Public-Key Generation with Verifiable Randomness

[4] Olivier Blazy, [1,*] Patrick Towa and [2,3] Damien Vergnaud

1: ETH zürich

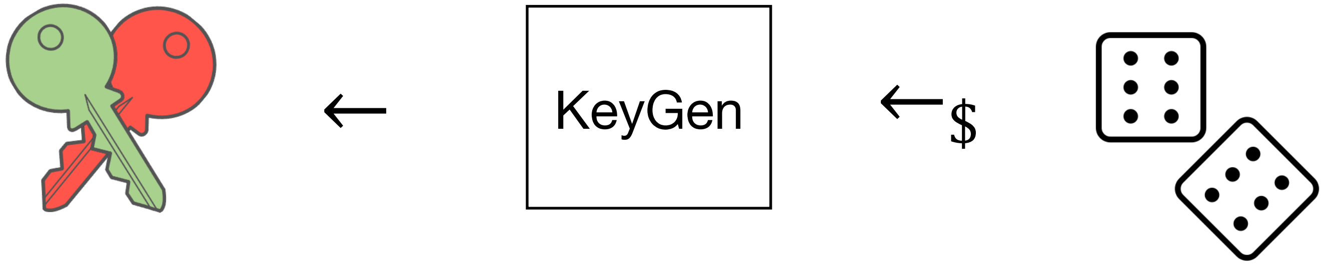
2: IUF institut universitaire de France

3: SORBONNE UNIVERSITÉ

4: ul Université de Limoges

* Work done while at IBM Research – Zurich and ENS and PSL Research University

Randomness in Key Generation

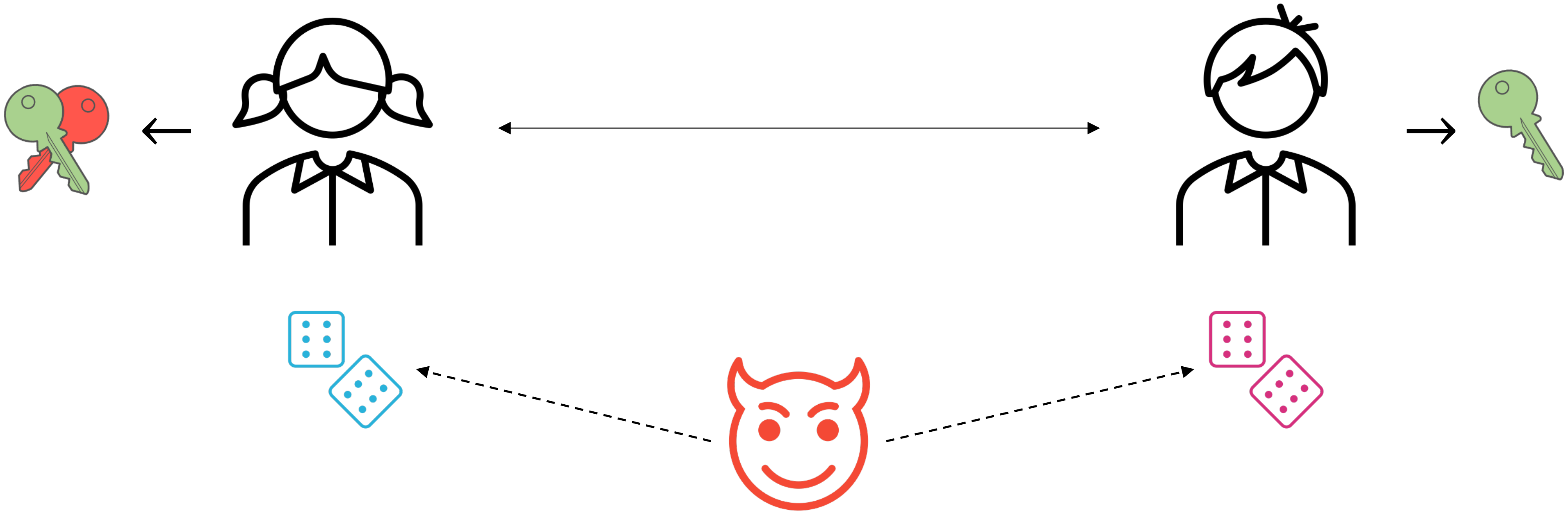


$\leftarrow$ KeyGen $\leftarrow_{\$}$

- True randomness (if it exists) is expensive
- [LHABKW12] 0.5% of RSA keys on the internet shared common primes
- [HDWH12] cause: low-entropy TLS and SSH keys generated at boot time
- [NSSKNM17] ROCA vulnerability: efficiently recovers (factoring-based) private keys from public ones – Estonian and Slovakian smartcards compromised

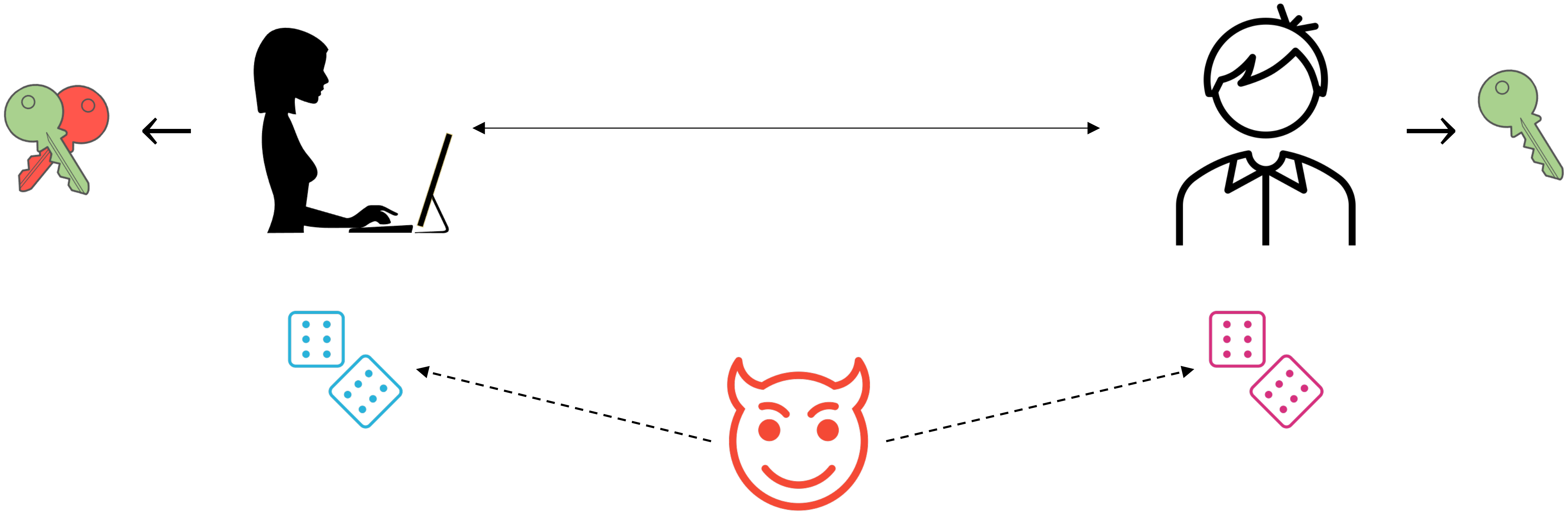
How to Certify Randomness in Key Generation

Juels and Guajardo, PKC *2002*



How to Certify Randomness in Key Generation

Goal: certify to the *end user* that her key was generated with high-entropy randomness



How to Certify Randomness in KeyGen – Requirements

1. Alice has high-entropy randomness ⇒ Bob has no info about sk
2. Alice or Bob has high-entropy randomness ⇒ No adversary (other than Bob) has more info about sk than with KG
3. Bob has high-entropy randomness ⇒ Alice's computer cannot influence the generation, no *covert channel*

How to Certify Keys – Requirements

1. Alice or Bob has high-entropy randomness ⇒ Keys indistinguishable from KG

2. Alice has high-entropy randomness ⇒ Bob has no info about sk

3. Bob has high-entropy randomness ⇒ Alice's computer cannot influence the generation, no *covert channel* ROCA

How to Certify Keys – Requirements

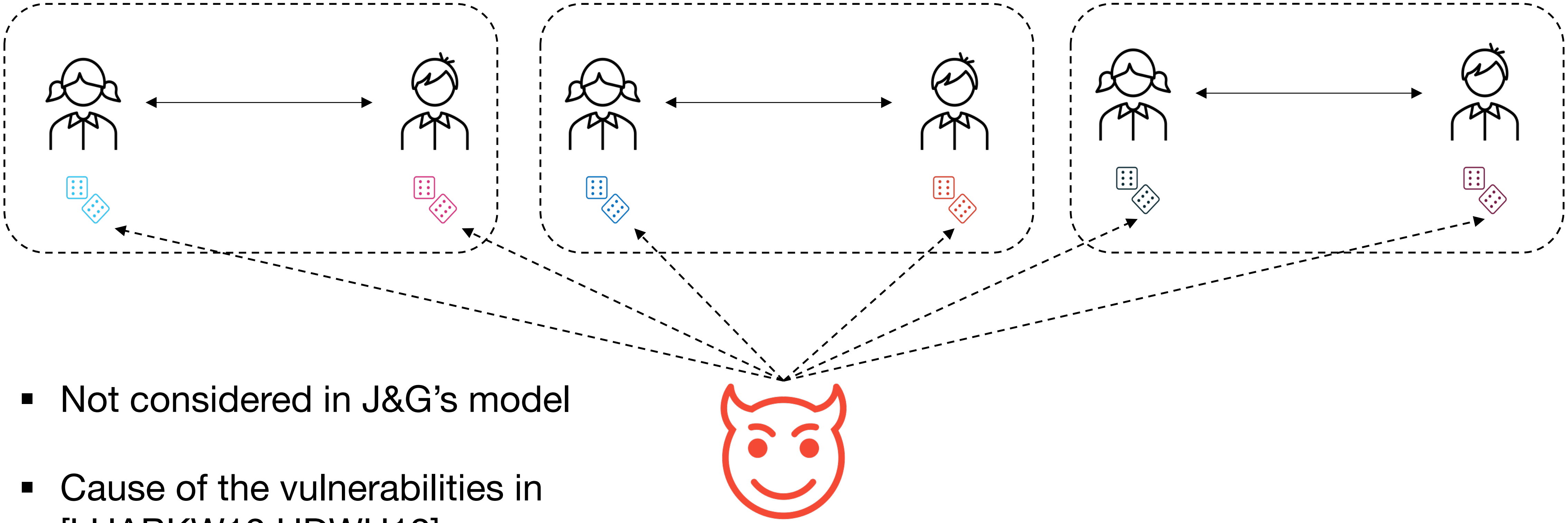
1. Alice or Bob has high-entropy randomness ⇒ Keys indistinguishable from KG

