Non-Interactive Secure Computation of Inner-Product from LPN and LWE

Geoffroy Couteau, Maryam Zarezadeh

cnrs irif

barkhausen institut

Université de Paris

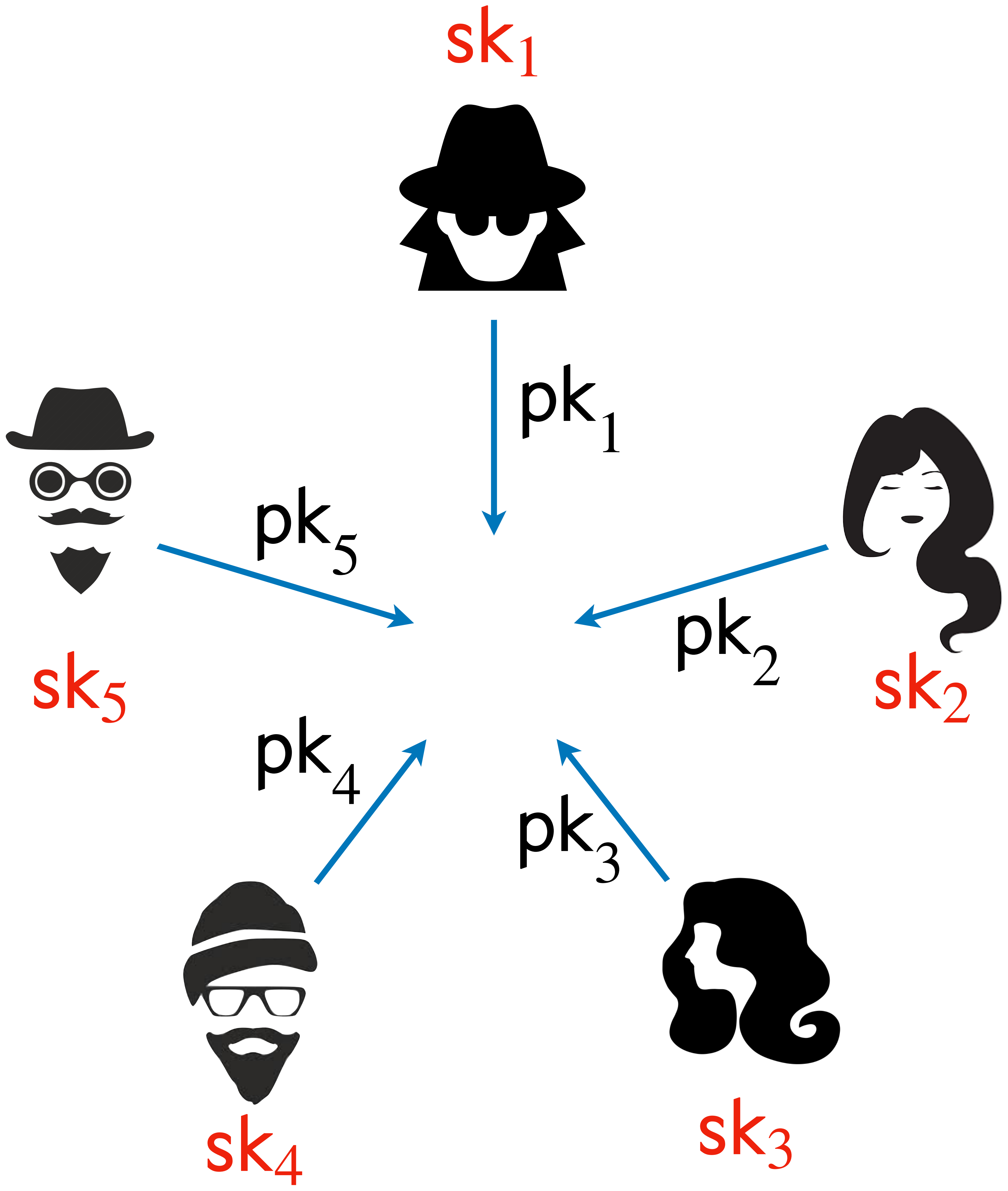
Non-Interactive Key Exchange

A very appealing interaction pattern:

- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work



Non-Interactive Key Exchange

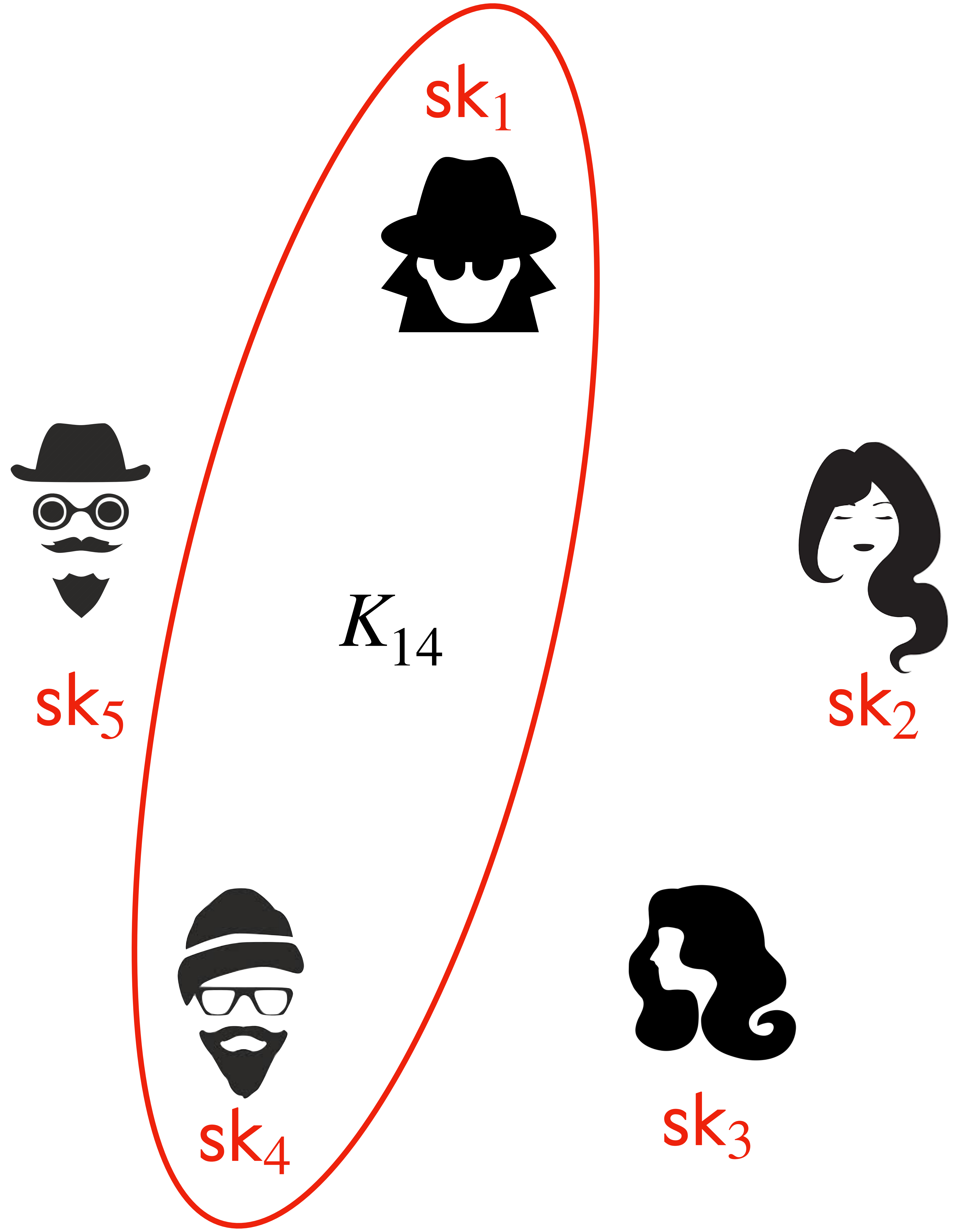
A very appealing interaction pattern:

- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work

$K_{14} \leftarrow \mathsf{Key}(\mathsf{sk}_1, \mathsf{pk}_4)$



$K_{14} \leftarrow \mathsf{Key}(\mathsf{sk}_4, \mathsf{pk}_1)$

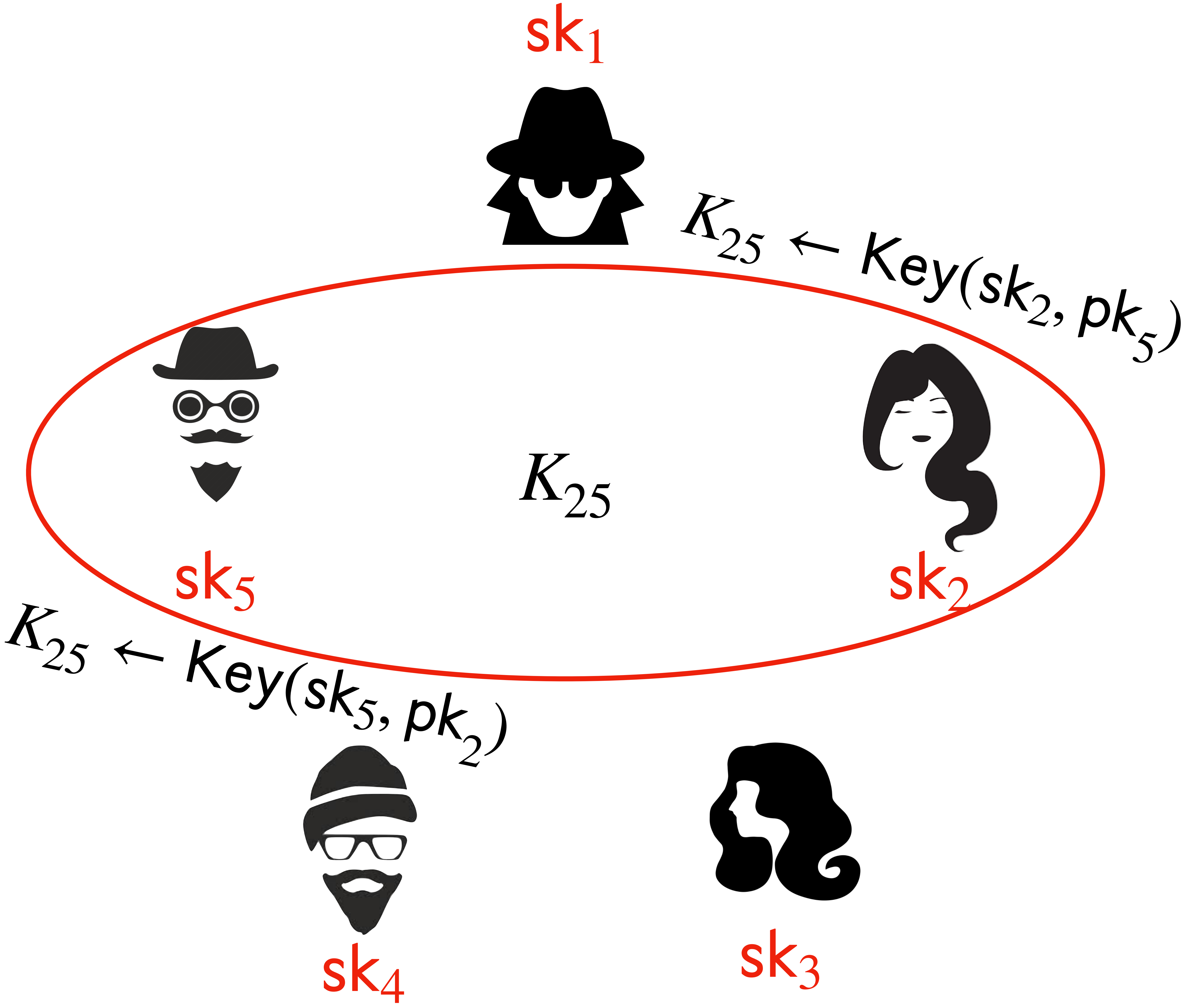
Non-Interactive Key Exchange

A very appealing interaction pattern:

- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work


sk_1
$K_{25} \leftarrow \mathsf{Key}(\mathsf{sk}_2, \mathsf{pk}_5)$
sk_5
K_{25}
sk_2
$K_{25} \leftarrow \mathsf{Key}(\mathsf{sk}_5, \mathsf{pk}_2)$
sk_4
sk_3

Non-Interactive Key Exchange

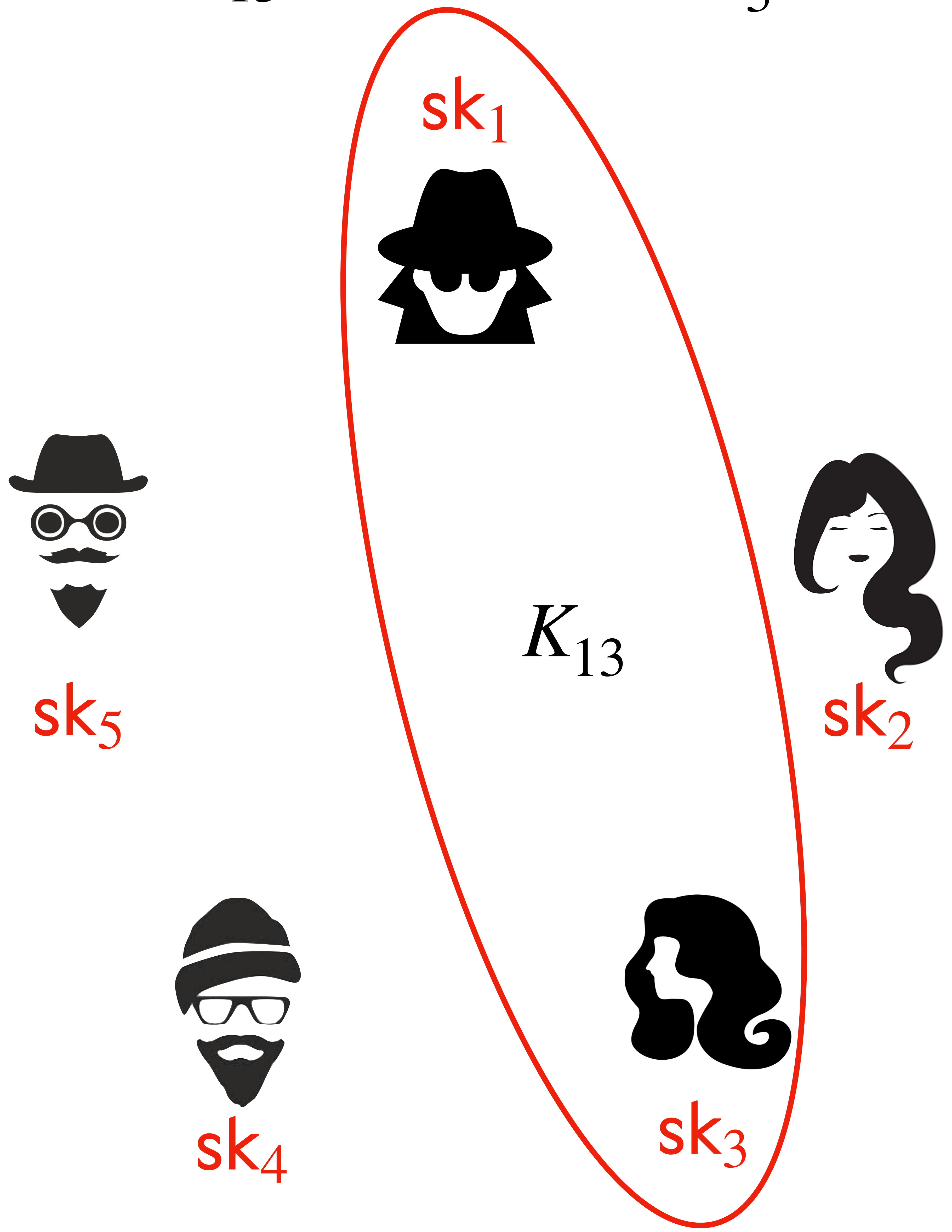
A very appealing interaction pattern:

- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work

$K_{13} \leftarrow \mathsf{Key}(\mathsf{sk}_1, \mathsf{pk}_3)$



$K_{13} \leftarrow \mathsf{Key}(\mathsf{sk}_3, \mathsf{pk}_1)$

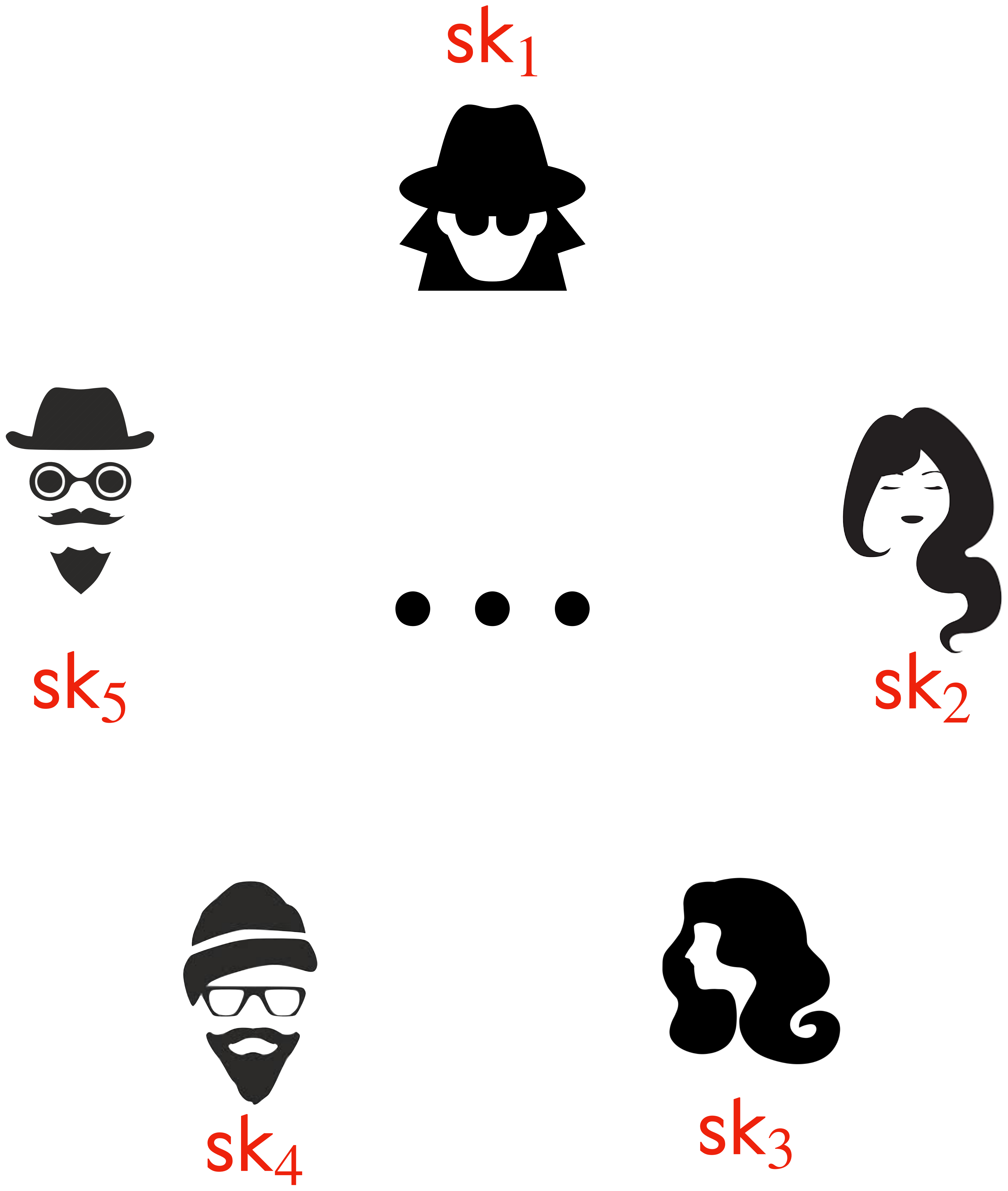
Non-Interactive Key Exchange

A very appealing interaction pattern:

- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work



Non-Interactive Key Exchange

A very appealing interaction pattern:

