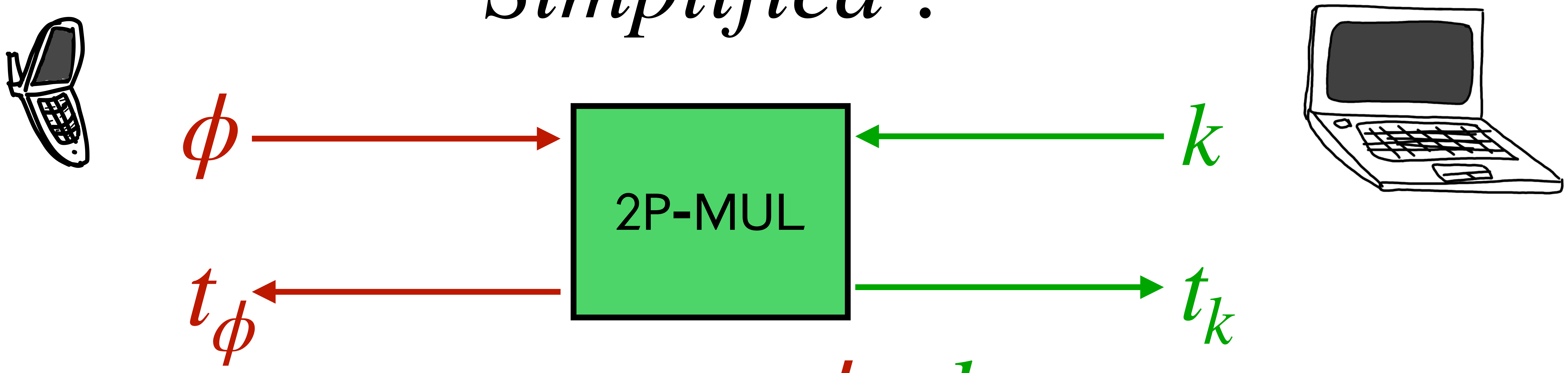
$$t_\phi + t_k = \phi \cdot k$$

Claim: $[k] \cdot G = R$

Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

Simplified :



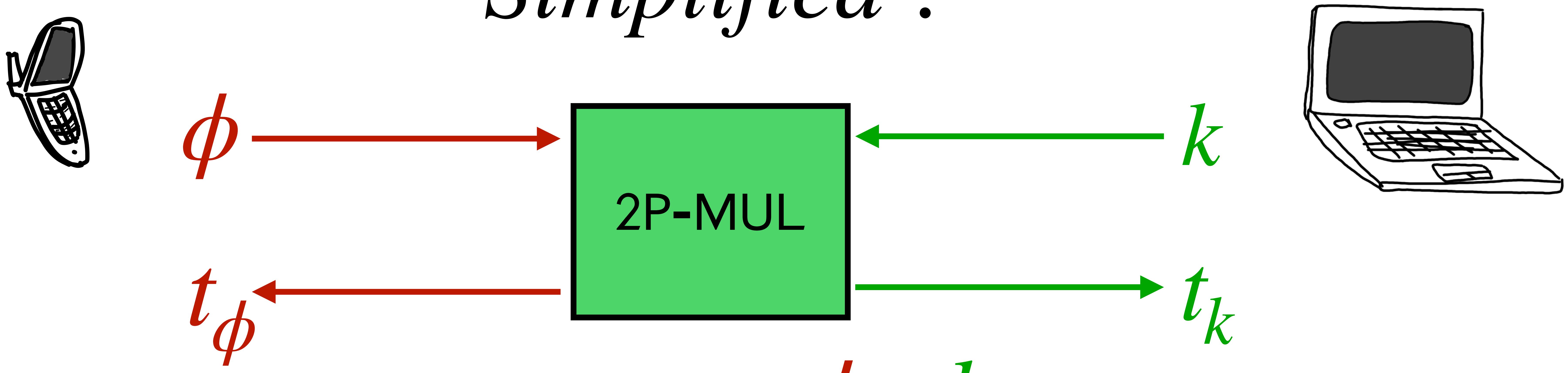
$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claim: $[k] \cdot G = R$

Verifying Consistency w.r.t. $\mathbb{G}$

$\text{MULT}([\phi], [k])$

Simplified :



$$t_\phi + t_k = \phi \cdot k$$

$$T_\phi = t_\phi \cdot G \qquad T_k = t_k \cdot G$$

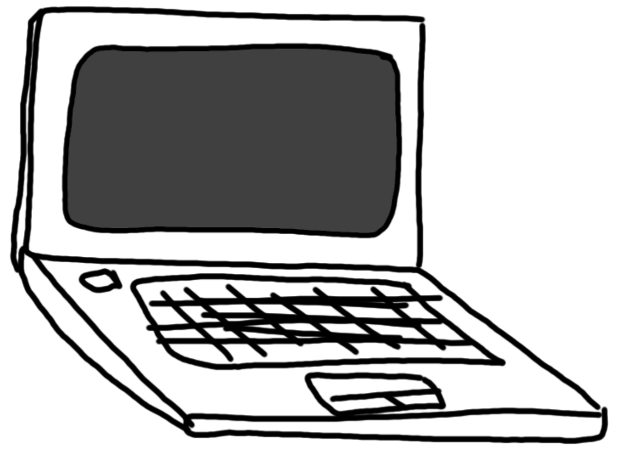
Claim: $[k] \cdot G = R$

$$T_k = \phi R - T_\phi$$

Verifying Consistency w.r.t. $\mathbb{G}$

MULT([ϕ], [k])

Simplified :



ϕ → 2P-MUL ← k

t_ϕ ← 2P-MUL → t_k

$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claim: $[k] \cdot G = R$

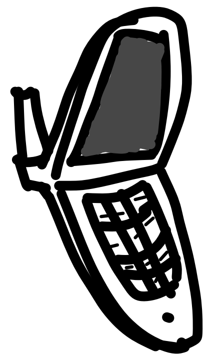
$$T_k = \phi R - T_\phi$$

T_k ←

Verifying Consistency w.r.t. $\mathbb{G}$

MULT([ϕ], [k])

Simplified :



$\phi \rightarrow$ 2P-MUL $\leftarrow k$

$t_\phi \leftarrow$ 2P-MUL $\rightarrow t_k$

$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claim: $[k] \cdot G = R$

$$\boxed{T_k} = \phi R - T_\phi$$

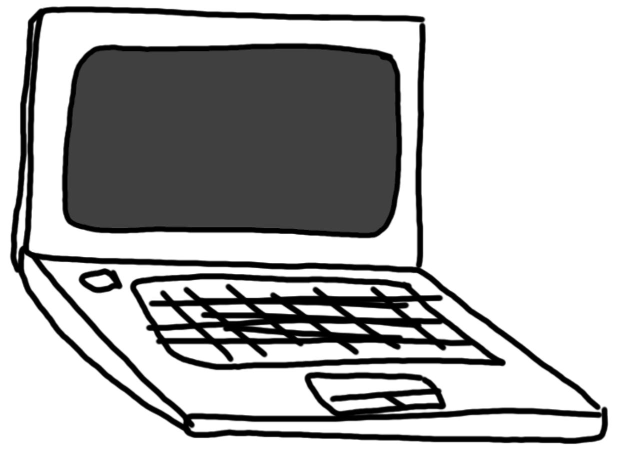
Match?

$\boxed{T_k}$

Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

Simplified :



$\phi \longrightarrow$ 2P-MUL $\longleftarrow k$

$t_\phi \longleftarrow$ 2P-MUL $\longrightarrow t_k$

No information about k

$T_\phi = t_\phi \cdot G$

$t_\phi + t_k = \phi \cdot k$

$T_k = t_k \cdot G$

Claim: $[k] \cdot G = R$

$\boxed{T_k} = \phi R - T_\phi$

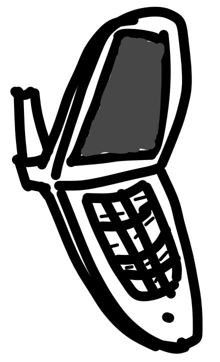
Match?

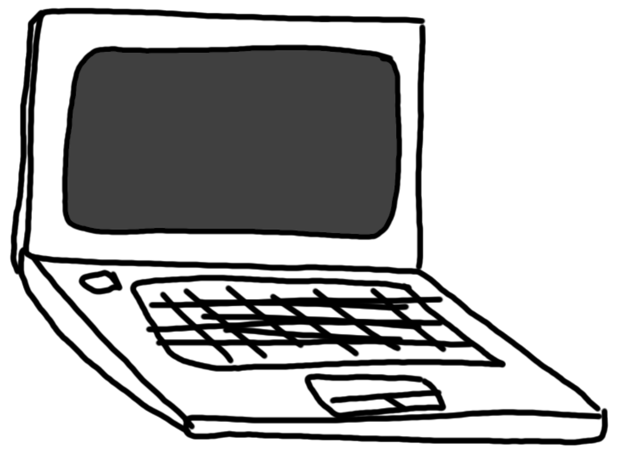
$\boxed{T_k}$

Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

Simplified :





No information about k

$\phi \longrightarrow$ 2P-MUL $\longleftarrow k$

$t_\phi \longleftarrow$ 2P-MUL $\longrightarrow t_k$

$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claim: $[k] \cdot G = R$

$\boxed{T_k} = \phi R - T_\phi$

Match?

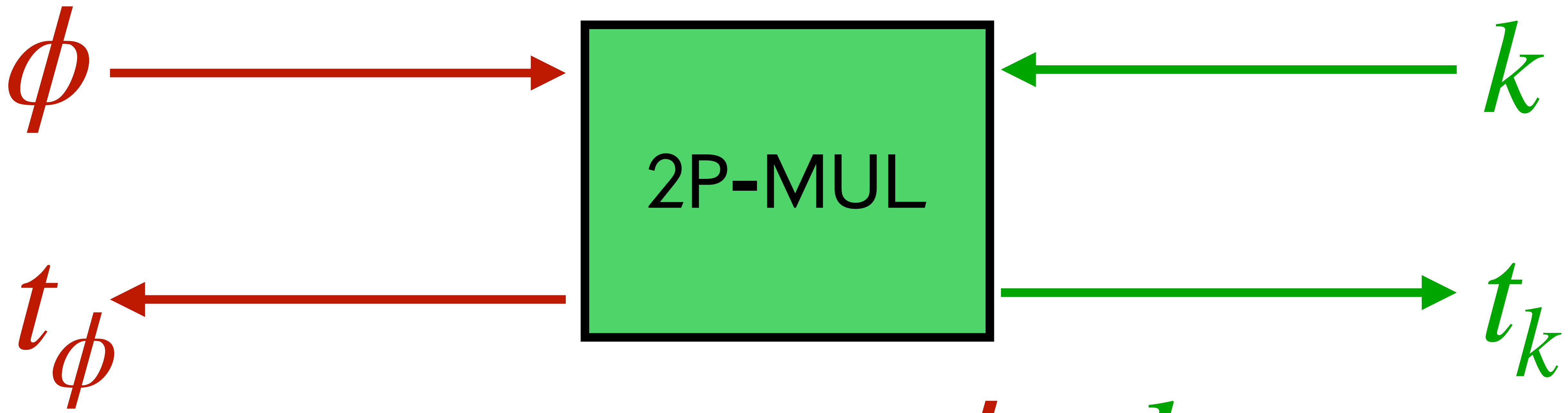
$\boxed{T_k}$

In case of cheat:
have to guess ϕ
(2^{-256} chance)

Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

Simplified :