2. Alice has high-entropy randomness ⇒ Bob has no info about sk

3. Bob has high-entropy randomness ⇒ Alice's computer cannot influence the generation, no *covert channel*

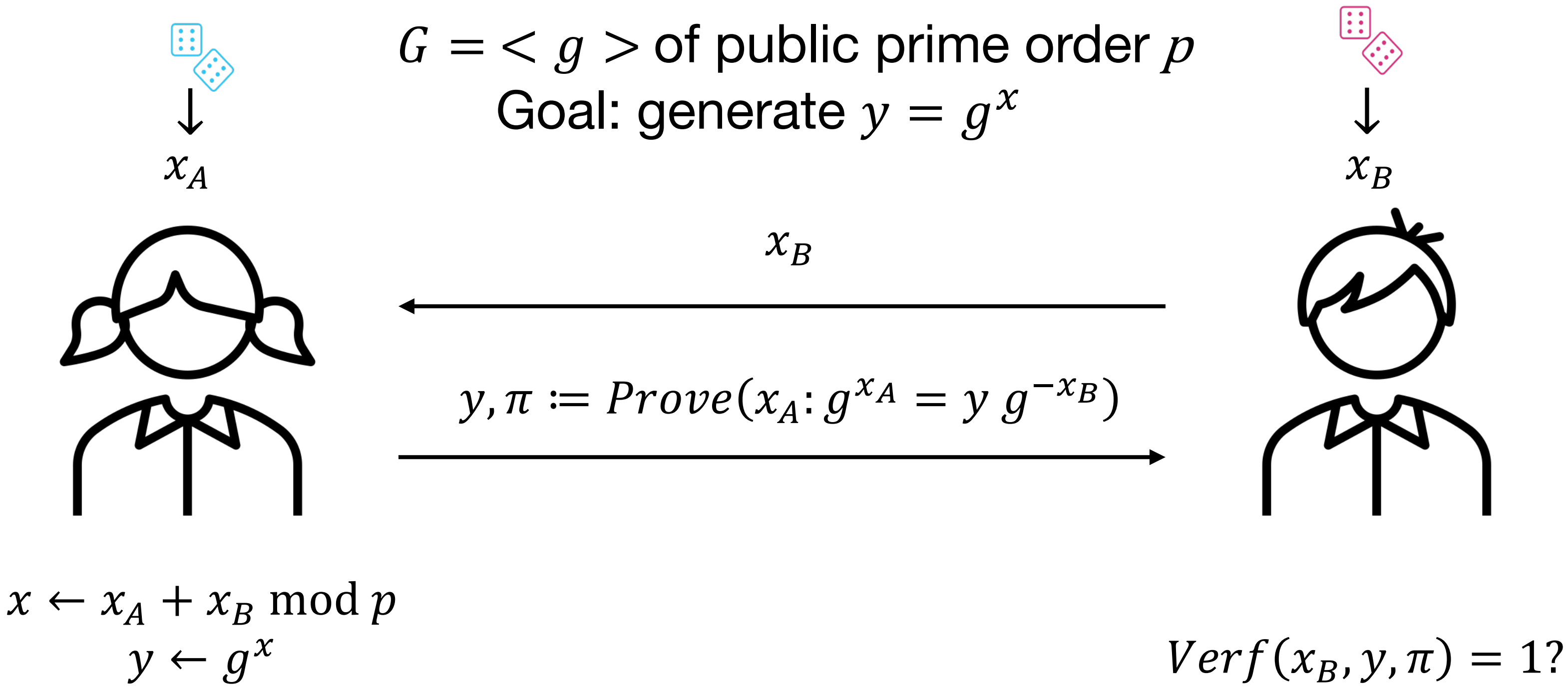
With J&G's protocol, $\log(\lambda)$ bit-capacity channels possible

Multi-Sessions with Correlated Randomness



- Not considered in J&G's model
- Cause of the vulnerabilities in [LHABKW12,HDWH12]