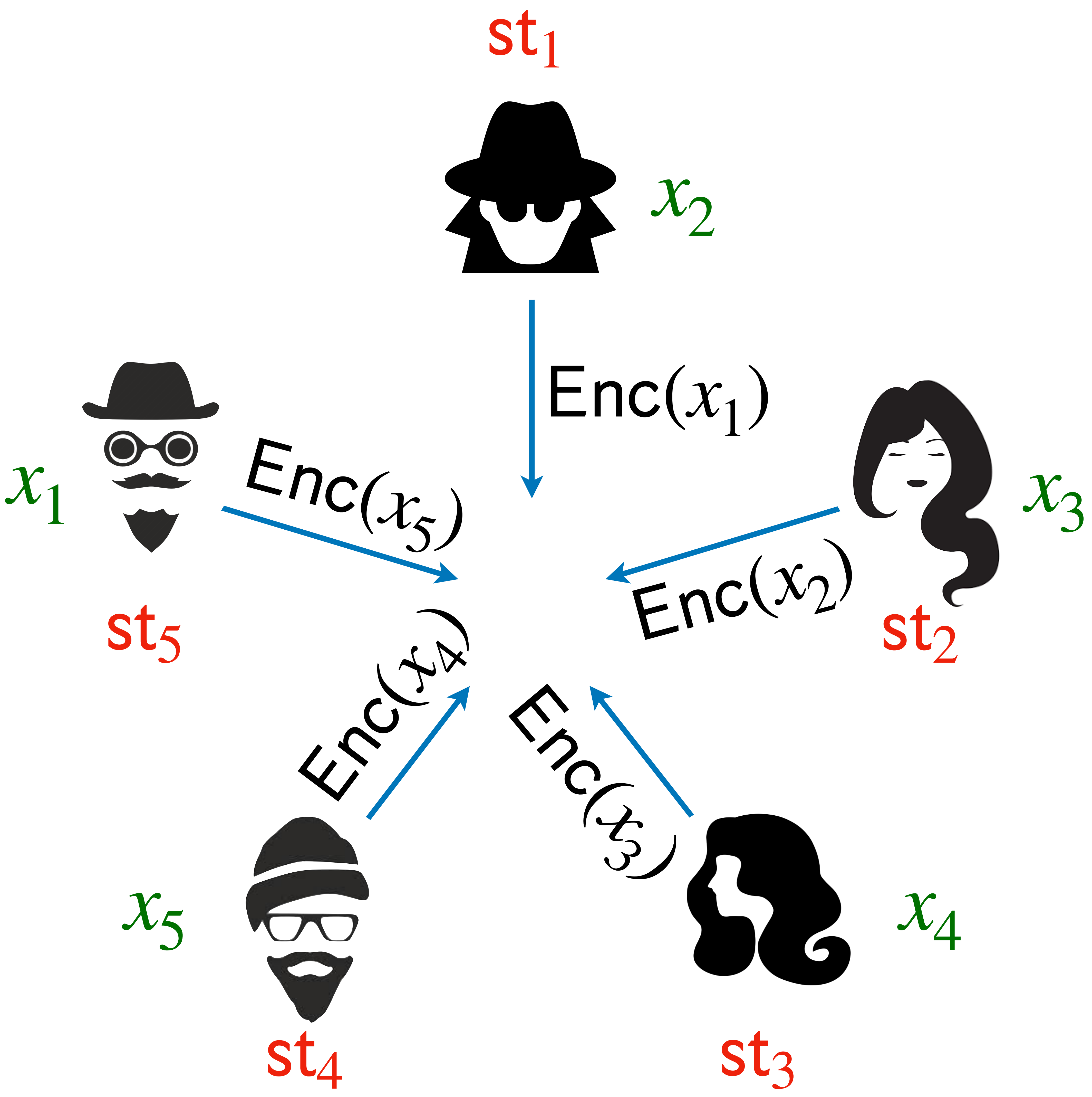
- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

Can we get a similar pattern for some simple MPC?

- n parties broadcast an *encoding* of their input
- Pairs (P_i, P_j) can compute $f_i(x_i, x_j)$ and $f_j(x_i, x_j)$ from their state and the other party's encoding
- Avoids the $\Omega(n^2)$ overhead

This Work



Non-Interactive Key Exchange

A very appealing interaction pattern:

- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

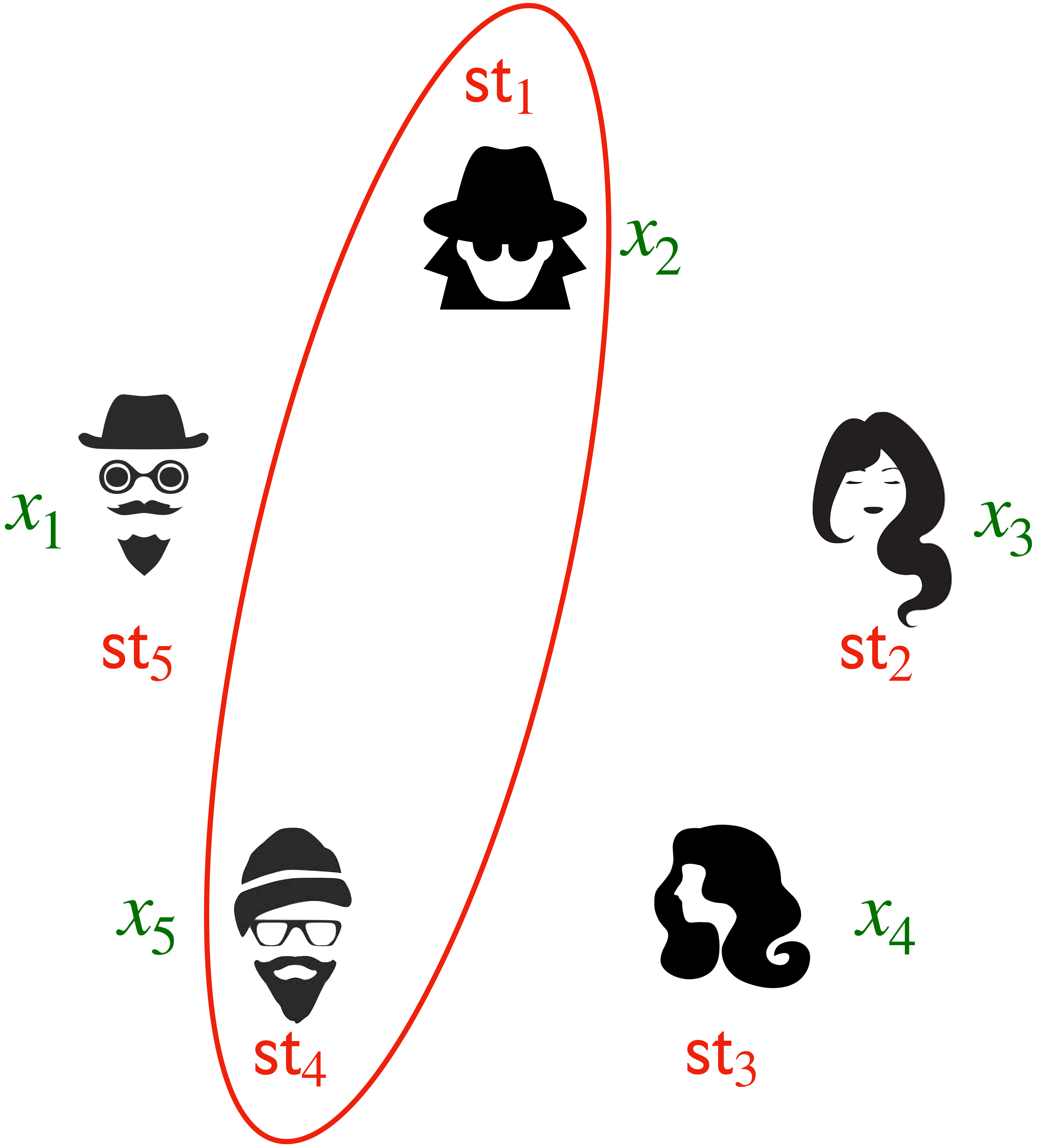
Non-Interactive Secure Computation

Can we get a similar pattern for some simple MPC?

- n parties broadcast an *encoding* of their input
- Pairs (P_i, P_j) can compute $f_i(x_i, x_j)$ and $f_j(x_i, x_j)$ from their state and the other party's encoding
- Avoids the $\Omega(n^2)$ overhead

This Work

$$f_1(x_1, x_4) = \mathsf{Out}(\mathsf{st}_1, \mathsf{Enc}(x_4))$$



$$f_1(x_1, x_4) = \mathsf{Out}(\mathsf{st}_4, \mathsf{Enc}(x_1))$$

Non-Interactive Key Exchange

A very appealing interaction pattern:

- n parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

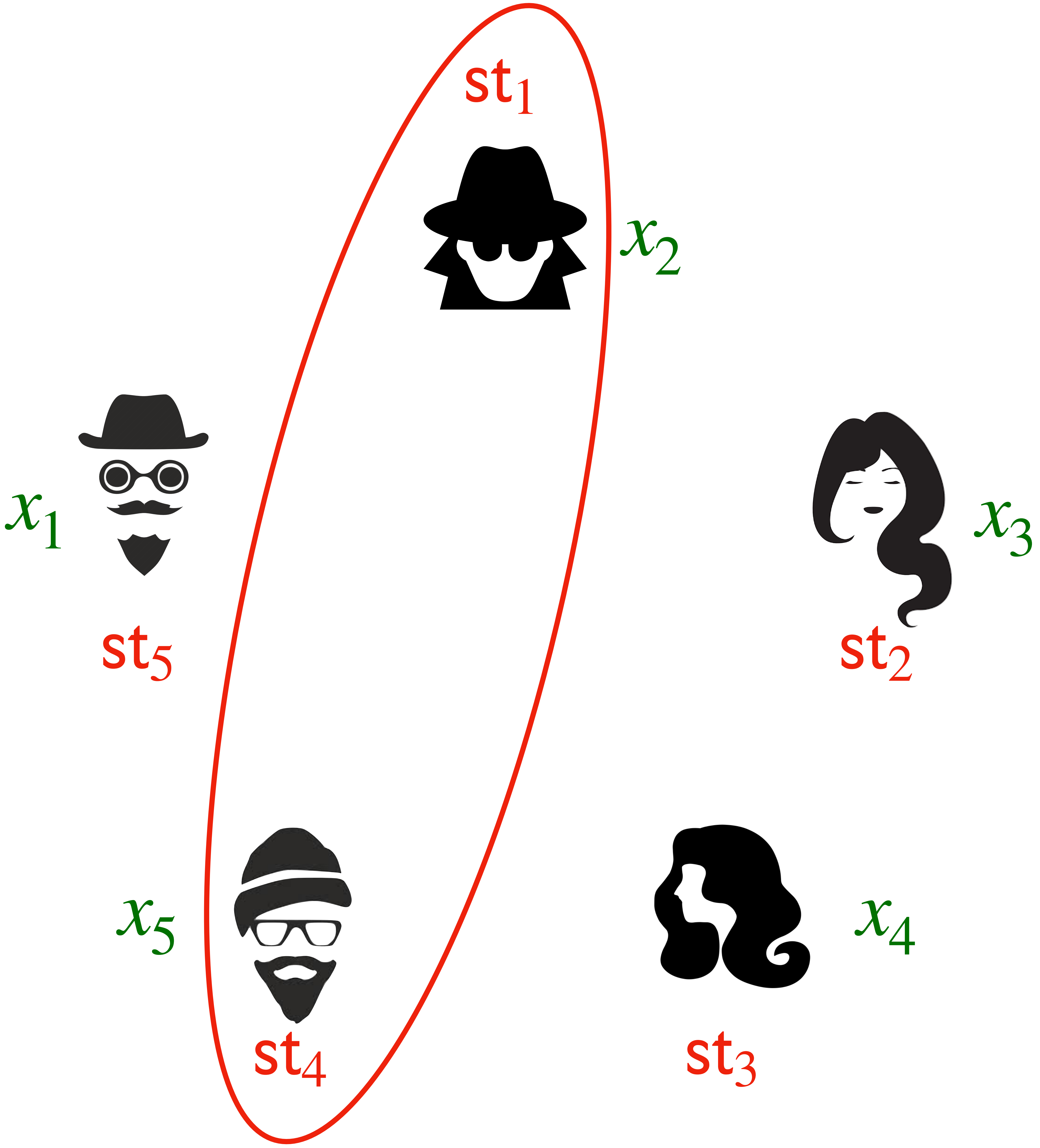
Can we get a similar pattern for some simple MPC?

- n parties broadcast an *encoding* of their input
- Pairs (P_i, P_j) can compute $f_i(x_i, x_j)$ and $f_j(x_i, x_j)$ from their state and the other party's encoding
- Avoids the $\Omega(n^2)$ overhead

This Work

- Non-interactive MPC for *shares of inner products:* $f_i(x_i, x_j), f_j(x_i, x_j)$ form shares of $\langle x_i, x_j \rangle$ over $\mathbb{F}$
- Reconstructing the result = sending a single element of $\mathbb{F}$

$$f_1(x_1, x_4) = \mathsf{Out}(\mathsf{st}_1, \mathsf{Enc}(x_4))$$

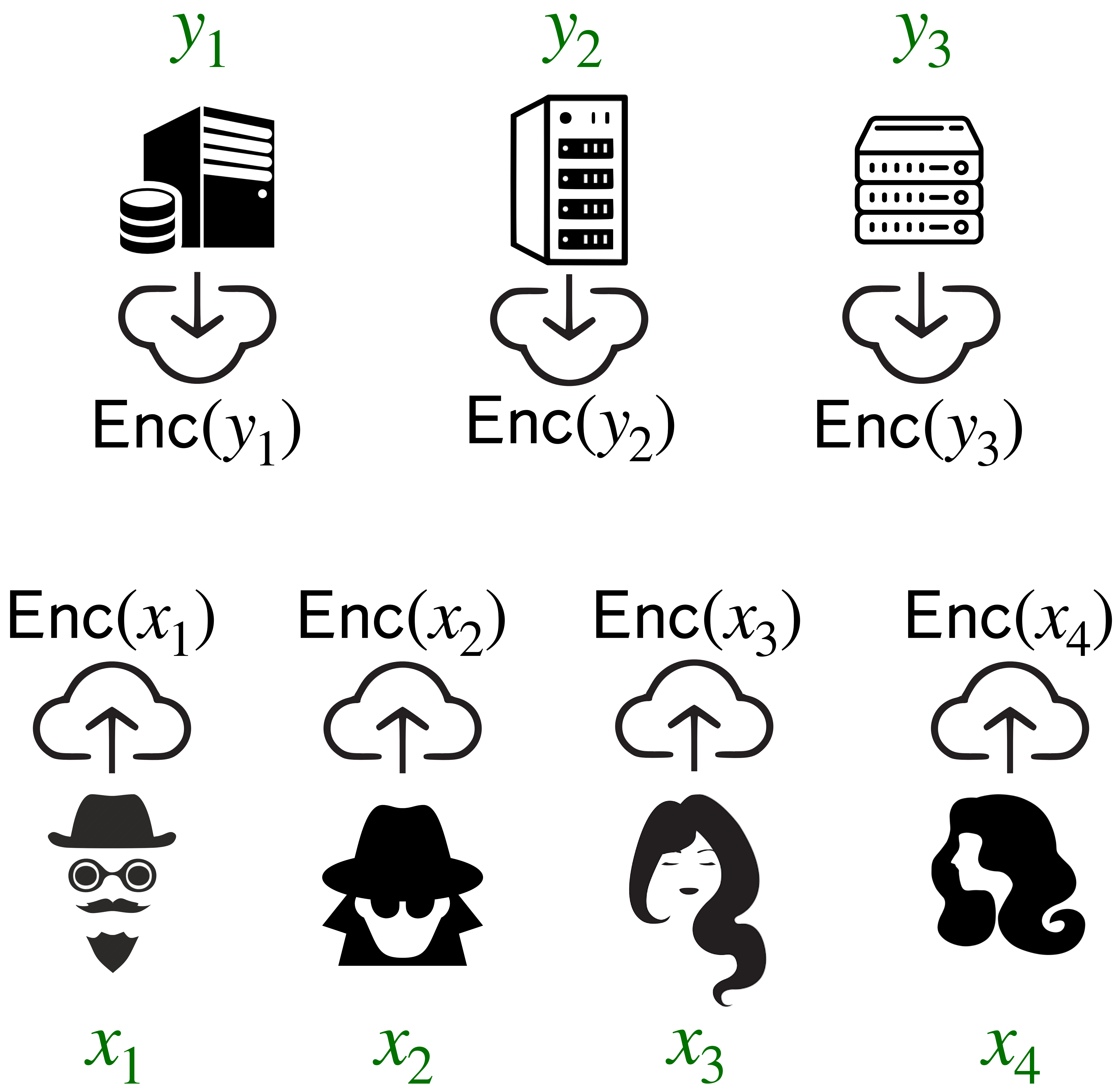


$$f_1(x_1, x_4) = \mathsf{Out}(\mathsf{st}_4, \mathsf{Enc}(x_1))$$

Non-Interactive Inner Product: Applications

Inner products is a simple, but very useful function:

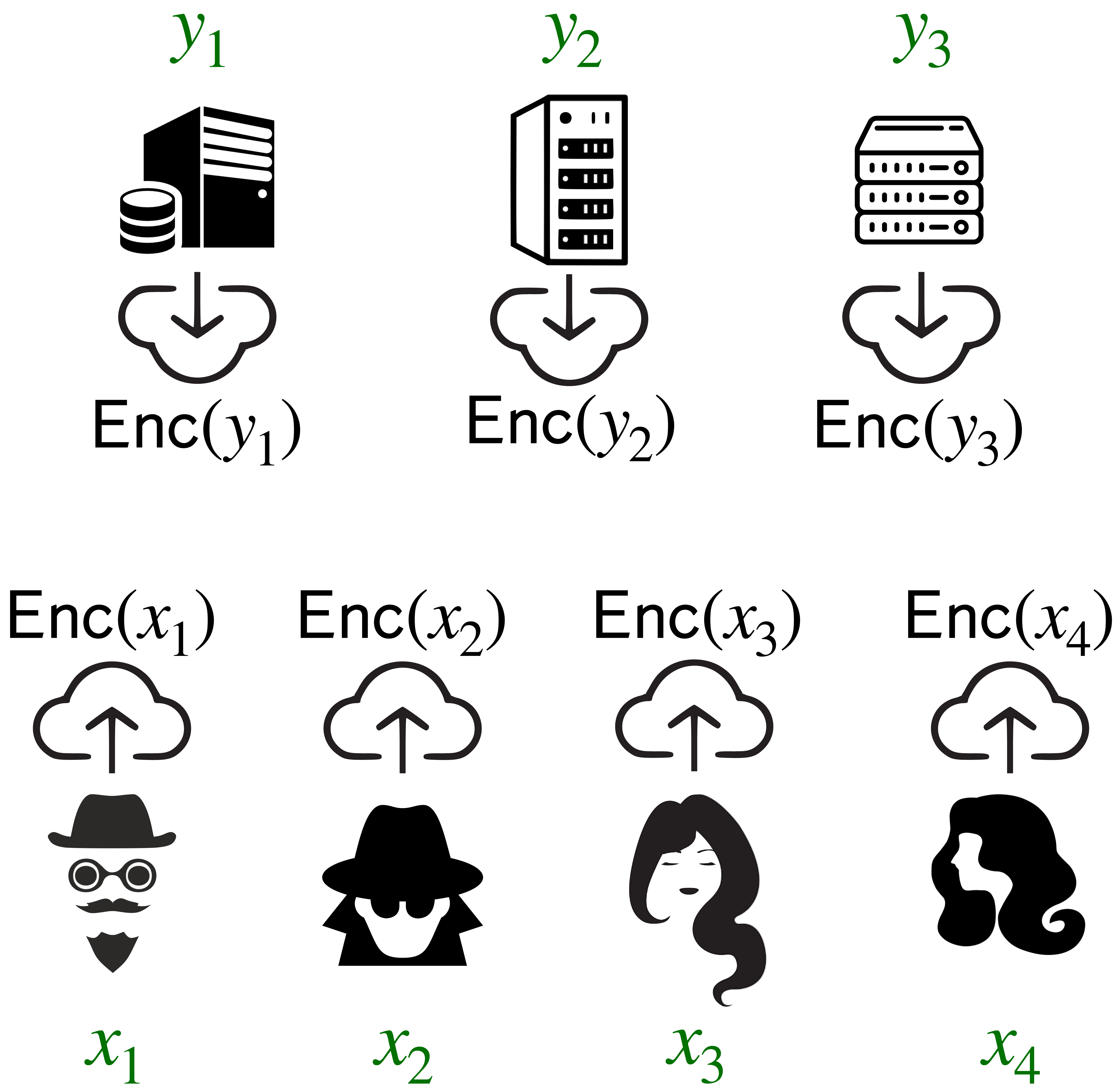
- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours,SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …



Non-Interactive Inner Product: Applications

Inner products is a simple, but very useful function:

- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours,SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …