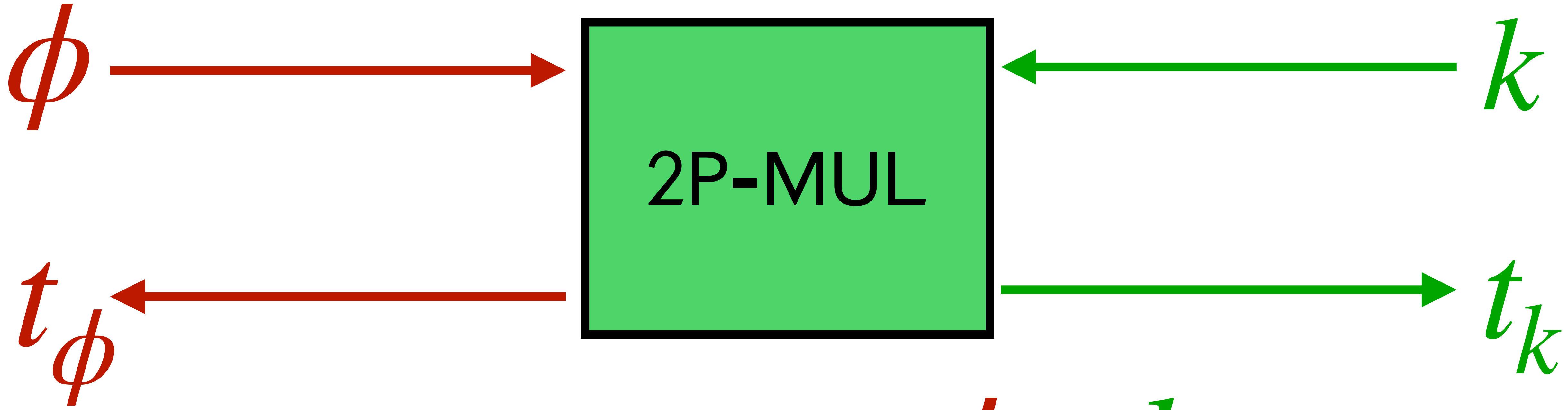
$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claimed $R^* = \Delta + k \cdot G$

Verifying Consistency w.r.t. $\mathbb{G}$

MULT($[\phi]$, $[k]$)

Simplified :



$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claimed $R^* = \Delta + k \cdot G$

$$T_k^* = \phi R^* - T_\phi$$

Verifying Consistency w.r.t. $\mathbb{G}$

MULT([ϕ], [k])

Simplified :

$$
\begin{tikzcd}
\phi \arrow[r] & \boxed{\text{2P-MUL}} \arrow[dl] \arrow[dr] & k \arrow[l] \\
t_\phi & & t_k
\end{tikzcd}
$$

$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claimed $R^* = \Delta + k \cdot G$

$$
\begin{aligned}
T_k^* &= \phi R^* - T_\phi \\
&= T_k + \phi\Delta
\end{aligned}
$$

Verifying Consistency w.r.t. $\mathbb{G}$

$\mathsf{MULT}([\phi], [k])$

Simplified :

$\phi \rightarrow$ 2P-MUL $\leftarrow k$

$t_\phi \leftarrow$ 2P-MUL $\rightarrow t_k$

$$t_\phi + t_k = \phi \cdot k$$

$$T_\phi = t_\phi \cdot G \qquad T_k = t_k \cdot G$$

Claimed $R^* = \Delta + k \cdot G$

$$T_k^* = \phi R^* - T_\phi$$
$$= T_k + \phi\Delta$$

No information about ϕ
$\Rightarrow$ distributed uniformly in $\mathbb{Z}_q$

Verifying Consistency w.r.t. $\mathbb{G}$

MULT($[\phi]$, $[k]$)

Simplified :

$$\phi \longrightarrow \boxed{\text{2P-MUL}} \longleftarrow k$$

$$t_\phi \longleftarrow \boxed{\text{2P-MUL}} \longrightarrow t_k$$

$$T_\phi = t_\phi \cdot G \qquad t_\phi + t_k = \phi \cdot k \qquad T_k = t_k \cdot G$$

Claimed $R^* = \Delta + k \cdot G$

$$T_k^* = \phi R^* - T_\phi$$
$$= T_k + \phi\Delta$$

No information about ϕ
$\Rightarrow$ distributed uniformly in $\mathbb{Z}_q$

Statistically unlikely to send correct T_k^*

Notes on Consistency Check

- Case 1: Inconsistent k^*—almost certainly fails
  Case 2: Consistent k— nothing about ϕ leaked
  $\Rightarrow$ ϕ is a MAC key, but also safe to (re)use in ECDSA tuple

- Costs 3 exponentiations, transmits single $\mathbb{G}$ element, one round
  All costs are superseded by 2P-MUL

- Exact same structure for $[\phi \mathsf{sk}]$ verification with pk

- Actual check: each party validates 2P-MUL inputs
  (i.e. shares of k, sk) used by every counterparty

3 Round ECDSA Signing

[This work]

Sample $[k]$

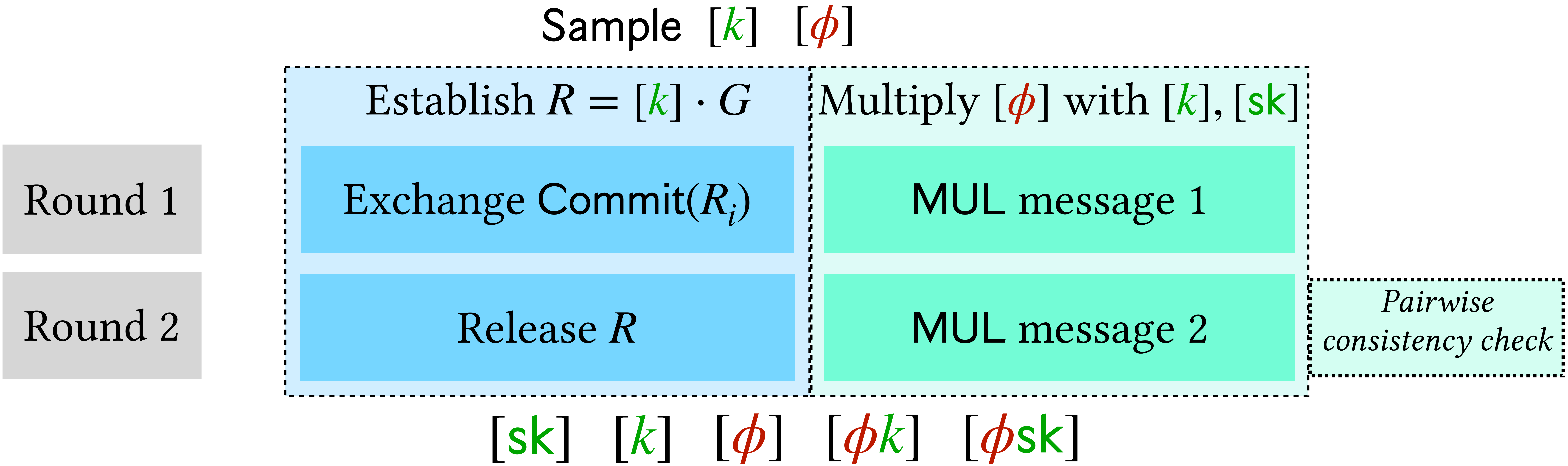
Establish $R = [k] \cdot G$

Round 1 — Exchange Commit(R_i)

Round 2 — Release R

3 Round ECDSA Signing

[This work]



3 Round ECDSA Signing

[This work]

Sample $[k]$ $[\phi]$

Establish $R = [k] \cdot G$

Multiply $[\phi]$ with $[k], [\mathsf{sk}]$

Round 1

Exchange $\mathsf{Commit}(R_i)$

MUL message 1

Round 2

Release R

MUL message 2

Pairwise consistency check

$[\mathsf{sk}]$ $[k]$ $[\phi]$ $[\phi k]$ $[\phi \mathsf{sk}]$

Round 3

Reveal $\alpha = e[\phi] + r_x[\phi\mathsf{sk}]$ and $\beta = [\phi k]$

Output $(R, \sigma = \alpha/\beta)$

Intro		How to distribute ECDSA			Tradeoffs	
MPC for Schnorr is easy	but not ECDSA	Rewriting ECDSA + 3 round protocol	ECDSA Tuples	2P-MUL + Consistency	OT vs AHE	Modes of operation

Instantiating Multiplication

- Secure n-party mult can be reduced to $2n$ instances of 2P-MUL
- 2P-MUL inherently requires public key crypto
- Broadly two approaches:
  - Additively Homomorphic Encryption (low bandwidth, high computation)
  - Oblivious Transfer (low computation, high bandwidth)

2P-MUL from Additively Homomorphic Encryption

- Additive Homomorphism: $\alpha \cdot \mathsf{Enc}(x) + \mathsf{Enc}(\beta) = \mathsf{Enc}(\alpha x + \beta)$

  [Gilboa 99]: Conceptually simple protocol for MUL from AHE
  [CGGMP 20]: Hardened for active security through ZK proofs

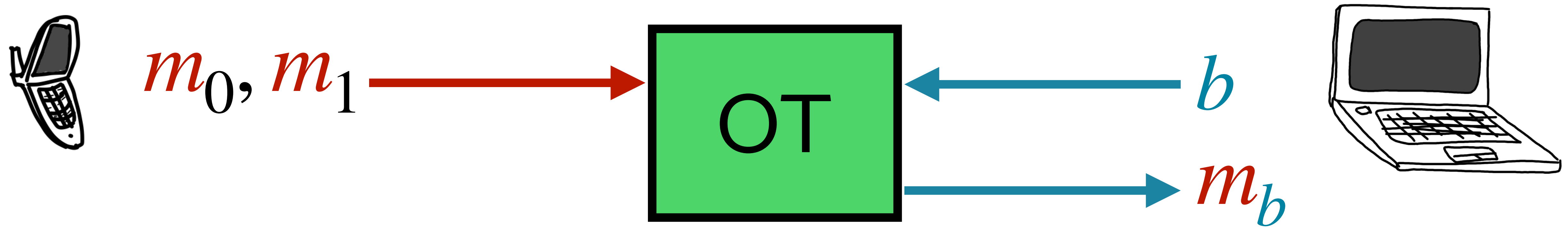
- Instantiations from factoring based cryptography (e.g. [Paillier 99]) and class groups [Castagnos Laguillaumie 15]

- <u>Advantages</u>: Parties exchange (relatively) compact ciphertexts

- <u>Downsides</u>:
  — Ciphertext operations are heavy (2 orders of magnitude slower than EC)
  — Seem to require ZK proofs to prevent misuse

2P-MUL from Oblivious Transfer

- Oblivious Transfer (OT):



[Gilboa 99]: Elegant protocol for MUL from OT
[DKLs 18,19, HMRT 22]: active security by randomized encoding+statistical checks

- Instantiable with ECDSA curve (think DH key exchange)

- Advantages: By OT Extension [IKNP03, Roy22] public key operations can be moved to one-time key generation phase, so only hashes when signing (1 order of magnitude slower than single party signing)

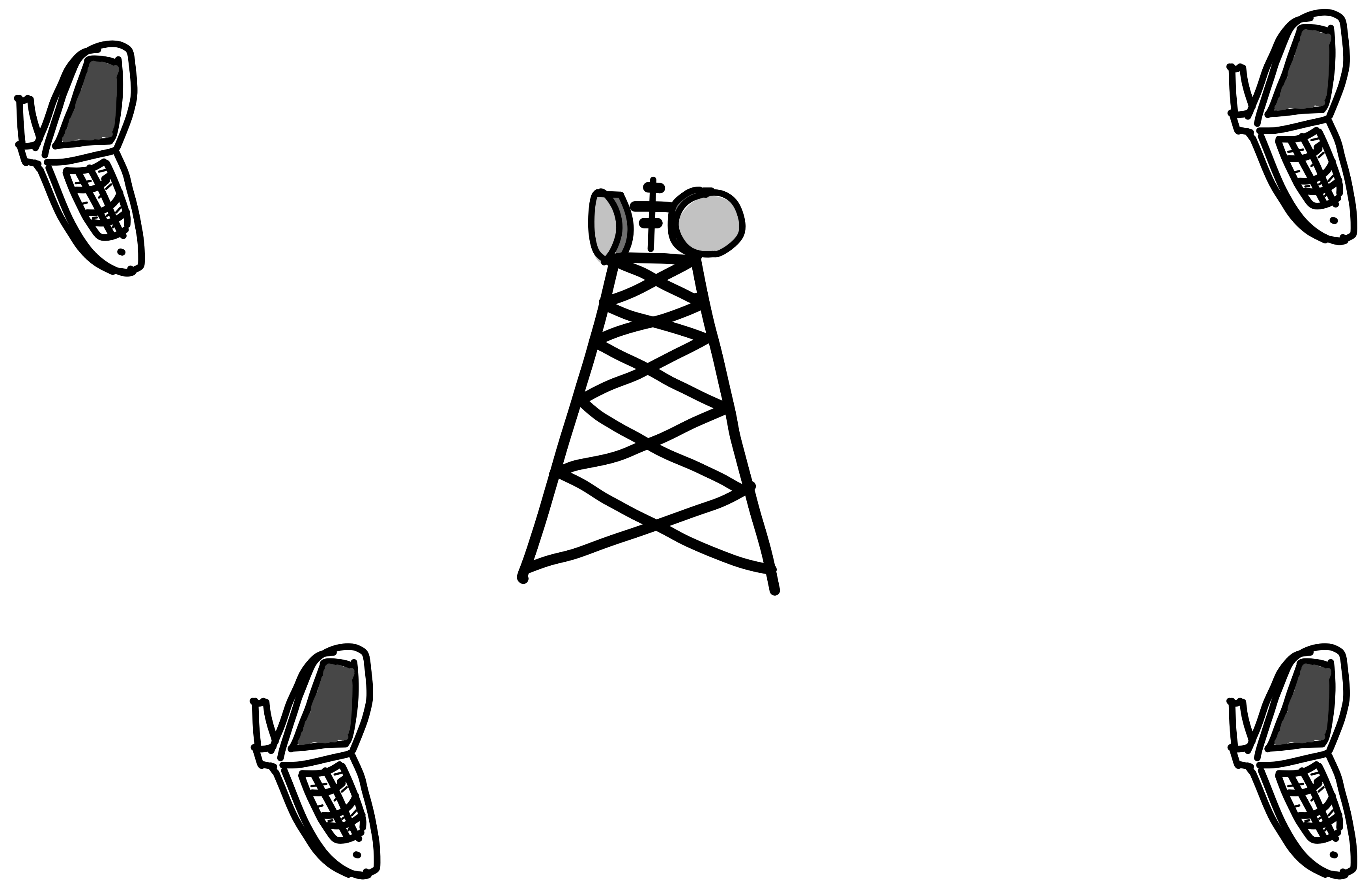
- Downsides: ~1000 OTs/sig, each transmits two $\mathbb{Z}_q$ elements

