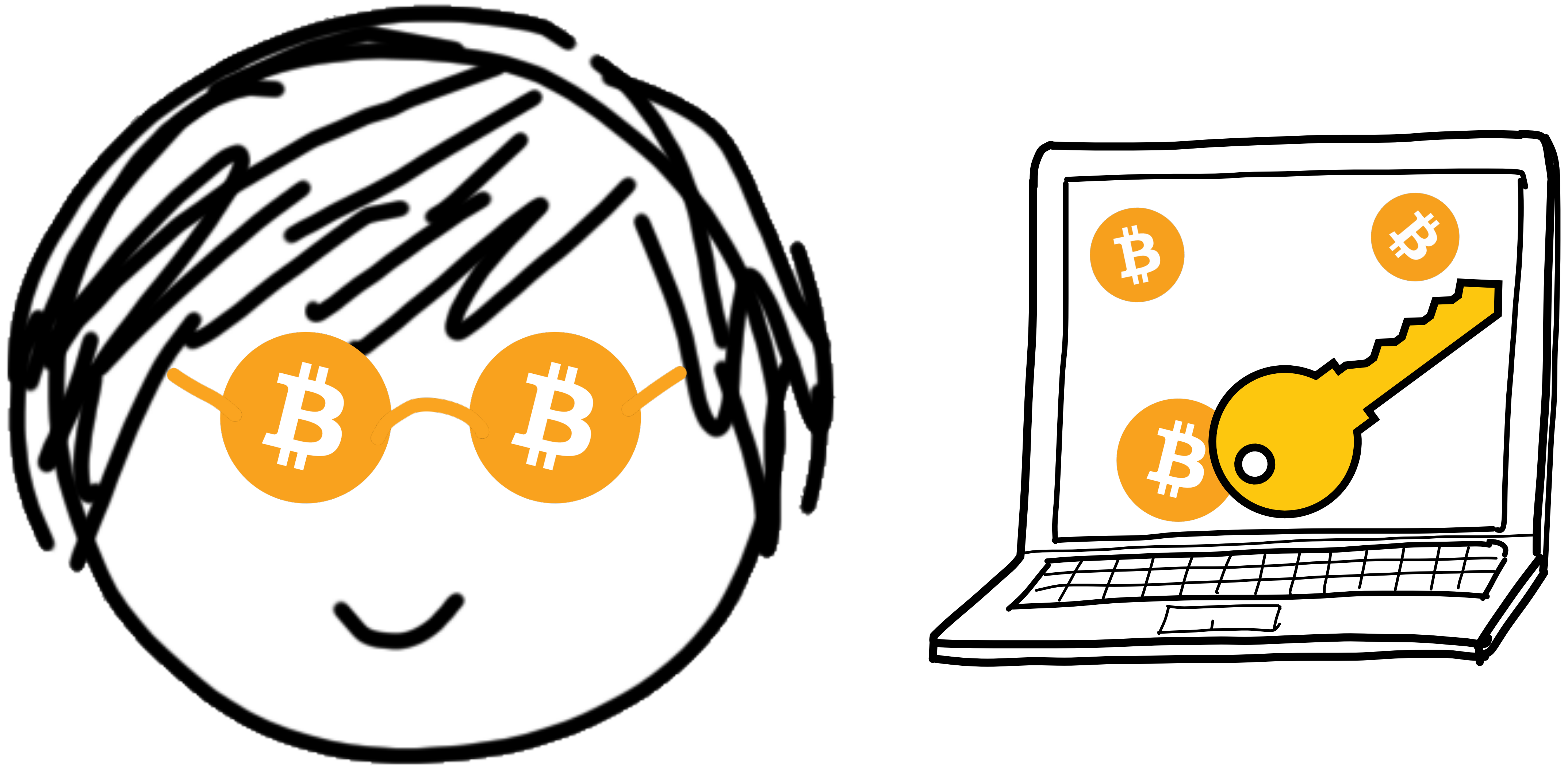
Two-Round Stateless Deterministic Two-Party Schnorr Signatures *from* Pseudorandom Correlation Functions

Yashvanth Kondi Claudio Orlandi Lawrence Roy

AARHUS UNIVERSITY

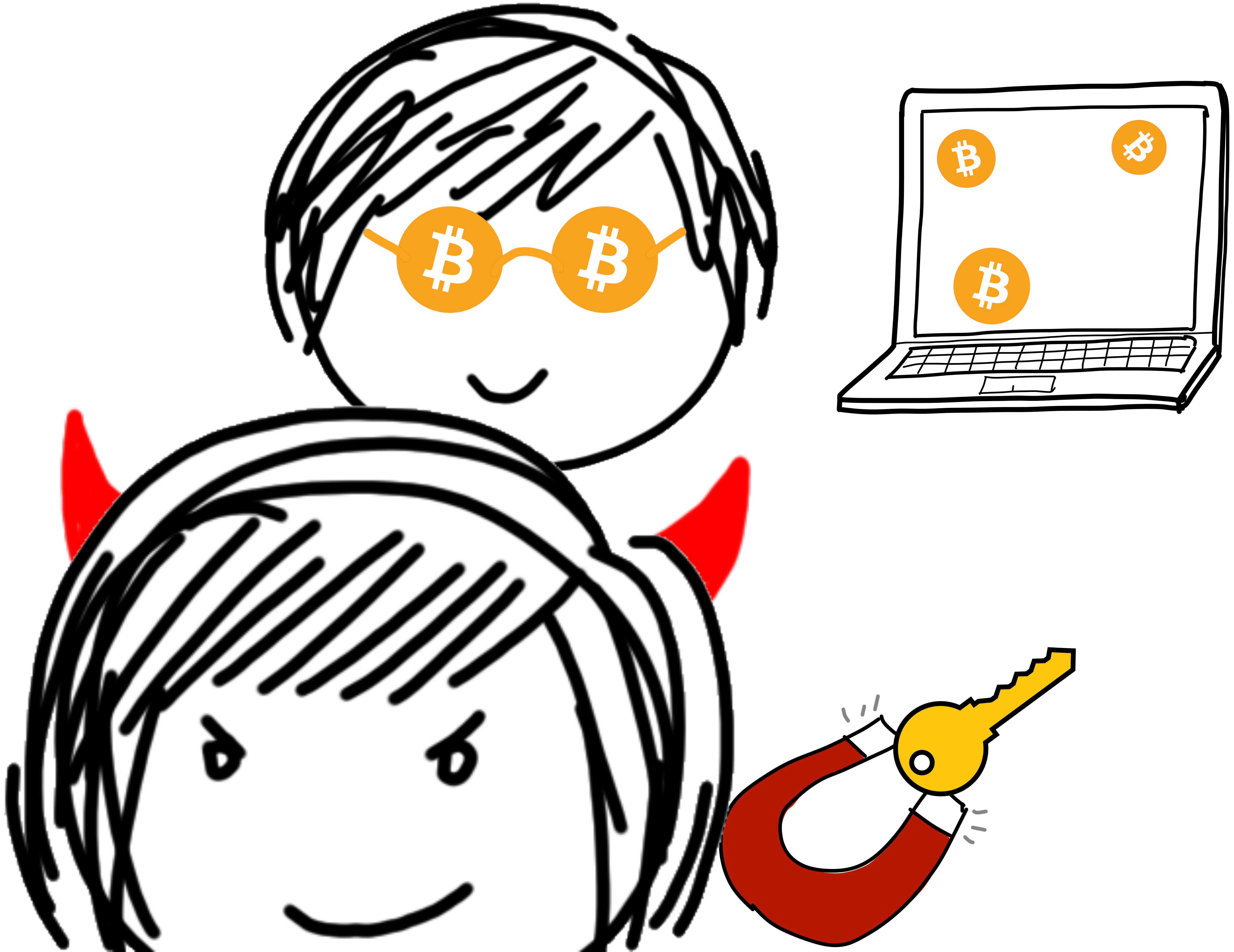
v10

Cryptographic Keys: Valuable Targets



Single point of failure

Cryptographic Keys: Valuable Targets



Single point of failure

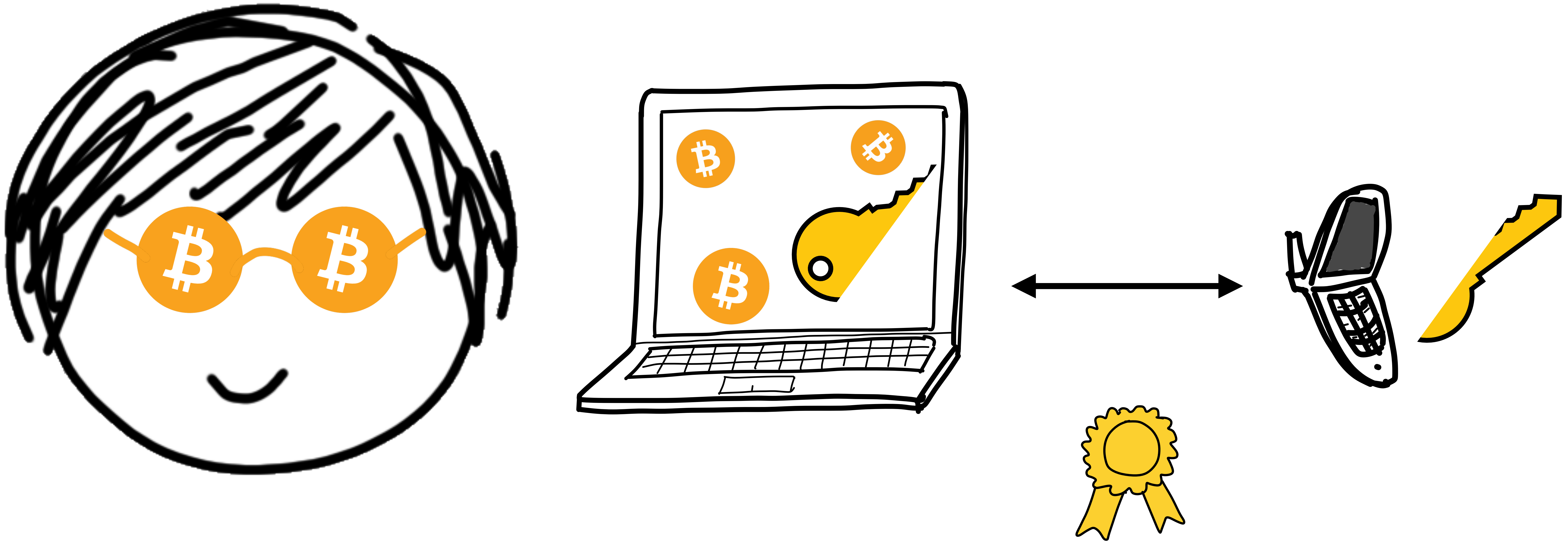
Cryptographic Keys: Valuable Targets

Single point of failure

Threshold Signatures

Distributed signing: Distribute the risk

Threshold Signatures



Distributed signing: Distribute the risk

This Work

- Derandomized Two-party Schnorr Signing w. resilience to state resets

- Conceptual insight: Just as PRFs derandomize plain signing, Pseudorandom *Correlation* Functions natively derandomize distributed signing