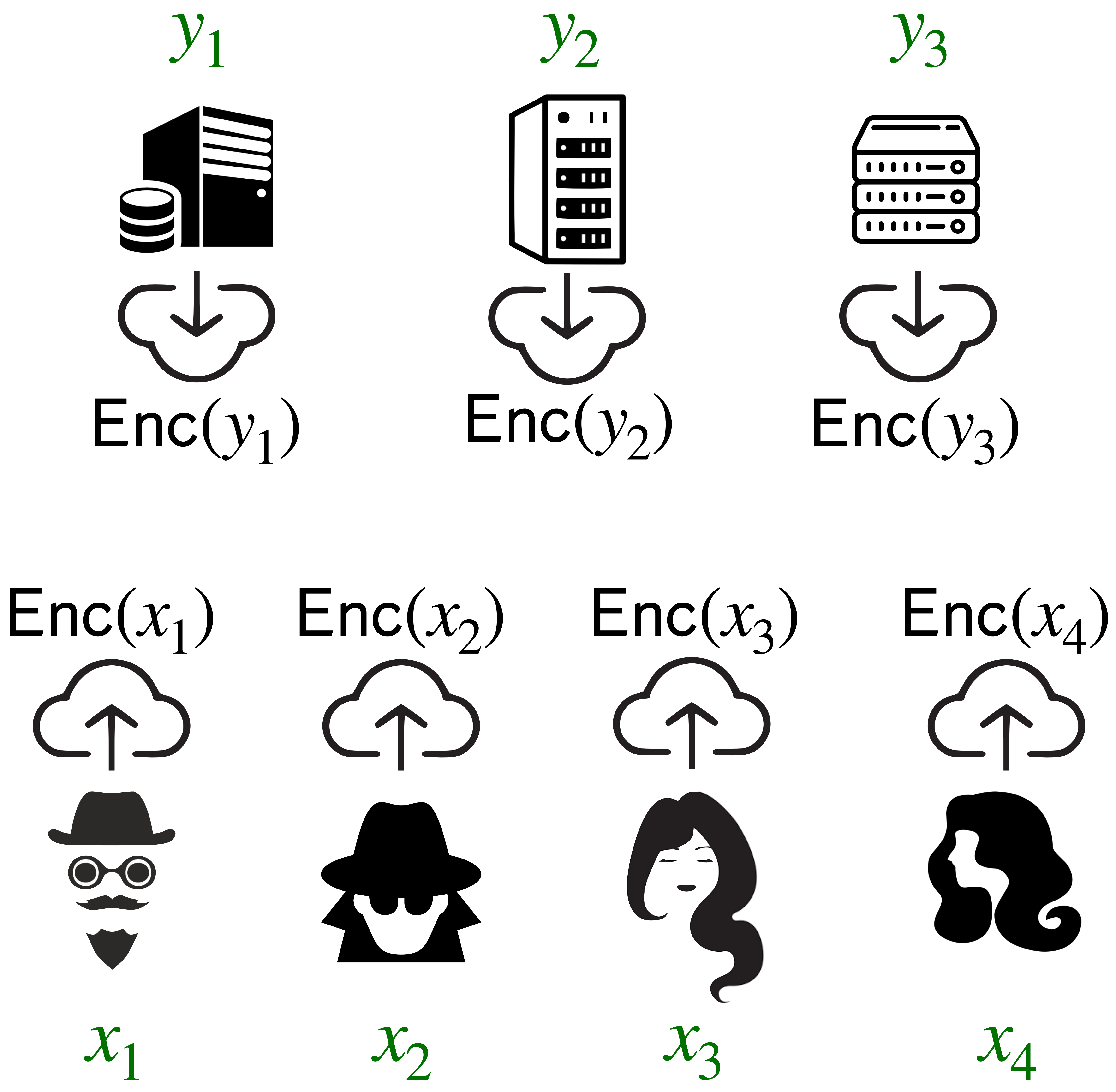
Non-Interactive Inner Product: Applications

Inner products is a simple, but very useful function:

- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours,SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …

Toy Example: Biometrics

- n clients and m servers with a fingerprint stored.
- Ahead of time, each party publishes an encoding of its fingerprint.



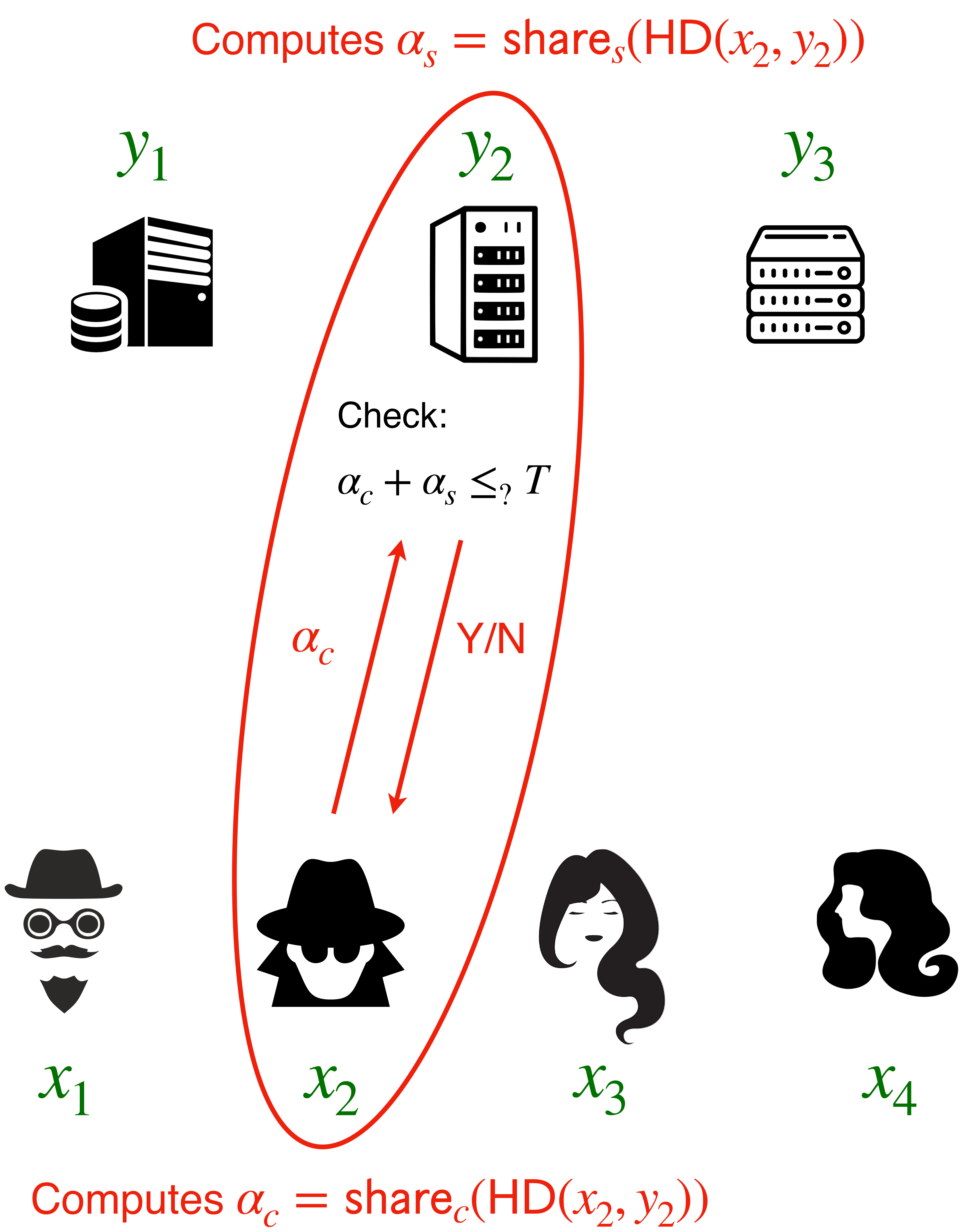
Non-Interactive Inner Product: Applications

Inner products is a simple, but very useful function:

- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours,SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …

Toy Example: Biometrics

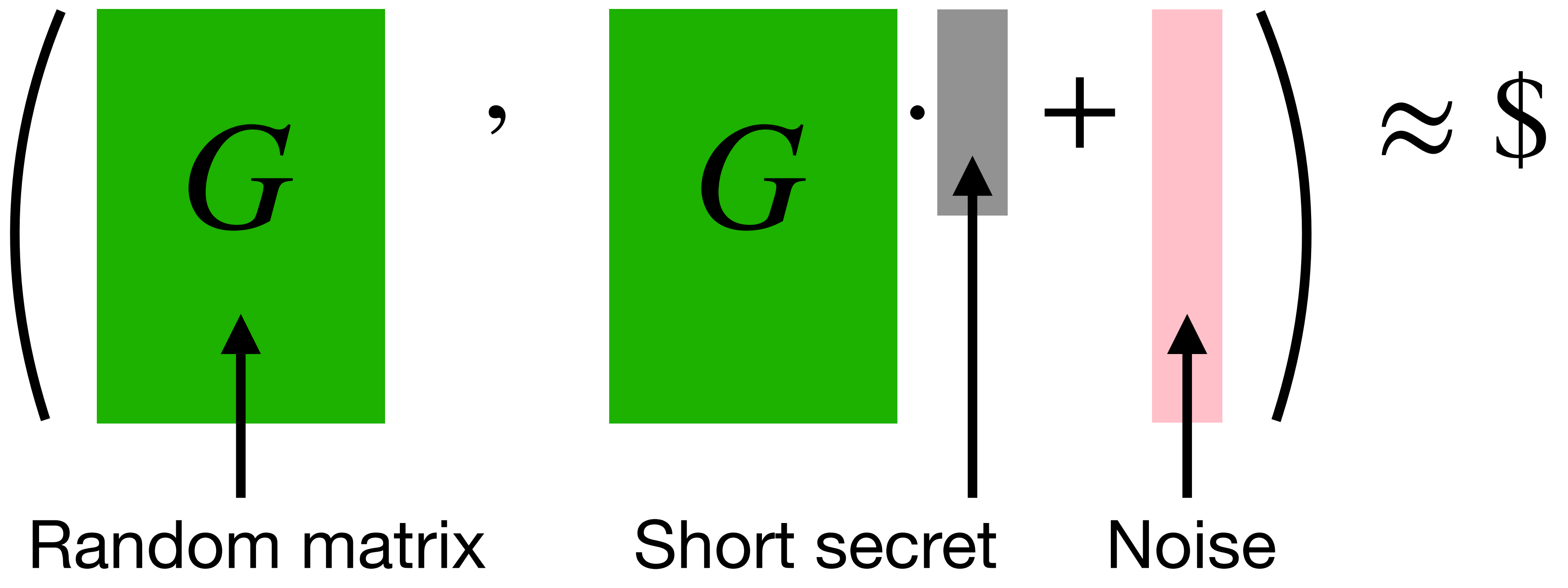
- n clients and m servers with a fingerprint stored.
- Ahead of time, each party publishes an encoding of its fingerprint.
- Later, a client C_i can authenticate to a server S_j by locally computing and sending his share of the Hamming distance, a single element of $\mathbb{F}$.



Computes $\alpha_s = \text{share}_s(\text{HD}(x_2, y_2))$
y_1
y_2
y_3
Check:
$\alpha_c + \alpha_s \leq_? T$
α_c
Y/N
x_1
x_2
x_3
x_4
Computes $\alpha_c = \text{share}_c(\text{HD}(x_2, y_2))$

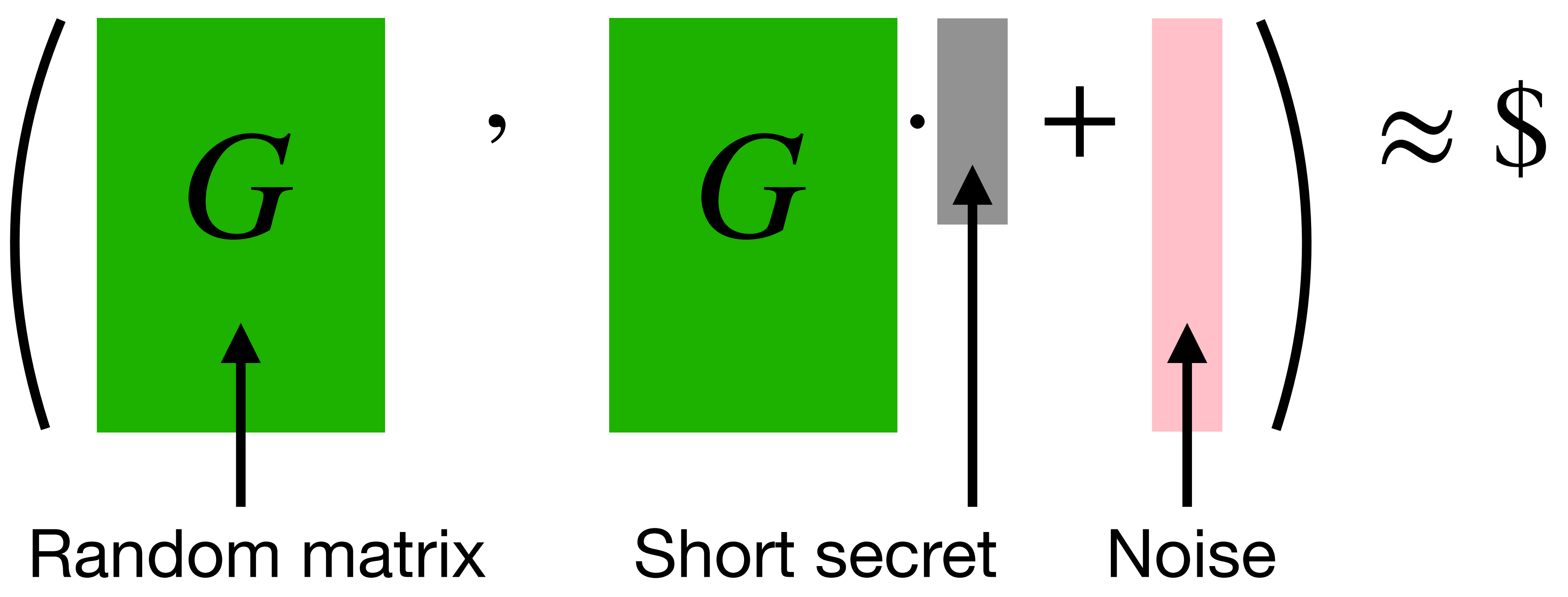
Preliminaries: LPN and LWE

LPN and LWE — Primal Form



Preliminaries: LPN and LWE

LPN and LWE — Primal Form



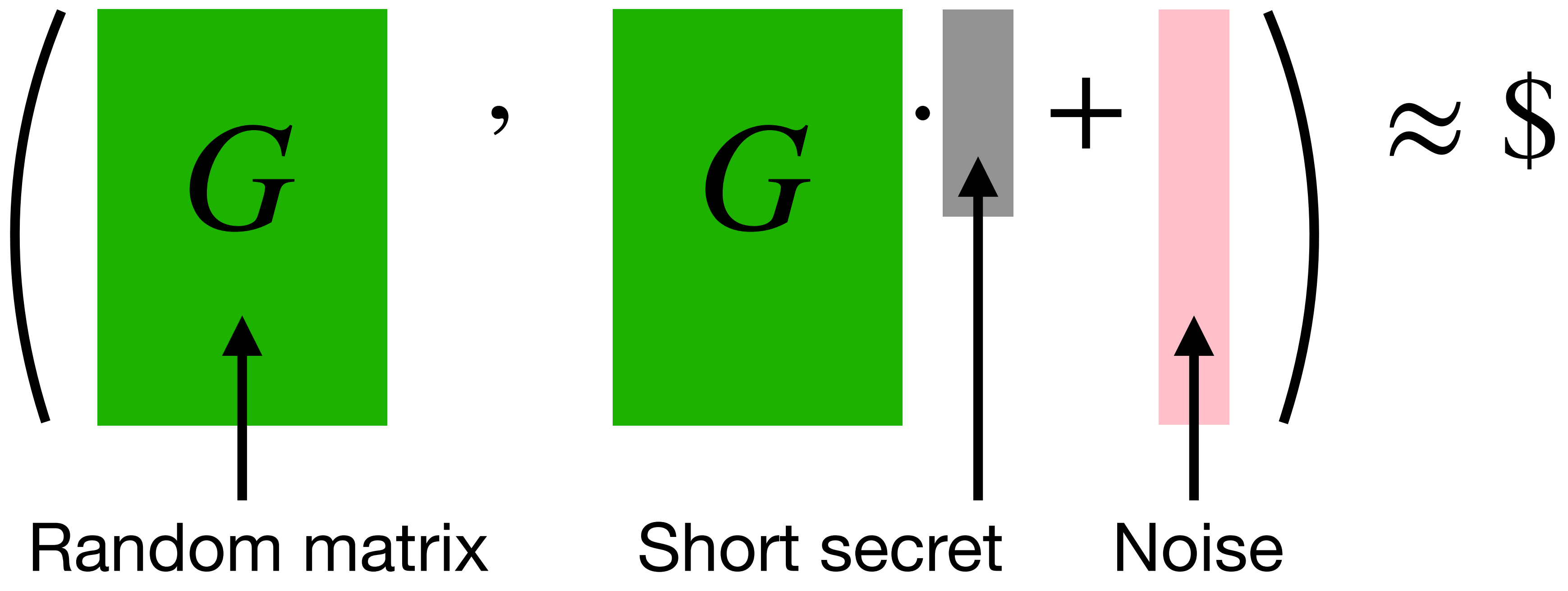
LPN($\mathbb{F}_2$): $G \leftarrow_{\$} \mathbb{F}_2^{m\times n}$, ▮ $\leftarrow_{\$} \mathbb{F}_2^{n}$, ▯ $\leftarrow_{\$}$ Ber($\mathbb{F}_2$)^n

LPN($\mathbb{F}_p$): $G \leftarrow_{\$} \mathbb{F}_p^{m\times n}$, ▮ $\leftarrow_{\$} \mathbb{F}_p^{n}$, ▯ $\leftarrow_{\$}$ Ber($\mathbb{F}_p$)^n

‘Sparse’

Preliminaries: LPN and LWE

LPN and LWE — Primal Form



$\left(G \;,\; G \cdot \text{▨} + \text{🟥} \right) \approx \$$
Random matrix
Short secret
Noise

LPN($\mathbb{F}_2$): $G \leftarrow_\$ \mathbb{F}_2^{m\times n}$, ▨ $\leftarrow_\$ \mathbb{F}_2^n$, 🟥 $\leftarrow_\$$ Ber($\mathbb{F}_2$)^n

LPN($\mathbb{F}_p$): $G \leftarrow_\$ \mathbb{F}_p^{m\times n}$, ▨ $\leftarrow_\$ \mathbb{F}_p^n$, 🟥 $\leftarrow_\$$ Ber($\mathbb{F}_p$)^n

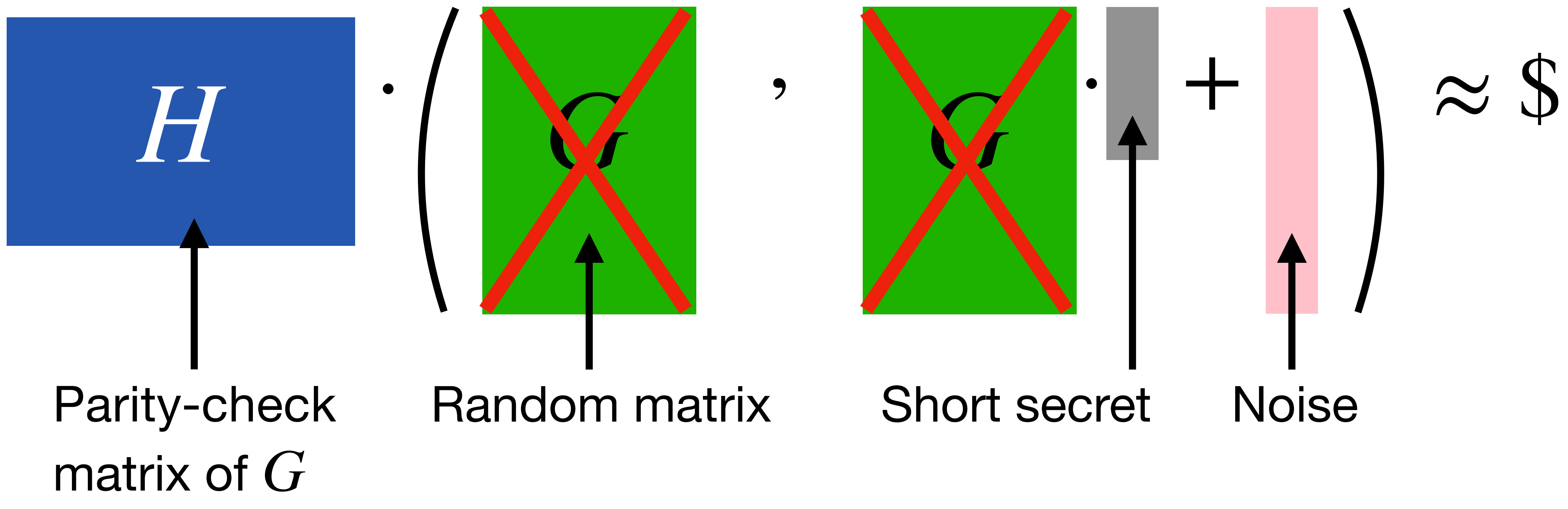
'Sparse'