2P-MUL: AHE vs OT

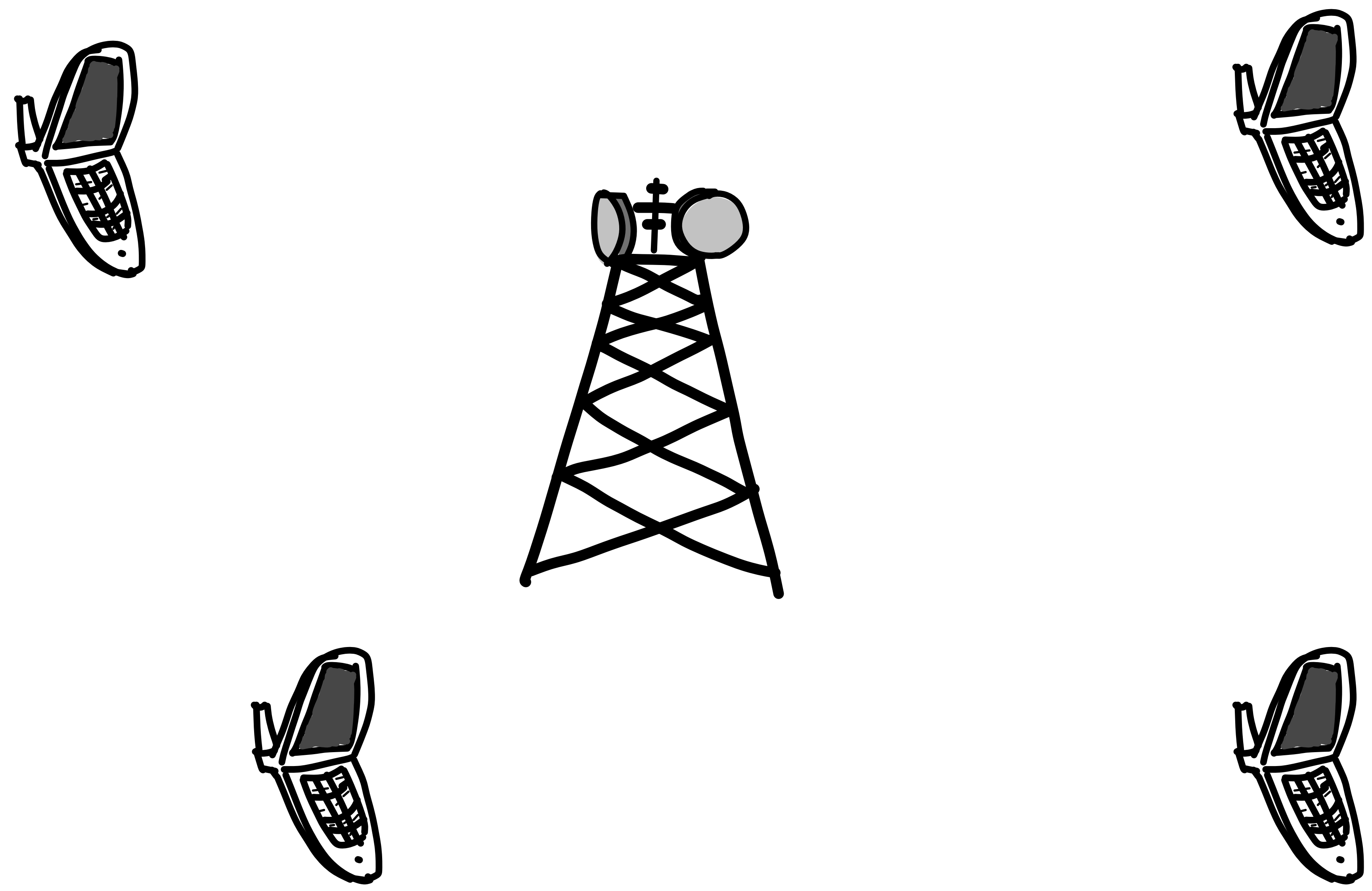
- Tradeoff to make: Computation vs. Bandwidth during signing time
- Rough costs with 256-bit curve, for each additional party (computation aggregated across [Gavenda 21, XAXYC 21, BMP 22]):

	Bandwidth	Computation
OT [DKLs 23]	60 KB	Few milliseconds
Paillier [CGGMP 20]	15 KB	Hundreds of milliseconds
Paillier [GG 18]	7 KB	Hundreds of milliseconds
Class Groups [CCLST20, YCX21]	4.5 KB	> 1 second

Is communication the bottleneck?



Is communication the bottleneck?



- **Mobile applications (human-initiated)**:

Is communication the bottleneck?

- **Mobile applications (human-initiated):**

Is communication the bottleneck?

- **Mobile applications (human-initiated)**:
  - eg. t=4, ~2Mbits transmitted per party

Is communication the bottleneck?

- **Mobile applications (human-initiated)**:
  - eg. t=4, ~2Mbits transmitted per party
  - Well within LTE envelope for responsivity