A not so Simple Example: Discrete-Log Keys



$G = < g >$ of public prime order p

Goal: generate $y = g^x$

x_A

x_B

x_B

$y, \pi \coloneqq Prove(x_A : g^{x_A} = y\ g^{-x_B})$

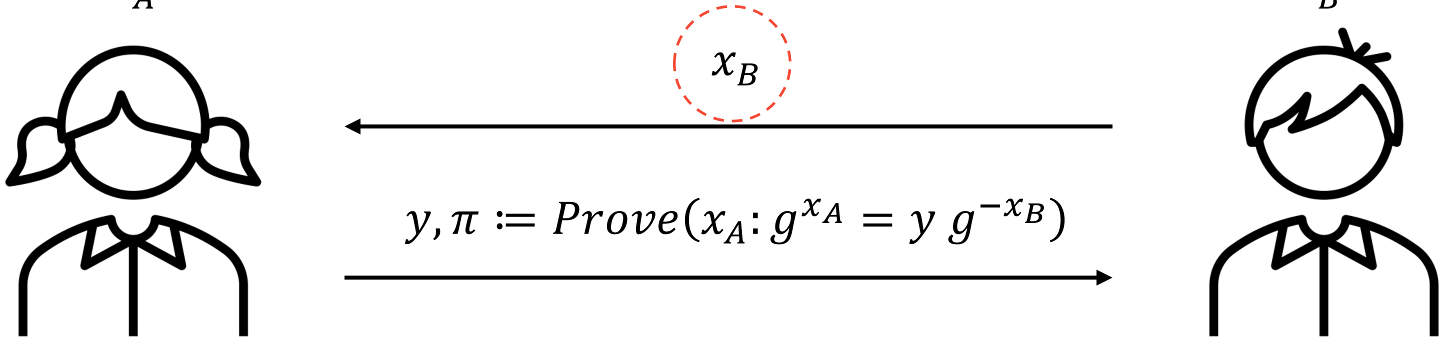
$x \leftarrow x_A + x_B \bmod p$

$y \leftarrow g^x$

$Verf(x_B, y, \pi) = 1?$

A not so Simple Example: Discrete-Log Keys

Alice can fix x to the value she wants
⇒ (3) violated



x_A

x_B

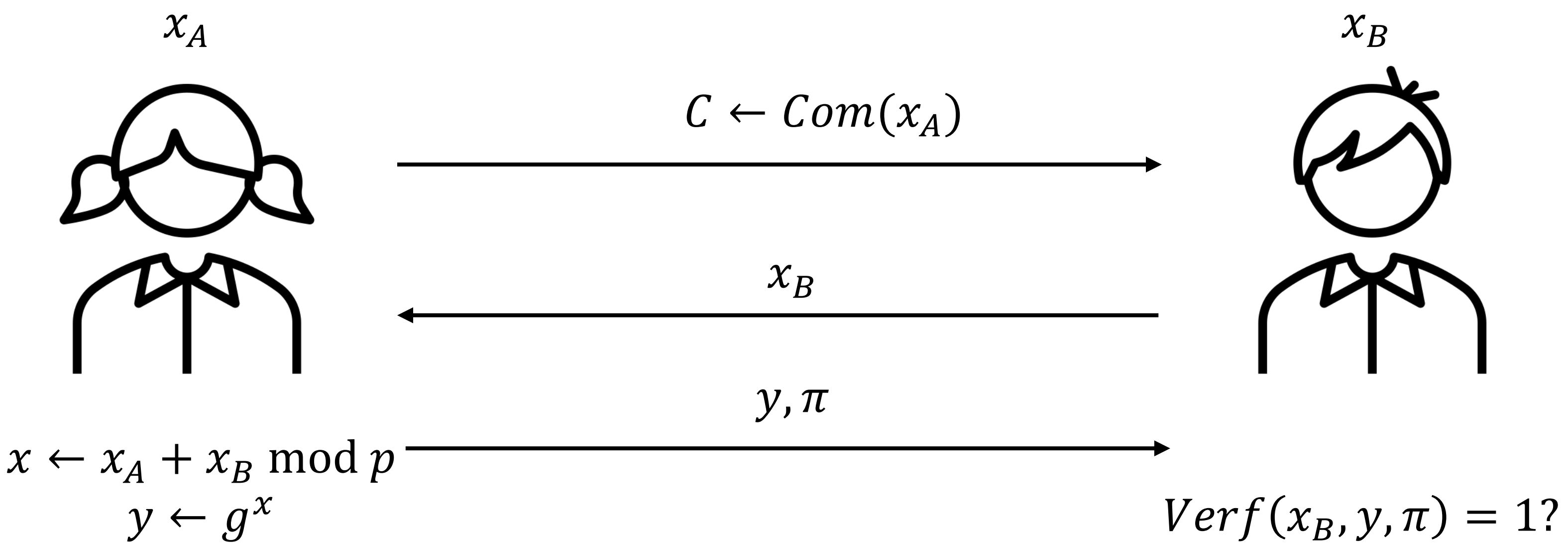
x_B

$y, \pi \coloneqq Prove(x_A : g^{x_A} = y\, g^{-x_B})$

$x \leftarrow x_A + x_B \bmod p$
$y \leftarrow g^x$

$Verf(x_B, y, \pi) = 1?$

A not so Simple Example: Discrete-Log Keys



x_A

x_B

$C \leftarrow Com(x_A)$

x_B

y, π

$x \leftarrow x_A + x_B \bmod p$

$y \leftarrow g^x$

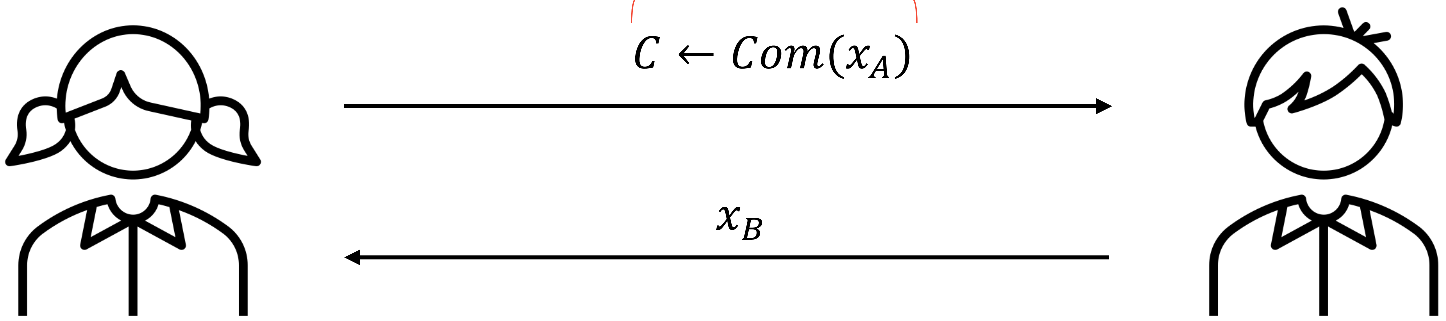
$Verf(x_B, y, \pi) = 1?$

$$\pi \coloneqq Prove(x_A : C = Com(x_A) \ \wedge \ g^{x_A} = y \ g^{-x_B})$$

A not so Simple Example: Discrete-Log Keys

With which randomness?
(2) might be violated

Overlooked in J&G's protocols



x_A

x_B

$\overbrace{C \leftarrow Com(x_A)}$

x_B

y, π

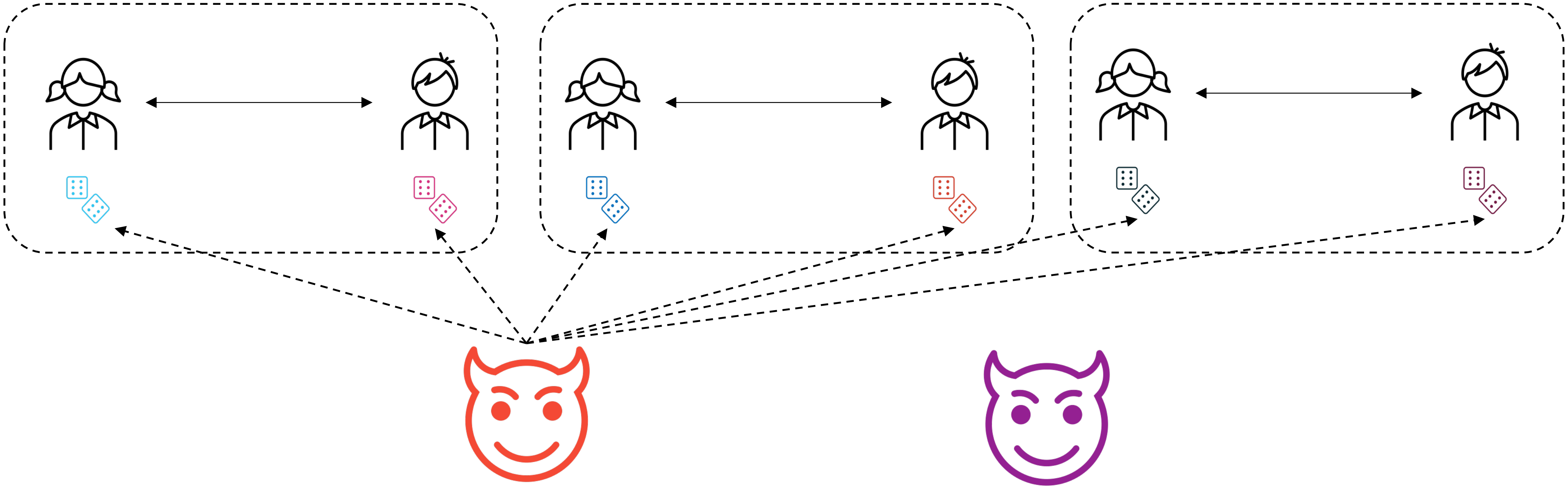
$x \leftarrow x_A + x_B \bmod p$
$y \leftarrow g^x$

$Verf(x_B, y, \pi) = 1?$

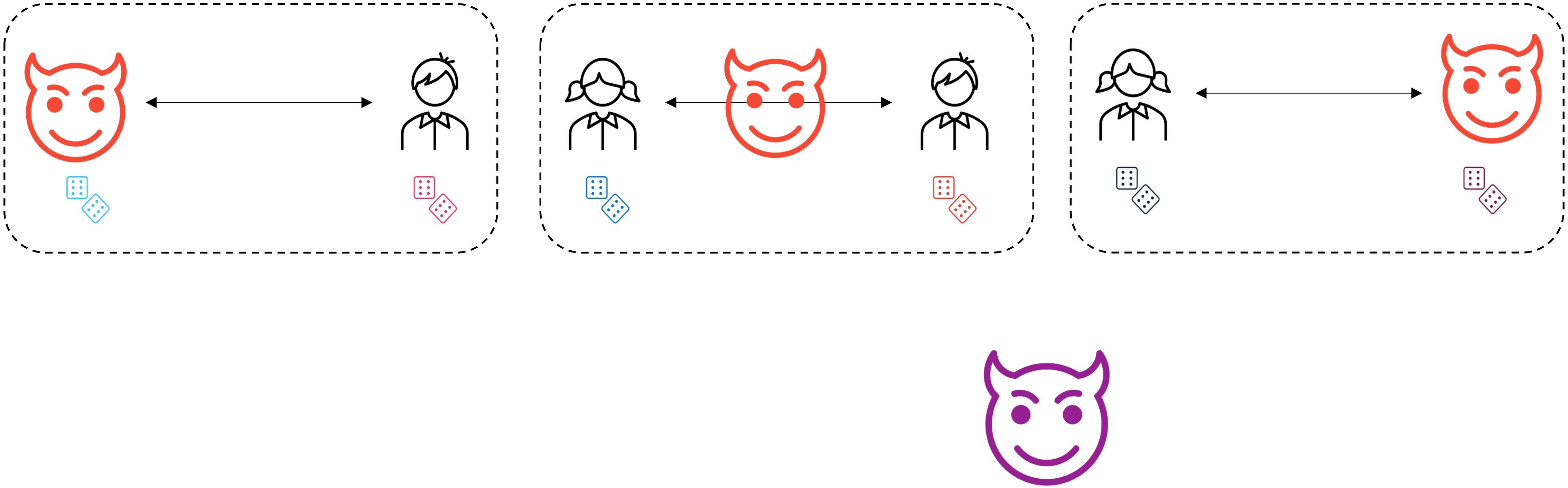
$$\pi \coloneqq \underbrace{Prove}(x_A : C = Com(x_A) \ \wedge \ g^{x_A} = y \ g^{-x_B})$$

With which randomness?
(2) might be violated

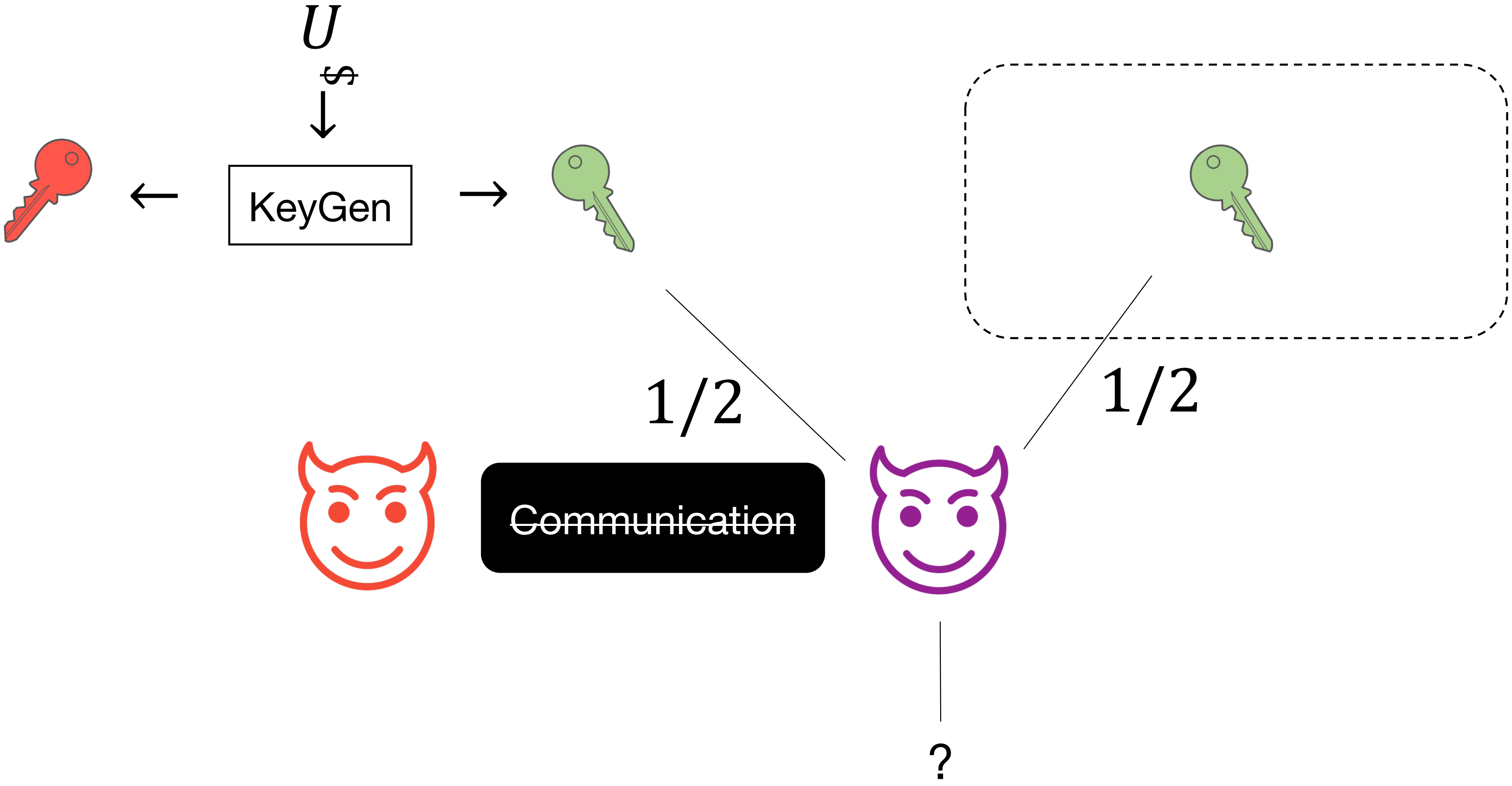
A new Model for Randomness Certification