LWE($\mathbb{F}_p$): $G \leftarrow_\$ \mathbb{F}_p^{m\times n}$, ▨ $\leftarrow_\$ \mathbb{F}_p^n$, 🟥 $\leftarrow_\$ [-B, B]^n$

'Small'

Preliminaries: LPN and LWE

LPN and LWE — Dual Form



$H \cdot \left(\cancel{G} \;,\; \cancel{G} \cdot \blacksquare + \square \right) \approx \$$

Parity-check matrix of G

Random matrix

Short secret

Noise

LPN($\mathbb{F}_2$): 🟩G $\leftarrow_\$ \mathbb{F}_2^{m\times n}$, ▨ $\leftarrow_\$ \mathbb{F}_2^{n}$, 🟥 $\leftarrow_\$$ Ber($\mathbb{F}_2$)^n

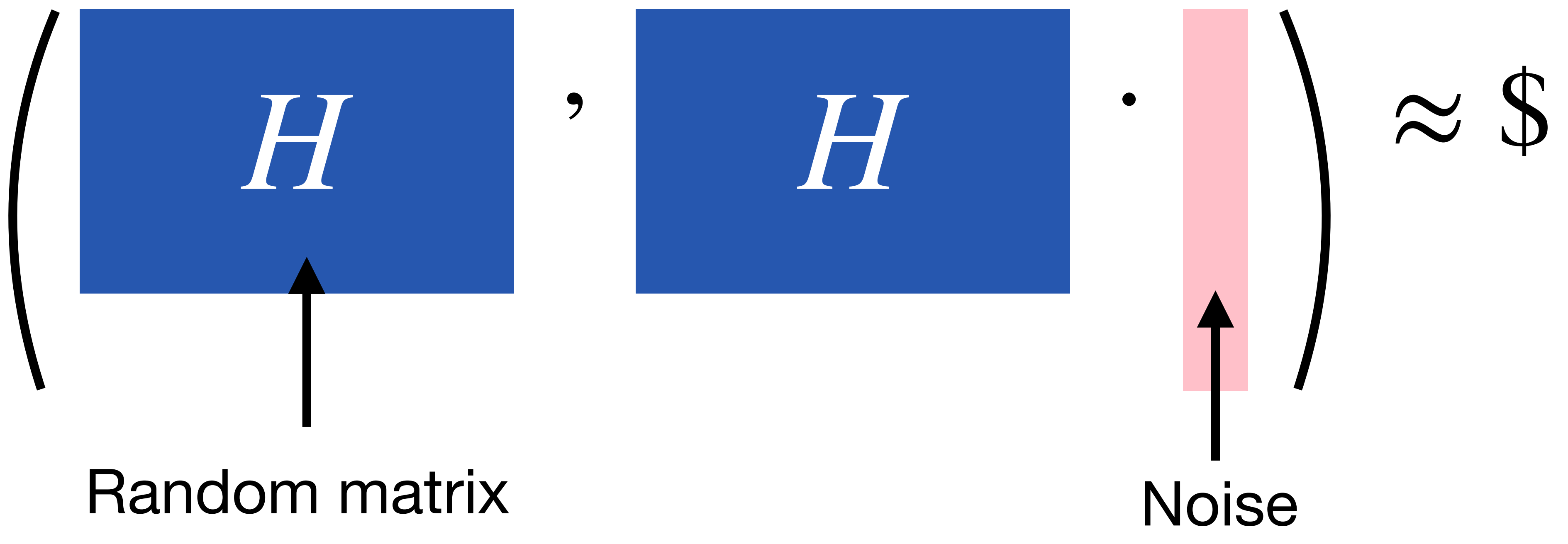
LPN($\mathbb{F}_p$): 🟩G $\leftarrow_\$ \mathbb{F}_p^{m\times n}$, ▨ $\leftarrow_\$ \mathbb{F}_p^{n}$, 🟥 $\leftarrow_\$$ Ber($\mathbb{F}_p$)^n

'Sparse'

LWE($\mathbb{F}_p$): 🟩G $\leftarrow_\$ \mathbb{F}_p^{m\times n}$, ▨ $\leftarrow_\$ \mathbb{F}_p^{n}$, 🟥 $\leftarrow_\$ [-B, B]^n$ ← *'Small'*

Preliminaries: LPN and LWE

LPN and LWE — Dual Form



LPN($\mathbb{F}_2$): $\boxed{H} \leftarrow_{\$} \mathbb{F}_2^{m\times n}$, ▮ $\leftarrow_{\$}$ Ber$(\mathbb{F}_2)^n$

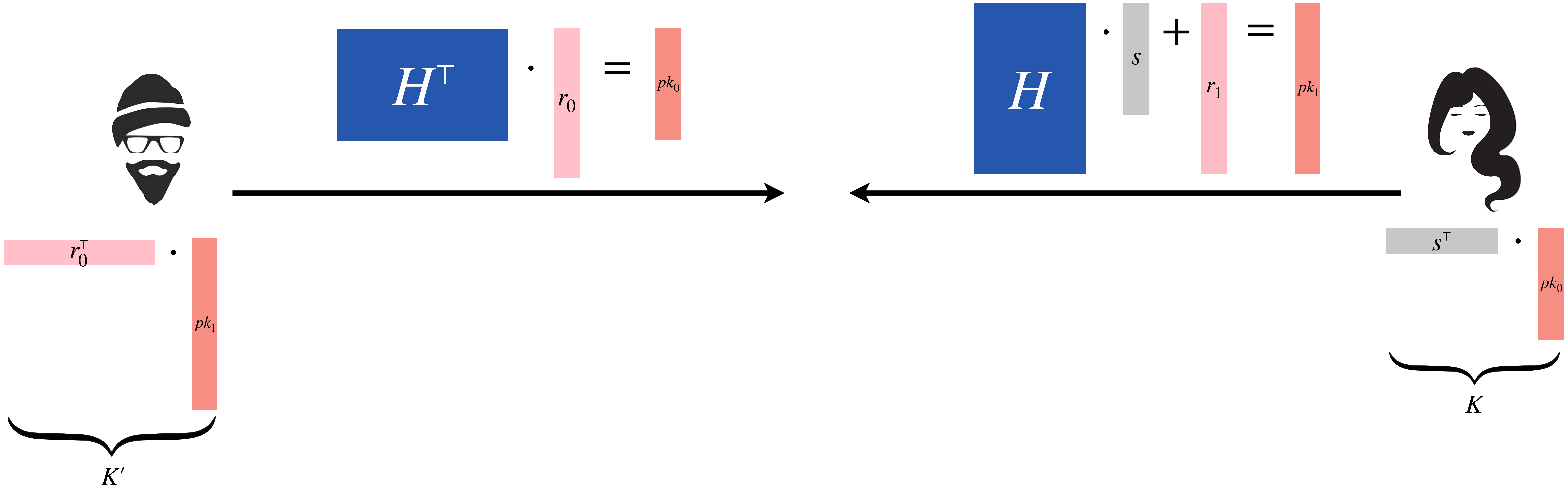
LPN($\mathbb{F}_p$): $\boxed{H} \leftarrow_{\$} \mathbb{F}_p^{m\times n}$, ▮ $\leftarrow_{\$}$ Ber$(\mathbb{F}_p)^n$

LWE($\mathbb{F}_p$): $\boxed{H} \leftarrow_{\$} \mathbb{F}_p^{m\times n}$, ▮ $\leftarrow_{\$} [-B, B]^n$

‘Sparse’

‘Small’

Alekhnovich Key Exchange



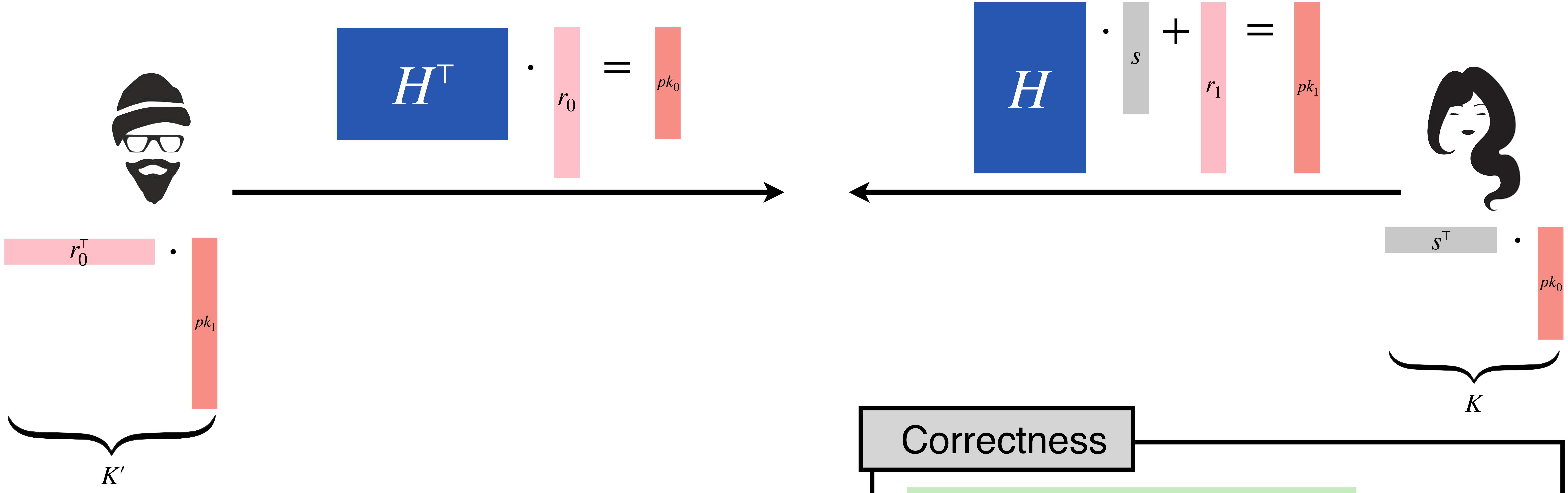
$$H^\top \cdot r_0 = pk_0$$

$$H \cdot s + r_1 = pk_1$$

$$\underbrace{r_0^\top \cdot pk_1}_{K'}$$

$$\underbrace{s^\top \cdot pk_0}_{K}$$

Alekhnovich Key Exchange



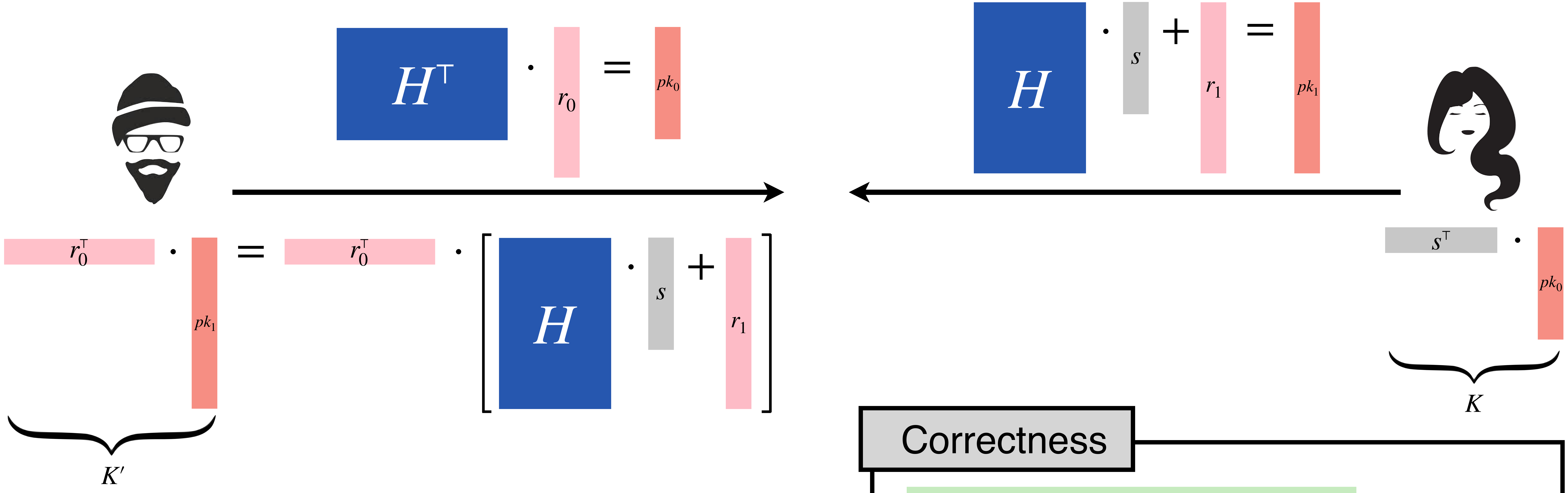
Correctness

Claim: $\Pr[K = K'] \approx t^2/n \ll 1$

$$K' = r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1$$
$$= \mathsf{pk}_0^\top \cdot s + r_0^\top \cdot r_1 = K + e$$

Where $\Pr[e = 1] \approx t^2/n \ll 1$

Alekhnovich Key Exchange


$H^\top \cdot r_0 = pk_0$
$H \cdot s + r_1 = pk_1$
$r_0^\top \cdot pk_1 = r_0^\top \cdot [H \cdot s + r_1]$
K'
$s^\top \cdot pk_0$
K

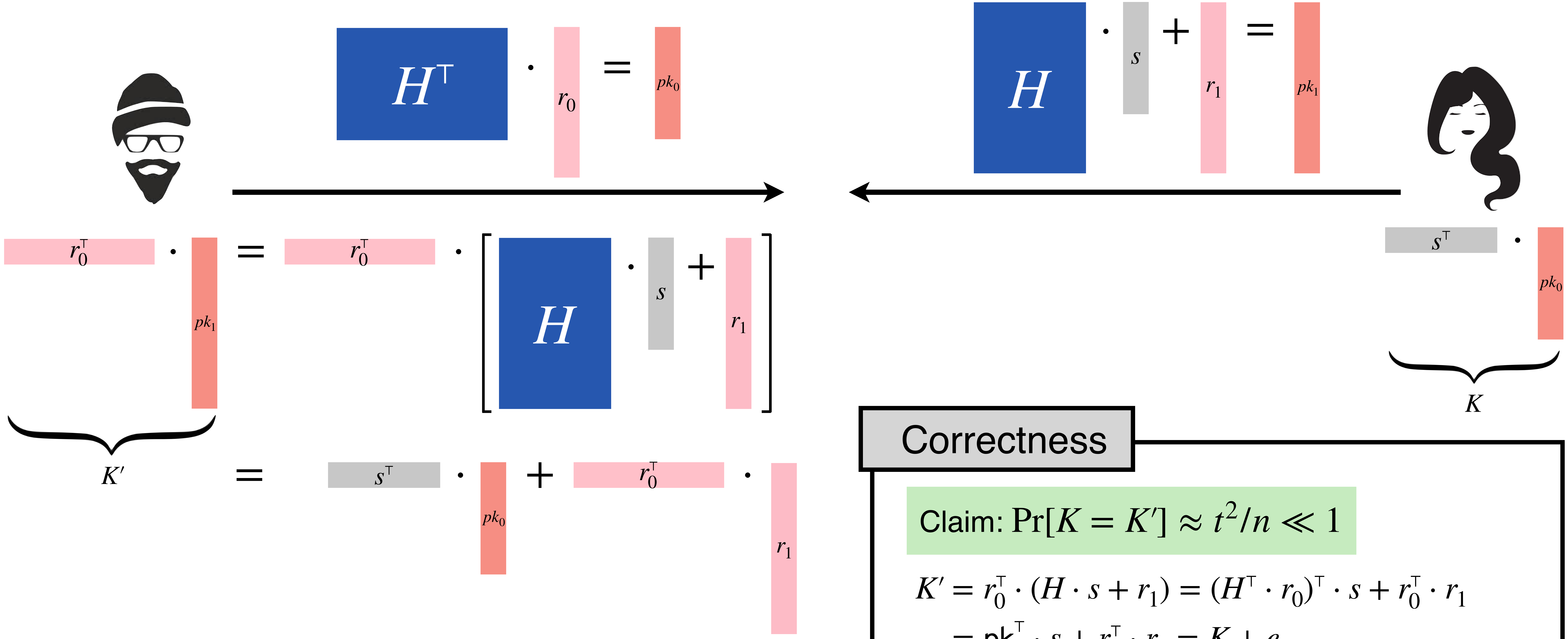
Correctness

Claim: $\Pr[K = K'] \approx t^2/n \ll 1$

$$\begin{aligned} K' &= r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1 \\ &= \mathrm{pk}_0^\top \cdot s + r_0^\top \cdot r_1 = K + e \end{aligned}$$

Where $\Pr[e = 1] \approx t^2/n \ll 1$

Alekhnovich Key Exchange


$H^\top \cdot r_0 = pk_0$
$H \cdot s + r_1 = pk_1$
$\underbrace{r_0^\top \cdot pk_1}_{K'} = r_0^\top \cdot [H \cdot s + r_1]$
$= s^\top \cdot pk_0 + r_0^\top \cdot r_1$
$\underbrace{s^\top \cdot pk_0}_{K}$

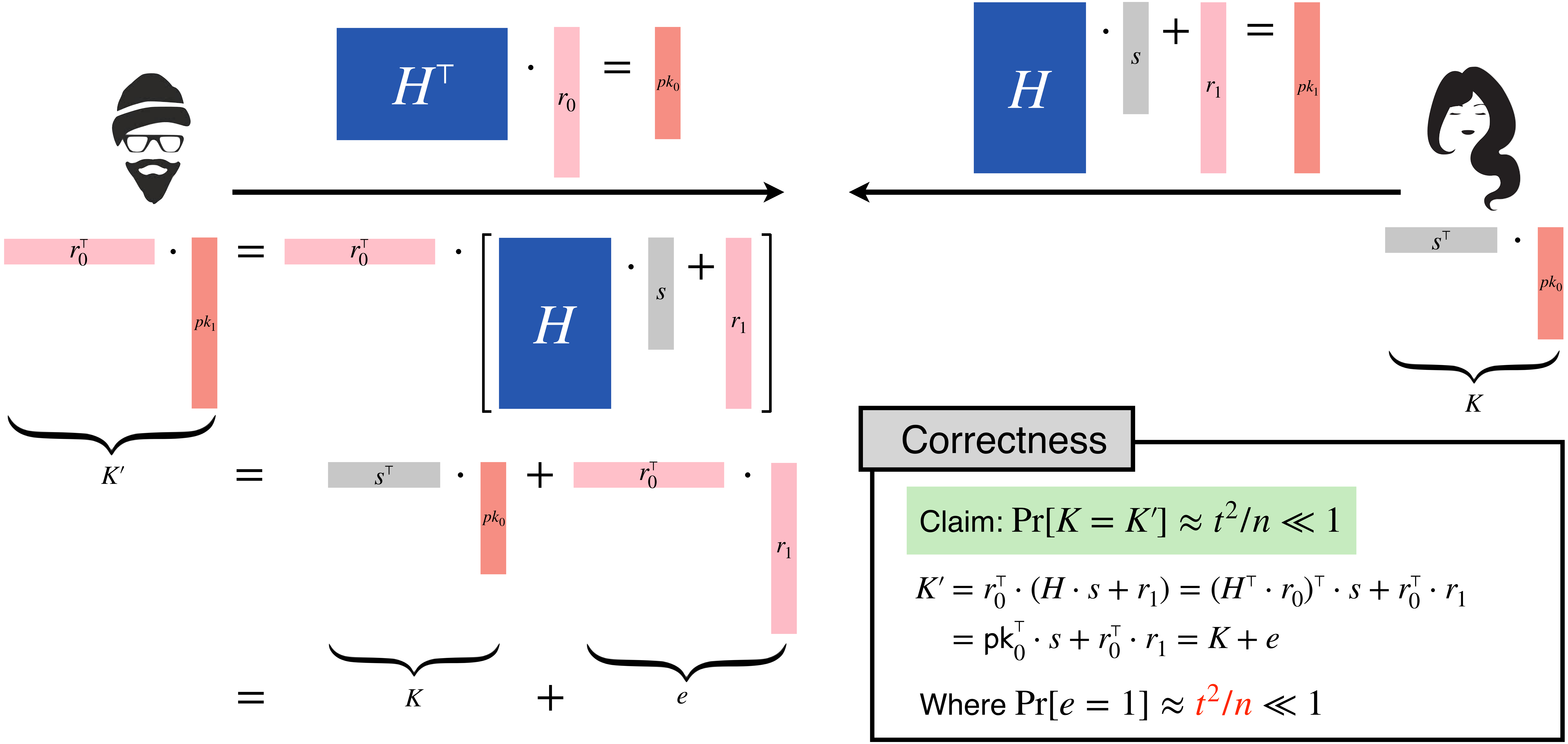
Correctness

Claim: $\Pr[K = K'] \approx t^2/n \ll 1$

$$K' = r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1$$
$$= \mathrm{pk}_0^\top \cdot s + r_0^\top \cdot r_1 = K + e$$