Example 1: Mobile Wallet

Multiplier: OT-based
Parties: 4
Curve: 256-bit

2 Mbits

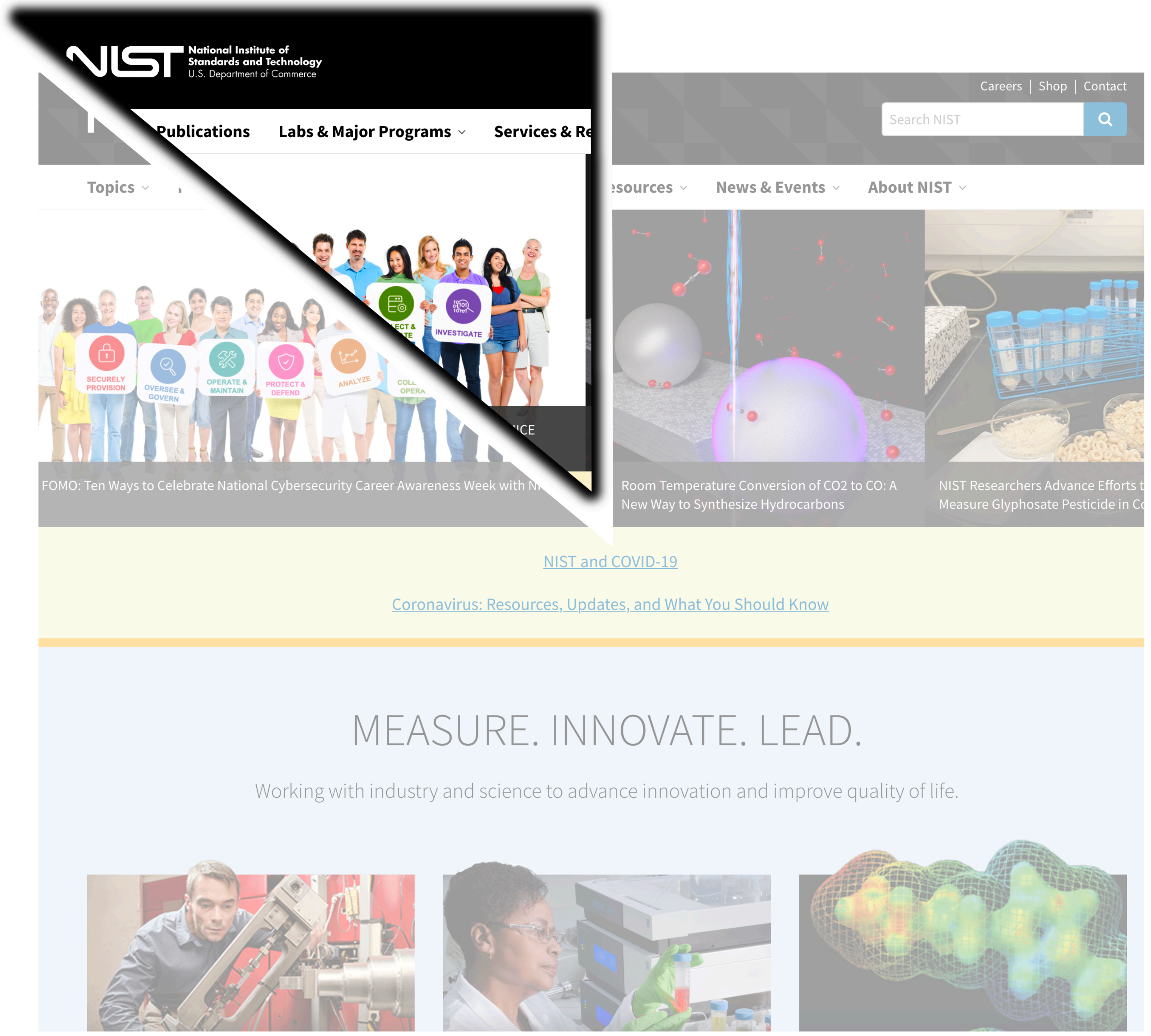
sent per party

Example 1: Mobile Wallet

Multiplier: OT-based
Parties: 4
Curve: 256-bit

2 Mbits
sent per party

$\approx$

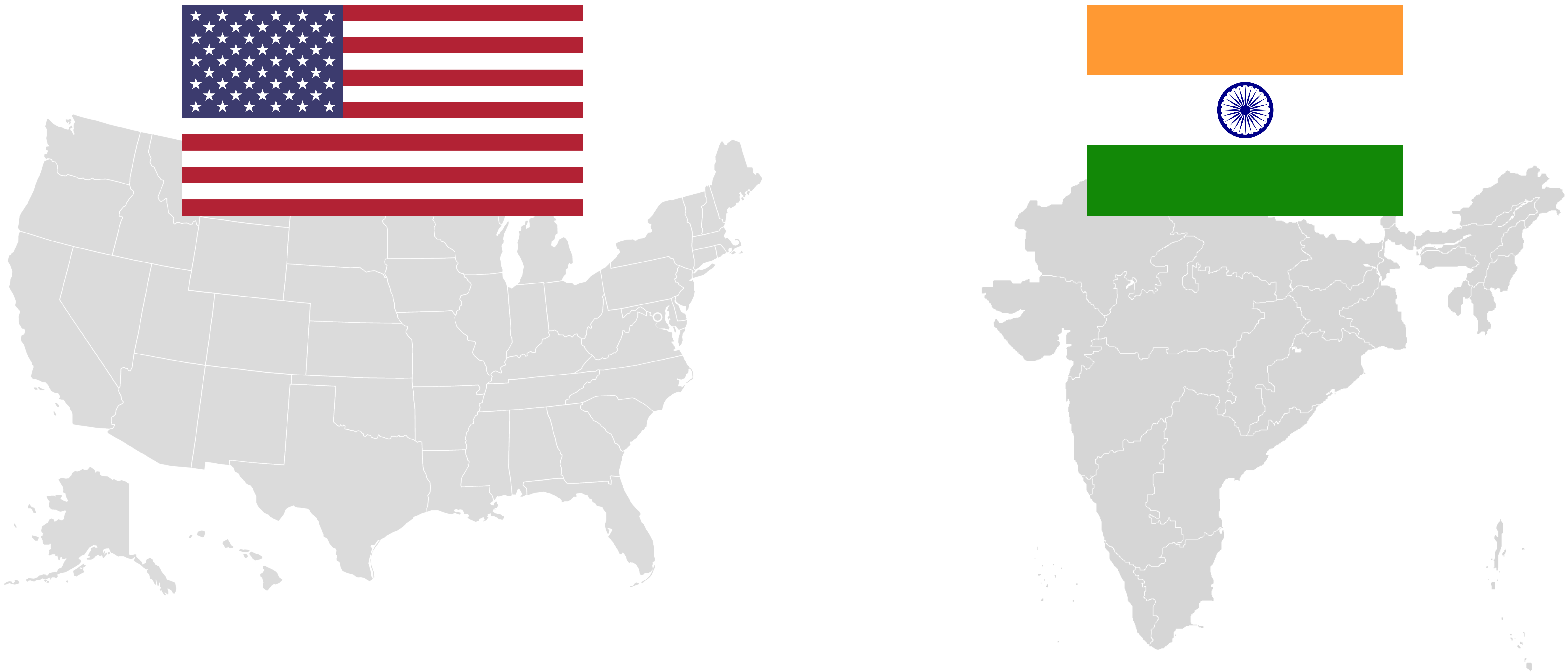


Example 1: Mobile Wallet

Multiplier: OT-based
Parties: 4
Curve: 256-bit

2 Mbits

sent per party

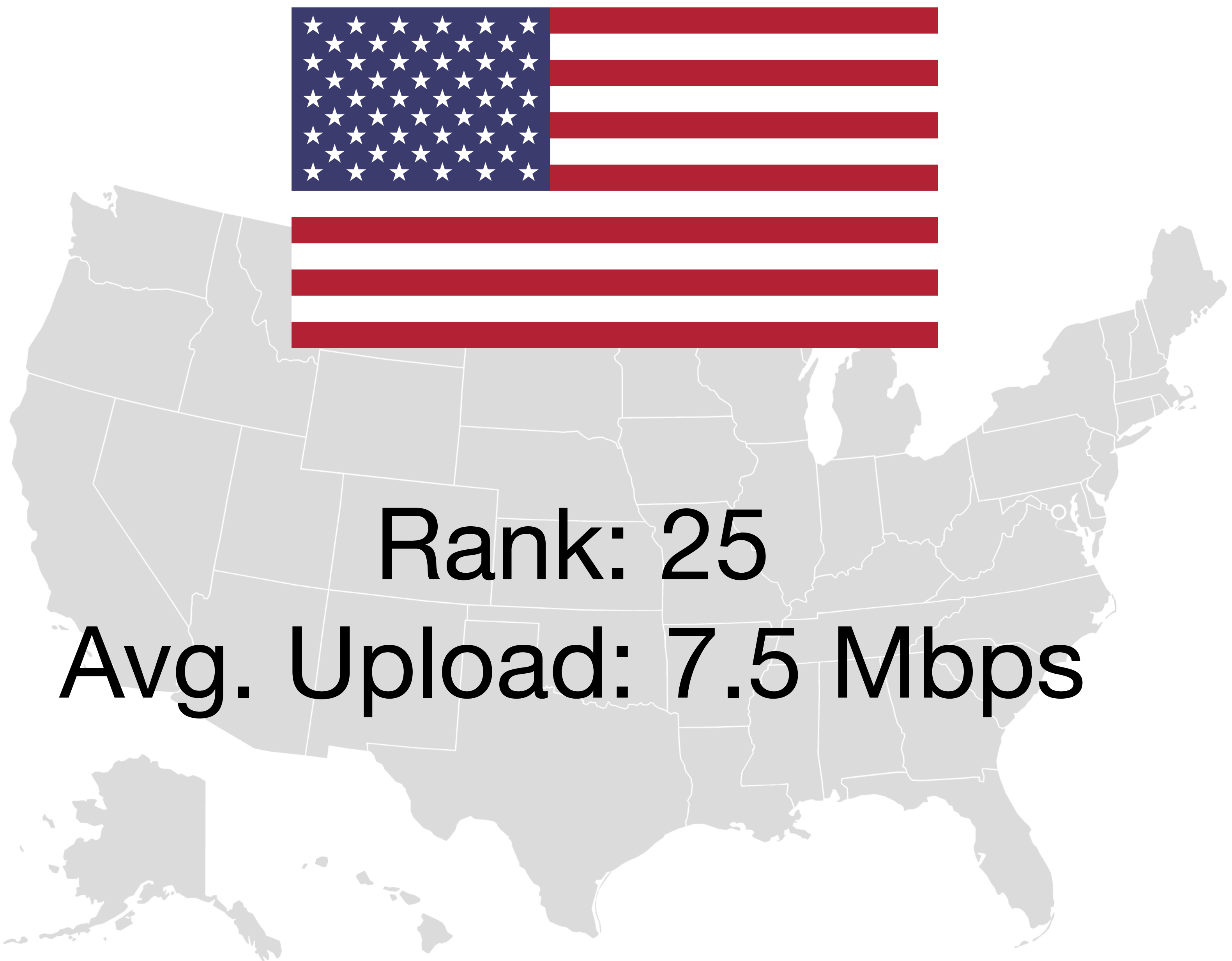


Example 1: Mobile Wallet

Multiplier: OT-based