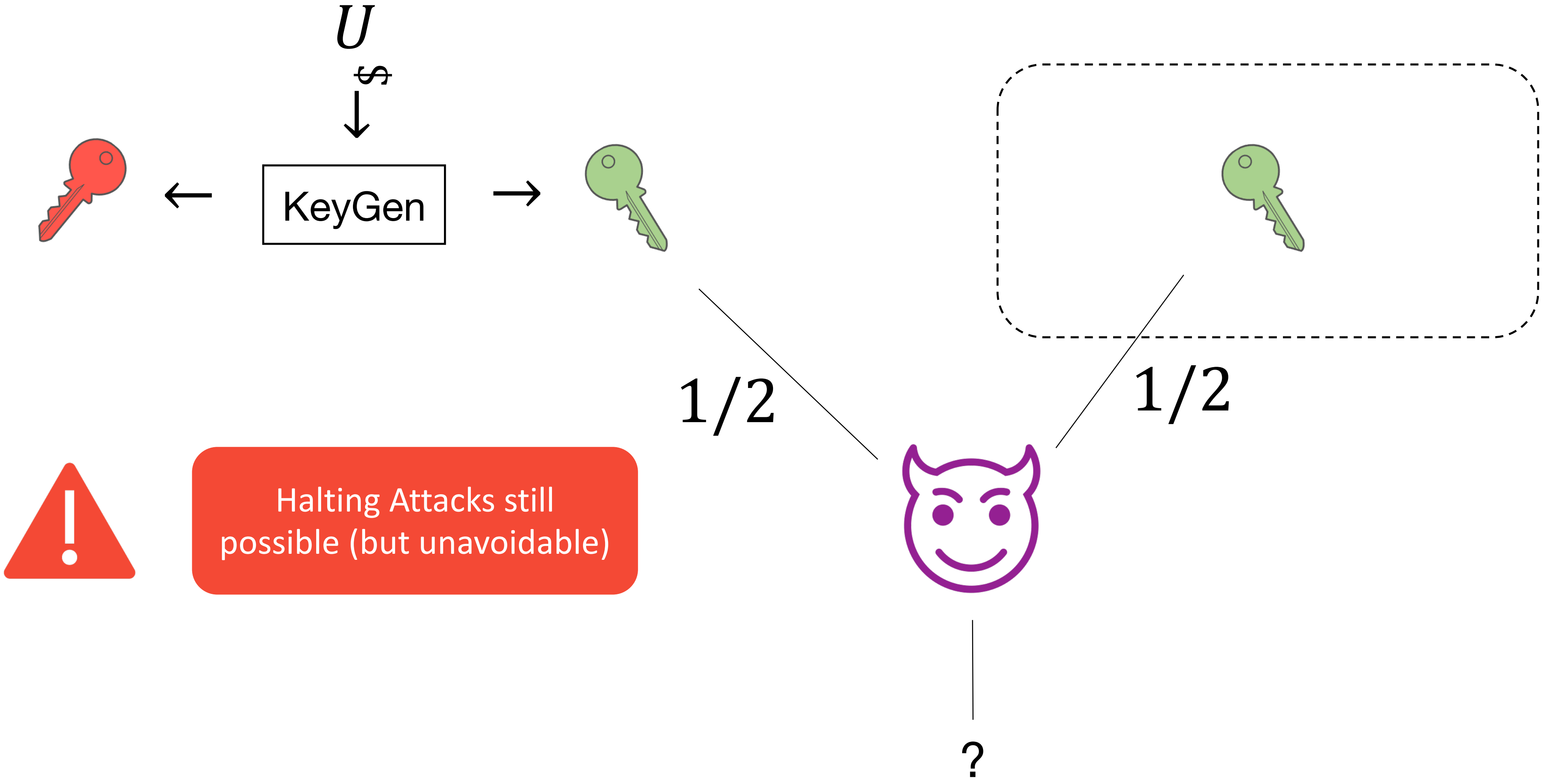
A new Model for Randomness Certification



A new Model for Randomness Certification



A new Model for Randomness Certification



Halting Attacks still possible (but unavoidable)