Where $\Pr[e = 1] \approx t^2/n \ll 1$

Alekhnovich Key Exchange



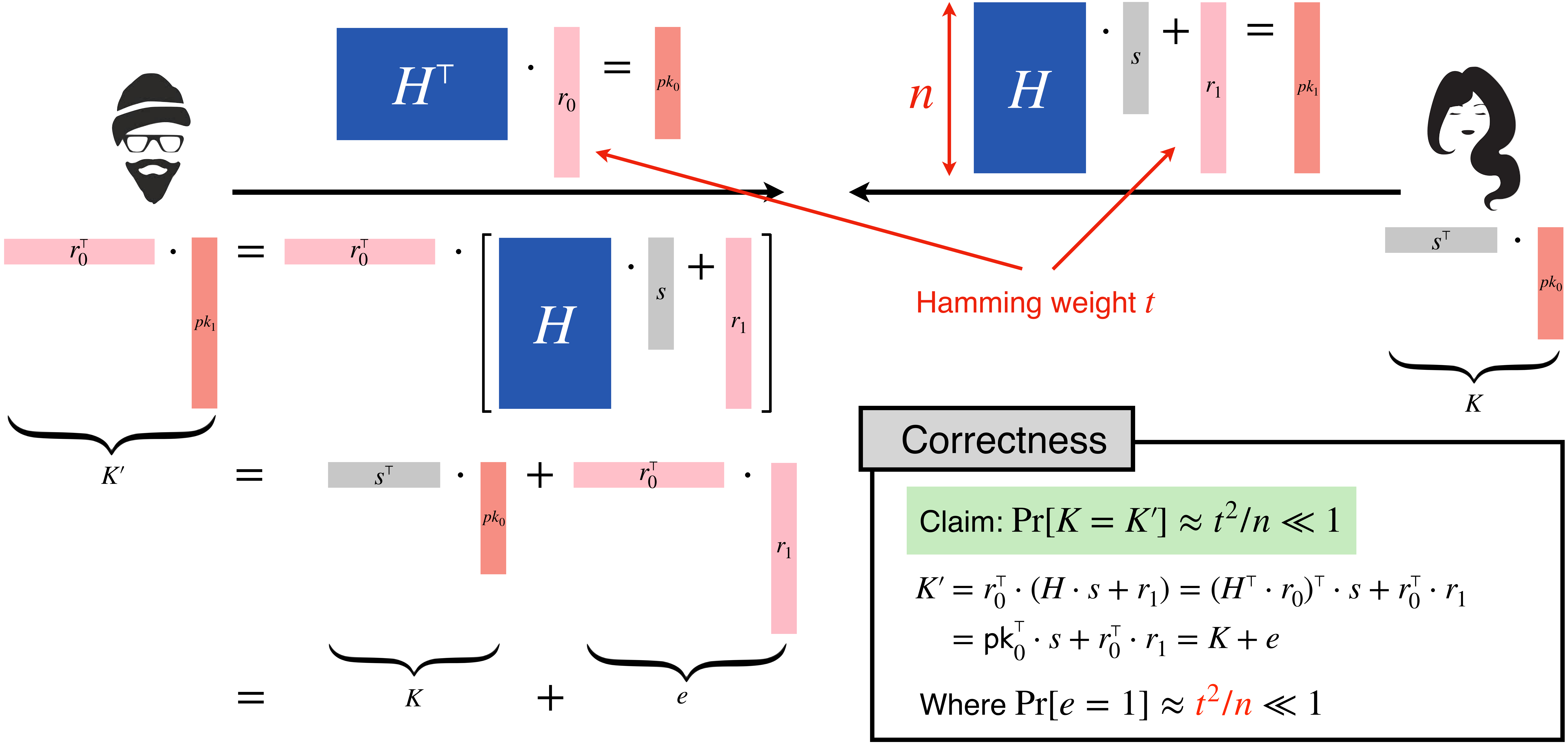
Correctness

Claim: $\Pr[K = K'] \approx t^2/n \ll 1$

$$K' = r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1$$
$$= \mathrm{pk}_0^\top \cdot s + r_0^\top \cdot r_1 = K + e$$

Where $\Pr[e = 1] \approx t^2/n \ll 1$

Alekhnovich Key Exchange



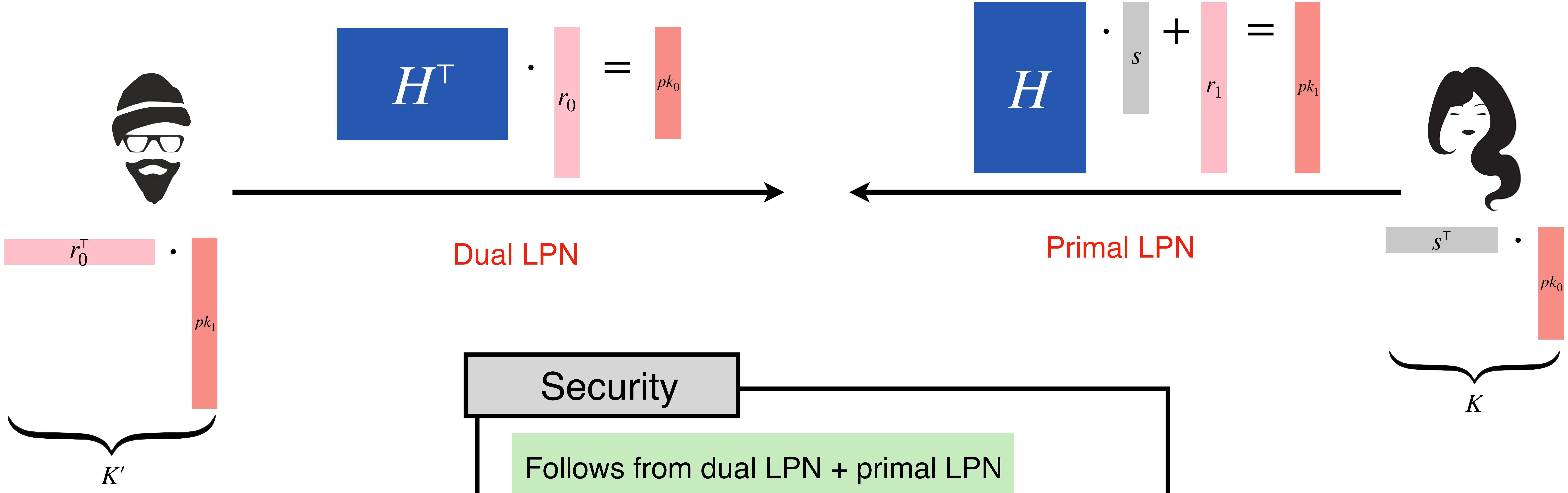
Correctness

Claim: $\Pr[K = K'] \approx t^2/n \ll 1$

$$K' = r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1$$
$$= \mathrm{pk}_0^\top \cdot s + r_0^\top \cdot r_1 = K + e$$

Where $\Pr[e = 1] \approx t^2/n \ll 1$

Alekhnovich Key Exchange



$H^{\top} \cdot r_0 = pk_0$

$H \cdot s + r_1 = pk_1$

Dual LPN

Primal LPN

$\underbrace{r_0^{\top} \cdot pk_1}_{K'}$

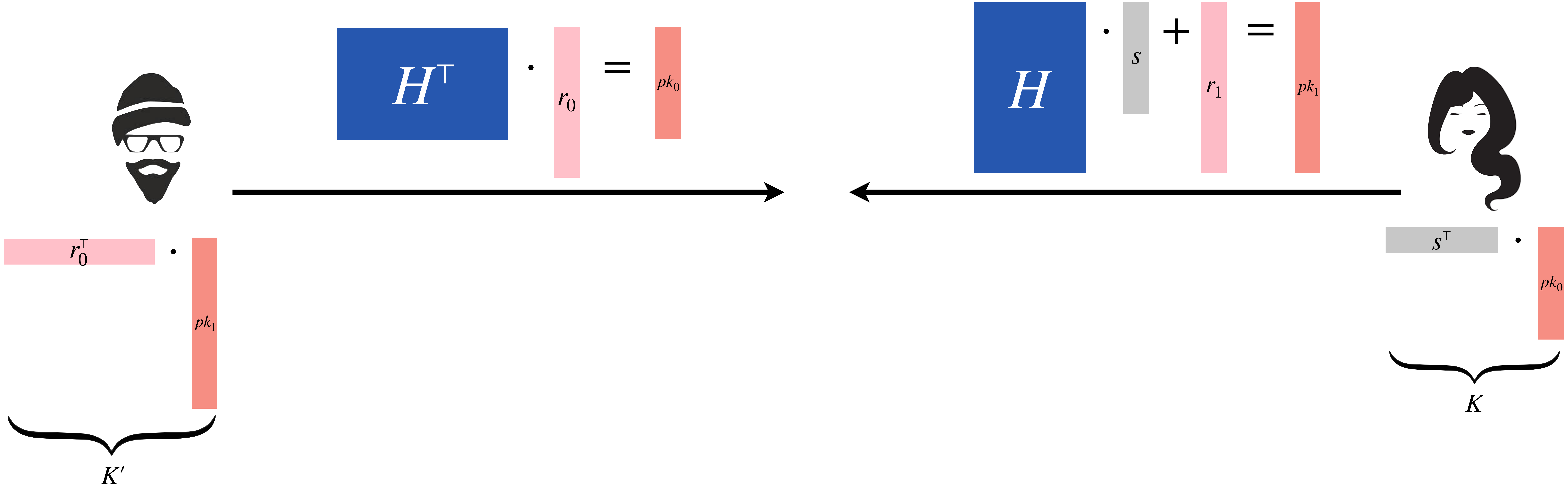
$\underbrace{s^{\top} \cdot pk_0}_{K}$

Security

Follows from dual LPN + primal LPN

(The proof is standard)

Embedding an Inner Product in Alekhnovich’s Key Exchange



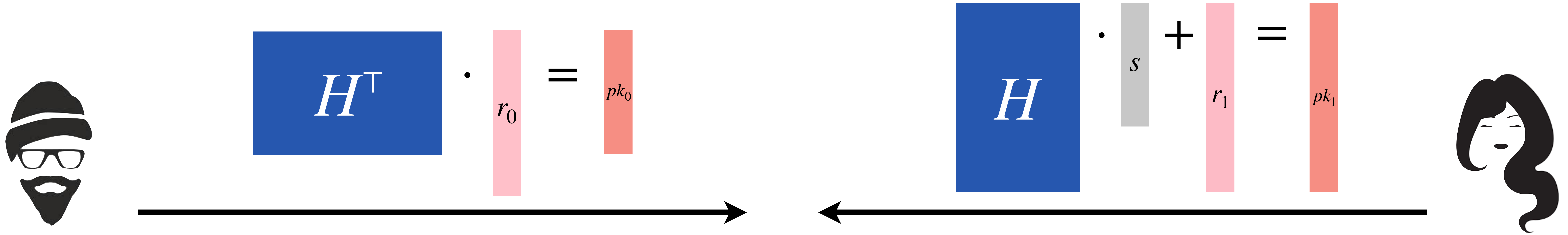
$H^{\top} \cdot r_0 = pk_0$

$H \cdot s + r_1 = pk_1$

$\underbrace{r_0^{\top} \cdot pk_1}_{K'}$

$\underbrace{s^{\top} \cdot pk_0}_{K}$

Embedding an Inner Product in Alekhnovich’s Key Exchange

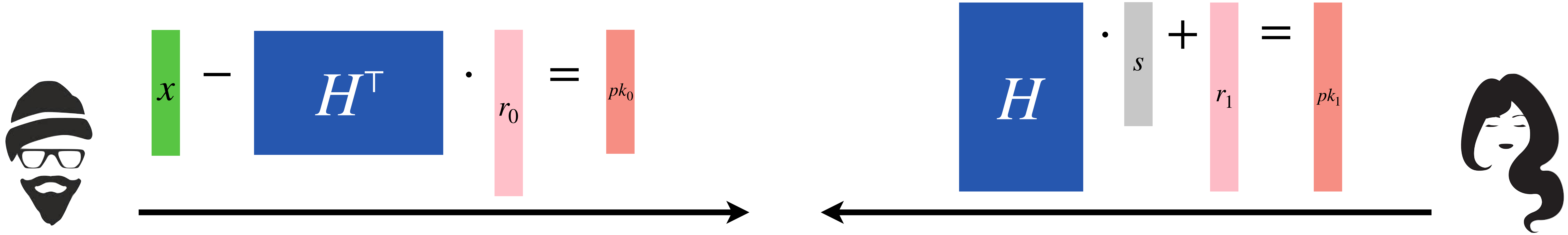


Since the parties are computing an inner product to get K, could we *embed* an inner product computation?

$$\underbrace{r_0^\top \cdot pk_1}_{K'}$$

$$\underbrace{s^\top \cdot pk_0}_{K}$$

Embedding an Inner Product in Alekhnovich’s Key Exchange



Since the parties are computing an inner product to get K, could we *embed* an inner product computation?

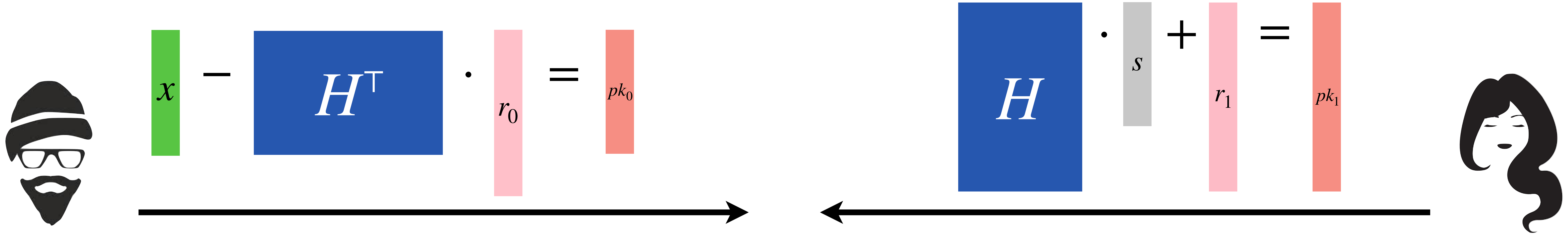
Input: x

Warmup attempt: only Bob has an input

Bob computes:

Alice computes:

Embedding an Inner Product in Alekhnovich's Key Exchange



Since the parties are computing an inner product to get K, could we *embed* an inner product computation?

Input: x

Warmup attempt: only Bob has an input

Bob computes:

Alice computes:

$$\underbrace{s^\top \cdot pk_0}_{K}$$

Embedding an Inner Product in Alekhnovich’s Key Exchange

$$x - H^\top \cdot r_0 = pk_0 \longrightarrow$$

$$\longleftarrow H \cdot s + r_1 = pk_1$$

Since the parties are computing an inner product to get K, could we *embed* an inner product computation?

Input: x

Warmup attempt: only Bob has an input

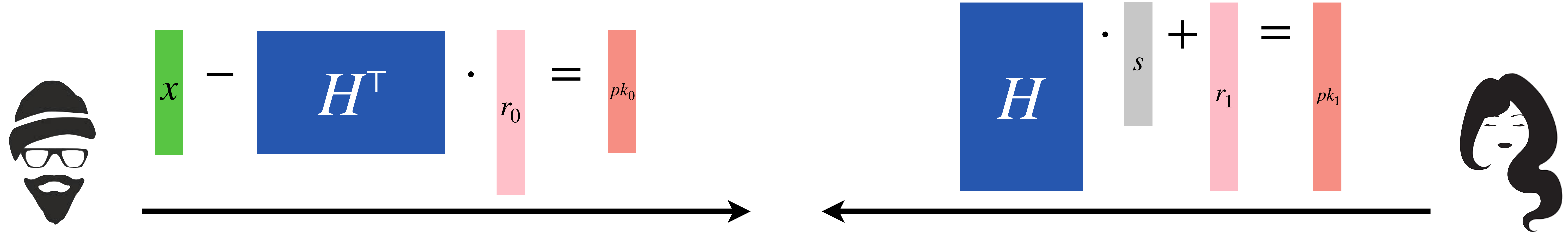
Bob computes:

$$\underbrace{r_0^\top \cdot pk_1}_{K'} = - \underbrace{s^\top \cdot pk_0}_{-K} + s^\top \cdot x + e$$

Alice computes:

$$\underbrace{s^\top \cdot pk_0}_{K}$$

Embedding an Inner Product in Alekhnovich’s Key Exchange



$$x - H^\top \cdot r_0 = pk_0$$

$$H \cdot s + r_1 = pk_1$$

Since the parties are computing an inner product to get K, could we *embed* an inner product computation?

Input: x

Warmup attempt: only Bob has an input

Bob computes:

$$\underbrace{r_0^\top \cdot pk_1}_{K'} = -\underbrace{s^\top \cdot pk_0}_{-K} + s^\top \cdot x + e$$

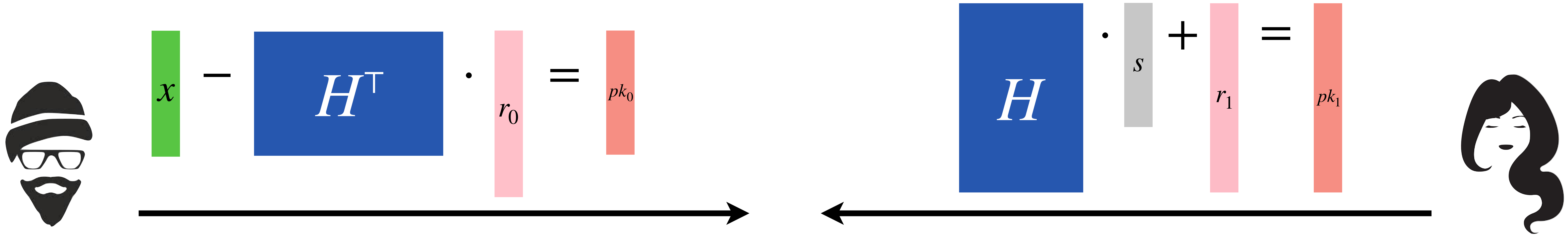
Alice computes:

$$\underbrace{s^\top \cdot pk_0}_{K}$$

K and K' form (noisy) additive share of $\langle x, s \rangle$

We are making progress — but s has to be random for primal LPN to hold!

Embedding an Inner Product in Alekhnovich's Key Exchange

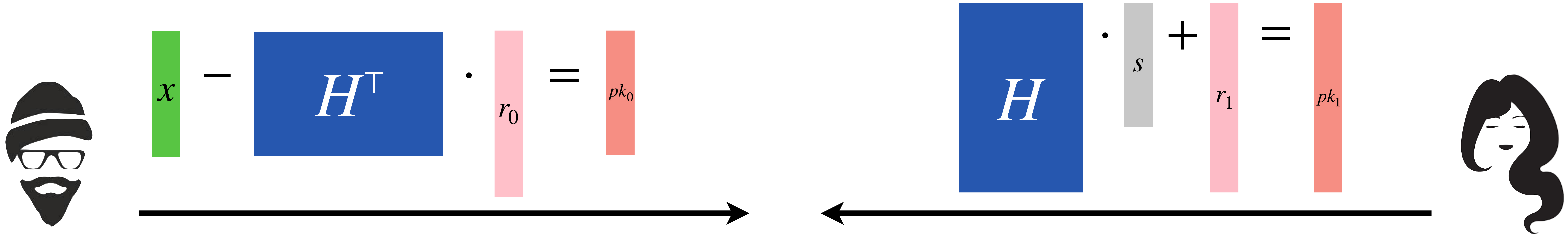


$$x - H^{\top} \cdot r_0 = pk_0$$

$$H \cdot s + r_1 = pk_1$$

How to embed Alice's input in s?