Parties: 4
Curve: 256-bit

2 Mbits

sent per party



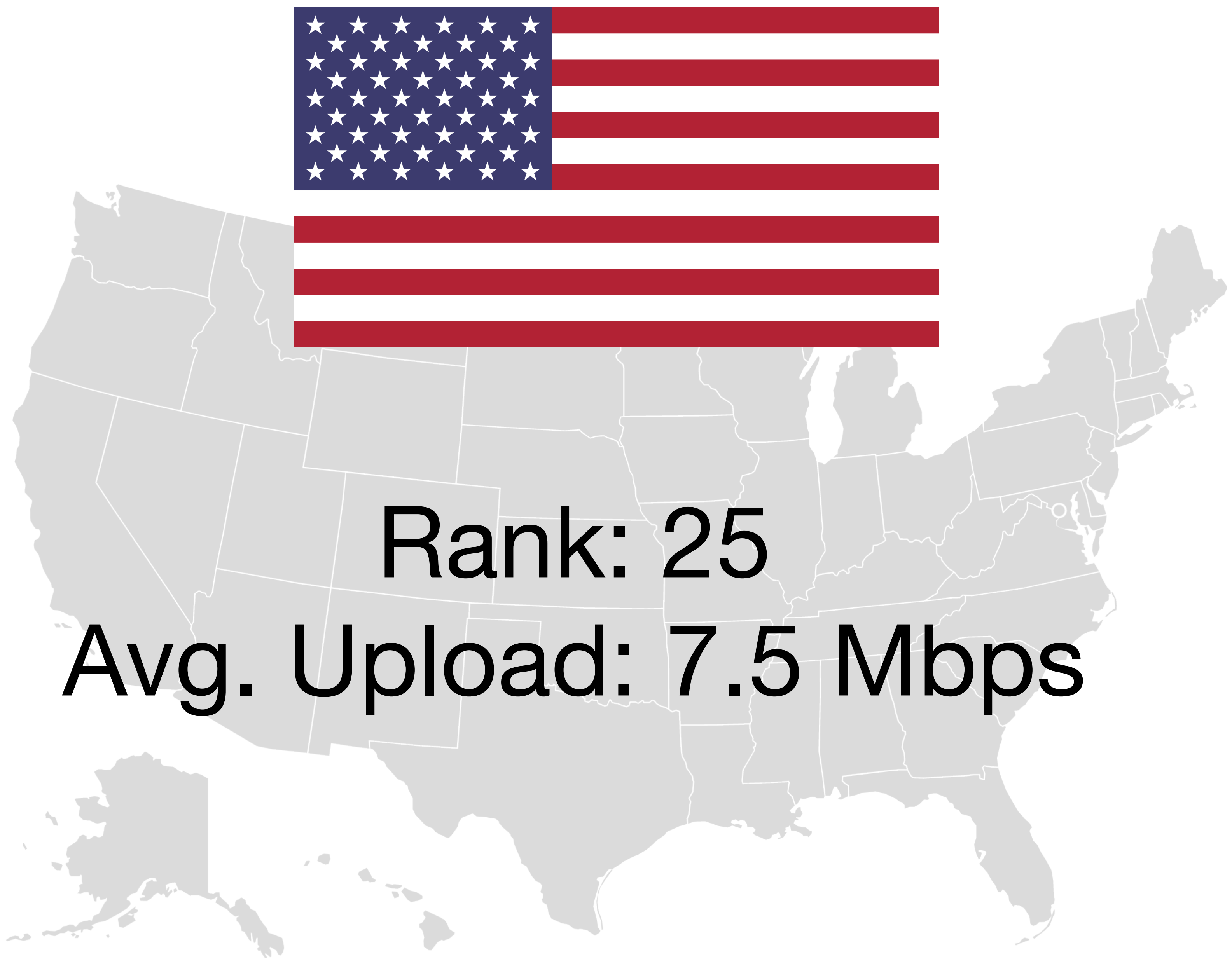
Rank: 25
Avg. Upload: 7.5 Mbps

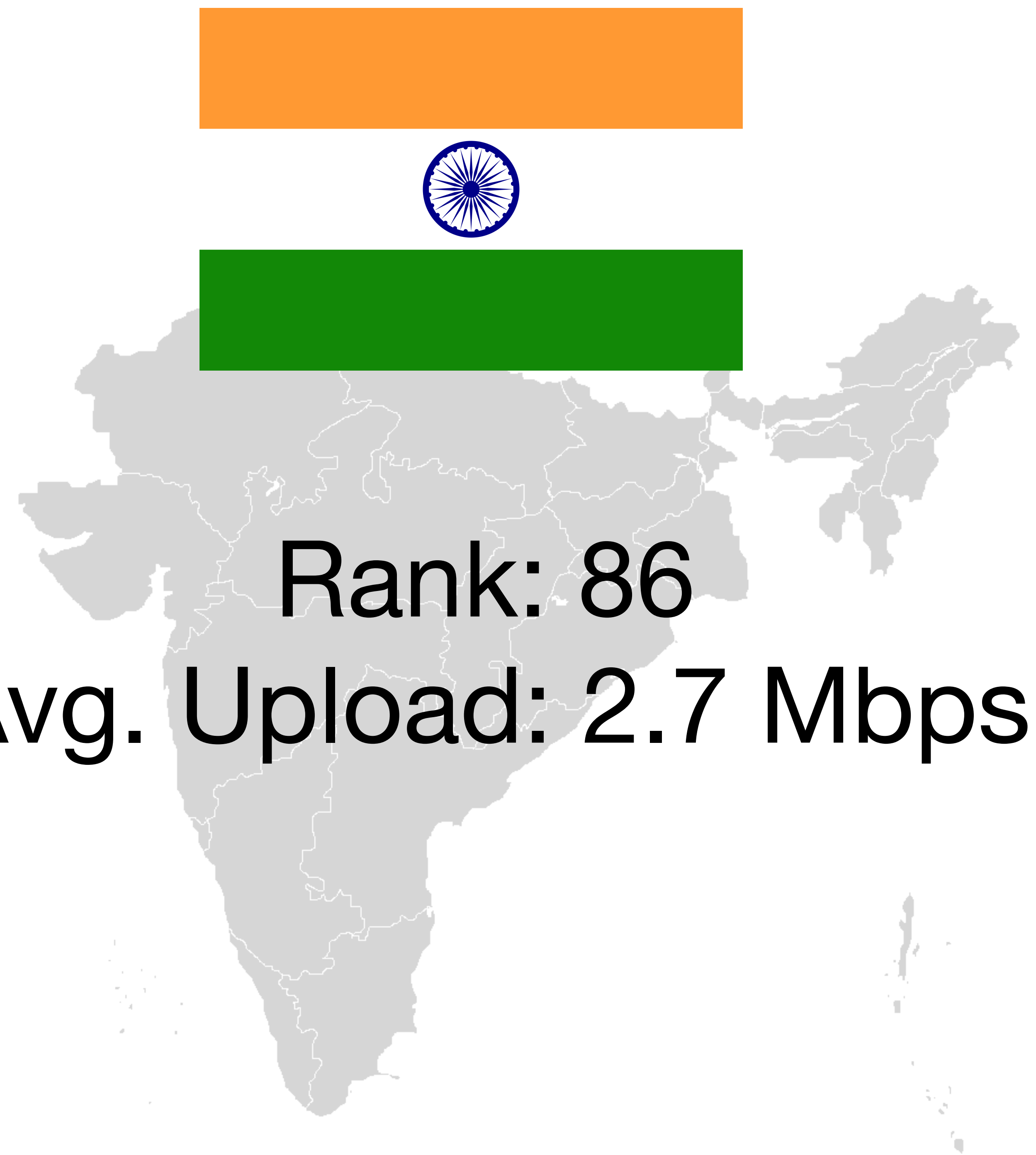
source: opensignal

Example 1: Mobile Wallet

Multiplier: OT-based
Parties: 4
Curve: 256-bit

2 Mbits
sent per party



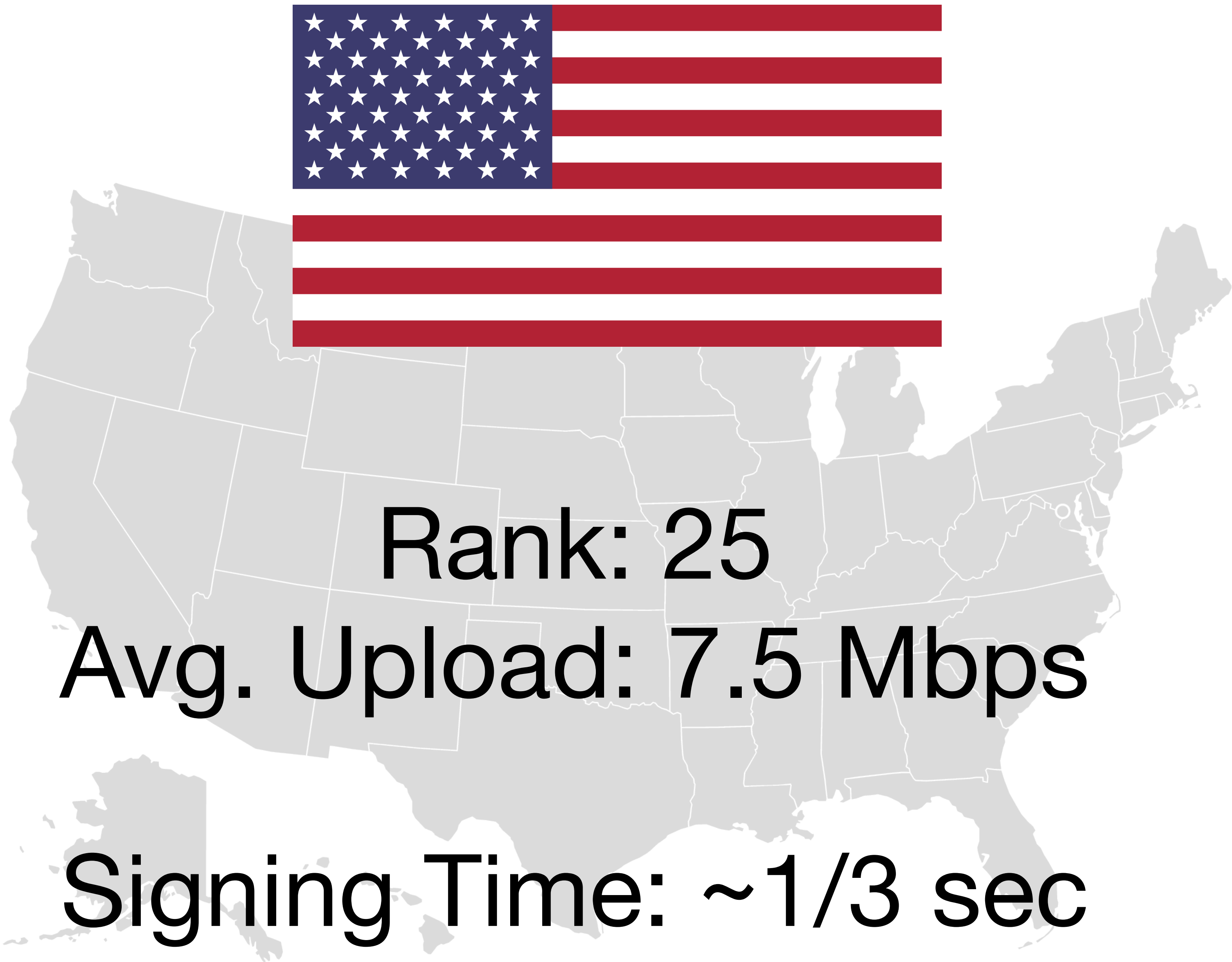


source: opensignal

Example 1: Mobile Wallet

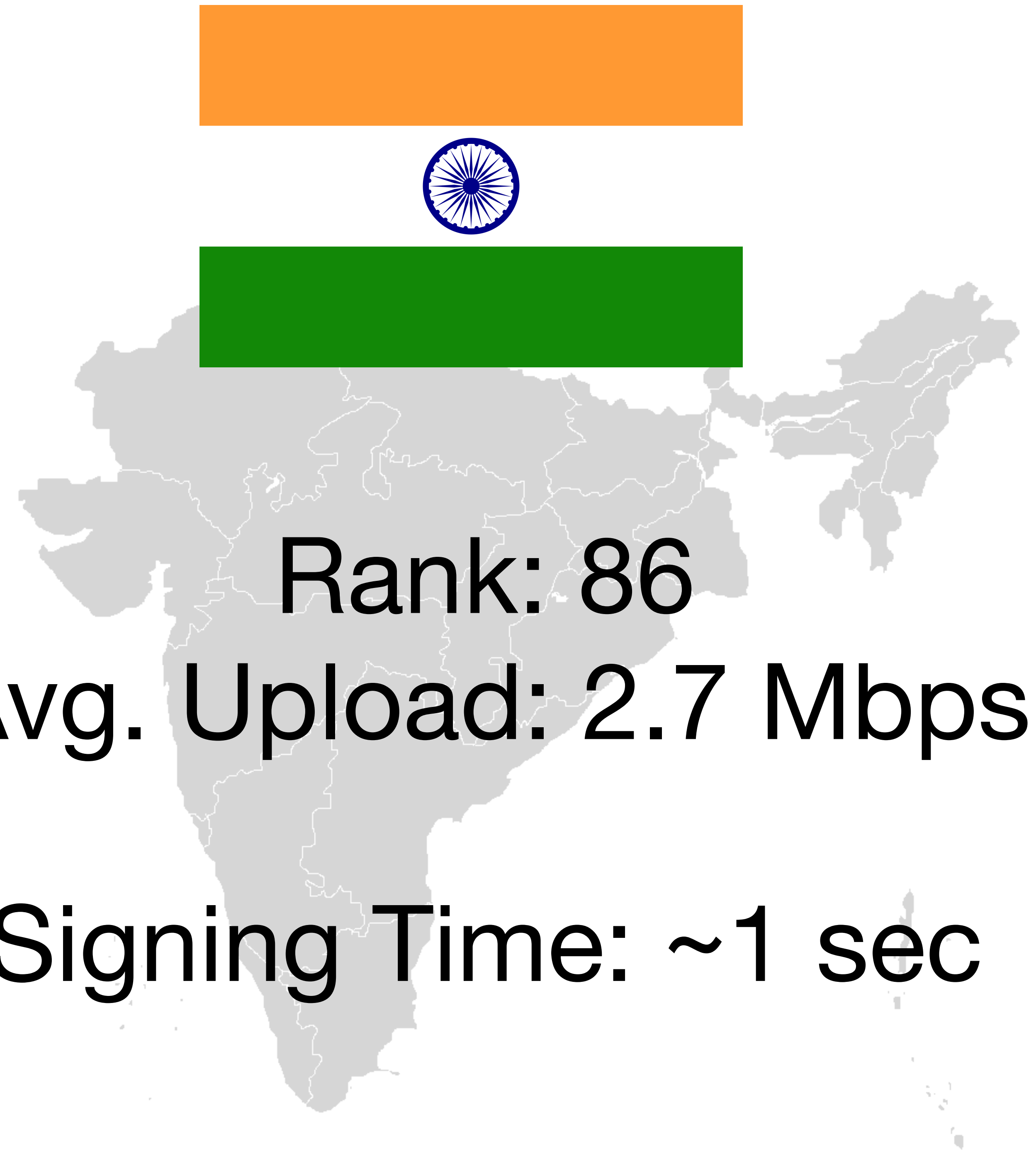
Multiplier: OT-based
Parties: 4
Curve: 256-bit