- Two constructions, useful tradeoffs relative to prior work

- Bonus (not explored in this talk): two-round signing w. standard assumptions

Schnorr Key Generation

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

secret key: kept private

Public Key: exposed to the outside world

Schnorr Signing

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

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

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

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

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

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

$$R = k \cdot G$$

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

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

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

$$\text{output } \sigma$$

NONCE
One-time use
value

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

Distributing Schnorr Signing

```
SchnorrSign(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 $\sigma$
```

Any linear secret sharing

Linear function of k, sk
Easy to distribute with most natural (i.e. linear) secret sharing schemes

EdDSA

- **Ed**wards-curve **D**igital **S**ignature **A**lgorithm
- Devised by Bernstein, Duif, Lange, Schwabe, and Yang in 2011
- Variant of Schnorr’s signature instantiated with careful choice of parameters
- Widely deployed, and increasing in use

EdDSA is a little different…

- (Distributed) KeyGeneration of EdDSA is identical to Schnorr
- EdDSA signing involves some non-linearity

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

EdDSA is a little different…

- (Distributed) KeyGeneration of EdDSA is identical to Schnorr
- EdDSA signing involves some non-linearity

```
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 $\sigma$
```

```
EdDSASign($\mathsf{sk}, m$) :
    $e = H(R \| m)$
    $s = k - \mathsf{sk} \cdot e$
    $\sigma = (s, R)$
    output $\sigma$
```

EdDSA is a little different…

- (Distributed) KeyGeneration of EdDSA is identical to Schnorr
- EdDSA signing involves some non-linearity

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

```
$\text{EdDSASign}(\mathsf{sk}, m):$
    $k = F(\mathsf{sk}, m)$
    $R = k \cdot G$
    $e = H(R \| m)$
    $s = k - \mathsf{sk} \cdot e$
    $\sigma = (s, R)$
    output $\sigma$
```

EdDSA is a little different…

- (Distributed) KeyGeneration of EdDSA is identical to Schnorr
- EdDSA signing involves some non-linearity

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

```
$\mathrm{EdDSASign}(\mathsf{sk}, m):$
    $k = F(\mathsf{sk}, m)$
    $R = k \cdot G$
    $e = H(R \| m)$
    $s = k - \mathsf{sk} \cdot e$
    $\sigma = (s, R)$
    output $\sigma$
```

Pseudorandom Function

Painful to distribute

Why does EdDSA have non-linear signing?

- Each Schnorr signature requires a fresh, one-time nonce (k, R)

- Security is extremely sensitive to the distribution of k
  [Boneh Venkatesan 96][Howgrave-Graham Smart 01][Bleichenbacher 00]
  [Aranha Novaes Takahashi Tibouchi Yarom 20][Albrecht Heninger 21]

- Major concern in practice: “true” randomness is a scarce resource
  - Errors in implementation
  - Poorly seeded Random Number Generators
  - eg. Sony Playstation hack, Bitcoin theft via repeated nonces

Stateful PRNG?

- Simple derandomization: keep counter, use PRF_{sd}(counter)
  Fresh state ⇒ fresh nonce, but Reused state ⇒ repeated nonce

- Stale state hard to detect in crypto API context

- State reuse can be accidental, or maliciously triggered
  - think of stale snapshots in VMs, power supply interrupts, etc.

- "State continuity" is non-trivial even with trusted hardware

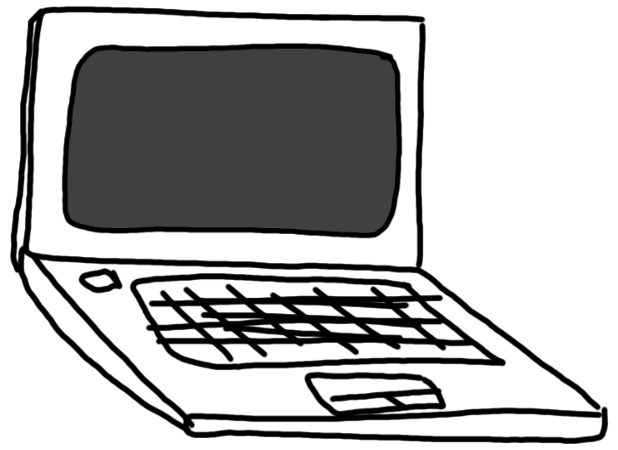
- Ideally, signing should be **stateless**

Stateless Derandomization

- Just as simple:
  - During keygen: $\mathsf{sd} \leftarrow \{0,1\}^\kappa$
  - To sign m: $k = \mathsf{PRF}(\mathsf{sd}, m)$
- Classic idea [M'Raïhi Naccache Pointcheval Vaudenay 98] [Wigley 97] [Barwood 97] that is employed by EdDSA
- Undetectable outside the system
  ⇒ Verification is unchanged
- Stateless derandomized *threshold* Schnorr signing?

Threshold Setting: Simple Attempt

sk_A



sk_B

$$k_\mathsf{A} \leftarrow \mathbb{Z}_q \qquad\qquad k_\mathsf{B} \leftarrow \mathbb{Z}_q$$

$$R_\mathsf{A} = k_\mathsf{A} \cdot G \qquad\qquad R_\mathsf{B} = k_\mathsf{B} \cdot G$$

$$R = R_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R = R_\mathsf{A} + R_\mathsf{B}$$

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

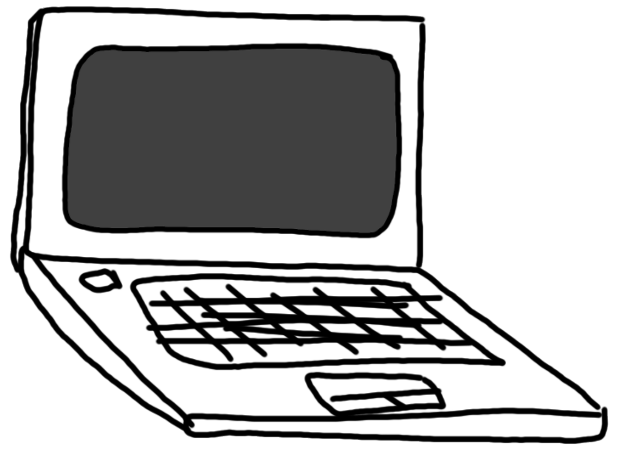
$$s_\mathsf{A} = k_\mathsf{A} - \mathsf{sk}_\mathsf{A} \cdot e \qquad\qquad s_\mathsf{B} = k_\mathsf{B} - \mathsf{sk}_\mathsf{B} \cdot e$$

$$s = s_\mathsf{A} + s_\mathsf{B} \longleftrightarrow s = s_\mathsf{A} + s_\mathsf{B}$$

Threshold Setting: Simple Attempt

sk_A

sk_B



Like plain signing, this is the only randomized step

$k_A \leftarrow \mathbb{Z}_q$ $\qquad$ $k_B \leftarrow \mathbb{Z}_q$

$R_A = k_A \cdot G$ $\qquad$ $R_B = k_B \cdot G$

$$R = R_A + R_B \longleftrightarrow R = R_A + R_B$$

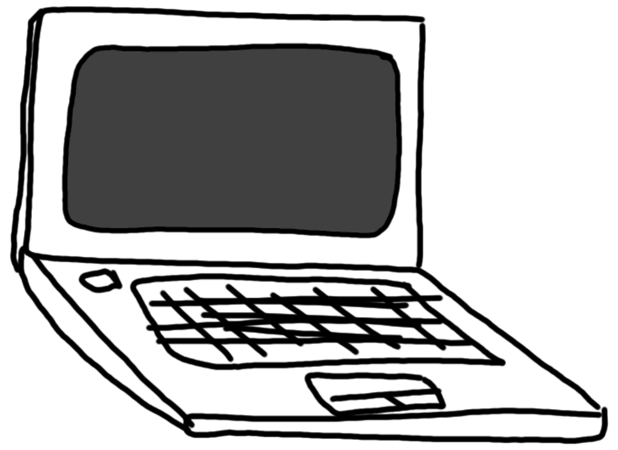
$e = H(R \| m)$ $\qquad$ $e = H(R \| m)$

$s_A = k_A - \mathsf{sk}_A \cdot e$ $\qquad$ $s_B = k_B - \mathsf{sk}_B \cdot e$

$$s = s_A + s_B \longleftrightarrow s = s_A + s_B$$

Threshold Setting: Simple Attempt

sk_A sd_A | sd_B sk_B



Like plain signing, this is the only randomized step

$$k_\mathsf{A} \leftarrow \mathbb{Z}_q \qquad\qquad k_\mathsf{B} \leftarrow \mathbb{Z}_q$$

$$R_\mathsf{A} = k_\mathsf{A} \cdot G \qquad\qquad R_\mathsf{B} = k_\mathsf{B} \cdot G$$

$$R = R_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R = R_\mathsf{A} + R_\mathsf{B}$$