Embedding an Inner Product in Alekhnovich’s Key Exchange



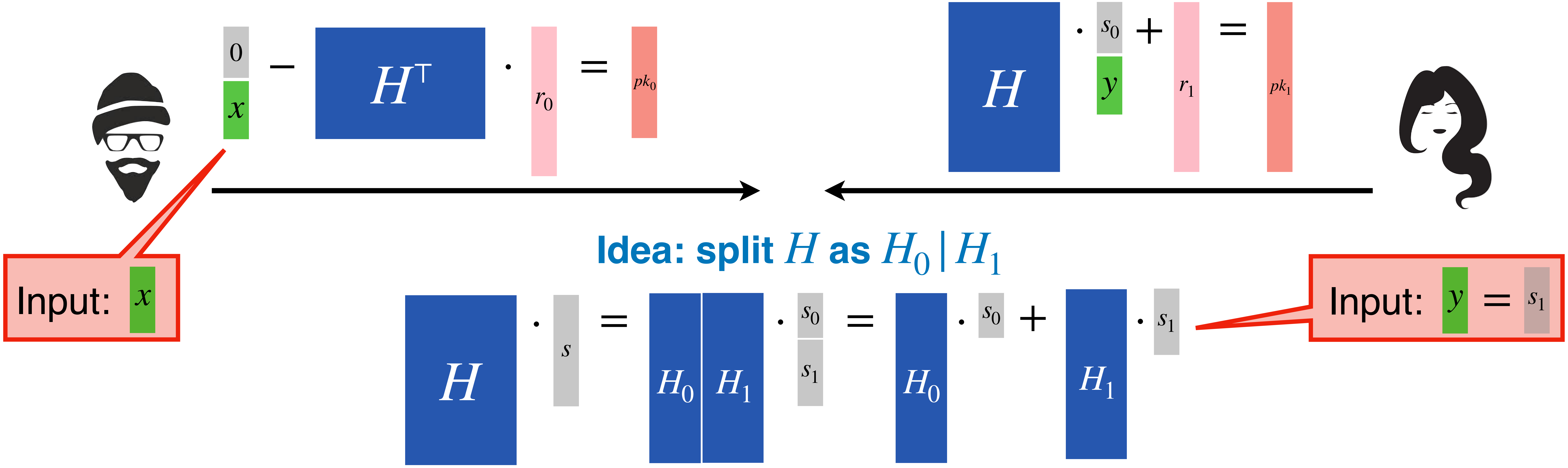
$$x - H^\top \cdot r_0 = pk_0$$

$$H \cdot s + r_1 = pk_1$$

Idea: split H as $H_0 \,|\, H_1$

$$H \cdot s = \begin{bmatrix} H_0 & H_1 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} = H_0 \cdot s_0 + H_1 \cdot s_1$$

Embedding an Inner Product in Alekhnovich's Key Exchange



$\begin{pmatrix}0\\x\end{pmatrix} - H^\top \cdot r_0 = pk_0$

$H \cdot \begin{pmatrix}s_0\\y\end{pmatrix} + r_1 = pk_1$

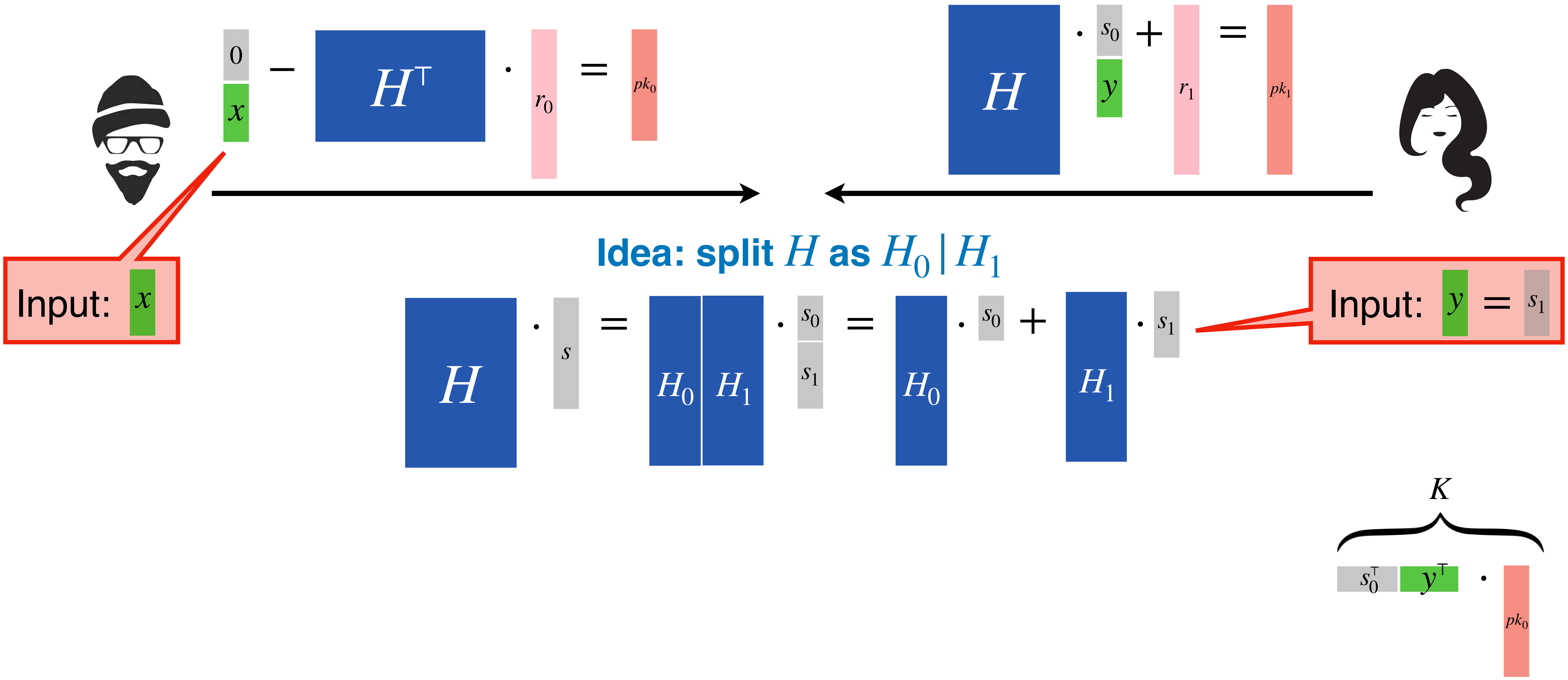
Input: x

Input: $y = s_1$

Idea: split H as $H_0 \,|\, H_1$

$$H \cdot s = \left(H_0 \,\middle|\, H_1\right) \cdot \begin{pmatrix}s_0\\s_1\end{pmatrix} = H_0 \cdot s_0 + H_1 \cdot s_1$$

Embedding an Inner Product in Alekhnovich's Key Exchange



$\begin{pmatrix}0\\x\end{pmatrix} - H^\top \cdot r_0 = pk_0$

$H \cdot \begin{pmatrix}s_0\\y\end{pmatrix} + r_1 = pk_1$

Input: x

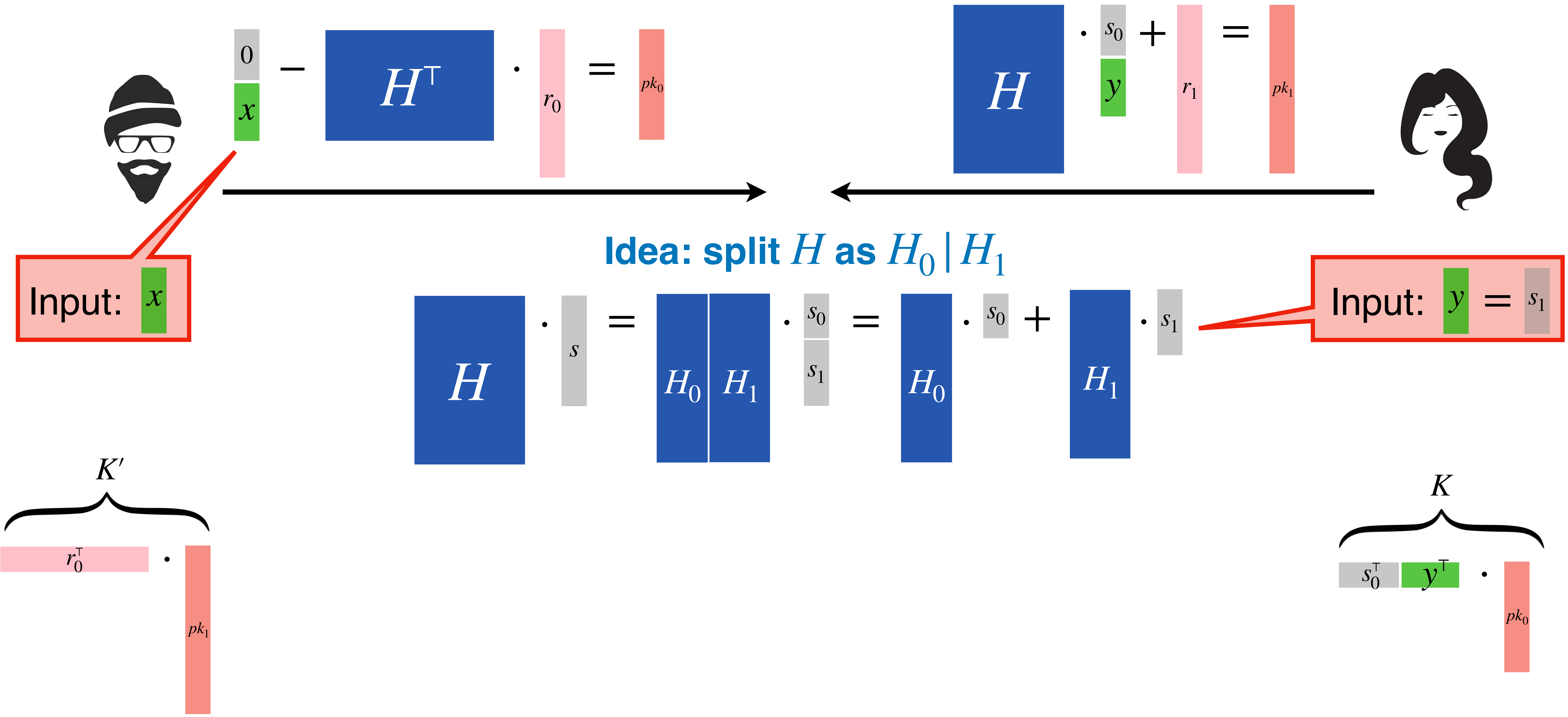
Idea: split H as $H_0 \mid H_1$

$H \cdot s = \begin{pmatrix}H_0 & H_1\end{pmatrix} \cdot \begin{pmatrix}s_0\\s_1\end{pmatrix} = H_0 \cdot s_0 + H_1 \cdot s_1$

Input: $y = s_1$

$\overbrace{\begin{pmatrix}s_0^\top & y^\top\end{pmatrix} \cdot pk_0}^{K}$

Embedding an Inner Product in Alekhnovich's Key Exchange



$$\begin{bmatrix}0\\x\end{bmatrix} - H^{\top} \cdot r_0 = pk_0$$

$$H \cdot \begin{bmatrix}s_0\\y\end{bmatrix} + r_1 = pk_1$$

Input: x

Idea: split H as $H_0 \mid H_1$

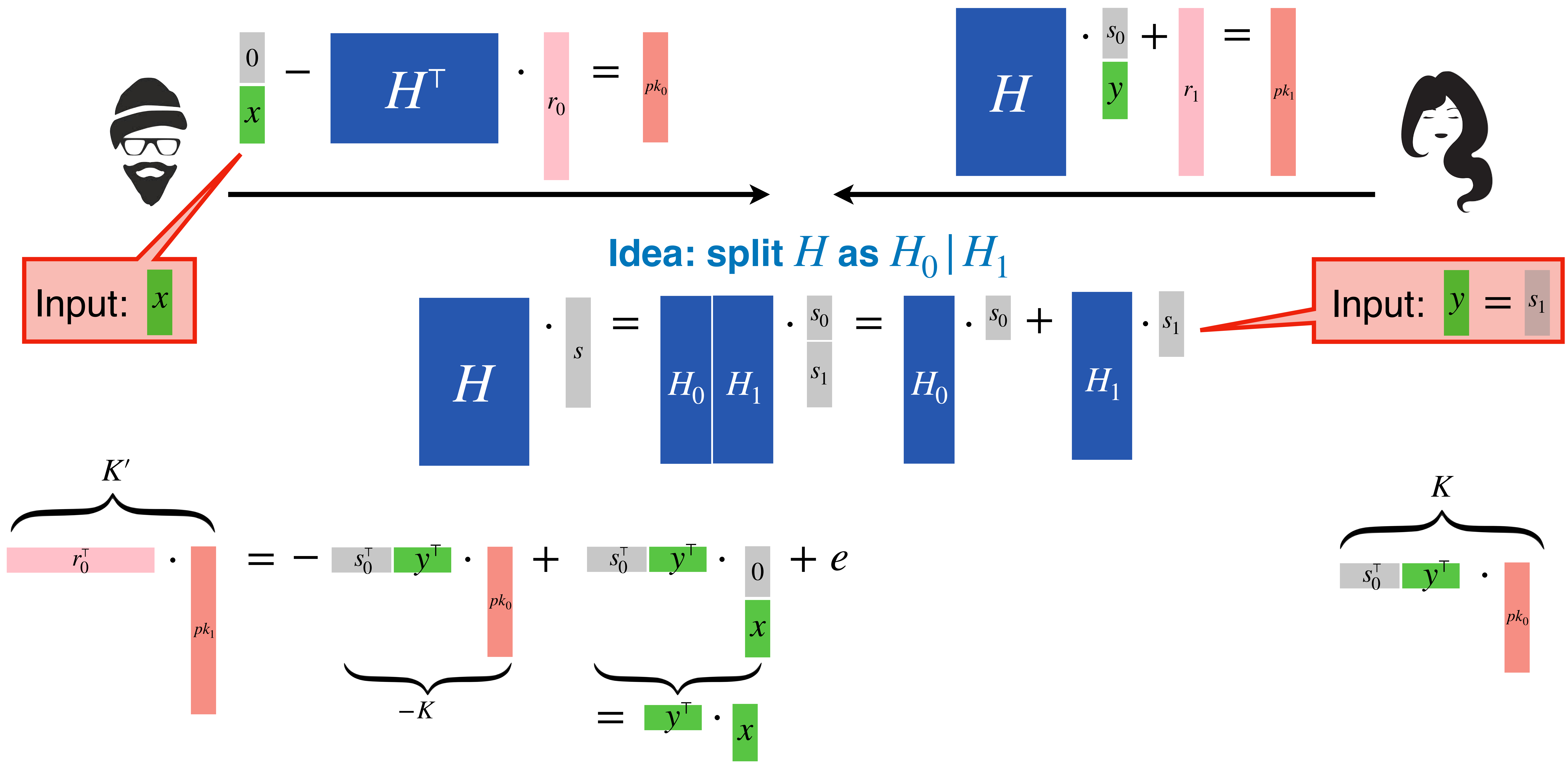
$$H \cdot s = \begin{bmatrix}H_0 & H_1\end{bmatrix} \cdot \begin{bmatrix}s_0\\s_1\end{bmatrix} = H_0 \cdot s_0 + H_1 \cdot s_1$$

Input: $y = s_1$

$$\overbrace{r_0^{\top} \cdot pk_1}^{K'}$$

$$\overbrace{\begin{bmatrix}s_0^{\top} & y^{\top}\end{bmatrix} \cdot pk_0}^{K}$$

Embedding an Inner Product in Alekhnovich's Key Exchange



$$\begin{pmatrix}0\\ x\end{pmatrix} - H^\top \cdot r_0 = pk_0$$

$$H \cdot \begin{pmatrix}s_0\\ y\end{pmatrix} + r_1 = pk_1$$

Input: x

Idea: split H as $H_0 \,|\, H_1$

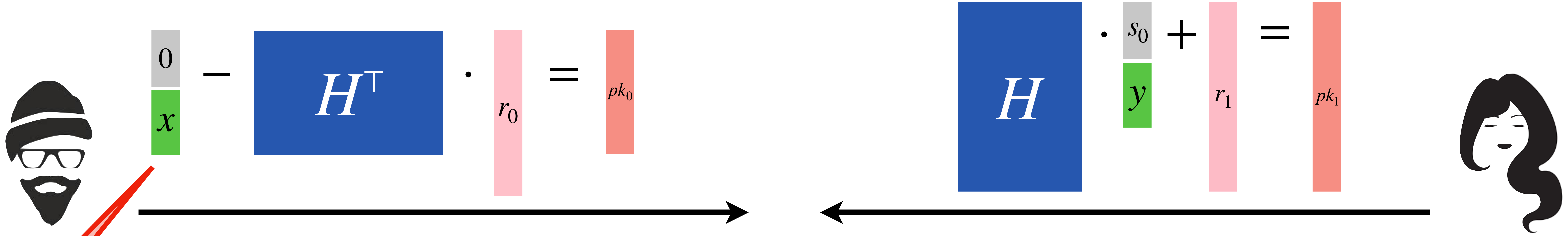
$$H \cdot s = \begin{pmatrix}H_0 & H_1\end{pmatrix} \cdot \begin{pmatrix}s_0\\ s_1\end{pmatrix} = H_0 \cdot s_0 + H_1 \cdot s_1$$

Input: $y = s_1$

$$\overbrace{r_0^\top \cdot pk_1}^{K'} = -\underbrace{\begin{pmatrix}s_0^\top & y^\top\end{pmatrix} \cdot pk_0}_{-K} + \underbrace{\begin{pmatrix}s_0^\top & y^\top\end{pmatrix} \cdot \begin{pmatrix}0\\ x\end{pmatrix}}_{= y^\top \cdot x} + e$$

$$\overbrace{\begin{pmatrix}s_0^\top & y^\top\end{pmatrix} \cdot pk_0}^{K}$$

Embedding an Inner Product in Alekhnovich’s Key Exchange



$$\begin{pmatrix}0\\x\end{pmatrix} - H^\top \cdot r_0 = pk_0$$

$$H \cdot \begin{pmatrix}s_0\\y\end{pmatrix} + r_1 = pk_1$$

Input: x

Input: $y = s_1$

Idea: split H as $H_0 \,|\, H_1$

$$H \cdot s = \left(H_0 \,|\, H_1\right) \cdot \begin{pmatrix}s_0\\s_1\end{pmatrix} = H_0 \cdot s_0 + H_1 \cdot s_1$$

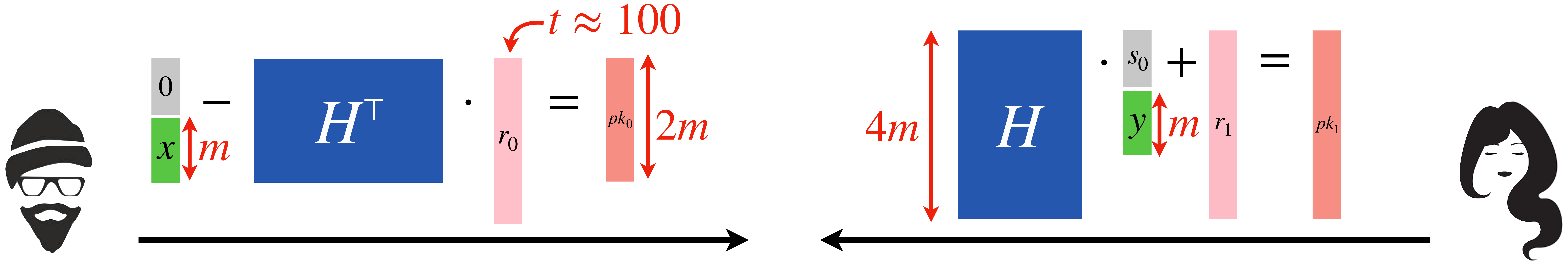
$$\overbrace{r_0^\top \cdot pk_1}^{K'} = -\underbrace{\left(s_0^\top \; y^\top\right) \cdot pk_0}_{-K} + \underbrace{\left(s_0^\top \; y^\top\right) \cdot \begin{pmatrix}0\\x\end{pmatrix}}_{= y^\top \cdot x} + e$$

$$\overbrace{\left(s_0^\top \; y^\top\right) \cdot pk_0}^{K}$$

Security

Use primal LPN with matrix H_0 for Alice, and dual LPN with matrix $H^\top$ for Bob

Embedding an Inner Product in Alekhnovich’s Key Exchange


$t \approx 100$
$\begin{pmatrix} 0 \\ x \end{pmatrix} - H^\top \cdot r_0 = pk_0$
m
$2m$
$4m$
$H \cdot \begin{pmatrix} s_0 \\ y \end{pmatrix} + r_1 = pk_1$
m