A Protocol for Discrete-Log Keys

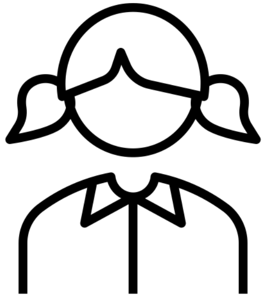
$G = < g >$ of public prime order p

Goal: generate $y = g^x$

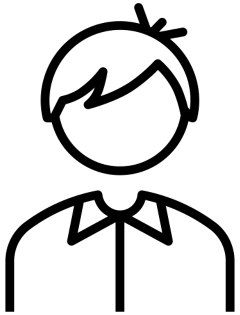
Extract $\leftarrow$ x_A

$\swarrow$ $\searrow$

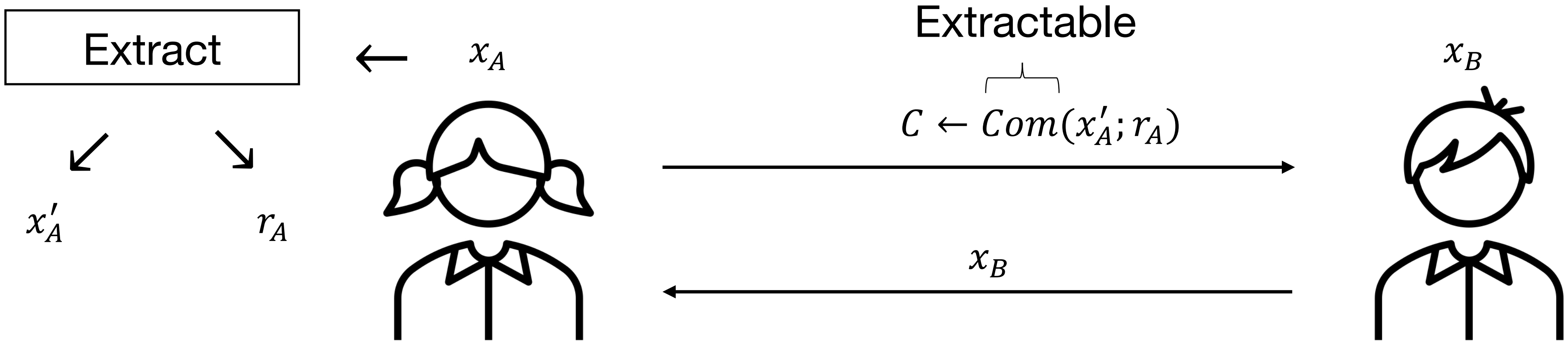
x'_A r_A



x_B



A Protocol for Discrete-Log Keys



Extract ← x_A

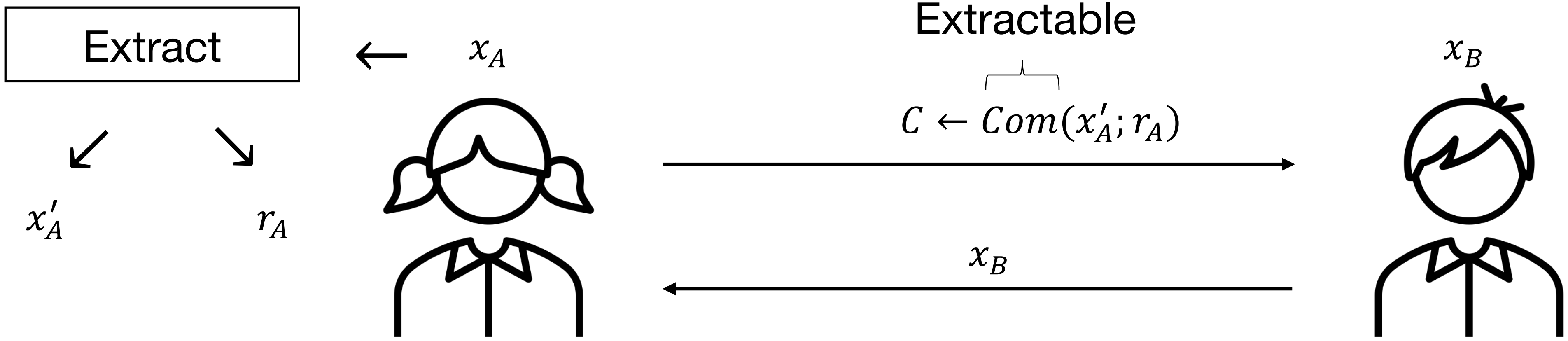
x'_A r_A

x_B

$C \leftarrow \overbrace{Com}^{\text{Extractable}}(x'_A; r_A)$

x_B

A Protocol for Discrete-Log Keys



Extract ← x_A

x'_A r_A

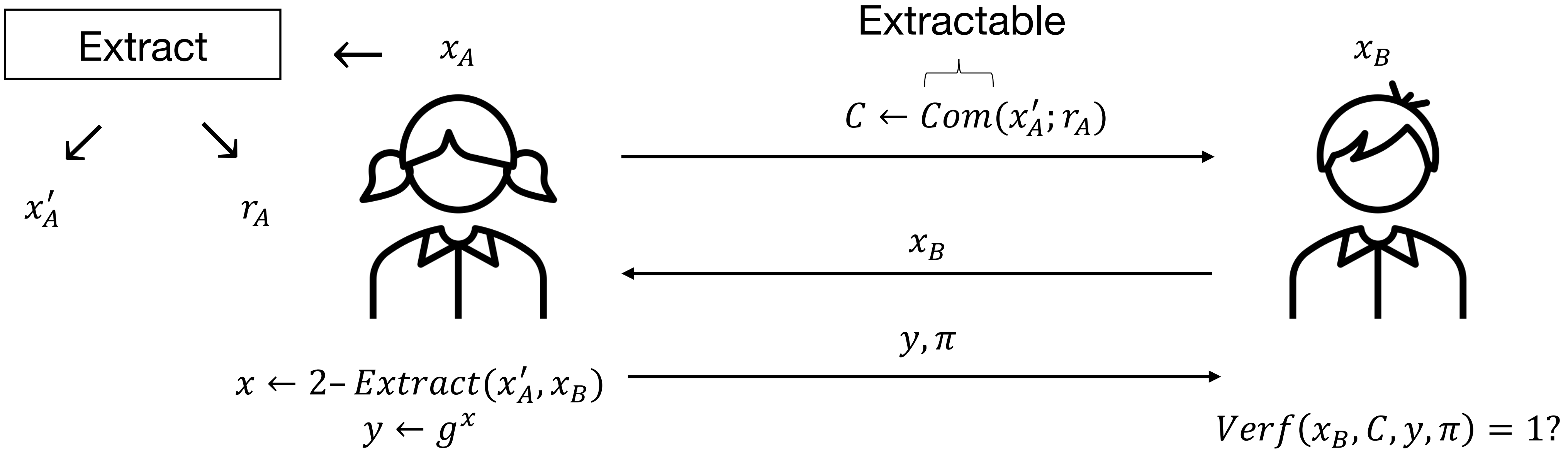
$$C \leftarrow \overbrace{Com}^{\text{Extractable}}(x'_A; r_A)$$

x_B

x_B

$$x \leftarrow 2\text{–}Extract(x'_A, x_B)$$
$$y \leftarrow g^x$$

A Protocol for Discrete-Log Keys



Extract ← x_A

↙ ↘

x'_A r_A

x_B

$C \leftarrow \overbrace{Com}^{\text{Extractable}}(x'_A; r_A)$ →

x_B ←

y, π →

$x \leftarrow 2\text{-}Extract(x'_A, x_B)$

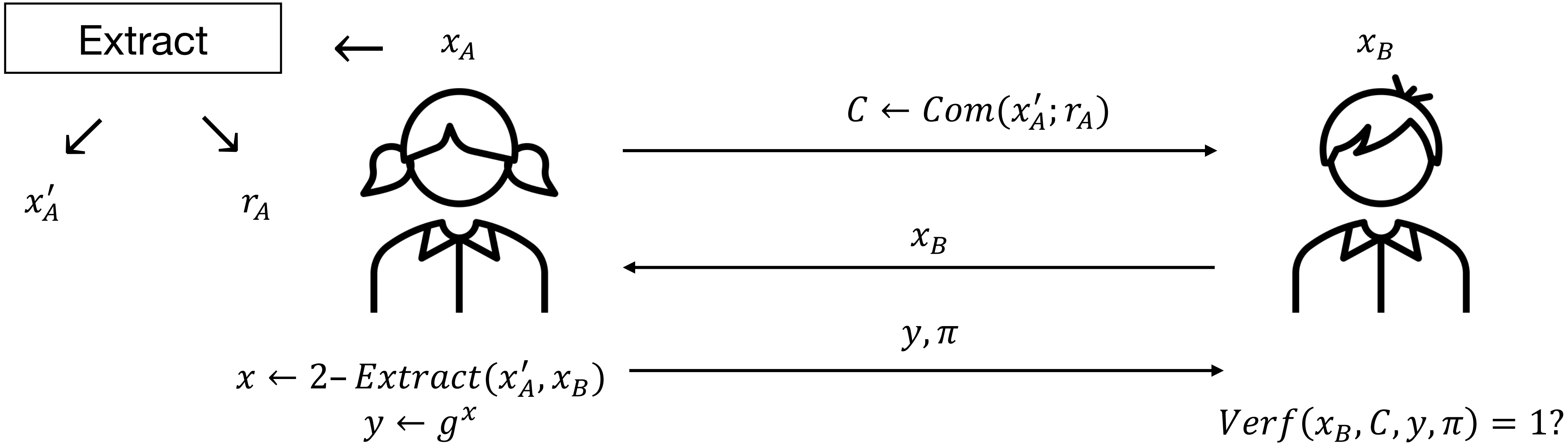
$y \leftarrow g^x$

$Verf(x_B, C, y, \pi) = 1?$

$$\pi \coloneqq Prove\left(x'_A, r_A : C = Com(x'_A; r_A) \ \wedge \ y = g^{2-Extract(x'_A, x_B)}\right)$$

A Protocol for Discrete-Log Keys

Deterministic extractors for all sources do not exist



Extract ← x_A

x'_A r_A

x_B

$C \leftarrow Com(x'_A; r_A)$

x_B

y, π

$x \leftarrow 2\text{–}Extract(x'_A, x_B)$

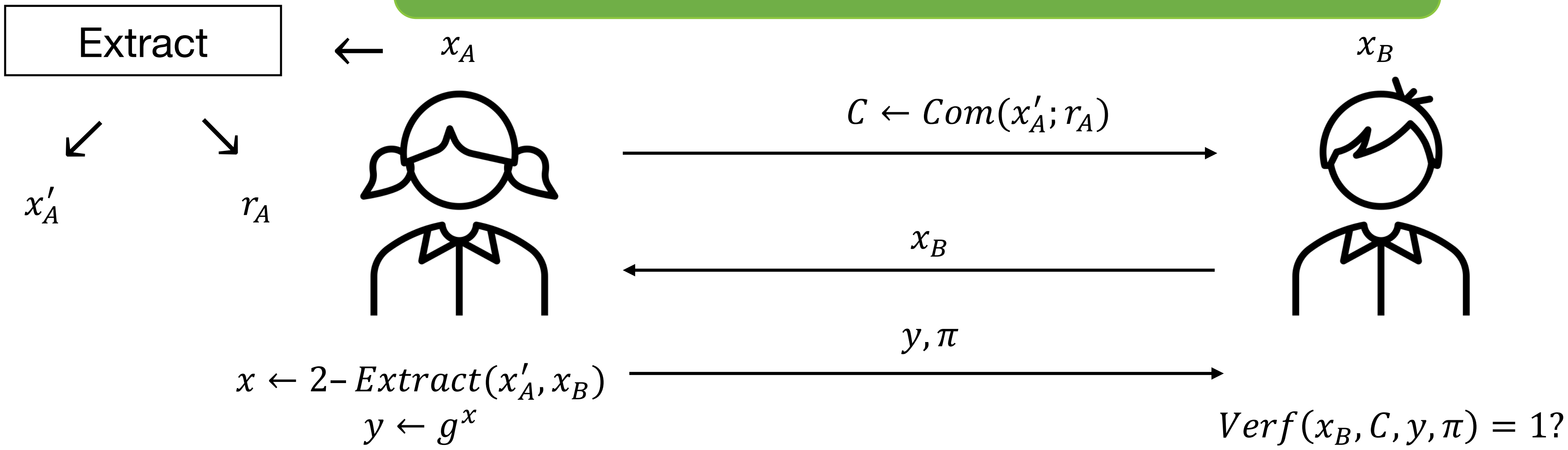
$y \leftarrow g^x$

$Verf(x_B, C, y, \pi) = 1?$

$$\pi \coloneqq Prove\left(x'_A, r_A : C = Com(x'_A; r_A) \wedge y = g^{2\text{–}Extract(x'_A, x_B)}\right)$$

A Protocol for Discrete-Log Keys

Ideas can be generalized to all probabilistic circuits



Extract $\leftarrow$ x_A

$\swarrow$ $\searrow$

x'_A r_A

x_B

$C \leftarrow Com(x'_A; r_A)$

x_B

y, π

$x \leftarrow 2\text{–}Extract(x'_A, x_B)$

$y \leftarrow g^x$

$Verf(x_B, C, y, \pi) = 1?$

$$\pi \coloneqq Prove\left(x'_A, r_A : C = Com(x'_A; r_A) \ \wedge \ y = g^{2\text{–}Extract(x'_A, x_B)}\right)$$

RSA Key Generation [NIST Standard]

- Choose at random two distinct large primes p and q
- $N \leftarrow pq$ and $\varphi(N) = (p-1)(q-1)$
- Choose $2^{16} < e \ < 2^{256}$ such that $\mathrm{gcd}\big(e, \varphi(N)\big) = 1$; $d \leftarrow [e^{-1} \bmod \varphi(N)]$
- $pk \leftarrow (N, e)$ and $sk \leftarrow (N, d)$ (or (p, q, e))

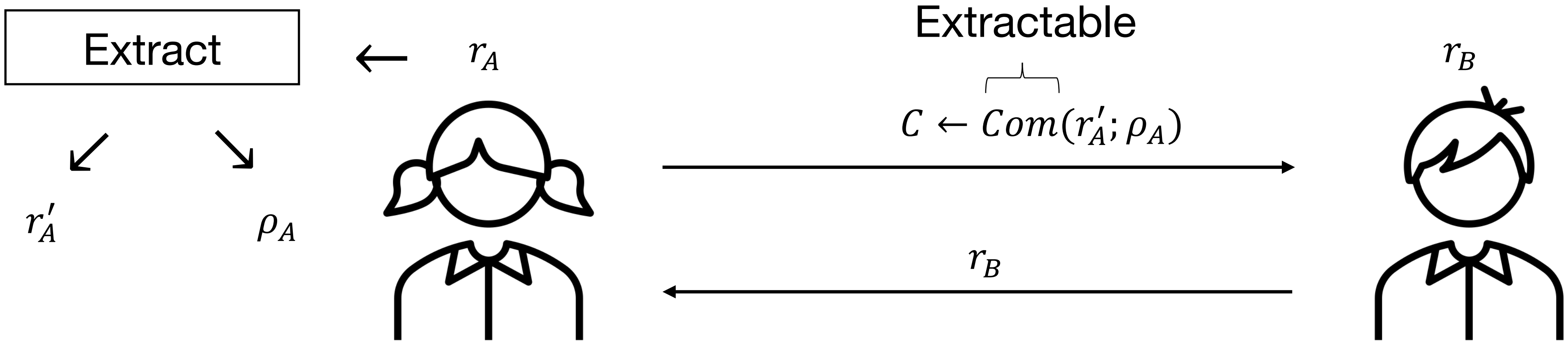
RSA Key Generation [NIST Standard – Interpretation]