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

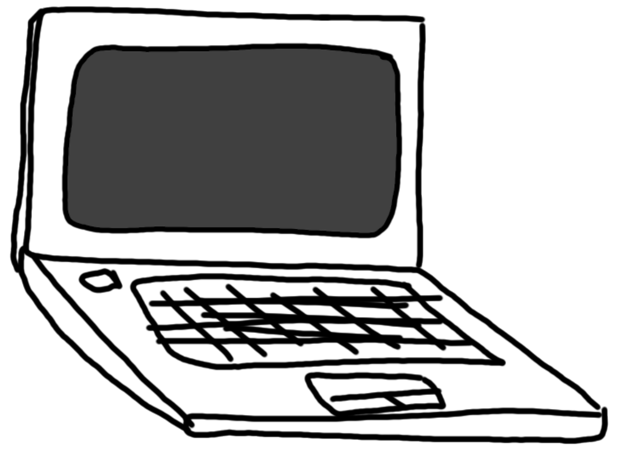
$$s_\mathsf{A} = k_\mathsf{A} - \mathsf{sk}_\mathsf{A} \cdot e \qquad\qquad s_\mathsf{B} = k_\mathsf{B} - \mathsf{sk}_\mathsf{B} \cdot e$$

$$s = s_\mathsf{A} + s_\mathsf{B} \longleftrightarrow s = s_\mathsf{A} + s_\mathsf{B}$$

Threshold Setting: Simple Attempt

sk_A sd_A

sd_B sk_B



Like plain signing, this is the only randomized step

$k_\mathsf{A} = F(\mathsf{sd}_\mathsf{A}, m)$ $\quad$ $k_\mathsf{B} = F(\mathsf{sd}_\mathsf{B}, m)$

$R_\mathsf{A} = k_\mathsf{A} \cdot G$ $\quad$ $R_\mathsf{B} = k_\mathsf{B} \cdot G$

$R = R_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R = R_\mathsf{A} + R_\mathsf{B}$

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

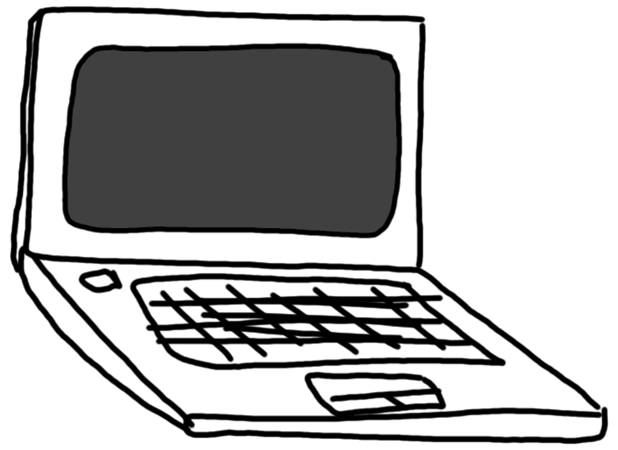
$s_\mathsf{A} = k_\mathsf{A} - \mathsf{sk}_\mathsf{A} \cdot e$ $\quad$ $s_\mathsf{B} = k_\mathsf{B} - \mathsf{sk}_\mathsf{B} \cdot e$

$s = s_\mathsf{A} + s_\mathsf{B} \longleftrightarrow s = s_\mathsf{A} + s_\mathsf{B}$

Threshold Setting: Simple Attempt

sk_A sd_A

sd_B sk_B



Like plain signing, this is the only randomized step

$k_\mathsf{A} = F(\mathsf{sd}_\mathsf{A}, m)$ $\qquad$ $k_\mathsf{B} = F(\mathsf{sd}_\mathsf{B}, m)$

$R_\mathsf{A} = k_\mathsf{A} \cdot G$ $\qquad$ $R_\mathsf{B} = k_\mathsf{B} \cdot G$

$R = R_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R = R_\mathsf{A} + R_\mathsf{B}$

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

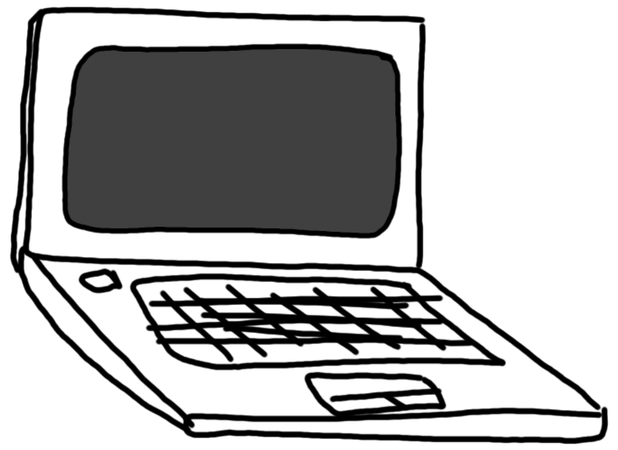
$s_\mathsf{A} = k_\mathsf{A} - \mathsf{sk}_\mathsf{A} \cdot e$ $\qquad$ $s_\mathsf{B} = k_\mathsf{B} - \mathsf{sk}_\mathsf{B} \cdot e$

$s = s_\mathsf{A} + s_\mathsf{B} \longleftrightarrow s = s_\mathsf{A} + s_\mathsf{B}$

Threshold Setting: Simple Attempt

sk_A sd_A

sd_B sk_B



Like plain signing, this is the only randomized step

$$k_\mathsf{A} = F(\mathsf{sd}_\mathsf{A}, m)$$

$$R_\mathsf{A} = k_\mathsf{A} \cdot G$$

$$R = R_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R = R_\mathsf{A} + R_\mathsf{B}$$

$$\boxed{\begin{aligned} k_\mathsf{B} &= F(\mathsf{sd}_\mathsf{B}, m) \\ R_\mathsf{B} &= k_\mathsf{B} \cdot G \end{aligned}}$$

Sign same m again

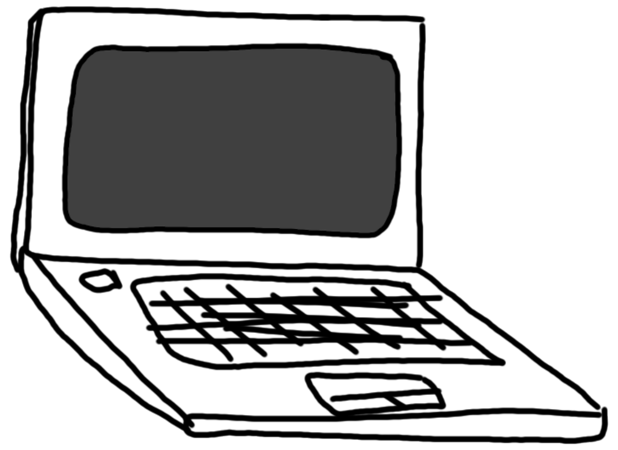
These stay the same

$$\boxed{\begin{aligned} k_\mathsf{B} &= F(\mathsf{sd}_\mathsf{B}, m) \\ R_\mathsf{B} &= k_\mathsf{B} \cdot G \end{aligned}}$$

Threshold Setting: Simple Attempt

sk_A sd_A

sd_B sk_B



Like plain signing, this is the only randomized step

$$k_\mathsf{A} = F(\mathsf{sd}_\mathsf{A}, m)$$
$$R_\mathsf{A} = k_\mathsf{A} \cdot G$$

$$\boxed{\begin{aligned} k_\mathsf{B} &= F(\mathsf{sd}_\mathsf{B}, m) \\ R_\mathsf{B} &= k_\mathsf{B} \cdot G \end{aligned}}$$

$$R = R_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R = R_\mathsf{A} + R_\mathsf{B}$$

Sign same m again

These stay the same

This changes

$$\boxed{\begin{aligned} k^*_\mathsf{A} &= F^*(\mathsf{sd}_\mathsf{A}, m) \\ R^*_\mathsf{A} &= k^*_\mathsf{A} \cdot G \end{aligned}}$$

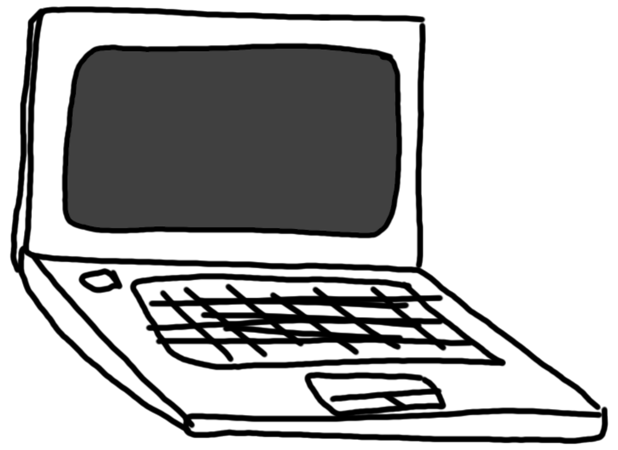
$$\boxed{\begin{aligned} k_\mathsf{B} &= F(\mathsf{sd}_\mathsf{B}, m) \\ R_\mathsf{B} &= k_\mathsf{B} \cdot G \end{aligned}}$$

$$R^* = R^*_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R^* = R^*_\mathsf{A} + R_\mathsf{B}$$

Threshold Setting: Simple Attempt

sk_A sd_A

sd_B sk_B



Like plain signing, this is the only randomized step

$$k_\mathsf{A} = F(\mathsf{sd}_\mathsf{A}, m)$$

$$R_\mathsf{A} = k_\mathsf{A} \cdot G$$

$$\boxed{\begin{aligned} k_\mathsf{B} &= F(\mathsf{sd}_\mathsf{B}, m) \\ R_\mathsf{B} &= k_\mathsf{B} \cdot G \end{aligned}}$$

$$R = R_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R = R_\mathsf{A} + R_\mathsf{B}$$