Efficiency

Communication: $2m$ from Bob and $4m$ from Alice with reasonable parameters ($m = |x| = |y|$), $(1 + \varepsilon)m$ from each party asymptotically (which is optimal)

Computation: cost dominated by $v \to H \cdot v$ ($\iff$ $v \to H^\top \cdot v$), can be $O(m \cdot \log m)$ (LPN with quasi-cyclic codes, standard) or even $O(m)$ (Druk-Ishai codes, slightly more exotic)

$$\overbrace{r_0^\top \cdot pk_1}^{K'} = -\underbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}_{-K} + \underbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot \begin{pmatrix} 0 \\ x \end{pmatrix}}_{= y^\top \cdot x} + e$$

$$\overbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}^{K}$$

Security

Use primal LPN with matrix H_0 for Alice, and dual LPN with matrix $H^\top$ for Bob

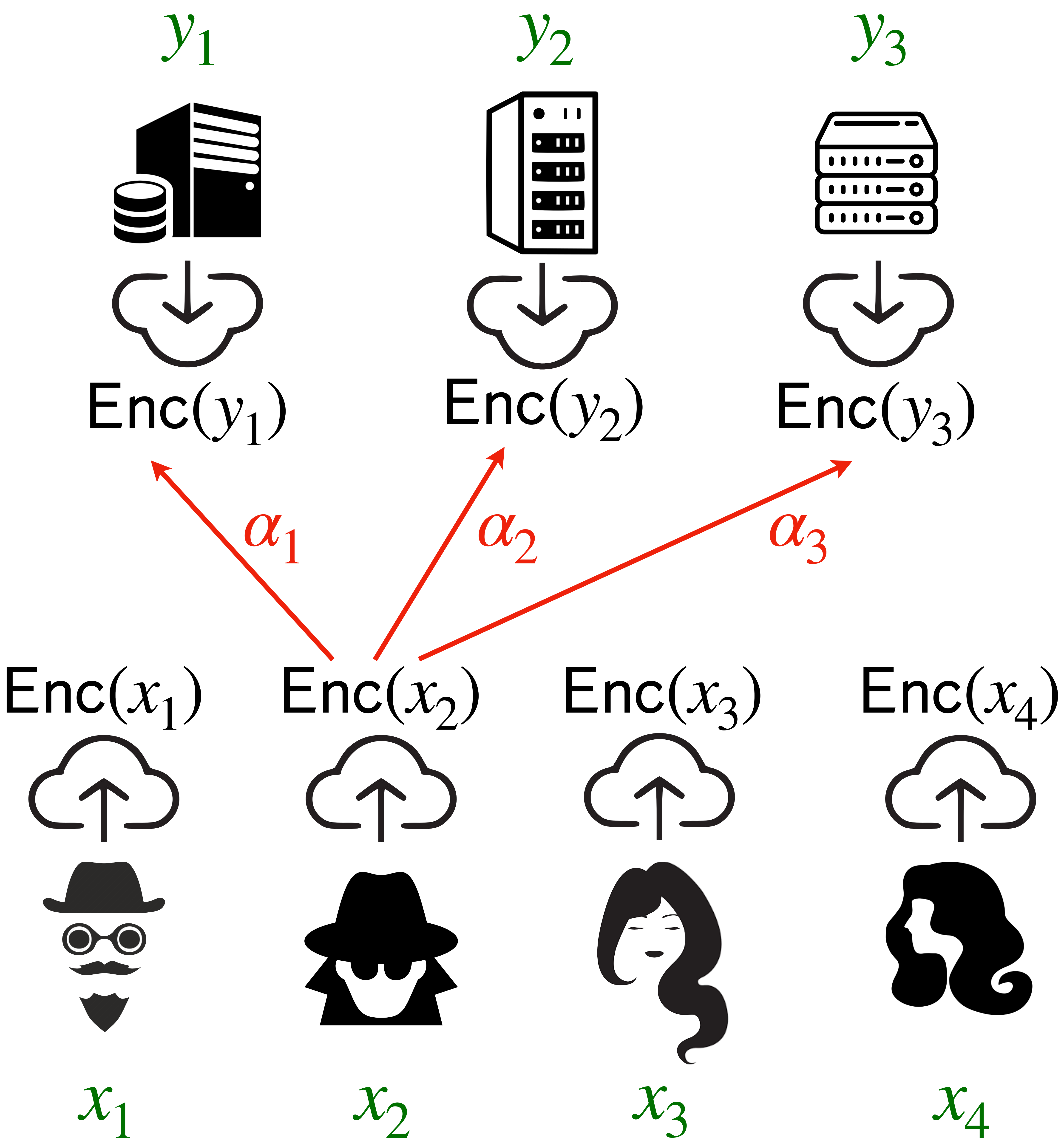
Multiparty Inner Product with Leakage

The protocol has t^2/n correctness error.

⚠ In MPC, correctness errors translate to *leakage* when a ‘detectable’ error occurs: the server learns an equation $\langle v, r\rangle$ with v known and r the noise vector.

$\Longrightarrow$ leaks $\approx N \cdot t^2/n$ linear equations in r if the client interacts with N servers.

$\Longrightarrow$ still secure under the **LPN with leakage** assumption (equivalent to standard LPN, but with a loss)



Multiparty Inner Product with Leakage

The protocol has t^2/n correctness error.

⚠ In MPC, correctness errors translate to *leakage* when a 'detectable' error occurs: the server learns an equation $\langle v, r \rangle$ with v known and r the noise vector.

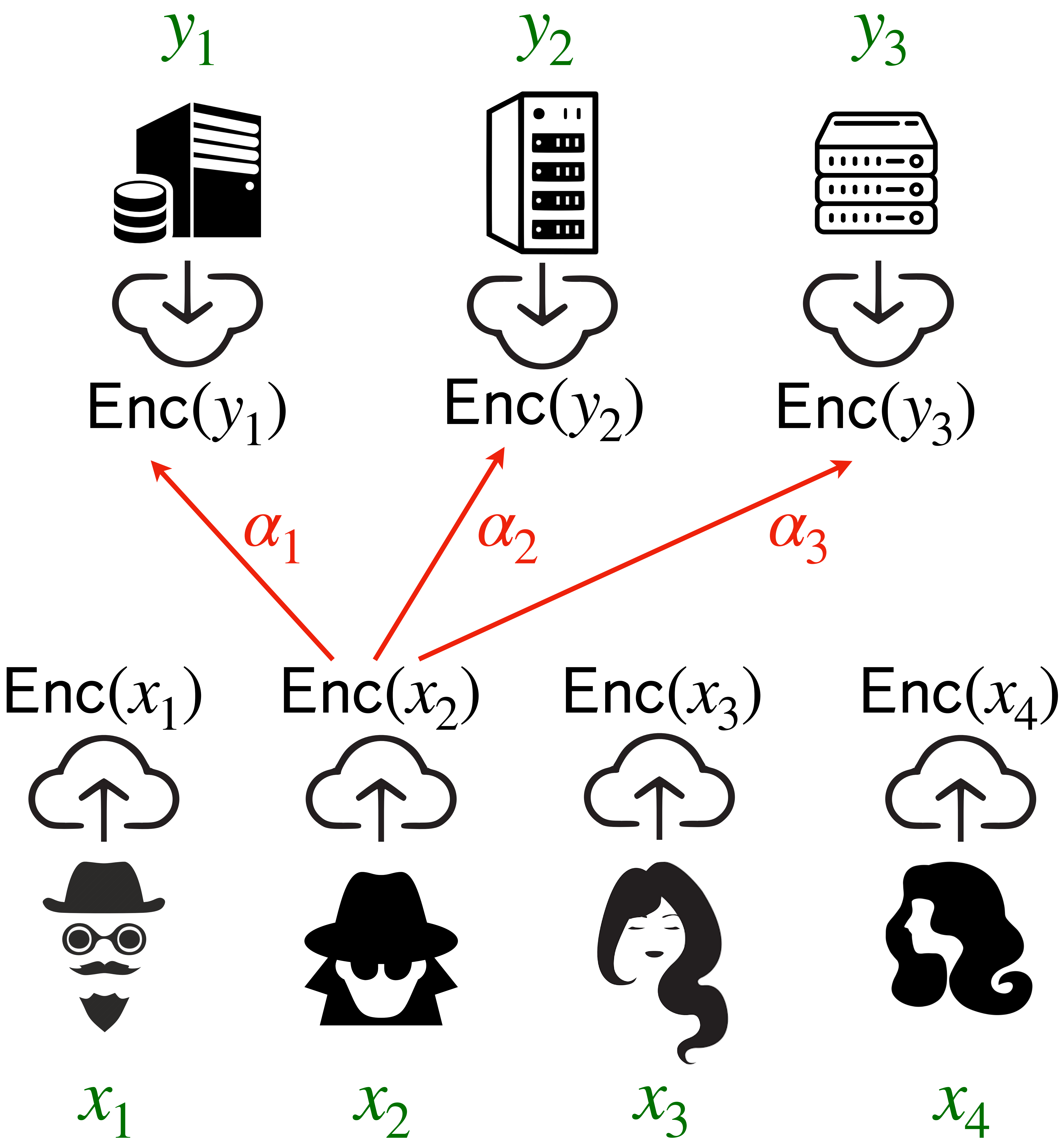
$\Longrightarrow$ leaks $\approx N \cdot t^2/n$ linear equations in r if the client interacts with N servers.

$\Longrightarrow$ still secure under the **LPN with leakage** assumption (equivalent to standard LPN, but with a loss)

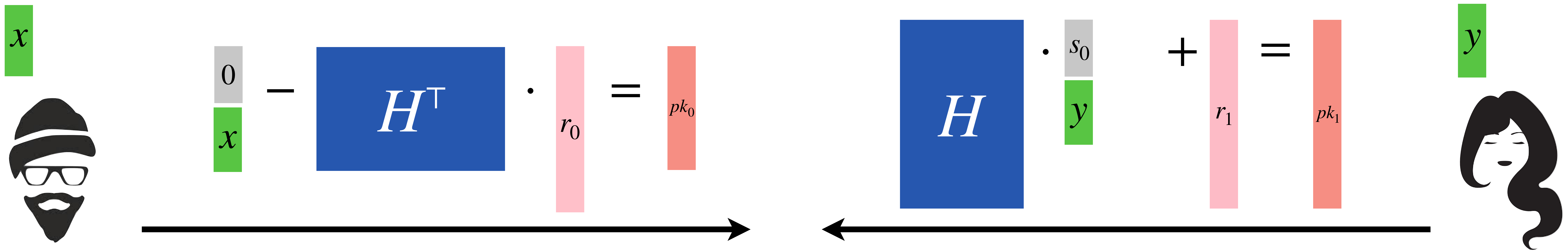
Alternatives

The above is fine when N is not too large. For large N, or when overwhelming correctness matters (e.g. for biometric authentication), two alternatives:

1. We give an LWE-based variant with negligible error
2. We describe a way to remove errors via a sublinear-communication preprocessing phase



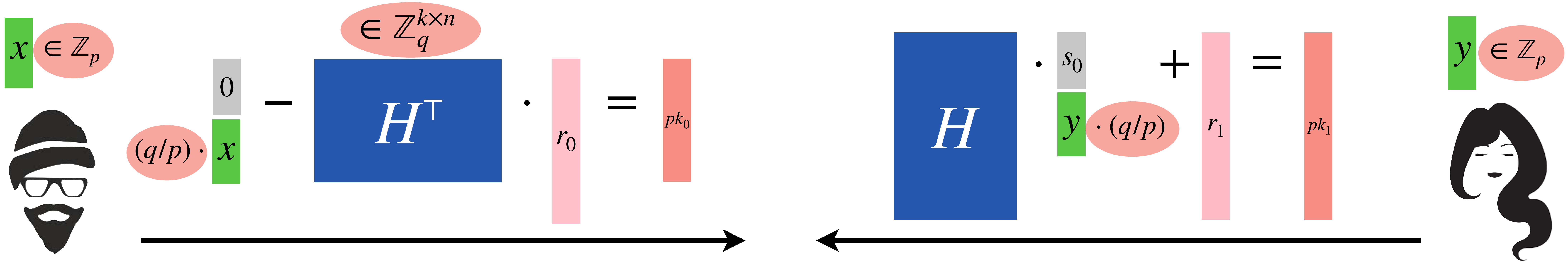
A Variant under LWE



$$\begin{pmatrix}0\\x\end{pmatrix} - H^{\top} \cdot r_0 = pk_0$$

$$H \cdot \begin{pmatrix}s_0\\y\end{pmatrix} + r_1 = pk_1$$

A Variant under LWE



$x \in \mathbb{Z}_p$

$$\begin{pmatrix} 0 \\ (q/p) \cdot x \end{pmatrix} - H^\top \cdot r_0 = pk_0 \qquad H^\top \in \mathbb{Z}_q^{k \times n}$$

$y \in \mathbb{Z}_p$

$$H \cdot \begin{pmatrix} s_0 \\ y \cdot (q/p) \end{pmatrix} + r_1 = pk_1$$

A Variant under LWE

$x \in \mathbb{Z}_p$

$$\begin{pmatrix} 0 \\ (q/p)\cdot x \end{pmatrix} - \overset{\in \mathbb{Z}_q^{k\times n}}{H^\top} \cdot r_0 = pk_0$$

$y \in \mathbb{Z}_p$

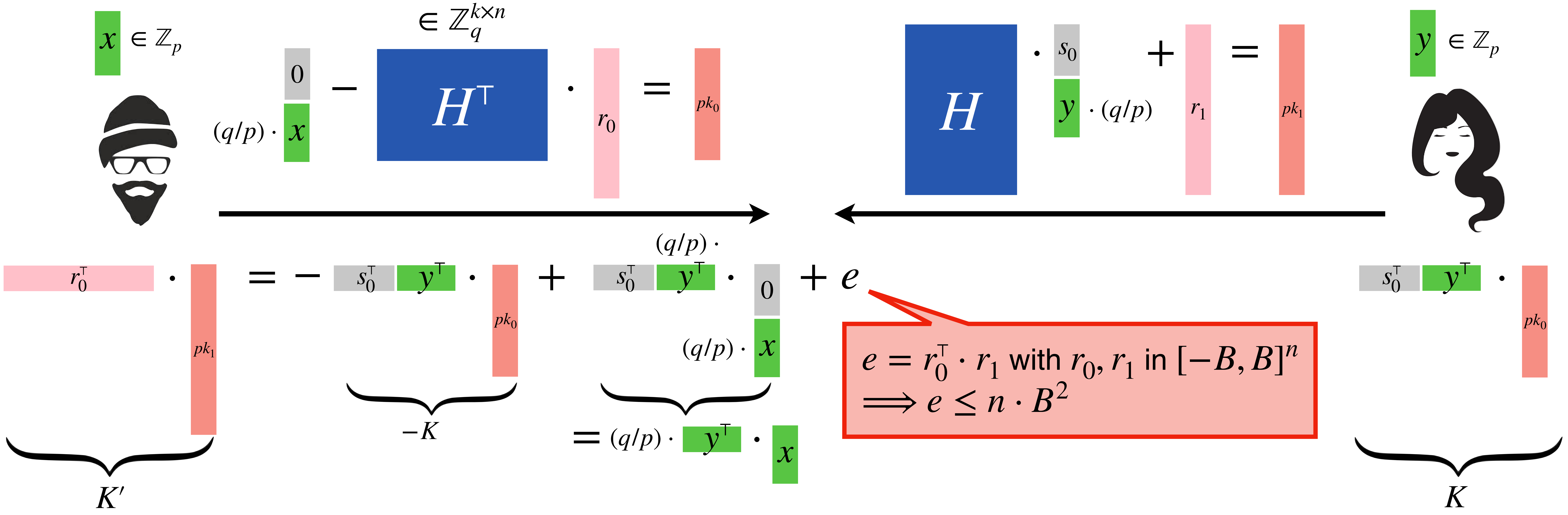
$$H \cdot \begin{pmatrix} s_0 \\ y \cdot (q/p) \end{pmatrix} + r_1 = pk_1$$

$$\longrightarrow \qquad \longleftarrow$$

$$\underbrace{r_0^\top \cdot pk_1}_{K'} = \underbrace{-\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}_{-K} + \underbrace{(q/p)\cdot \begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot \begin{pmatrix} 0 \\ (q/p)\cdot x \end{pmatrix}}_{= (q/p)\cdot y^\top \cdot x} + e$$

$$\underbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}_{K}$$

A Variant under LWE



$x \in \mathbb{Z}_p$

$$\begin{bmatrix} 0 \\ (q/p)\cdot x \end{bmatrix} - \underset{\in \mathbb{Z}_q^{k\times n}}{H^\top} \cdot r_0 = pk_0$$

$y \in \mathbb{Z}_p$

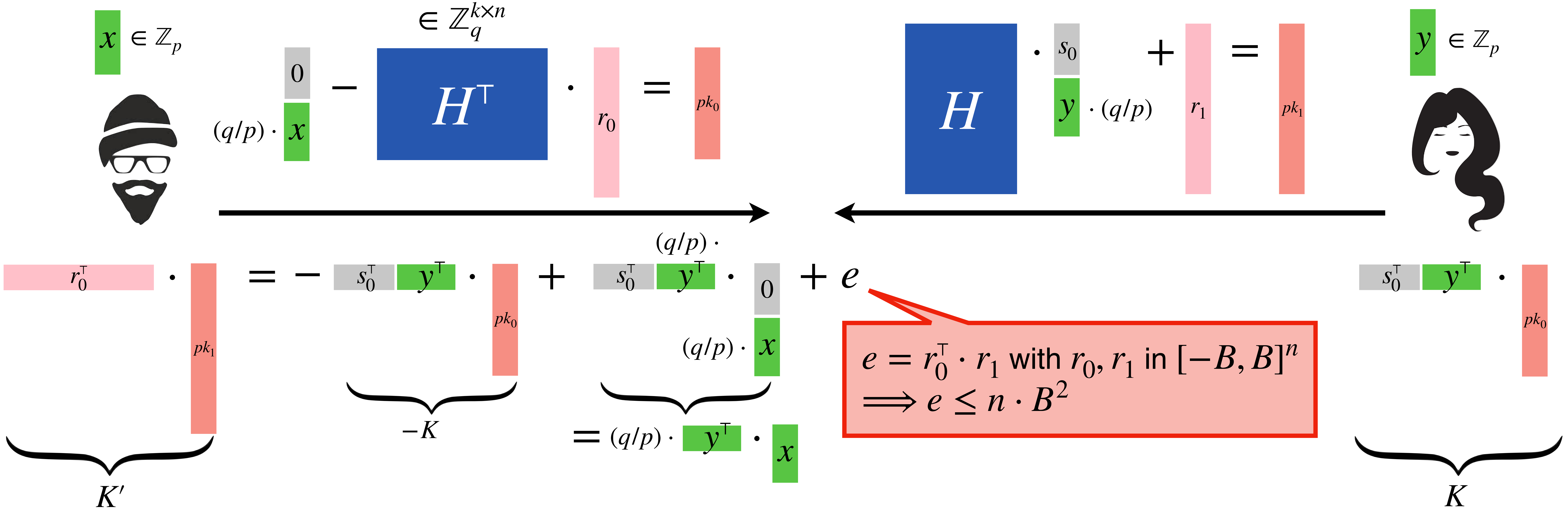
$$H \cdot \begin{bmatrix} s_0 \\ y \cdot (q/p) \end{bmatrix} + r_1 = pk_1$$

$$\underbrace{r_0^\top \cdot pk_1}_{K'} = -\underbrace{\begin{bmatrix} s_0^\top & y^\top \end{bmatrix} \cdot pk_0}_{-K} + \underbrace{(q/p)\cdot \begin{bmatrix} s_0^\top & y^\top \end{bmatrix} \cdot \begin{bmatrix} 0 \\ (q/p)\cdot x \end{bmatrix}}_{= (q/p)\cdot y^\top \cdot x} + e$$

$$\underbrace{\begin{bmatrix} s_0^\top & y^\top \end{bmatrix} \cdot pk_0}_{K}$$

$e = r_0^\top \cdot r_1$ with r_0, r_1 in $[-B, B]^n$
$\Longrightarrow e \leq n \cdot B^2$

A Variant under LWE



$x \in \mathbb{Z}_p$

$$\begin{pmatrix} 0 \\ (q/p)\cdot x \end{pmatrix} - \underbrace{H^\top}_{\in \mathbb{Z}_q^{k\times n}} \cdot r_0 = pk_0$$

$y \in \mathbb{Z}_p$

$$H \cdot \begin{pmatrix} s_0 \\ y \cdot (q/p) \end{pmatrix} + r_1 = pk_1$$

$$\underbrace{r_0^\top \cdot pk_1}_{K'} = -\underbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}_{-K} + \underbrace{(q/p)\cdot \begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot \begin{pmatrix} 0 \\ (q/p)\cdot x \end{pmatrix}}_{= (q/p)\cdot y^\top \cdot x} + e$$

$$\underbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}_{K}$$

$e = r_0^\top \cdot r_1$ with r_0, r_1 in $[-B, B]^n$
$\Longrightarrow e \leq n \cdot B^2$

Rounding Lemma: if $q/(p \cdot |e|) \geq \lambda^{\omega(1)}$, then $\lceil (p/q) \cdot K' \rfloor \bmod p$ and $\lceil (p/q) \cdot K' \rfloor \bmod p$ form additive shares of $\langle x, y \rangle$ with proba $1 - \mathsf{negl}(\lambda)$.