- Choose at random two distinct $\overbrace{\text{large}}^{[2^{b-1}; 2^{b}]}$ primes p and q
- $N \leftarrow pq$ and $\varphi(N) = (p-1)(q-1)$
- Choose $2^{16} < e \; < 2^{256}$ such that $\gcd\big(e, \varphi(N)\big) = 1$; $d \leftarrow [e^{-1} \bmod \varphi(N)]$
- $pk \leftarrow (N, e)$ and $sk \leftarrow (p, q, e)$

RSA Key Generation [NIST Standard – Interpretation]

- Choose at random $\underbrace{\text{two}}_{\text{The first two}}$ distinct large $\underbrace{\overbrace{\text{primes}}^{\text{Potentially some additional conditions, e.g., safe}}}_{PrimeTest\ \text{Algorithm}}$ p and q
- $N \leftarrow pq$ and $\varphi(N) = (p-1)(q-1)$
- Choose $2^{16} < e < 2^{256}$ such that $\underbrace{\gcd\big(e, \varphi(N)\big) = 1}_{\text{Part of } PrimeTest}$; $d \leftarrow [e^{-1} \bmod \varphi(N)]$
- $pk \leftarrow (N, e)$ and $sk \leftarrow (p, q, e)$

A Protocol for RSA Keys



Extract $\leftarrow$ r_A

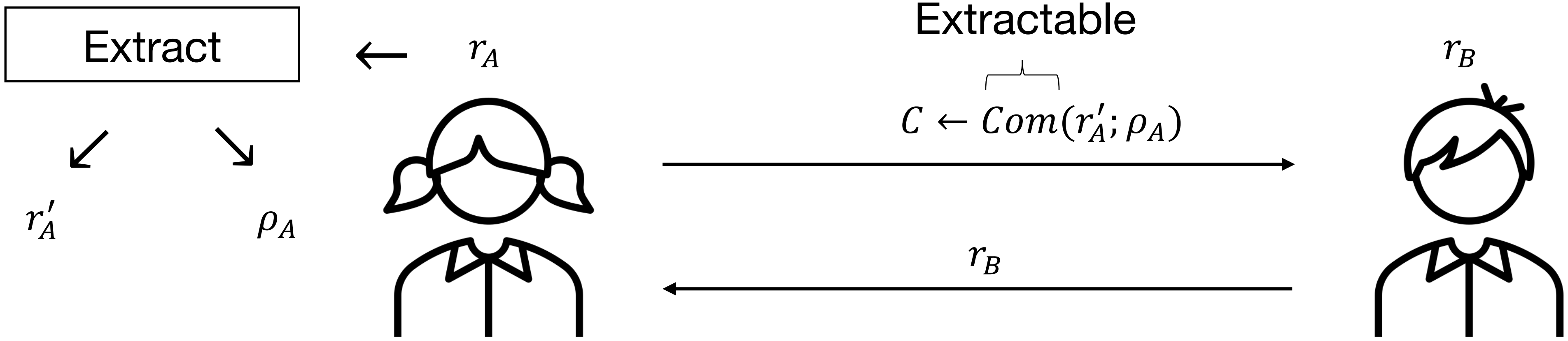
r'_A ρ_A

r_B

$C \leftarrow \overbrace{Com}^{\text{Extractable}}(r'_A; \rho_A)$

r_B

A Protocol for RSA Keys



Extract $\leftarrow$ r_A

$\swarrow$ $\searrow$

r'_A ρ_A

$$C \leftarrow \overbrace{Com}^{\text{Extractable}}(r'_A; \rho_A)$$

r_B

r_B

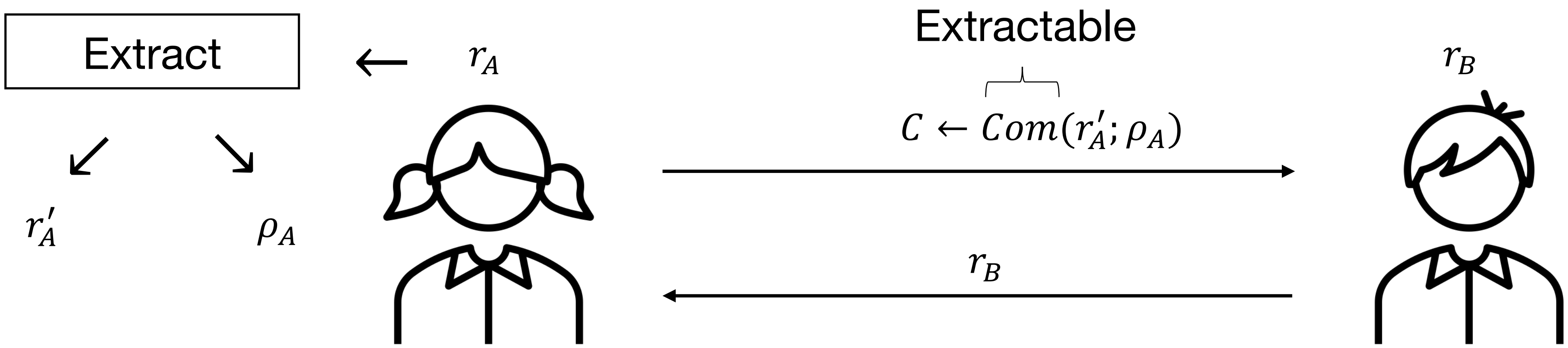
$$s \leftarrow 2\text{–}Extract(r'_A, r_B)$$

$$a_1, \dots, \underbrace{a_i}_{p}, \dots, \underbrace{a_j}_{q} \leftarrow PRF(s, \cdot)$$

counter mode

First two primes that pass $PrimeTest$

A Protocol for RSA Keys



Extract $\leftarrow$ r_A

$\swarrow$ $\searrow$

r'_A ρ_A

$$C \leftarrow Com(\overbrace{r'_A}^{\text{Extractable}}; \rho_A) \longrightarrow$$

$$\longleftarrow r_B$$

r_B

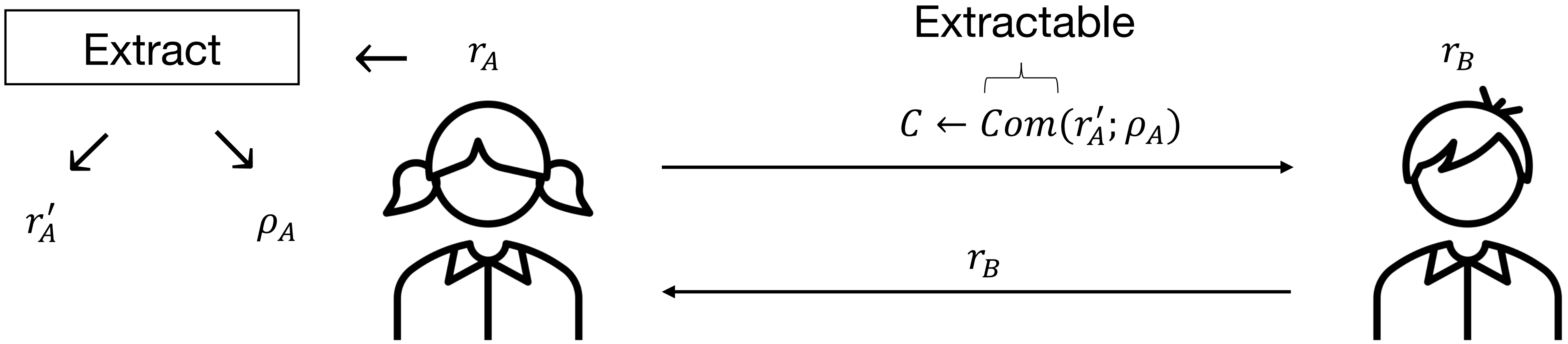
$$s \leftarrow 2\text{–}Extract(r'_A, r_B)$$

$$a_1, \ldots, p, \ldots, q \ \leftarrow PRF(s,\cdot)$$

$$N \leftarrow pq$$

$$\pi \leftarrow proof\ that\ N\ is\ the\ product\ of\ two\ \boldsymbol{integers}\ returned\ by\ PRF\ on\ seed\ s, that\ pass\ PrimeTest,\ and\ are\ of\ binary\ length\ b$$

A Protocol for RSA Keys



$$\boxed{\text{Extract}} \leftarrow r_A$$

$$\swarrow \quad \searrow$$

r'_A ρ_A

r_B

$$C \leftarrow Com(\overbrace{r'_A}^{\text{Extractable}}; \rho_A) \longrightarrow$$

$$\xleftarrow{r_B}$$

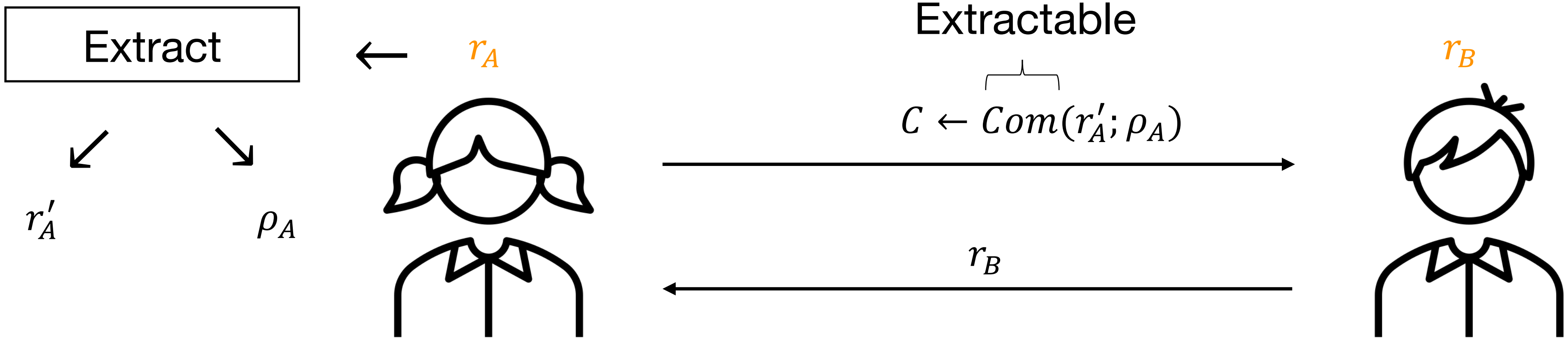
$$\xrightarrow{N, e, (a_k)_{\{k \neq i,j\}}, i, \pi}$$