Sign same m again

These stay the same

This changes

$$\boxed{\begin{aligned} k^*_\mathsf{A} &= F^*(\mathsf{sd}_\mathsf{A}, m) \\ R^*_\mathsf{A} &= k^*_\mathsf{A} \cdot G \end{aligned}}$$

$$\boxed{\begin{aligned} k_\mathsf{B} &= F(\mathsf{sd}_\mathsf{B}, m) \\ R_\mathsf{B} &= k_\mathsf{B} \cdot G \end{aligned}}$$

$$R^* = R^*_\mathsf{A} + R_\mathsf{B} \longleftrightarrow R^* = R^*_\mathsf{A} + R_\mathsf{B}$$

collects

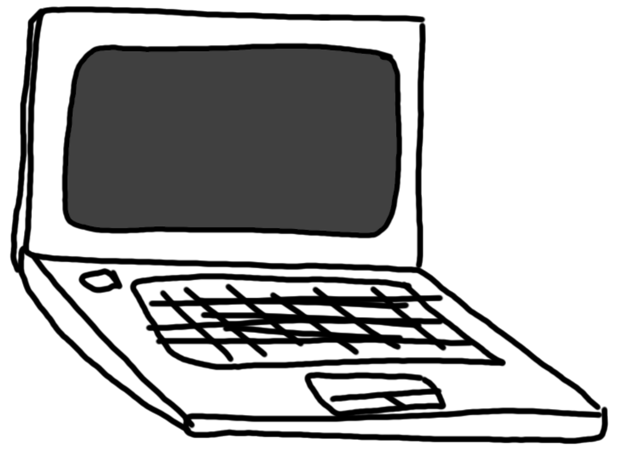
$$s_\mathsf{B} = k_\mathsf{B} - \mathsf{sk}_\mathsf{B} \cdot e$$

$$s^*_\mathsf{B} = k_\mathsf{B} - \mathsf{sk}_\mathsf{B} \cdot e^*$$

2 linear combinations of honest party's 2 secrets

[Maxwell Poelstra Seurin Wuille 19]

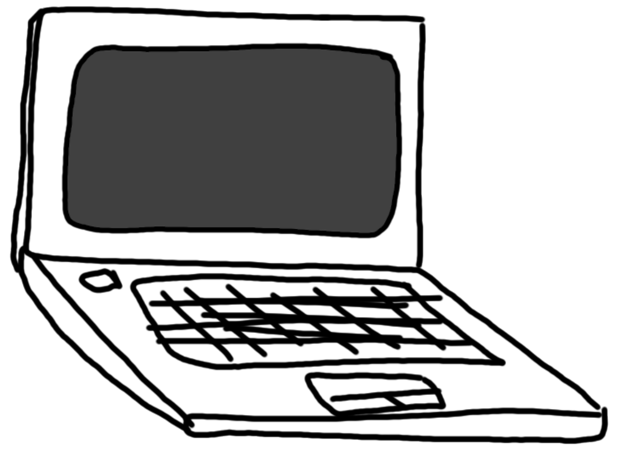
Threshold Setting: Take 2



Need to verify this is done correctly

sk_A sd_A		sd_B sk_B
$k_A = F(\mathsf{sd}_A, m)$		$k_B = F(\mathsf{sd}_B, m)$
$R_A = k_A \cdot G$		$R_B = k_B \cdot G$
$R = R_A + R_B$	⟷	$R = R_A + R_B$
$e = H(R \| m)$		$e = H(R \| m)$
$s_A = k_A - \mathsf{sk}_A \cdot e$		$s_B = k_B - \mathsf{sk}_B \cdot e$
$s = s_A + s_B$	⟷	$s = s_A + s_B$