2 Mbits
sent per party



Rank: 25
Avg. Upload: 7.5 Mbps

Signing Time: ~1/3 sec



Rank: 86
Avg. Upload: 2.7 Mbps

Signing Time: ~1 sec

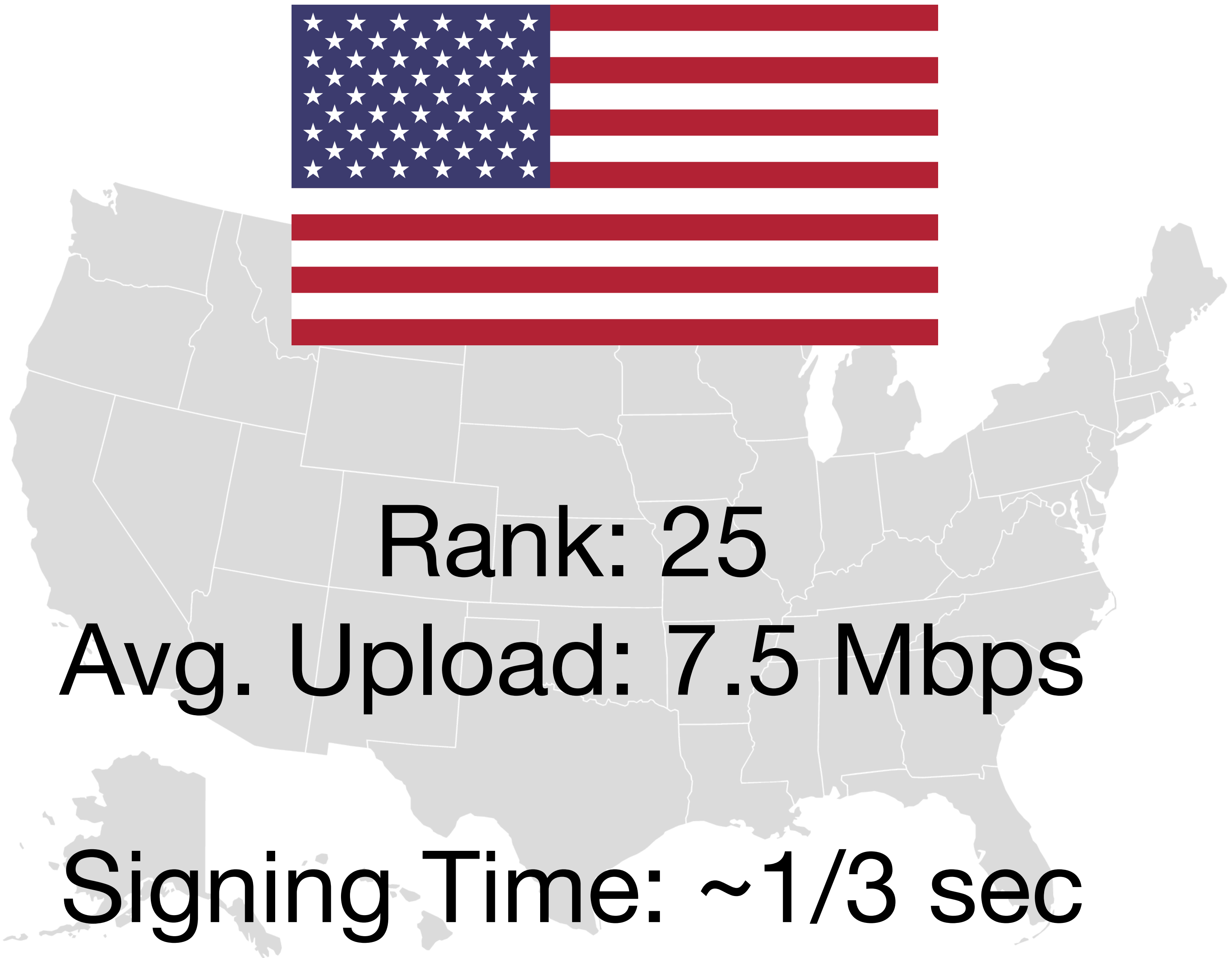
source: opensignal

Example 1: Mobile Wallet

Multiplier: OT-based
Parties: 4
Curve: 256-bit

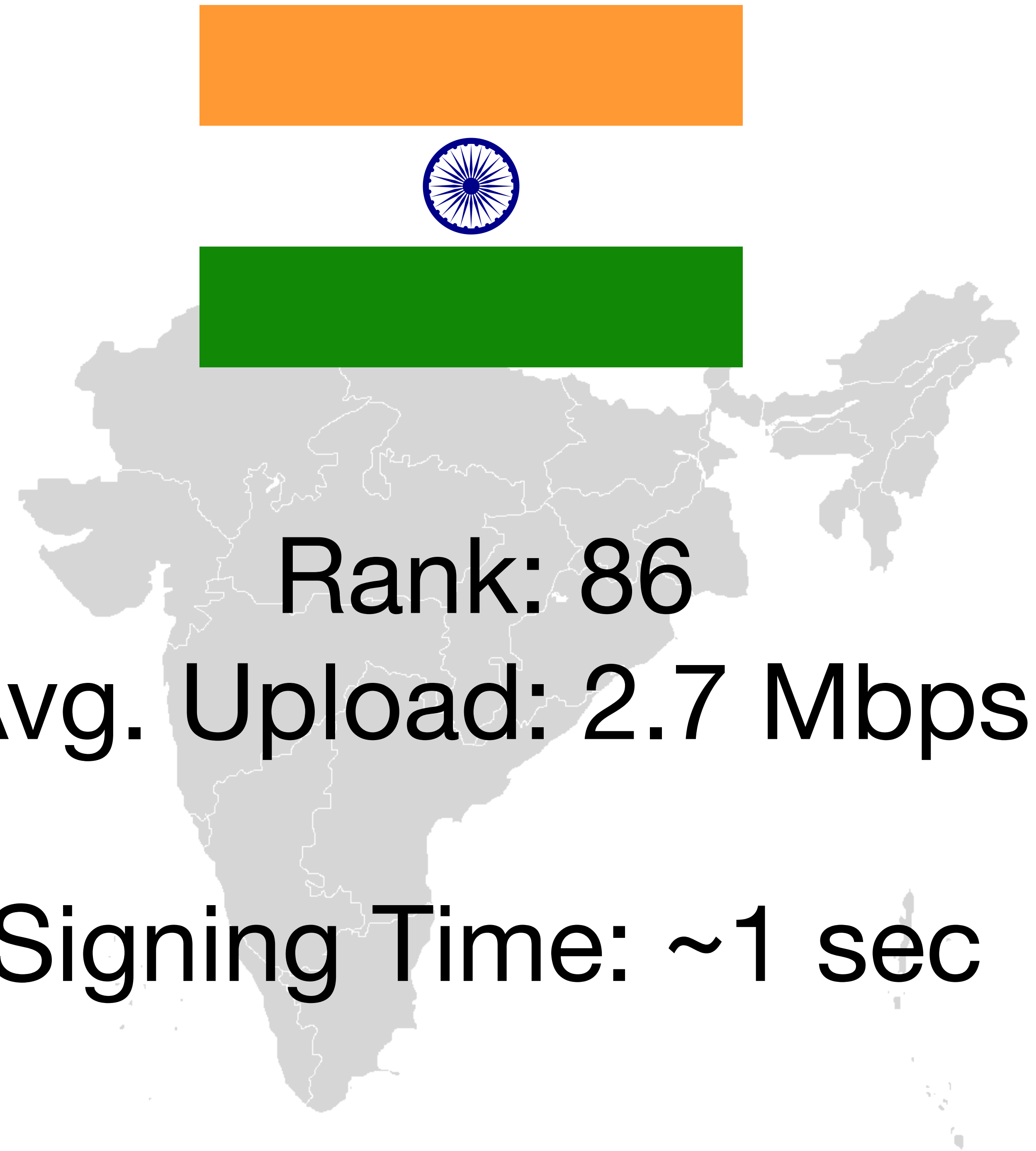
2 Mbits

sent per party



Rank: 25
Avg. Upload: 7.5 Mbps

Signing Time: ~1/3 sec



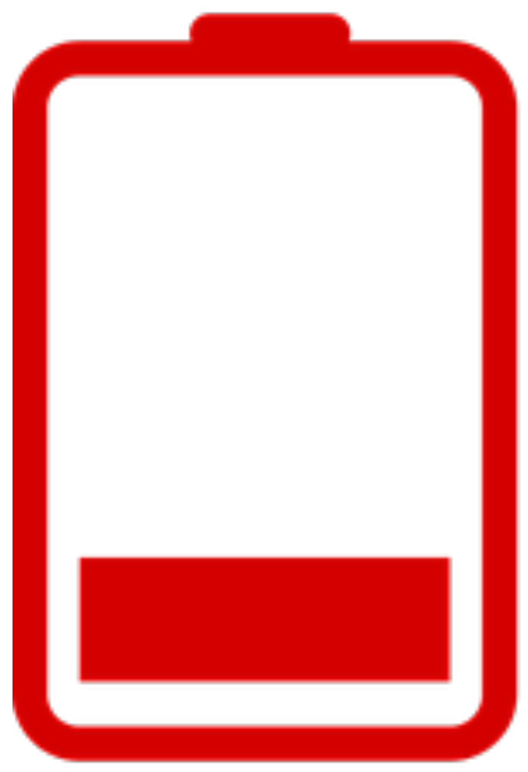
Rank: 86
Avg. Upload: 2.7 Mbps

Signing Time: ~1 sec

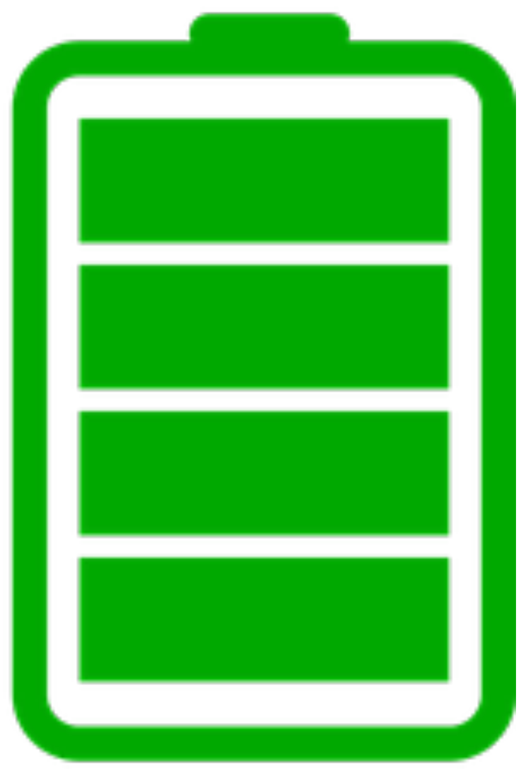
Similar to computation time for Paillier
on powerful hardware!

source: opensignal

On the Other Hand

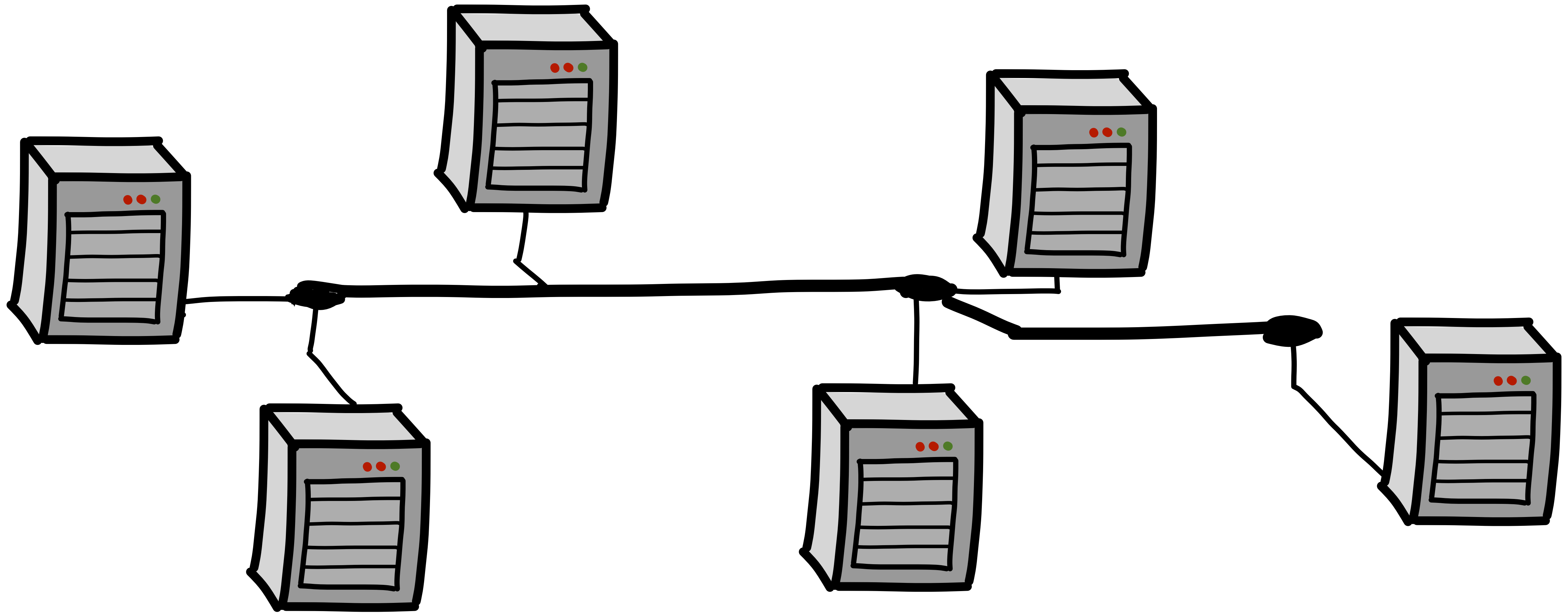


Paillier + ZK

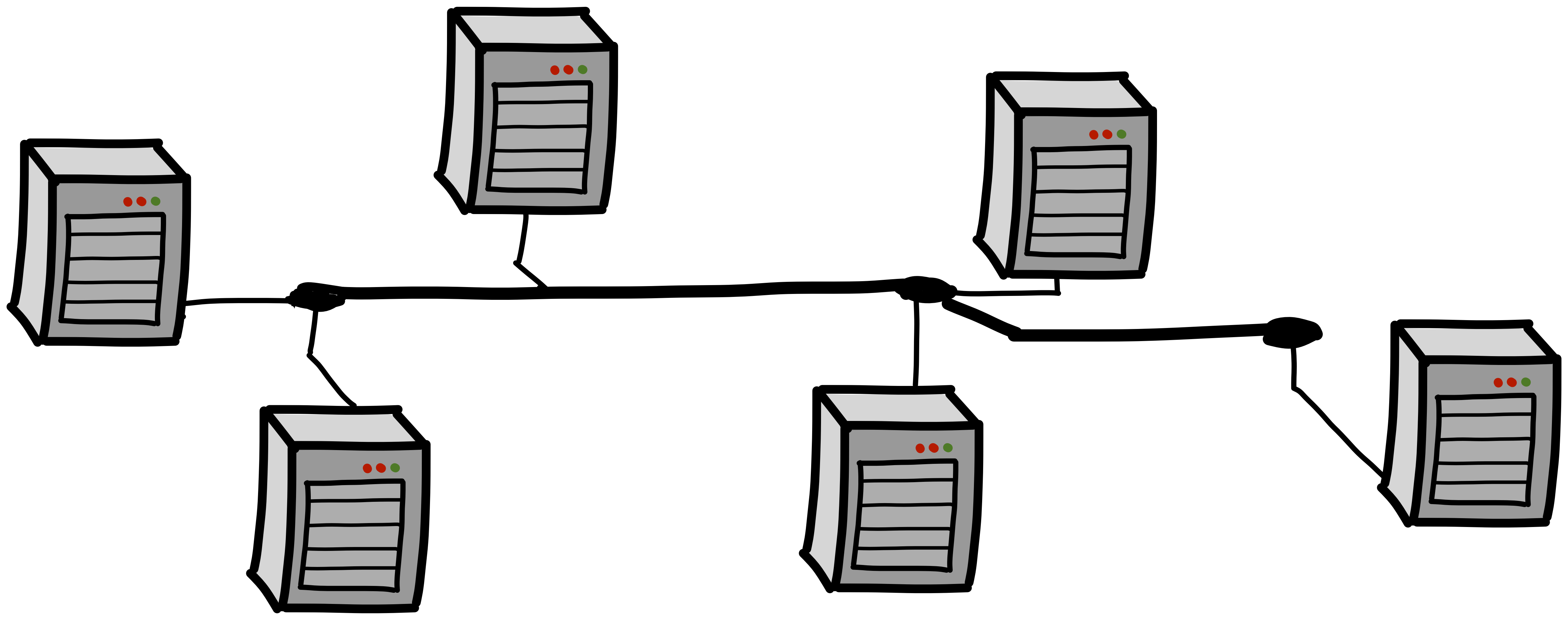


OT

Is communication the bottleneck?