$$s \leftarrow 2\text{–}Extract(r'_A, r_B)$$
$$a_1, \dots, p, \dots, q \leftarrow PRF(s, \cdot)$$
$$N \leftarrow pq$$
$$\pi \leftarrow proof\ that\ N\ is\ the\ product\ of\ two\ \boldsymbol{integers}$$
$$returned\ by\ PRF\ on\ seed\ s, that\ pass\ PrimeTest,$$
$$and\ are\ of\ binary\ length\ b$$
$$+(a_k)_{\{k \neq i,j\}}\ are\ returned\ by\ PRF$$

$$Verf(r_B, C, \pi, N, e) = 1?$$
$$\boxed{for\ k \neq i, j, PrimeTest(a_k) = 0?}$$

crucial to ensure that p and q really were the first two appropriate primes

A Protocol for RSA Keys



Extract ← r_A

↙ ↘

r'_A ρ_A

r_B

$C \leftarrow \overbrace{Com}^{\text{Extractable}}(r'_A; \rho_A)$ →

← r_B

$N, e, (a_k)_{\{k \neq i,j\}}, i, \pi$ →

$s \leftarrow 2\text{–}Extract(r'_A, r_B)$

$a_1, \ldots, p, \ldots, q \leftarrow PRF(s, \cdot)$

$N \leftarrow pq$

$\pi \leftarrow \boxed{proof}$ *that N is the product of two* ***integers***
returned by PRF on seed s, that pass $\boxed{PrimeTest,}$
and are of binary length b

$Verf(r_B, C, \pi, N, e) = 1?$

$for\ k \neq i, j, a_k is\ returned\ by\ PRF\ on\ s$

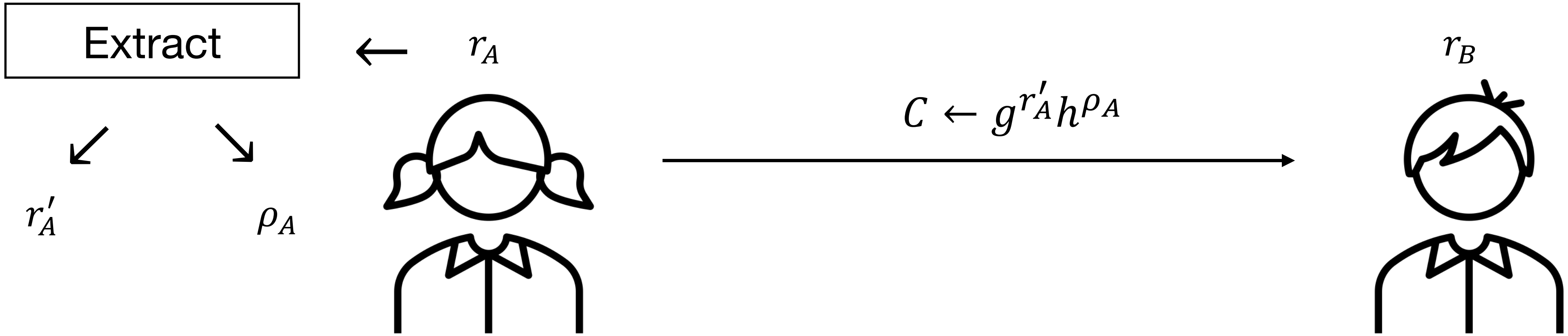
$and\ \boxed{PrimeTest}(a_k) = 0?$

require randomness

$\Rightarrow PRF'\ on\ seeds\ extracted\ from$

A Protocol for RSA Keys – Instantiation

$G = < g >$ $ord(G)$ public and prime



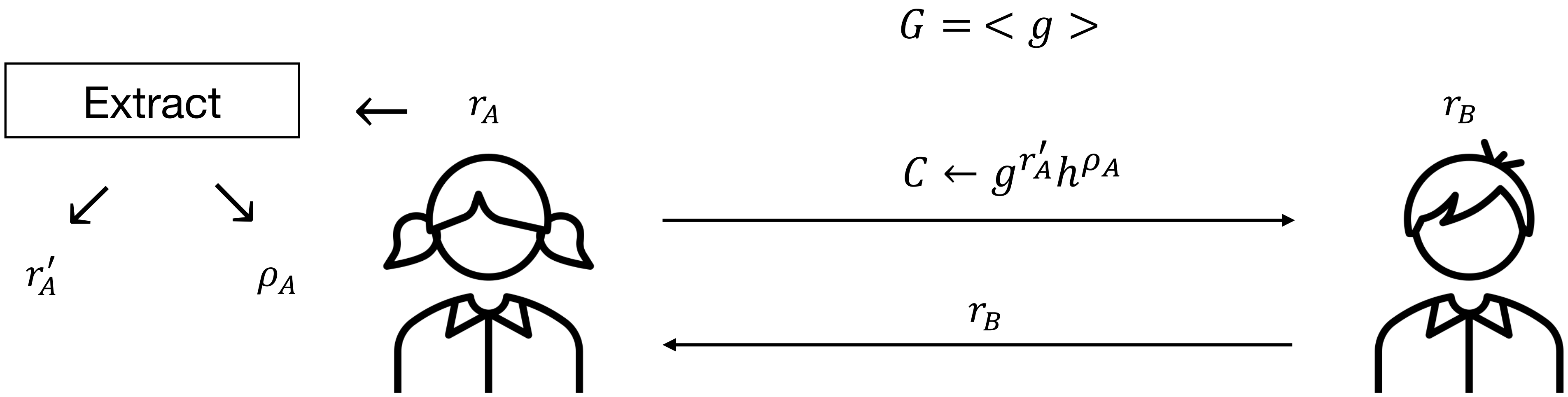
Extract $\leftarrow$ r_A

r_B

$C \leftarrow g^{r'_A} h^{\rho_A}$

r'_A ρ_A

A Protocol for RSA Keys – Instantiation



$G = < g >$

Extract $\leftarrow r_A$

r'_A ρ_A

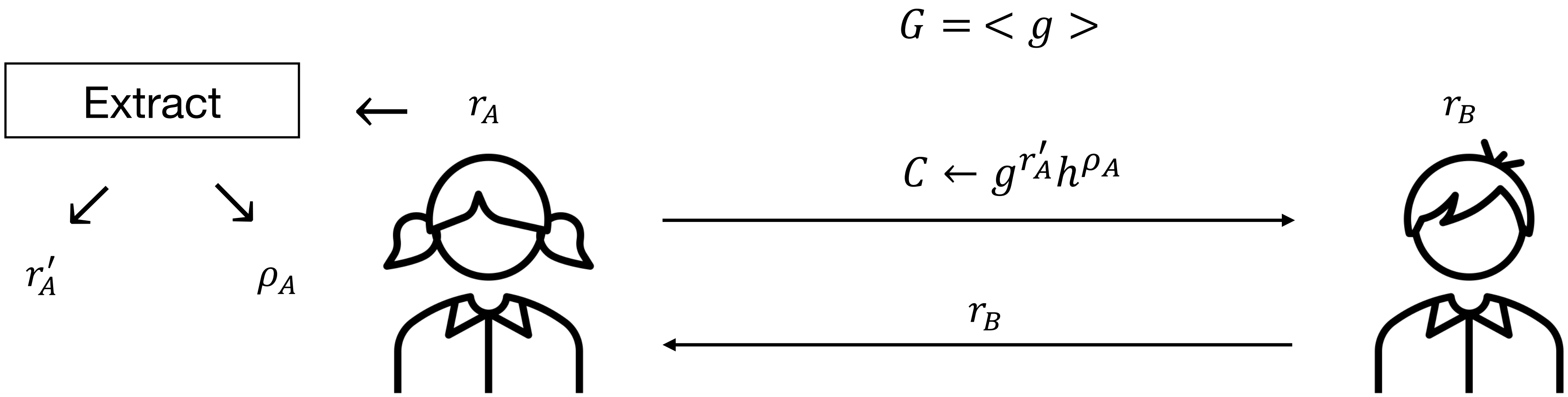
r_B

$C \leftarrow g^{r'_A} h^{\rho_A}$

r_B

$s \leftarrow r'_A + H(r_B) \; mod \; \ell$, with ℓ Sophie-Germain prime s.t. $\ell | \; ord(G) \; - 1$ and $ord(G) > (2\ell + 1)^2$

A Protocol for RSA Keys – Instantiation



$G = < g >$

Extract $\leftarrow$ r_A

r'_A ρ_A

r_B

$C \leftarrow g^{r'_A} h^{\rho_A}$

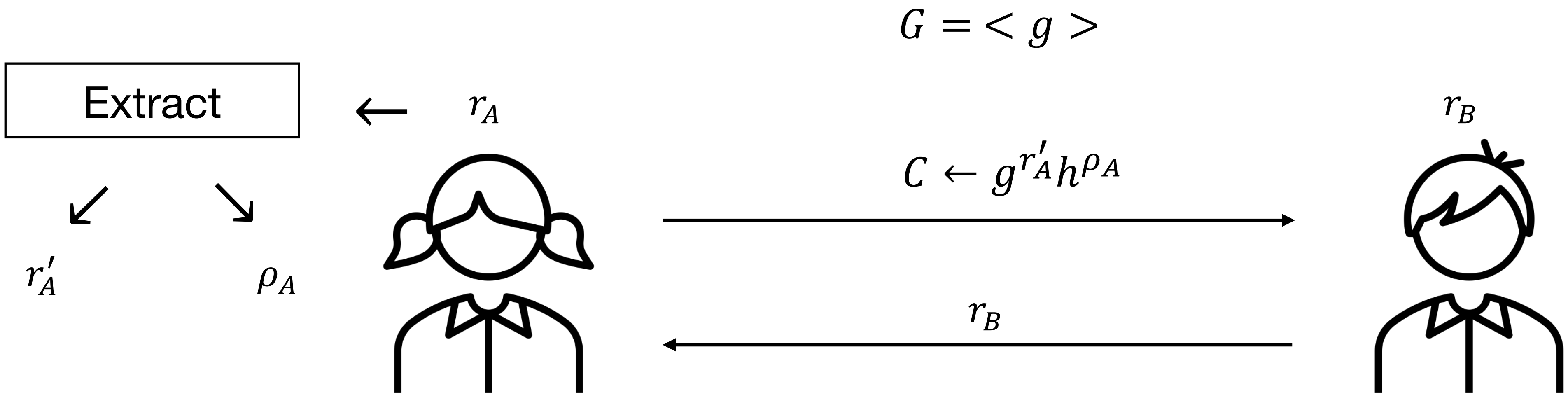
r_B

$s \leftarrow r'_A + H(r_B)\ mod\ \ell$, with ℓ Sophie-Germain prime s.t. $\ell |\ ord(G)\ - 1$ and $ord(G) > (2\ell + 1)^2$