Threshold Setting: Take 2



$\mathsf{sk_A}$ $\mathsf{sd_A}$ | $\mathsf{sd_B}$ $\mathsf{sk_B}$

$\mathsf{Com(sd_A)}$

$\mathsf{Com(sd_B)}$

Need to verify this is done correctly

$k_A = F(\mathsf{sd_A}, m)$ | $k_B = F(\mathsf{sd_B}, m)$

$R_A = k_A \cdot G$ | $R_B = k_B \cdot G$

$R = R_A + R_B \longleftrightarrow R = R_A + R_B$

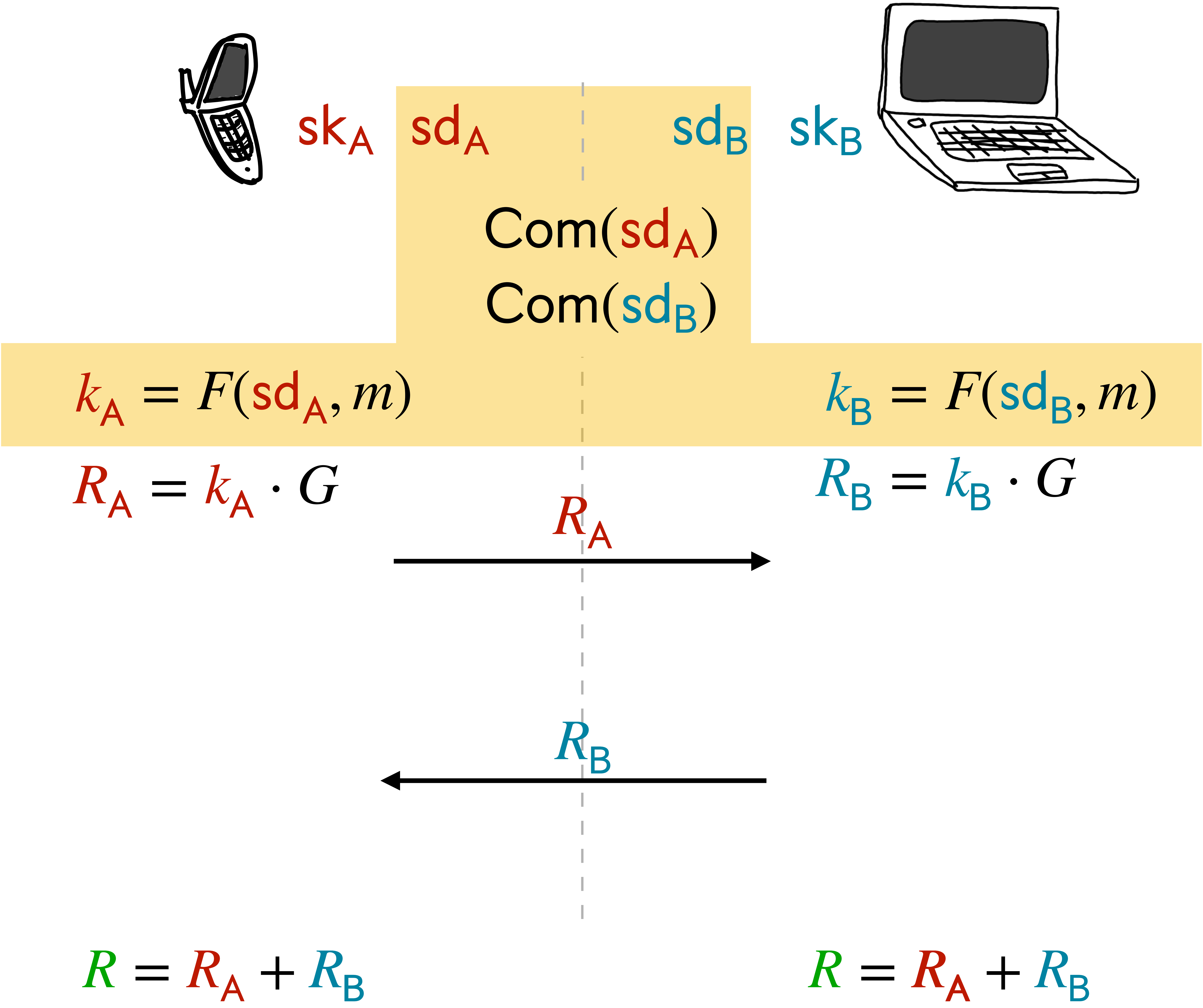
$e = H(R \| m)$ | $e = H(R \| m)$

$s_A = k_A - \mathsf{sk_A} \cdot e$ | $s_B = k_B - \mathsf{sk_B} \cdot e$

$s = s_A + s_B \longleftrightarrow s = s_A + s_B$

Threshold Setting: Take 2

Need to verify this is done correctly



$\mathsf{sk_A}$ $\mathsf{sd_A}$ $\mathsf{sd_B}$ $\mathsf{sk_B}$

$\mathsf{Com}(\mathsf{sd_A})$

$\mathsf{Com}(\mathsf{sd_B})$

$k_\mathsf{A} = F(\mathsf{sd_A}, m)$ $k_\mathsf{B} = F(\mathsf{sd_B}, m)$

$R_\mathsf{A} = k_\mathsf{A} \cdot G$ $R_\mathsf{B} = k_\mathsf{B} \cdot G$

$\xrightarrow{R_\mathsf{A}}$

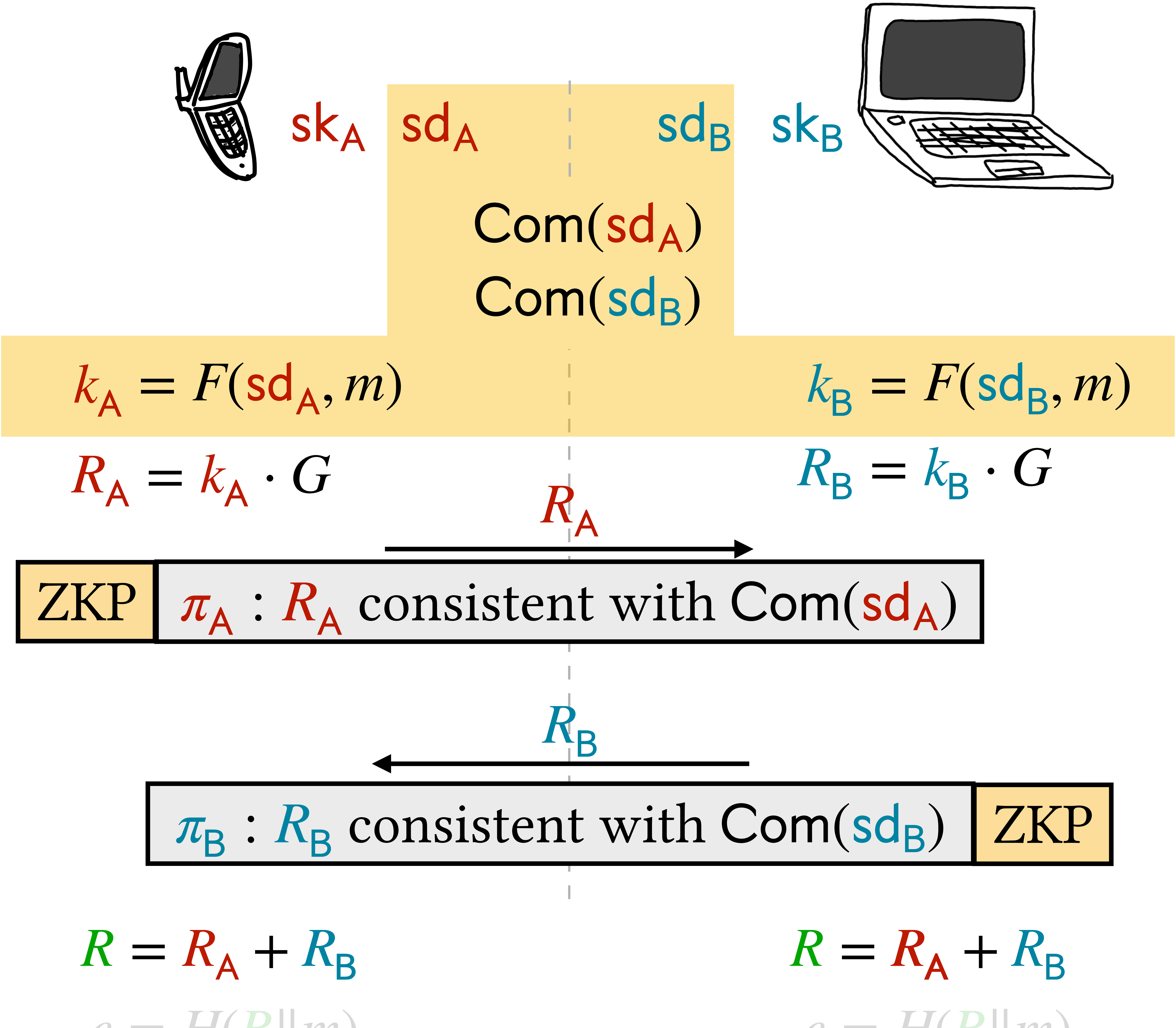
$\xleftarrow{R_\mathsf{B}}$

$R = R_\mathsf{A} + R_\mathsf{B}$ $R = R_\mathsf{A} + R_\mathsf{B}$

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

Threshold Setting: Take 2

Need to verify this is done correctly



sk_A sd_A sd_B sk_B

$\mathsf{Com}(\mathsf{sd}_\mathsf{A})$

$\mathsf{Com}(\mathsf{sd}_\mathsf{B})$

$k_\mathsf{A} = F(\mathsf{sd}_\mathsf{A}, m)$ $k_\mathsf{B} = F(\mathsf{sd}_\mathsf{B}, m)$

$R_\mathsf{A} = k_\mathsf{A} \cdot G$ $R_\mathsf{B} = k_\mathsf{B} \cdot G$

$\xrightarrow{R_\mathsf{A}}$

ZKP $\pi_\mathsf{A} : R_\mathsf{A}$ consistent with $\mathsf{Com}(\mathsf{sd}_\mathsf{A})$

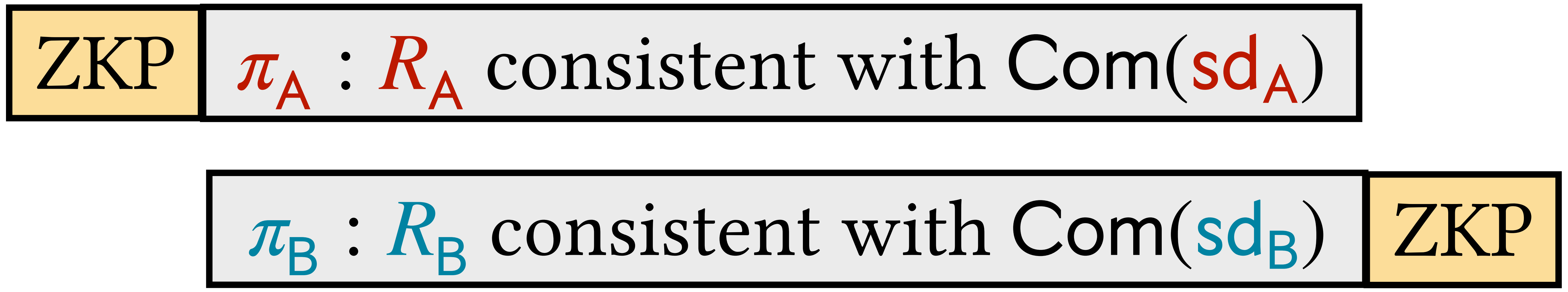
$\xleftarrow{R_\mathsf{B}}$

$\pi_\mathsf{B} : R_\mathsf{B}$ consistent with $\mathsf{Com}(\mathsf{sd}_\mathsf{B})$ ZKP

$R = R_\mathsf{A} + R_\mathsf{B}$ $R = R_\mathsf{A} + R_\mathsf{B}$

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

Threshold Setting: Take 2



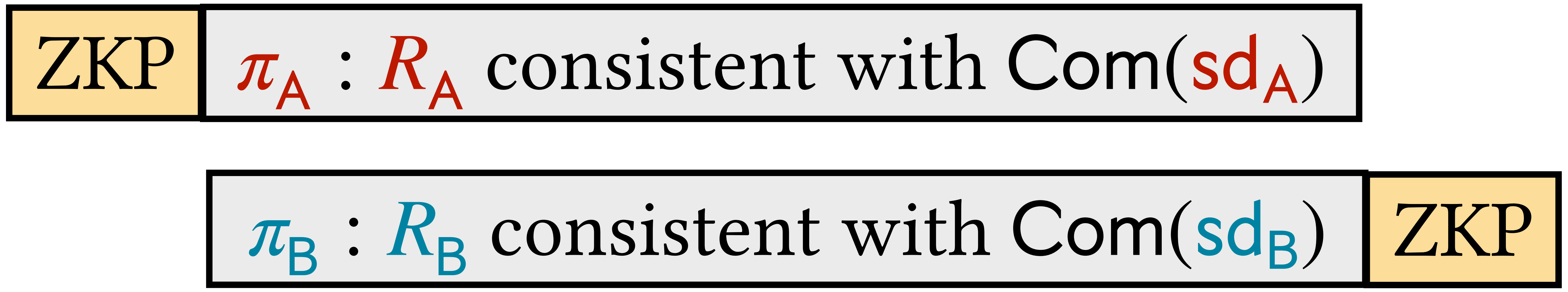
ZKP π_{A} : R_{A} consistent with $\mathsf{Com}(\mathsf{sd}_{\mathsf{A}})$

π_{B} : R_{B} consistent with $\mathsf{Com}(\mathsf{sd}_{\mathsf{B}})$ ZKP

- This "GMW-style" approach was taken in (the only) previous works [Nick Ruffing Seurin Wuille 20][Garillot K Mohassel Nikolaenko 21]

- The statement to be proven in ZK is non-trivial: $R_{\mathsf{A}} = F(\mathsf{sd}_{\mathsf{A}}, m) \cdot G$

Threshold Setting: Take 2



ZKP | π_{A} : R_{A} consistent with $\mathsf{Com}(\mathsf{sd}_{\mathsf{A}})$

π_{B} : R_{B} consistent with $\mathsf{Com}(\mathsf{sd}_{\mathsf{B}})$ | ZKP

- This "GMW-style" approach was taken in (the only) previous works [Nick Ruffing Seurin Wuille 20][Garillot K Mohassel Nikolaenko 21]

- The statement to be proven in ZK is non-trivial: $R_{\mathsf{A}} = \underbrace{F(\mathsf{sd}_{\mathsf{A}}, m)}_{\text{PRF evaluation}} \cdot \underbrace{G}_{\text{Exponentiation}}$

  - [NRSW 20]: Custom arithmetic PRF + Bulletproofs

  - [GKMN 21]: Standardized PRF (eg. AES) + Garbled Circuits

Is there a more "native" approach?

- Proving correct evaluation of F is inherently bottlenecked by circuit complexity of PRFs
- Ideally, we would like to avoid such non-blackbox use of crypto
- Central question in this paper:

> Can we design a distributed, stateless deterministic Schnorr signing scheme that makes **blackbox use** of cryptographic primitives?