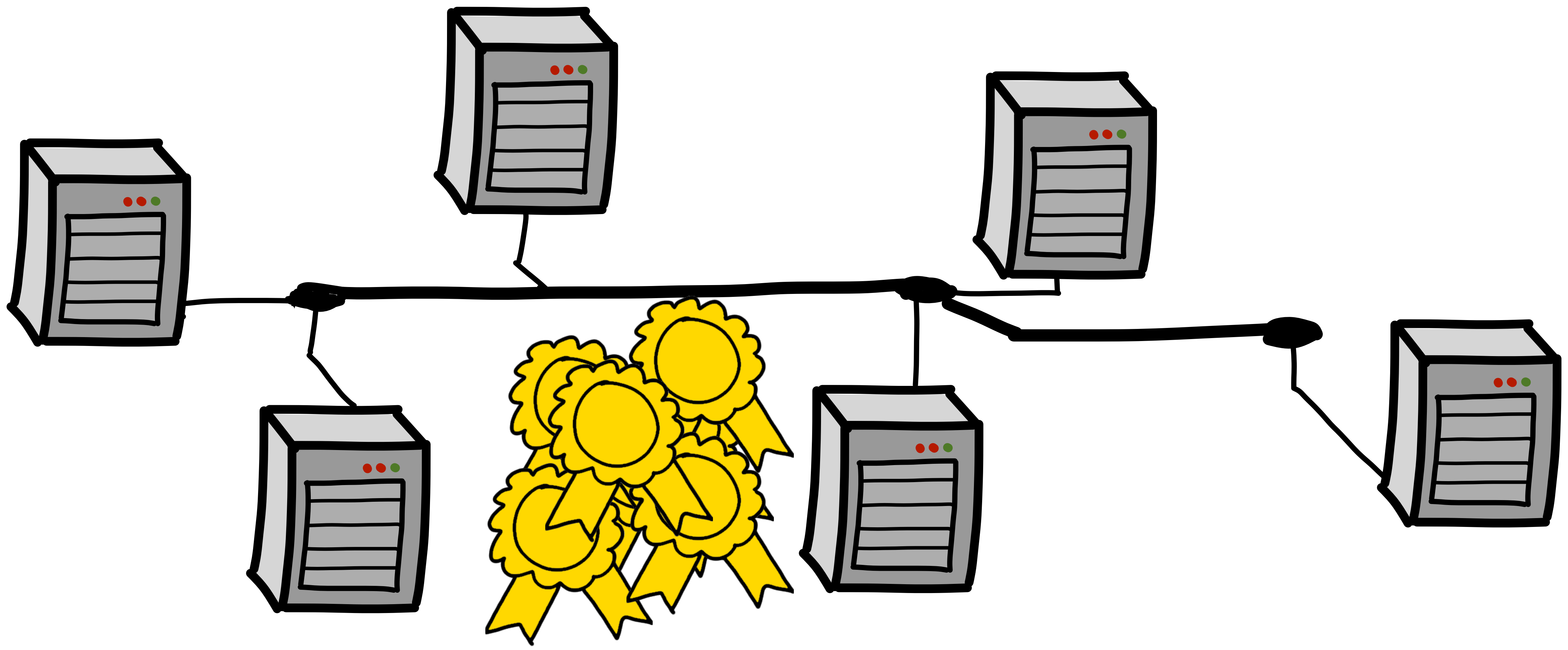


Is communication the bottleneck?



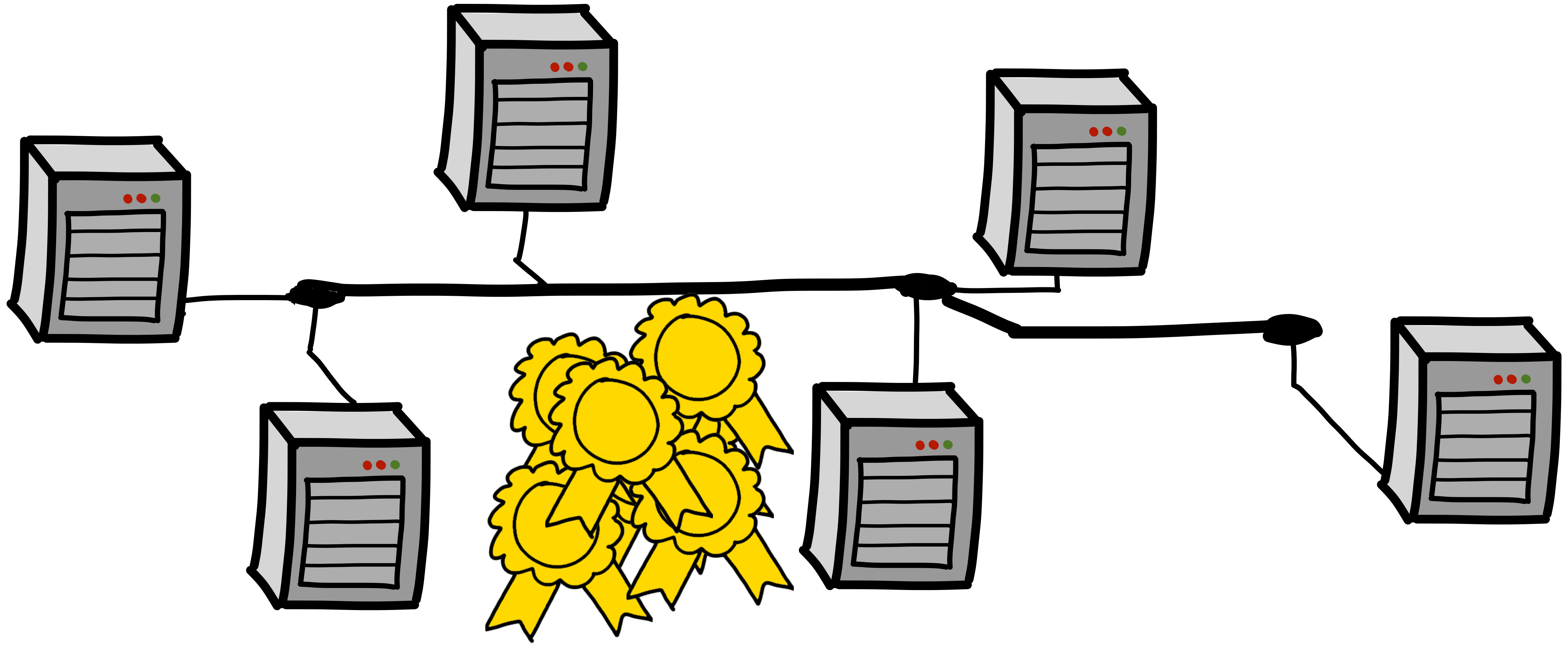
- **Large-scale automated distributed signing:**

Is communication the bottleneck?



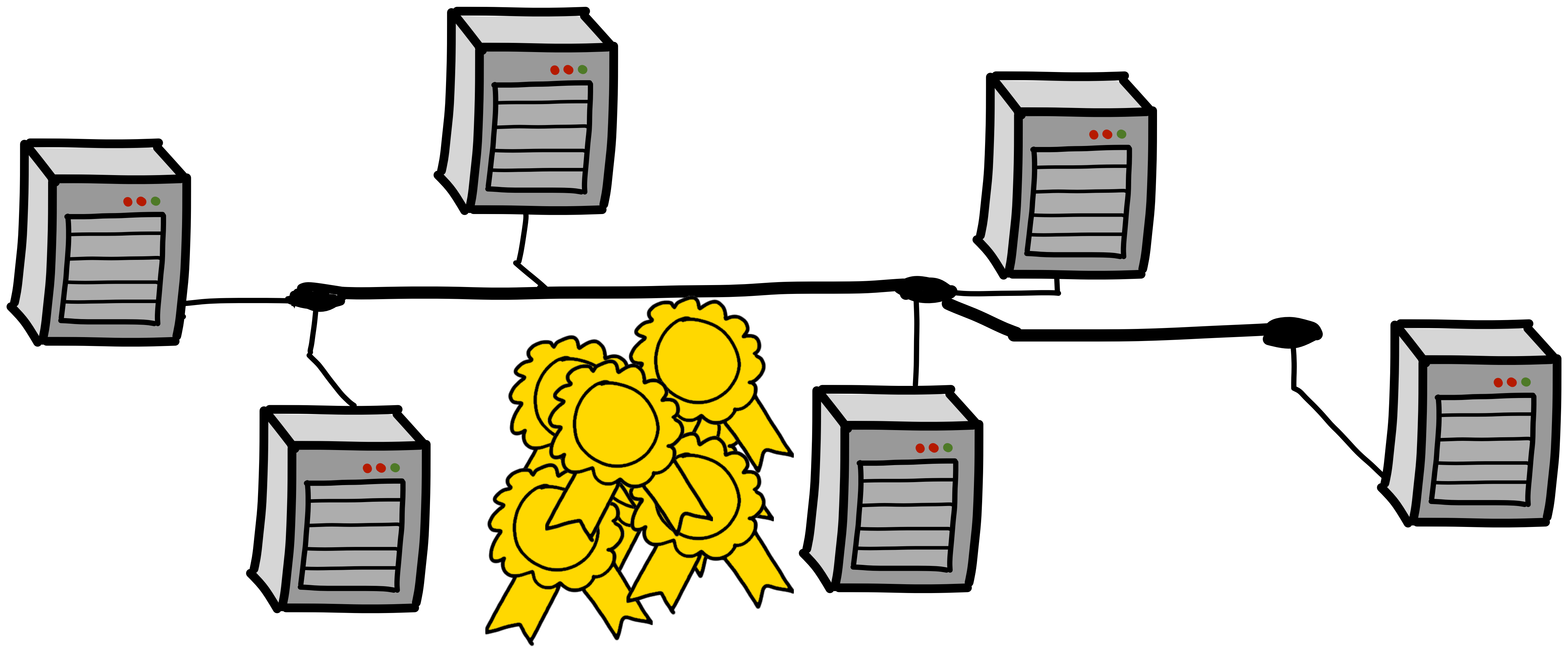
- **Large-scale automated distributed signing:**

Is communication the bottleneck?