PRF: Dodis–Yampolskiy in the group $QR_{2\ell+1} = < a >$

$$(s, x) \mapsto a^{1/(s+x)}\ mod\ \ell$$

A Protocol for RSA Keys – Instantiation



$G = < g >$

Extract $\leftarrow$ r_A

r'_A ρ_A

r_B

$C \leftarrow g^{r'_A} h^{\rho_A}$

r_B

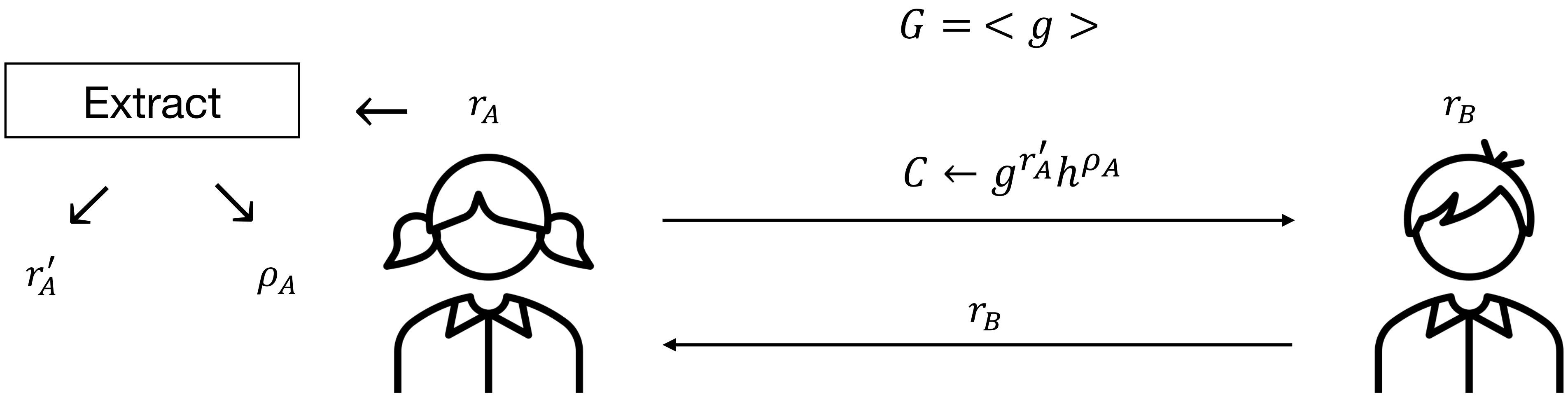
$s \leftarrow r'_A + H(r_B)\ mod\ \ell$

PRF: Dodis–Yampolskiy in the group $QR_{2\ell+1} = < a >$

$$(s, x) \mapsto a^{1/(s+x)}\ mod\ \ell$$

In π, compute $P \leftarrow g^p h^{r_P}$ and $Q \leftarrow g^q h^{r_q}$

A Protocol for RSA Keys – Instantiation



$G = < g >$

Extract $\leftarrow$ r_A

$\swarrow$ $\searrow$

r'_A ρ_A

r_B

$C \leftarrow g^{r'_A} h^{\rho_A}$ $\longrightarrow$

r_B $\longleftarrow$

≈ proving knowledge of x s.t. $y = g^{a^x}$

i.e., a “double discrete logarithm”

Double Discrete Logarithm $y = g^{a^x}$

- Introduced by Stadler at EUROCRYPT'96 for VSS
- Later used to build GS [CS97], e-cash [CG07], credentials [CGM12]
- Only method known so far to prove knowledge of DDLogs in ZK had $\Omega(\log ord(G))$ (prover) communication complexity because of $\{0,1\}$ challenges
- Using "Bulletproofs" [BBBPWM18] for arithmetic circuits, our proof has $O(\log\log ord(G))$ communication complexity

$$G = < g >, y = g^{a^x}$$

$$x = \sum_{0 \leq i \leq n} x_i 2^i \text{, with } x_i \in \{0,1\}$$

$$a^x = \prod_{0 \leq i \leq n} \left(a^{2^i}\right)^{x_i} = \prod_{0 \leq i \leq n} a_i \ mod\ ord(G), \text{with } a_i \in \left\{1, a^{2^i}\right\}$$

$$y = g^{\prod_i a_i}$$

$$G = < g >, y = g^{a^x}$$

over $\mathbb{Z}$ (instead of $\mathbb{Z}_{ord(G)}$)

$$a^x - \prod_{0 \le i \le n} a_i = 0$$

$$G = < g >, y = g^{a^x}$$

$$\left(a^x - \prod_{0 \le i \le n} a_i\right)^2 + \sum_{0 \le i \le n} \left((a_i - 1)\left(a_i - a^{2^i}\right)\right)^2 = 0$$

$$G = < g >, y = g^{a^x}$$

$$\left(a^x - \prod_{0 \le i \le n} a_i\right)^2 + \sum_{0 \le i \le n} \left((a_i - 1)\left(a_i - a^{2^i}\right)\right)^2 = 0$$

$$\begin{gathered} b_0 \leftarrow a_0 \\ b_1 \leftarrow b_0 a_1 \\ \vdots \\ b_{n-1} \leftarrow b_{n-2} a_{n-1} \end{gathered}$$

$$G = < g >, y = g^{a^x}$$

$$(b_0 - a_0)^2 + \sum_{1 \le i \le n-1} (b_i - a_i b_{i-1})^2 + (a^x - a_n b_{n-1})^2 + \sum_{0 \le i \le n} \left((a_i - 1) \left(a_i - a^{2^i} \right) \right)^2 = 0$$

$$\begin{gathered} b_0 \leftarrow a_0 \\ b_1 \leftarrow b_0 a_1 \\ \vdots \\ b_{n-1} \leftarrow b_{n-2} a_{n-1} \end{gathered}$$

$G = < g >, y = g^{a^x}$

$$(b_0 - a_0)^2 + \sum_{1 \le i \le n-1} (b_i - a_i b_{i-1})^2 + (a^x - a_n b_{n-1})^2$$
$$+ \sum_{0 \le i \le n} \left((a_i - 1) \left(a_i - a^{2^i} \right) \right)^2 = 0$$

$$u_i \leftarrow a_i - 1$$
$$v_i \leftarrow a_i - a^{2^i}$$

$G = < g >, y = g^{a^x}$

$$(b_0 - a_0)^2 + \sum_{1 \le i \le n-1} (b_i - a_i b_{i-1})^2 + (a^x - a_n b_{n-1})^2$$
$$+ \sum_{0 \le i \le n} (u_i v_i)^2 + \sum_{0 \le i \le n} (u_i - a_i + 1)^2 + \left(v_i - a_i + a^{2^i}\right)^2 = 0$$

$$u_i \leftarrow a_i - 1$$
$$v_i \leftarrow a_i - a^{2^i}$$

$$G = < g >, y = g^{a^x}$$

$$(b_0 - a_0)^2 + \sum_{1 \le i \le n-1} (b_i - a_i b_{i-1})^2 + (a^x - a_n b_{n-1})^2$$
$$+ \sum_{0 \le i \le n} (u_i v_i)^2 + \sum_{0 \le i \le n} (u_i - a_i + 1)^2 + \left(v_i - a_i + a^{2^i}\right)^2 = 0$$

$$G = < g >, y = g^{a^x}$$

$$(b_0 - a_0)^2 + \sum_{1 \leq i \leq n-1} (b_i - a_i b_{i-1})^2 + (a^x - a_n b_{n-1})^2$$
$$+ \sum_{0 \leq i \leq n} (u_i v_i)^2 + \sum_{0 \leq i \leq n} \underbrace{(u_i - a_i + 1)}_{}{}^2 + \underbrace{\left(v_i - a_i + a^{2^i}\right)}_{}{}^2 = 0$$

Linear

$$G = < g >, y = g^{a^x}$$

$$(b_0 - a_0 \ * 1)^2 + \sum_{1 \le i \le n-1} (b_i - a_i b_{i-1})^2 + (a^x - a_n b_{n-1})^2$$
$$+ \sum_{0 \le i \le n} (0 \ - u_i v_i)^2 + \sum_{0 \le i \le n} (u_i - a_i + 1)^2 + \left(v_i - a_i + a^{2^i}\right)^2 = 0$$

(over $\mathbb{Z}_{ord(G)}$) $\boldsymbol{a_L} \circ \boldsymbol{a_R} = \boldsymbol{a_O}$ and $\boldsymbol{W_L a_L} + \boldsymbol{W_R a_R} + \boldsymbol{W_O a_O} = \boldsymbol{W_V} \ [a^x] + \boldsymbol{C}$

Randomness Certification – Open Problems

- Would it be possible to use Bob's randomness to *amplify* Alice's instead of strictly requiring either to have high-entropy randomness?

Randomness Certification – Open Problems

- Would it be possible to use Bob's randomness to *amplify* Alice's instead of strictly requiring either to have high-entropy randomness?

- Can one devise a more realistic model in which
    - entropy is accumulated,
    - sources are not independent of the extractors? [CDKT19]