This work: a qualified "yes"

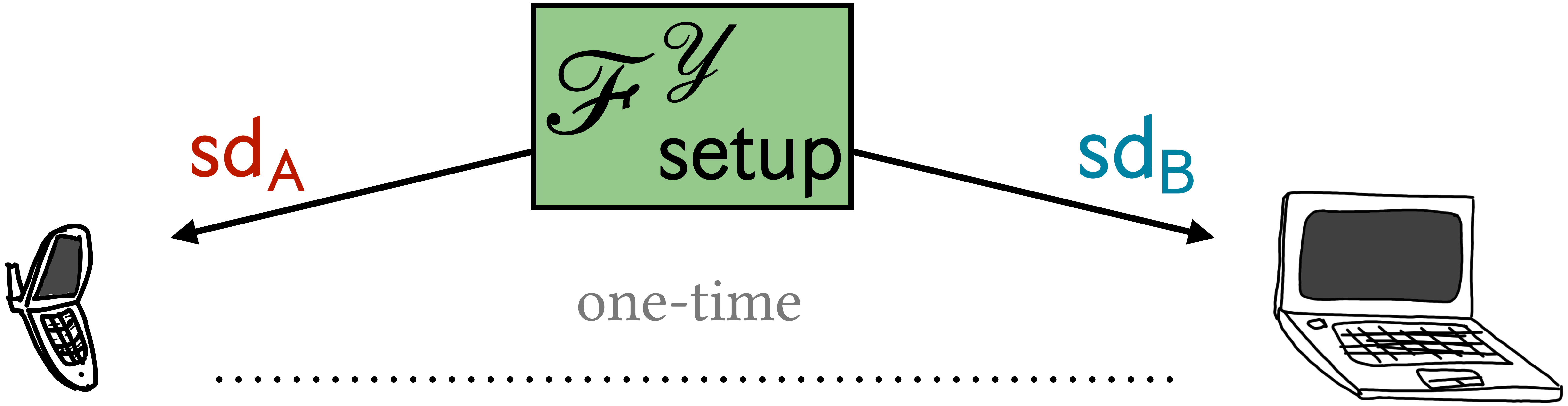
Our Results

- Main construction: blackbox use of Pseudorandom Correlation Function (PCF) for Vector Oblivious Linear Evaluation (VOLE) in $\mathbb{Z}_q$
  - Simple stateless derandomization pattern
  - PCFs are increasingly general, but it's not Oblivious Transfer
- Two concrete instantiations:
  1. Covert security from any PRF
  2. Full malicious security from Paillier

Pseudorandom Correlation Functions

[Boyle Couteau Gilboa Ishai Kohl Scholl 20]

For a correlation $\mathscr{Y}$:



$\mathscr{F}^{\mathscr{Y}}_{\mathsf{setup}}$

sd_{A}

sd_{B}

one-time

unbounded

$$y_{x,\mathsf{A}} = \mathsf{PCF}(\mathsf{sd}_{\mathsf{A}}, x) \qquad x \qquad y_{x,\mathsf{B}} = \mathsf{PCF}(\mathsf{sd}_{\mathsf{B}}, x)$$

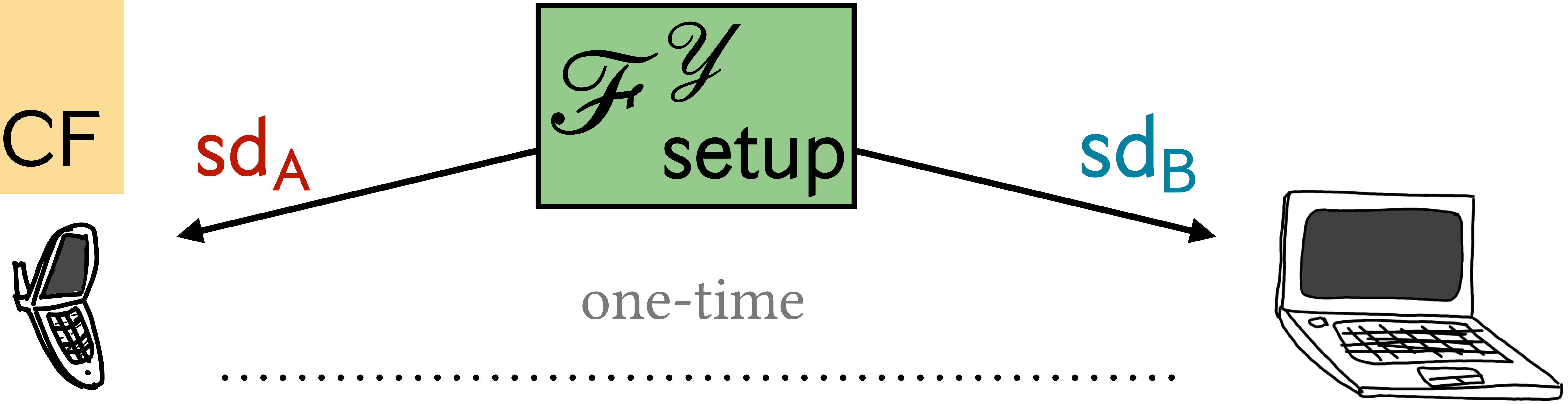
$$\left(y_{x,\mathsf{A}}, y_{x,\mathsf{B}}\right) \in \mathscr{Y}$$

Pseudorandom Correlation Functions

[Boyle Couteau Gilboa Ishai Kohl Scholl 20]

For a correlation $\mathscr{Y}$:

Complexity of $\mathscr{Y}$ determines efficiency of PCF



$\mathscr{F}^{\mathscr{Y}}$ setup

sd_A

sd_B

one-time

unbounded

$$x$$

$$y_{x,A} = \mathsf{PCF}(\mathsf{sd}_A, x) \qquad y_{x,B} = \mathsf{PCF}(\mathsf{sd}_B, x)$$

$$\left(y_{x,A}, y_{x,B}\right) \in \mathscr{Y}$$

“Good enough” Correlation for Schnorr

- simple enough for reasonably efficient PCFs
- powerful enough to build what we want

$$\mathscr{Y}^{\Delta}_{\mathsf{VOLE}} : \big((k, w = \Delta k + \beta), (\Delta, \beta)\big)$$

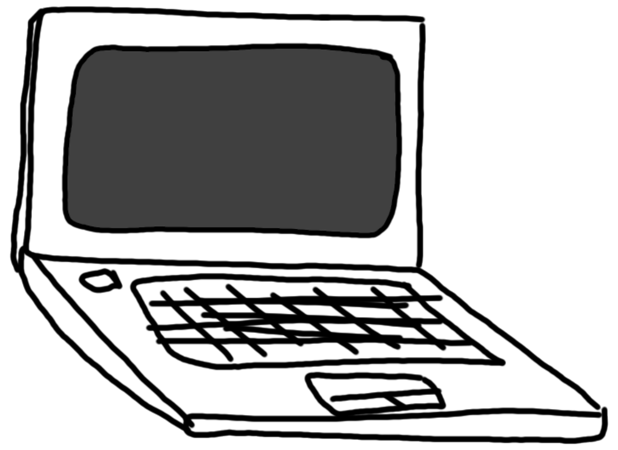
“Good enough” Correlation for Schnorr

private nonce

MAC on nonce

$$\mathcal{Y}^{\Delta}_{\mathsf{VOLE}} : \big((k, w = \Delta k + \beta), (\Delta, \beta)\big)$$

MAC verification key



“Good enough” Correlation for Schnorr

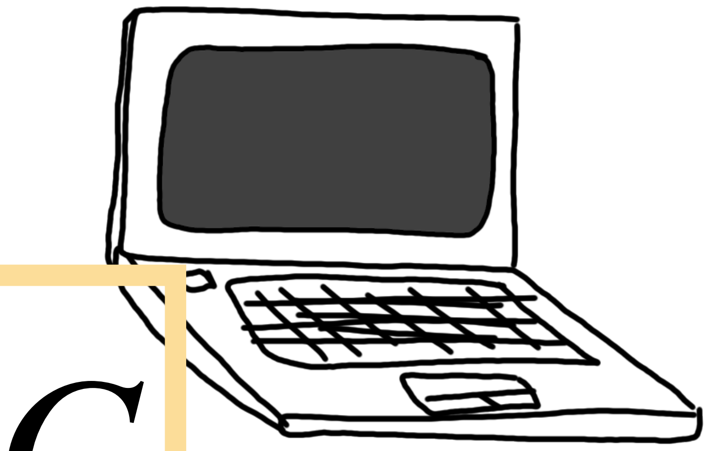
private nonce

MAC on nonce

$$\mathcal{Y}^{\Delta}_{\mathsf{VOLE}} : \big((k, w = \Delta k + \beta), (\Delta, \beta)\big)$$

MAC verification key

$$\xrightarrow[W = w \cdot G]{R = k \cdot G}$$



$$W \overset{?}{=} \Delta \cdot R + \beta \cdot G$$

Verify MAC in exponent

“Good enough” Correlation for Schnorr

private nonce

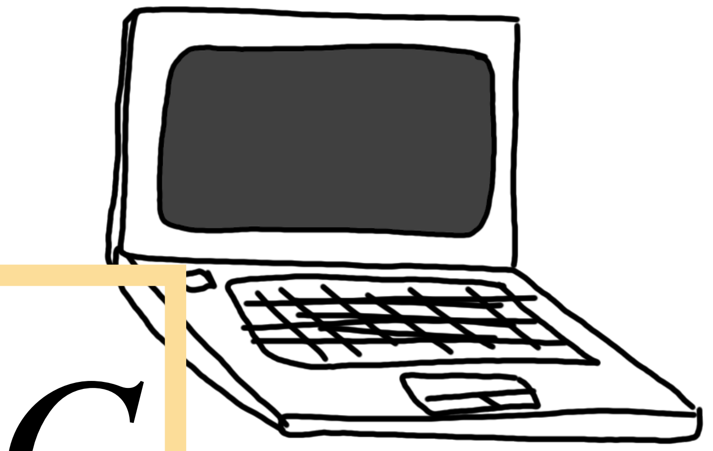
MAC on nonce

$$\mathcal{Y}^{\Delta}_{\mathsf{VOLE}} : \big((k, w = \Delta k + \beta), (\Delta, \beta)\big)$$