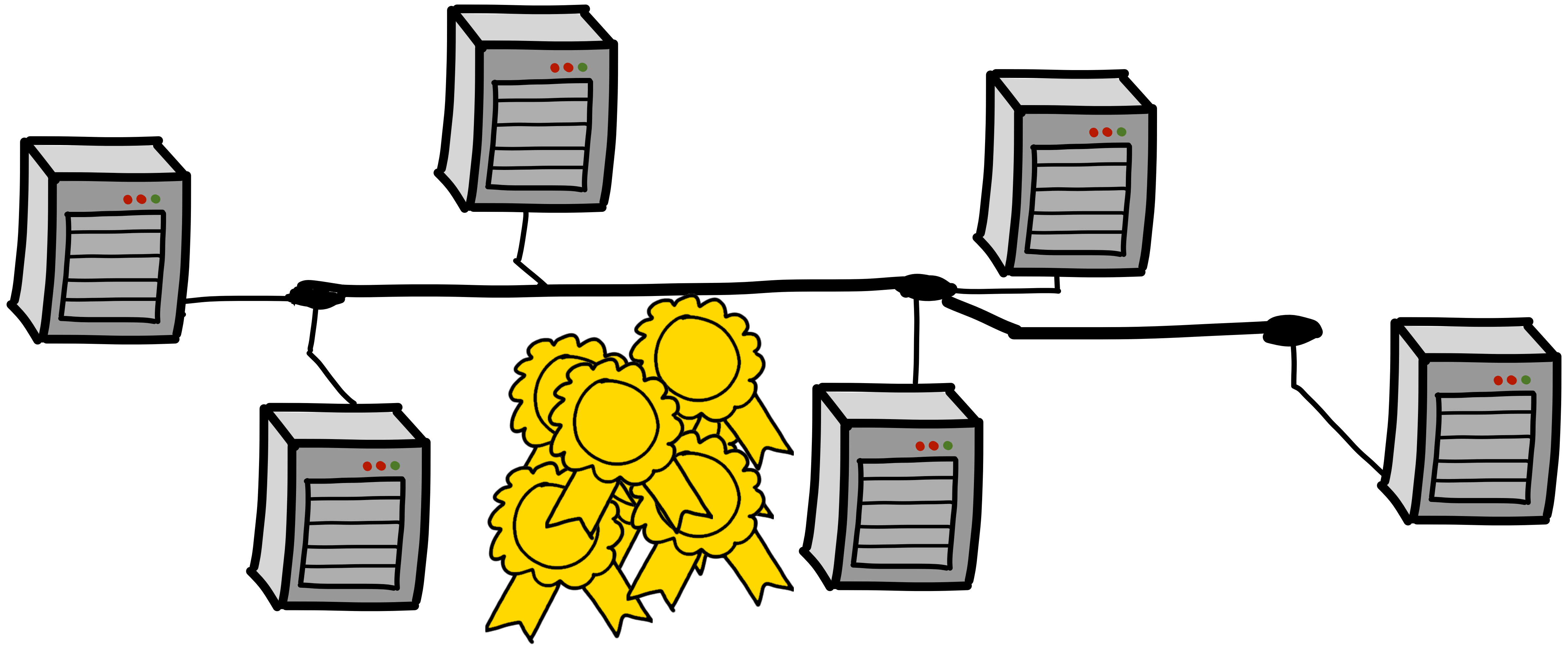
- **Large-scale automated distributed signing:**
  - Threshold 2: 3.8ms/sig <= ~263 sig/second

Is communication the bottleneck?



- **Large-scale automated distributed signing:**
  - Threshold 2: 3.8ms/sig <= ~263 sig/second
  - Threshold 20: 31.6ms/sig <= ~31 sig/second

Is communication the bottleneck?



- **Large-scale automated distributed signing:**
  - Threshold 2: 3.8ms/sig <= ~263 sig/second
  - Threshold 20: 31.6ms/sig <= ~31 sig/second
- Neither setting saturates a gigabit connection

Example 2: Datacenter Signing

How much bandwidth to be CPU bound?
(including preprocessing)

2 Parties
~250 sigs/second

256 Parties
~3 sigs/second

using GCP n1-highcpu nodes

Example 2: Datacenter Signing

How much bandwidth to be CPU bound?
(including preprocessing)

2 Parties
~250 sigs/second

Each party sends:
~700 Kbits per sig

256 Parties
~3 sigs/second

Each party sends:
~185 Mbits per sig

using GCP n1-highcpu nodes

Example 2: Datacenter Signing

How much bandwidth to be CPU bound?
(including preprocessing)

2 Parties
~250 sigs/second

Each party sends:
~700 Kbits per sig

Bandwidth required:
~180 Mbps symmetric

256 Parties
~3 sigs/second

Each party sends:
~185 Mbits per sig

Bandwidth required:
~555 Mbps symmetric

using GCP n1-highcpu nodes

Non-interactive Online Signing

Most Threshold ECDSA protocols have this format ([DOKSS20, CGGMP20] were the first to “use” it)

$\mathsf{Sign}([\mathsf{sk}], m)$

Round 1

⋮

Round $r - 1$

Only this round needs m

Round r

Non-interactive Online Signing

Sign([sk],)

Most Threshold ECDSA protocols have this format ([DOKSS20, CGGMP20] were the first to "use" it)

Round 1

⋮

Round $r - 1$

m is now available

Only this round needs m

Round r

Non-interactive Online Signing

Sign([sk],)

Most Threshold ECDSA protocols have this format ([DOKSS20, CGGMP20] were the first to "use" it)

Round 1

⋮

Round $r-1$

m is now available

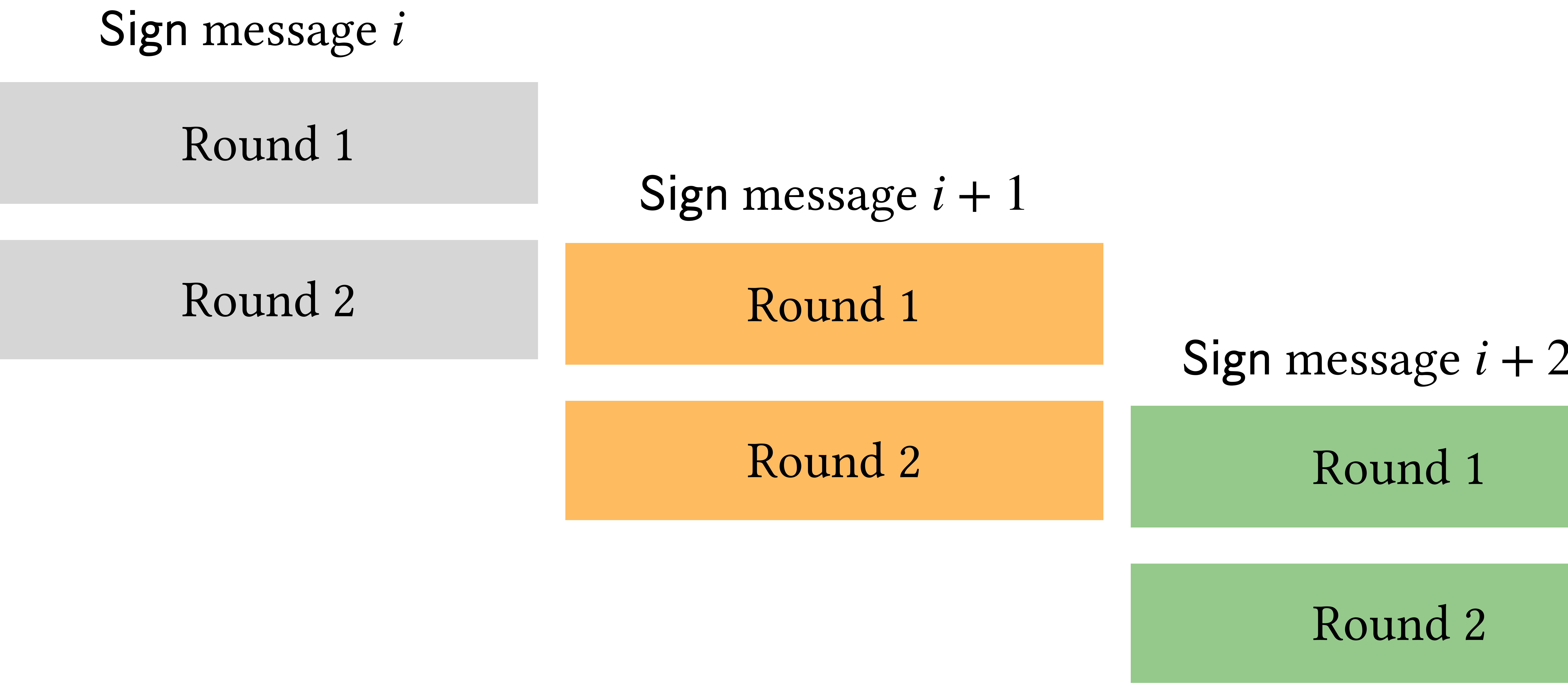
Only this round needs m

Round r

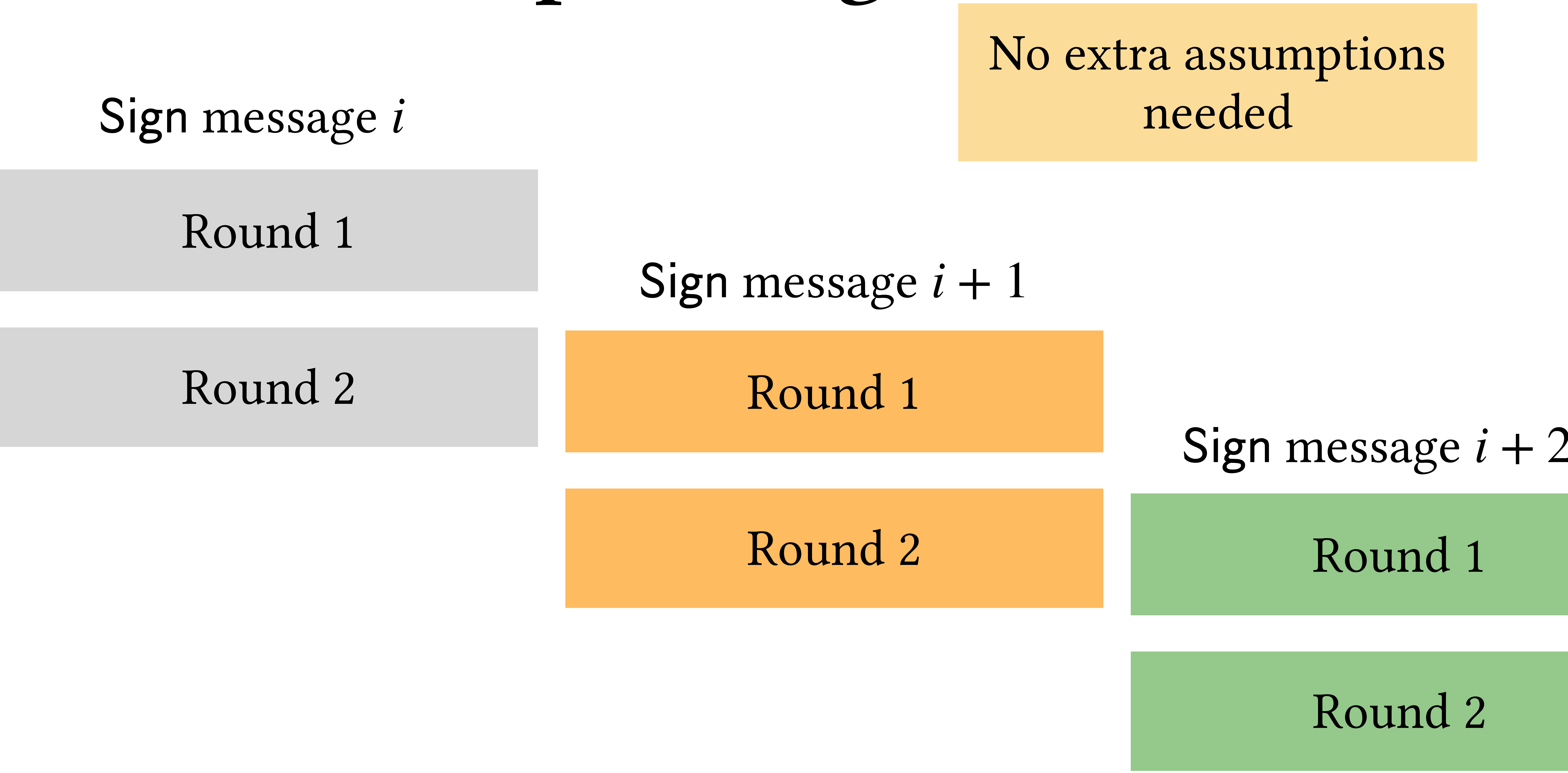
<u>Caveat</u>

Requires a stronger assumption on ECDSA, which is proven to hold in the GGM [Groth Shoup 22]

Pipelining


Sign message i
Round 1
Round 2
Sign message $i + 1$
Round 1
Round 2
Sign message $i + 2$
Round 1
Round 2

Pipelining


No extra assumptions needed
Sign message i
Round 1
Round 2
Sign message $i+1$
Round 1
Round 2
Sign message $i+2$
Round 1
Round 2

Pipelining

No extra assumptions needed

Sign message i

Round 1

Round 2

Sign message $i + 1$

Round 1

Round 2

Sign message $i + 2$

Round 1

Round 2

Saves a round on average

Intro		How to distribute ECDSA			Tradeoffs	
MPC for Schnorr is easy	but not ECDSA	Rewriting ECDSA + 3 round protocol	ECDSA Tuples	2P-MUL + Consistency	OT vs AHE	Modes of operation

In Conclusion

- Threshold ECDSA in Three Rounds: Now matches Schnorr
- Enabled by well-chosen correlation + simple new consistency check
- Blackbox use of UC 2-round 2P-MUL
  NOTE: OT-based protocols satisfy UC, but AHE is more complicated
- No ZK proofs during signing: light protocol and straightforward UC analysis

dkls.info

Thanks!

Thanks **Eysa Lee** for