MAC verification key

Need to guess Δ to subvert the check

$$\xrightarrow[W = w \cdot G]{R = k \cdot G}$$

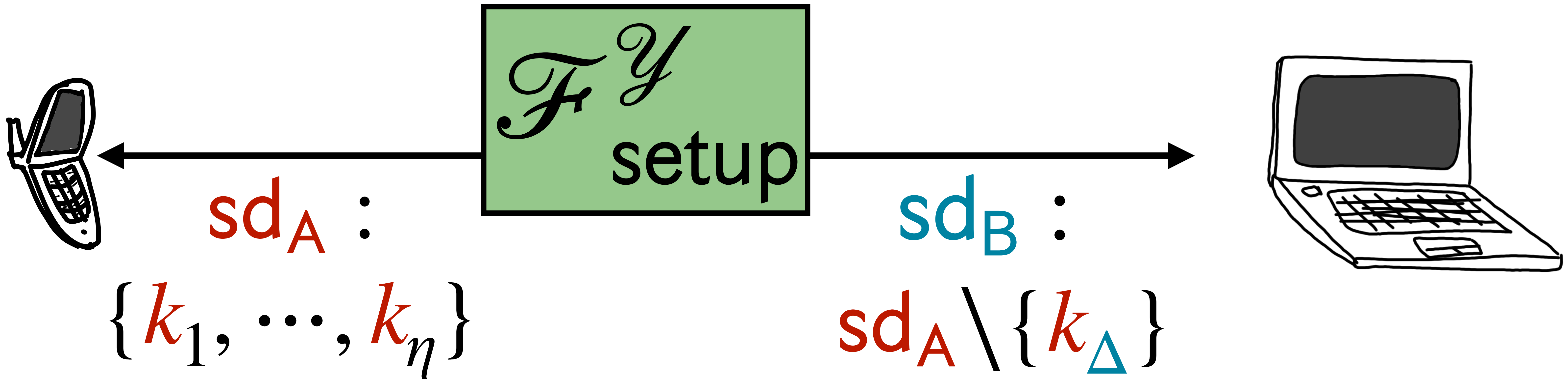


$$W \overset{?}{=} \Delta \cdot R + \beta \cdot G$$

Verify MAC in exponent

PCF for $\mathcal{Y}^{\Delta}_{\mathsf{VOLE}}$

- First construction: adapted from SoftSpoken VOLE [Roy22] (originally used for OT Extension)



$\mathsf{sd}_A : \{k_1, \cdots, k_\eta\}$

$\mathsf{sd}_B : \mathsf{sd}_A \backslash \{k_\Delta\}$

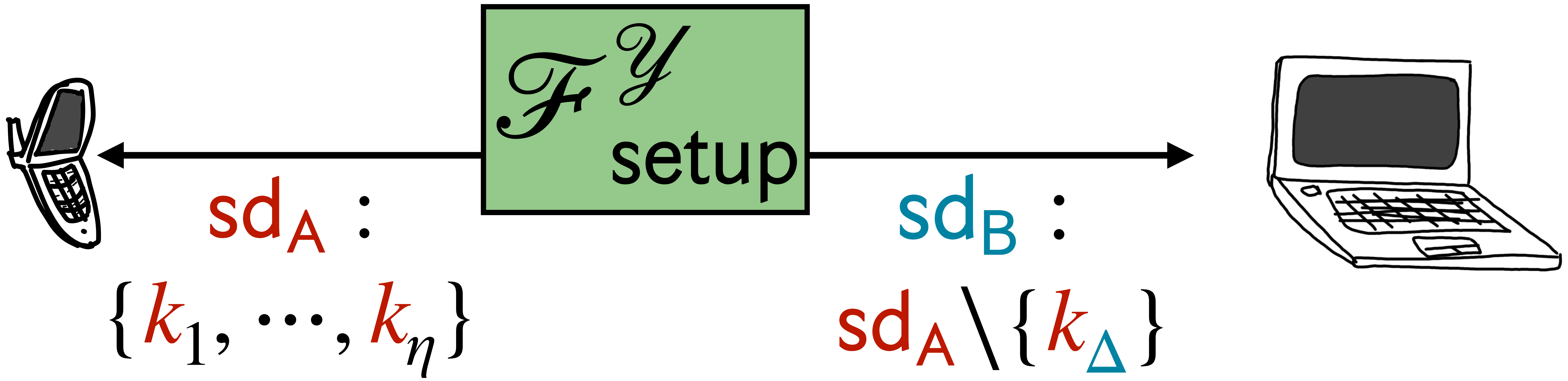
$$\mathsf{PCF}(x) :$$

$$k = \Sigma_i \ \mathsf{PRF}_{k_i}(x) \qquad \beta = \Sigma_i \ (i - \Delta) \cdot \mathsf{PRF}_{k_i}(x)$$

$$w = \Sigma_i \ i \cdot \mathsf{PRF}_{k_i}(x)$$

PCF for $\mathscr{Y}^{\Delta}_{\mathsf{VOLE}}$

- First construction: adapted from SoftSpoken VOLE [Roy22] (originally used for OT Extension)



$\mathscr{F}^{\mathscr{Y}}_{\mathsf{setup}}$

sd_{A} : $\{k_1, \cdots, k_\eta\}$

sd_{B} : $\mathsf{sd}_{\mathsf{A}} \backslash \{k_\Delta\}$

$$\mathsf{PCF}(x) :$$

$$k = \Sigma_i \ \mathsf{PRF}_{k_i}(x)$$

$$w = \Sigma_i \ i \cdot \mathsf{PRF}_{k_i}(x)$$

$$\beta = \Sigma_i \ (i - \Delta) \cdot \mathsf{PRF}_{k_i}(x)$$

$\Delta \in \mathsf{poly}(\kappa) \Rightarrow$ only *covert* security (eg. 2^{-10} soundness)

Fully Secure PCF for $\mathcal{Y}^{\Delta}_{\mathsf{VOLE}}$

- Unclear how to strengthen the SoftSpoken VOLE construction

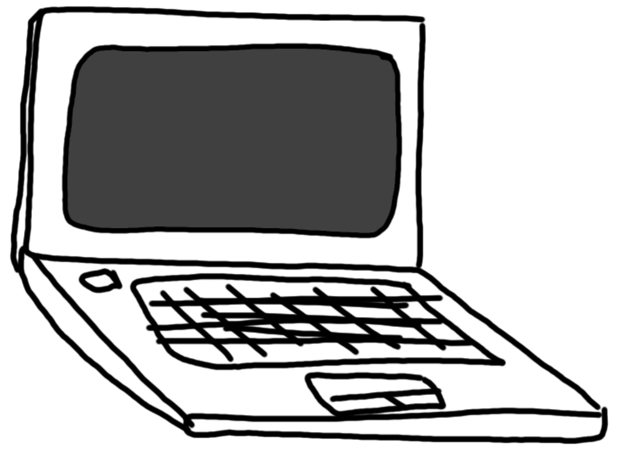
- [Orlandi Scholl Yakoubov 21]: Elegant VOLE PCF from Paillier, supports $\Delta \in \mathsf{exp}(\kappa)$

- Unfortunately, [OSY21] gives VOLE in the ring $\mathbb{Z}_N$
(*N* is a biprime of factorization unknown to verifier)

- We need to "translate" VOLE in $\mathbb{Z}_N$ to $\mathbb{Z}_q$
This turns out to be quite non-trivial, borrowed ideas from [OSY21, Roy Singh 21]

Securely Translating $\mathscr{Y}^{\Delta,N}_{\mathsf{VOLE}} \to \mathscr{Y}^{\Delta,q}_{\mathsf{VOLE}}$

$\mathscr{Y}^{\Delta,N}_{\mathsf{VOLE}}$



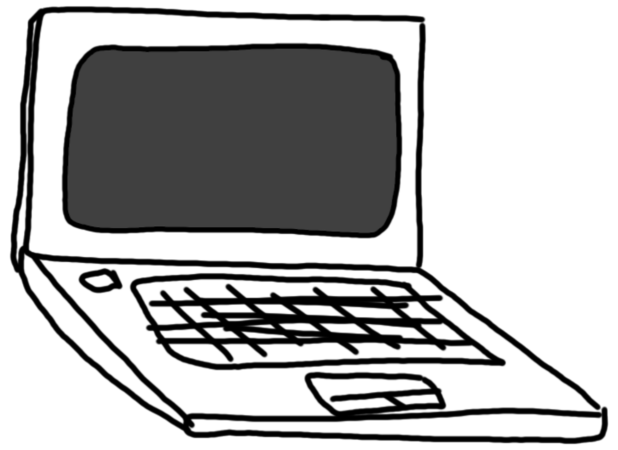
$k, w = \Delta k + \beta \pmod{N}$

Δ, β

Securely Translating $\mathcal{Y}^{\Delta,N}_{\mathsf{VOLE}} \to \mathcal{Y}^{\Delta,q}_{\mathsf{VOLE}}$

Public M s.t. $q \mid M$

$\mathcal{Y}^{\Delta,N}_{\mathsf{VOLE}}$



$k, w = \Delta k + \beta \pmod{N}$

Δ, β

Derive k_{lo}, k_{hi} : $k_{hi}M + k_{lo} = k$

IKNP-style "correction word"

$$\xrightarrow[k_{hi}]{}$$

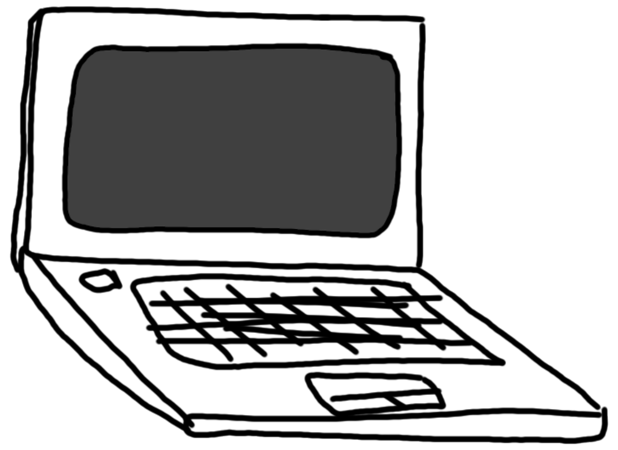
$$\beta' = \beta + \Delta(Mk_{hi})$$

$$\big((k_{lo}, w), (\Delta, \beta')\big) \in \mathcal{Y}^{\Delta,q}_{\mathsf{VOLE}}$$

Securely Translating $\mathcal{Y}^{\Delta,N}_{\mathsf{VOLE}} \to \mathcal{Y}^{\Delta,q}_{\mathsf{VOLE}}$

Public M s.t. $q \mid M$

$\mathcal{Y}^{\Delta,N}_{\mathsf{VOLE}}$



$k, w = \Delta k + \beta \pmod{N}$

Δ, β

IKNP-style “correction word”

$\xrightarrow[k^*_{hi}]{}$

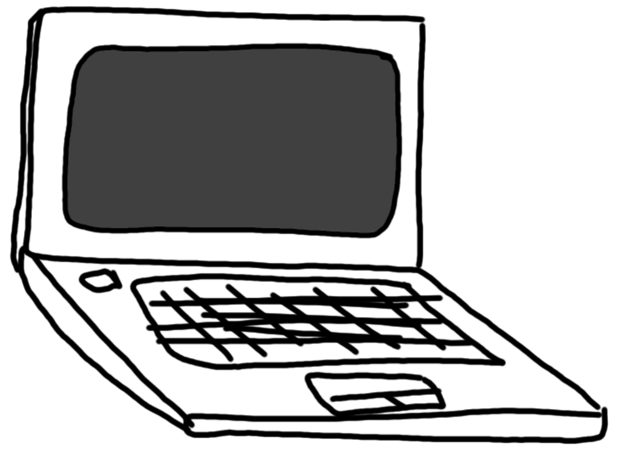
$\beta' = \beta + \Delta(Mk^*_{hi})$

$\Delta\boxed{??} + \beta' = w$

Securely Translating $\mathcal{Y}^{\Delta, N}_{\mathsf{VOLE}} \to \mathcal{Y}^{\Delta, q}_{\mathsf{VOLE}}$

Public M s.t. $q \mid M$

$\mathcal{Y}^{\Delta, N}_{\mathsf{VOLE}}$



$k, w = \Delta k + \beta \pmod{N}$

Δ, β

IKNP-style "correction word"

However, deriving a *correct* correlation isn't enough; we need reset resilience as well

$\xrightarrow{}$

k^*_{hi}

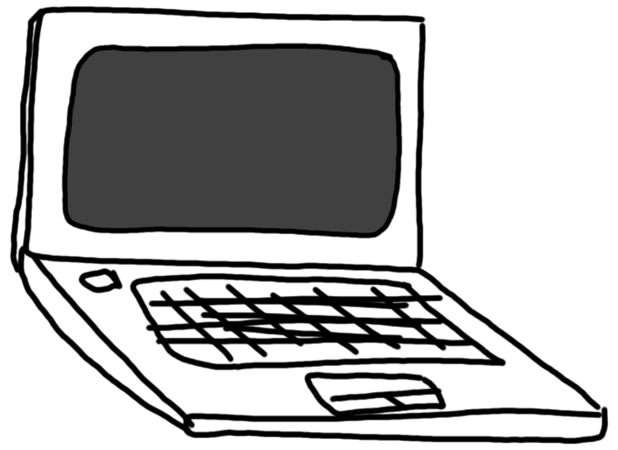
$\Delta \boxed{??} + \beta' = w$

$\beta' = \beta + \Delta(Mk^*_{hi})$

Securely Translating $\mathcal{Y}^{\Delta,N}_{\mathsf{VOLE}} \rightarrow \mathcal{Y}^{\Delta,q}_{\mathsf{VOLE}}$

Public M s.t. $q \mid M$

$\mathcal{Y}^{\Delta,N}_{\mathsf{VOLE}}$



$k, w = \Delta k + \beta \pmod{N}$

Δ, β

IKNP-style “correction word”

However, deriving a *correct* correlation isn’t enough; we need reset resilience as well

Same $k_{lo} \pmod{q}$ ∀ valid k^*_{hi}

$\xrightarrow{}$

k^*_{hi}

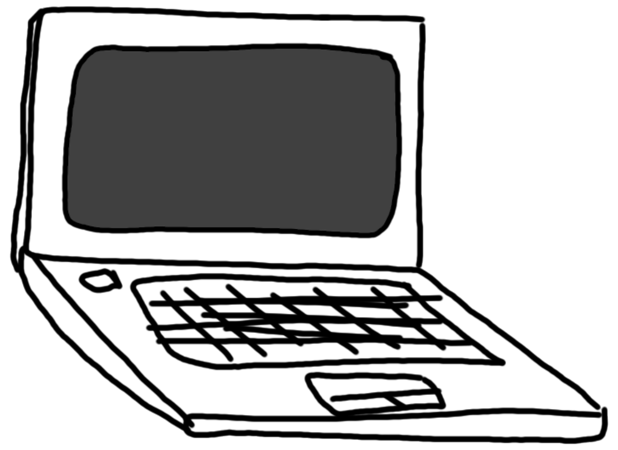
$\beta' = \beta + \Delta(Mk^*_{hi})$

$\boxed{\Delta k_{lo} + \beta' = w}$

Securely Translating $\mathcal{Y}^{\Delta,N}_{\text{VOLE}} \to \mathcal{Y}^{\Delta,q}_{\text{VOLE}}$

Public M s.t. $q \mid M$

$\mathcal{Y}^{\Delta,N}_{\text{VOLE}}$



$k, w = \Delta k + \beta \pmod N$

Δ, β

$$\xrightarrow[k^*_{hi}]{}$$

$$\beta' = \beta + \Delta(M k^*_{hi})$$

$$\Delta \underbrace{k_{lo}}_{\text{small}} + \beta' = w$$

IKNP-style “correction word”

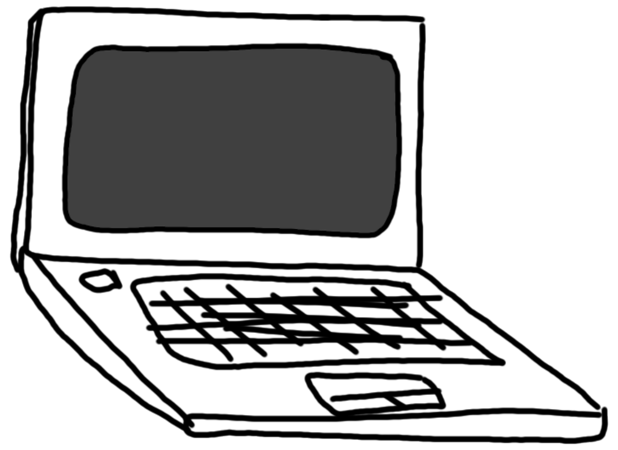
However, deriving a *correct* correlation isn’t enough; we need reset resilience as well

Same $k_{lo} \pmod q$ ∀ valid k^*_{hi}

Securely Translating $\mathcal{Y}^{\Delta,N}_{\text{VOLE}} \to \mathcal{Y}^{\Delta,q}_{\text{VOLE}}$

Public M s.t. $q \mid M$

$\mathcal{Y}^{\Delta,N}_{\text{VOLE}}$



$k, w = \Delta k + \beta \pmod{N}$

Δ, β

Check modulo auxiliary biprime

Similar to [DF02]

$$\xrightarrow[k^*_{hi}]{g^{k_{lo}},\, g^{w} \pmod{\tilde{N}}}$$

$$\beta' = \beta + \Delta(Mk^*_{hi})$$

IKNP-style “correction word”

However, deriving a *correct* correlation isn’t enough; we need reset resilience as well

Same $k_{lo} \pmod{q}$ $\forall$ valid k^*_{hi}

$\Delta \underbrace{k_{lo}}_{\text{small}} + \beta' = w$

$g^{\beta'}(g^{k_{lo}})^{\Delta} \overset{?}{=} g^{w} \pmod{\tilde{N}}$

Sound assuming Strong RSA

Signing Efficiency: PCF Overhead

- Covert construction only adds a single $\mathbb{G}$ element, comparable to semi-honest signing for reasonable deterrence

- Fully secure Paillier-based construction for 256-bit curve, this work (PCF) in comparison with [NRSW20] (Bulletproofs) and [GKMN21] (Garbled Circuits)

  - 451 bytes (including correction word+check)

    Bandwidth: PCF < Bulletproofs << Garbled Circuits
    0.5KB 1KB 100s of KB

  - 188ms to prove and verify

    Computation: Garbled Circuits < PCF < Bulletproofs
    tens of ms 188ms 950ms

Instantiating $\mathcal{F}_{\mathsf{setup}}$

- PCFs are defined with a trusted dealer, no standard setup protocol
  - This model may be enough for some applications [ANOSS22]

- Setup protocol for covert PCF is straightforward via OT

- Setup for Paillier PCF has to generate biprimes $N, \tilde{N}$
  - Prover knows factorization of N
  - Verifier can know factorization of $\tilde{N}$

- Each party could *potentially* choose its own modulus and prove well-formedness.
  We do not explore this further in this work as we focus on signing

In Conclusion

- We give a new approach to stateless deterministic 2P-Schnorr signing based on PCFs: towards blackbox use of cryptography

- Two instantiations based on PCFs for VOLE:
  - Covert security from PRF-based SoftSpoken VOLE [Roy22]
  - Malicious security from Paillier-based [OSY21, RS21]
    + Novel mechanism to translate VOLE from $\mathbb{Z}_N \to \mathbb{Z}_q$
    + Interesting tradeoffs relative to existing work

Thanks!

eprint: 2023/216

Thanks Eysa Lee for