A Variant under LWE

$x \in \mathbb{Z}_p$

$$\begin{pmatrix} 0 \\ (q/p) \cdot x \end{pmatrix} - \underbrace{H^\top}_{\in \mathbb{Z}_q^{k \times n}} \cdot r_0 = pk_0$$

$$H \cdot \begin{pmatrix} s_0 \\ y \cdot (q/p) \end{pmatrix} + r_1 = pk_1$$

$y \in \mathbb{Z}_p$

$$\underbrace{r_0^\top \cdot pk_1}_{K'} = - \underbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}_{-K} + \underbrace{(q/p) \cdot \begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot \begin{pmatrix} 0 \\ (q/p) \cdot x \end{pmatrix}}_{= (q/p) \cdot y^\top \cdot x} + e$$

$$\underbrace{\begin{pmatrix} s_0^\top & y^\top \end{pmatrix} \cdot pk_0}_{K}$$

$e = r_0^\top \cdot r_1$ with r_0, r_1 in $[-B, B]^n$
$\Longrightarrow e \leq n \cdot B^2$

Output

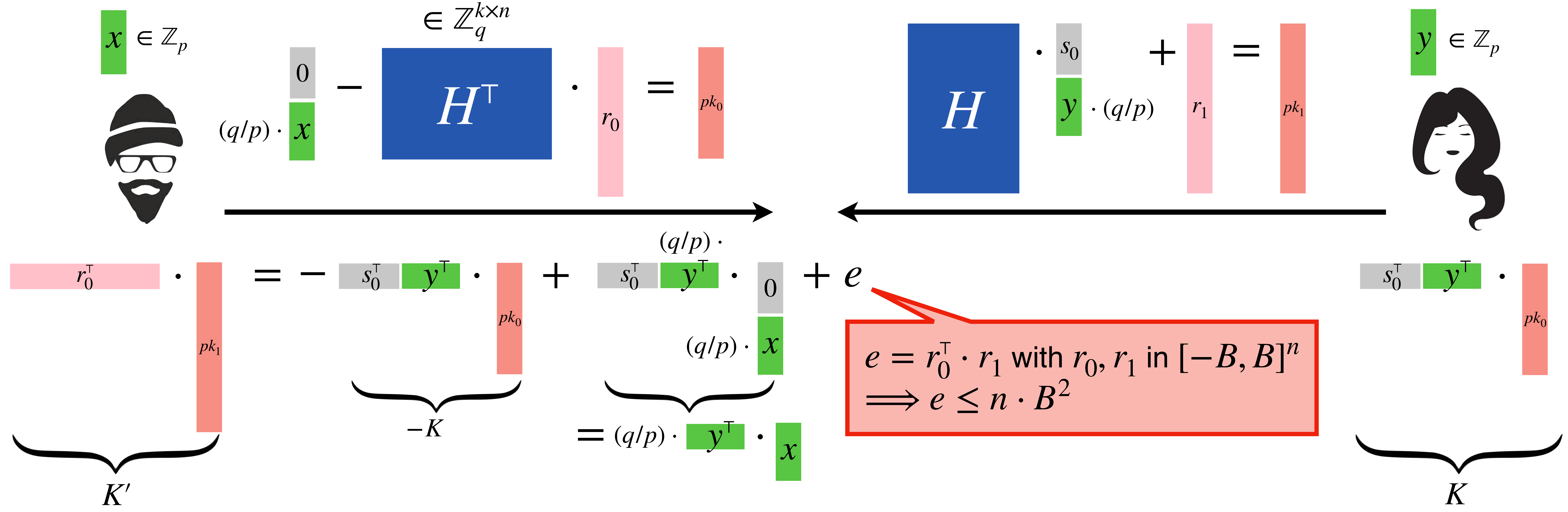
$\lceil (p/q) \cdot K' \rfloor \bmod p$

Rounding Lemma: if $q/(p \cdot |e|) \geq \lambda^{\omega(1)}$, then $\lceil (p/q) \cdot K' \rfloor \bmod p$ and $\lceil (p/q) \cdot K' \rfloor \bmod p$ form additive shares of $\langle x, y \rangle$ with proba $1 - \mathsf{negl}(\lambda)$.

Output

$\lceil (p/q) \cdot K \rfloor \bmod p$

A Variant under LWE



$x \in \mathbb{Z}_p$

$$\begin{pmatrix}0\\(q/p)\cdot x\end{pmatrix} - \underset{\in\,\mathbb{Z}_q^{k\times n}}{H^\top} \cdot r_0 = pk_0$$

$y \in \mathbb{Z}_p$

$$H \cdot \begin{pmatrix}s_0\\ y\cdot(q/p)\end{pmatrix} + r_1 = pk_1$$

$$\underbrace{r_0^\top \cdot pk_1}_{K'} = -\underbrace{\begin{pmatrix}s_0^\top & y^\top\end{pmatrix}\cdot pk_0}_{-K} + \underbrace{(q/p)\cdot\begin{pmatrix}s_0^\top & y^\top\end{pmatrix}\cdot\begin{pmatrix}0\\(q/p)\cdot x\end{pmatrix}}_{=(q/p)\cdot y^\top\cdot x} + e$$

$$\underbrace{\begin{pmatrix}s_0^\top & y^\top\end{pmatrix}\cdot pk_0}_{K}$$

$e = r_0^\top \cdot r_1$ with r_0, r_1 in $[-B, B]^n$
$\Longrightarrow e \le n \cdot B^2$

Output

$\lceil (p/q)\cdot K' \rfloor \bmod p$

Output

$\lceil (p/q)\cdot K \rfloor \bmod p$

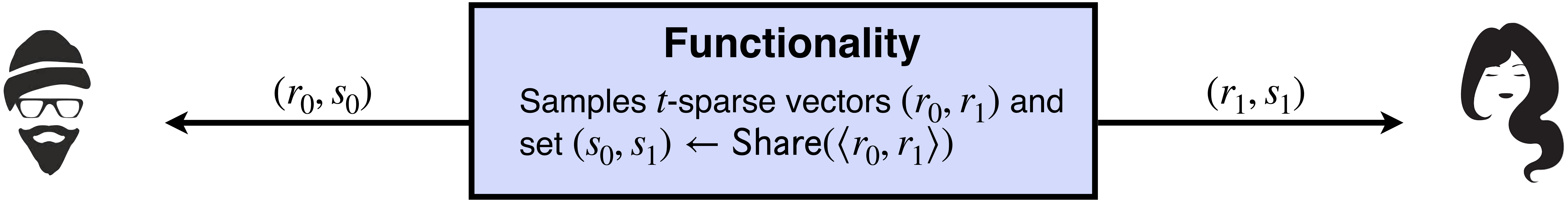
Rounding Lemma: if $q/(p\cdot|e|) \ge \lambda^{\omega(1)}$, then $\lceil (p/q)\cdot K' \rfloor \bmod p$ and $\lceil (p/q)\cdot K' \rfloor \bmod p$ form additive shares of $\langle x, y\rangle$ with proba $1-\mathsf{negl}(\lambda)$.

+ Negligible correctness error & leakage

− Larger communication overhead

Removing Errors via Preprocessing

Preprocessing phase

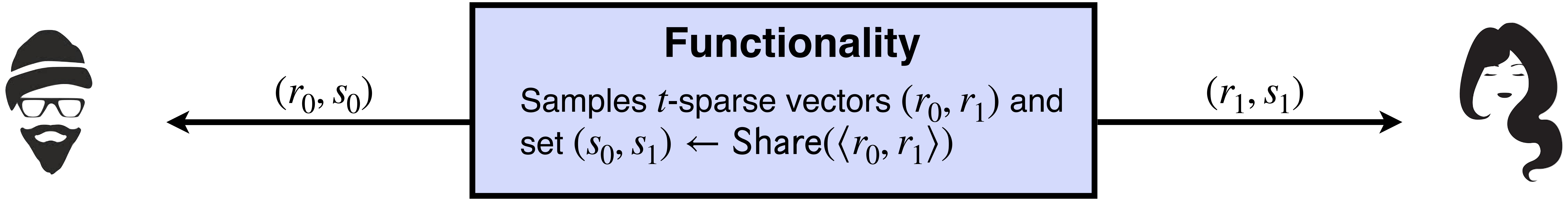


Functionality

Samples t-sparse vectors (r_0, r_1) and set $(s_0, s_1) \leftarrow \mathsf{Share}(\langle r_0, r_1 \rangle)$

Removing Errors via Preprocessing

Preprocessing phase



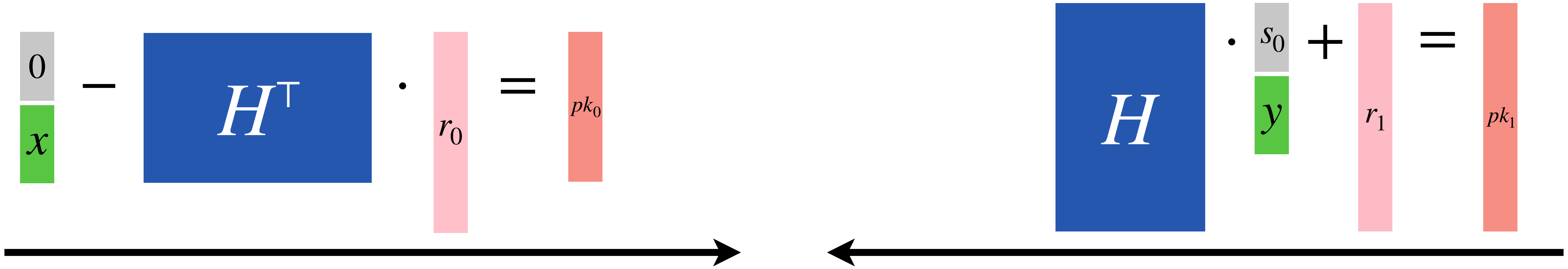
(r_0, s_0)

Functionality

Samples t-sparse vectors (r_0, r_1) and set $(s_0, s_1) \leftarrow \mathsf{Share}(\langle r_0, r_1 \rangle)$

(r_1, s_1)

Online phase



$$\begin{bmatrix}0\\x\end{bmatrix} - H^\top \cdot r_0 = pk_0$$

$$H \cdot \begin{bmatrix}s_0\\y\end{bmatrix} + r_1 = pk_1$$

Removing Errors via Preprocessing

Preprocessing phase

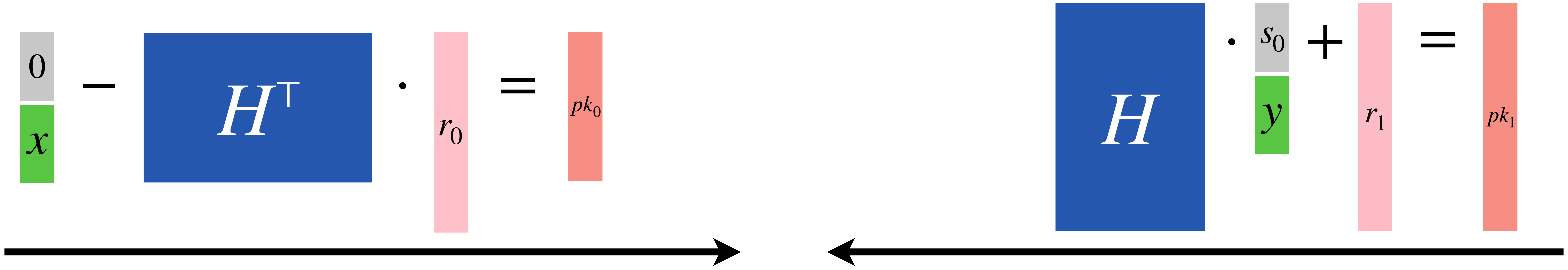
(r_0, s_0) ←

Functionality

Samples t-sparse vectors (r_0, r_1) and set $(s_0, s_1) \leftarrow \mathsf{Share}(\langle r_0, r_1 \rangle)$

→ (r_1, s_1)

Online phase



Output

Output

$\langle r_0, \mathsf{pk}_1 \rangle + s_0$

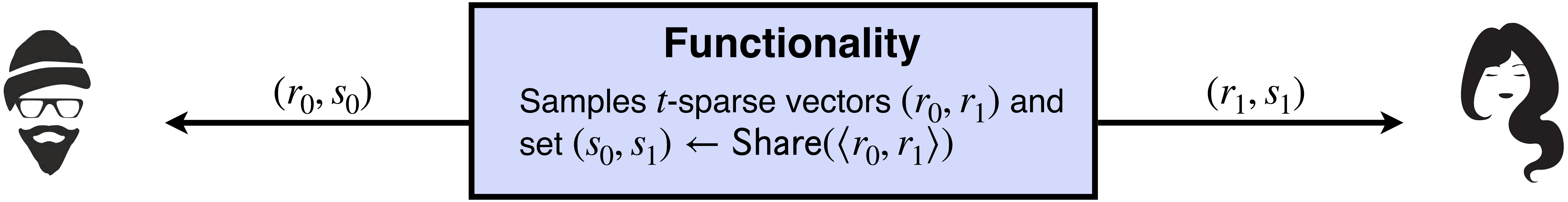
$$
\begin{aligned}
&\langle r_0, \mathsf{pk}_1 \rangle - s_0 + \langle s, \mathsf{pk}_0 \rangle - s_1 \\
=\ &\langle x, y \rangle - \langle r_0, r_1 \rangle + (s_0 + s_1) \\
=\ &\langle x, y \rangle
\end{aligned}
$$

Output

$\langle s, \mathsf{pk}_0 \rangle + s_1$

Removing Errors via Preprocessing

Preprocessing phase



(r_0, s_0)

Functionality

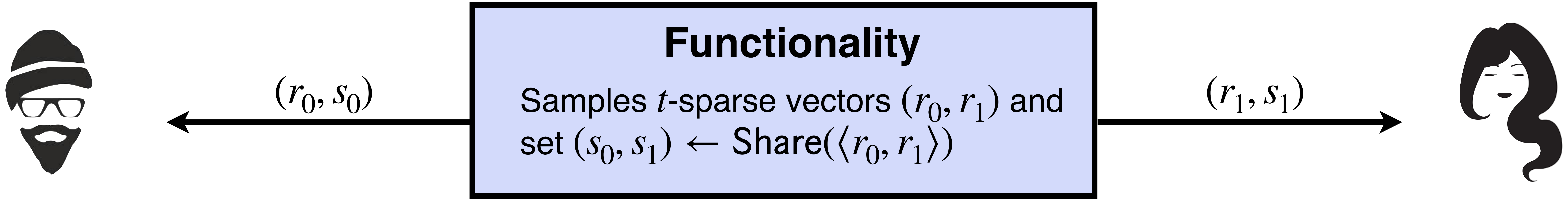
Samples t-sparse vectors (r_0, r_1) and set $(s_0, s_1) \leftarrow \text{Share}(\langle r_0, r_1 \rangle)$

(r_1, s_1)

Implementing the preprocessing

Removing Errors via Preprocessing

Preprocessing phase



(r_0, s_0)

Functionality

Samples t-sparse vectors (r_0, r_1) and set $(s_0, s_1) \leftarrow \mathsf{Share}(\langle r_0, r_1 \rangle)$

(r_1, s_1)

Implementing the preprocessing

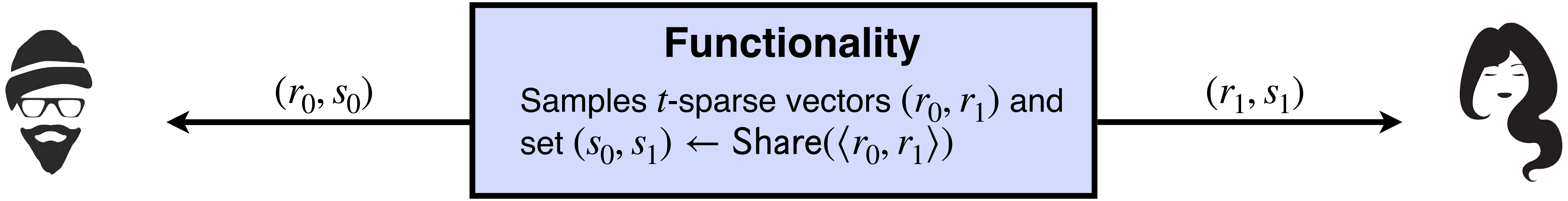
Write $r_\sigma = \sum_{i=1}^{t} r_\sigma^{(i)}$ where the $r_\sigma^{(i)}$ are unit vectors $\underset{\substack{\blacktriangle \\ k_\sigma^{(i)}}}{0 \cdots 0\ v_\sigma^{(i)}\ 0 \cdots 0}$

Then: $\langle r_0, r_1 \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \langle r_0^{(i)}, r_1^{(j)} \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \underbrace{[k_\sigma^{(i)} =_? k_\sigma^{(j)}]}_{\text{Secure Equality Test}} \cdot \underbrace{v_0^{(i)} v_1^{(j)}}_{\text{OLE}}$

$\Longrightarrow s_0, s_1$ can be securely computed using $O(t^2 \cdot \log n)$ communication

Removing Errors via Preprocessing

Preprocessing phase



Functionality

Samples t-sparse vectors (r_0, r_1) and set $(s_0, s_1) \leftarrow \text{Share}(\langle r_0, r_1 \rangle)$

(r_0, s_0)

(r_1, s_1)

Implementing the preprocessing

Write $r_\sigma = \sum_{i=1}^{t} r_\sigma^{(i)}$ where the $r_\sigma^{(i)}$ are unit vectors $\boxed{0 \cdots 0 \; \underset{\overset{\uparrow}{k_\sigma^{(i)}}}{v_\sigma^{(i)}} \; 0 \cdots 0}$

Then: $\langle r_0, r_1 \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \langle r_0^{(i)}, r_1^{(j)} \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \underbrace{[k_\sigma^{(i)} =_? k_\sigma^{(j)}]}_{\text{Secure Equality Test}} \cdot \underbrace{v_0^{(i)} v_1^{(j)}}_{\text{OLE}}$

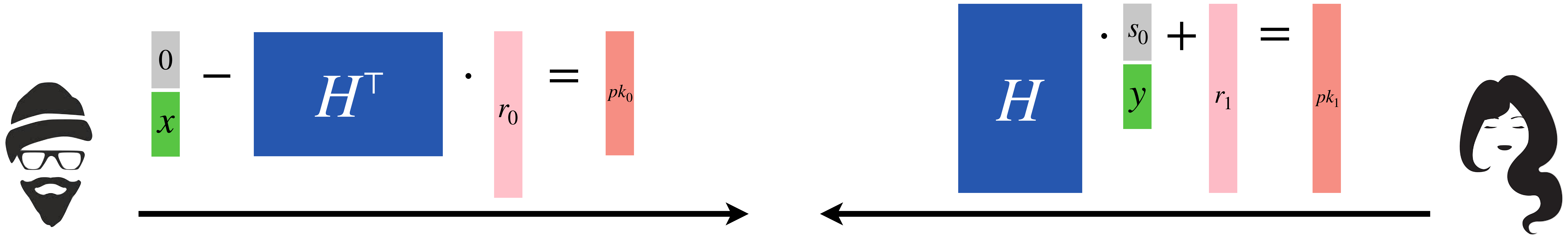
$\Longrightarrow$ s_0, s_1 can be securely computed using $O(t^2 \cdot \log n)$ communication

Improved version: using LPN with *regular noise* brings the cost down to $O(t \cdot \log n)$

$$r_\sigma = \boxed{0 \cdots 0 \; v_1 \; 0 \cdots 0} \boxed{0 \cdots 0 \; v_2 \; 0 \cdots 0} \boxed{0 \cdots 0 \; v_3 \; 0 \cdots 0} \boxed{0 \cdots 0 \; v_4 \; 0 \cdots 0}$$

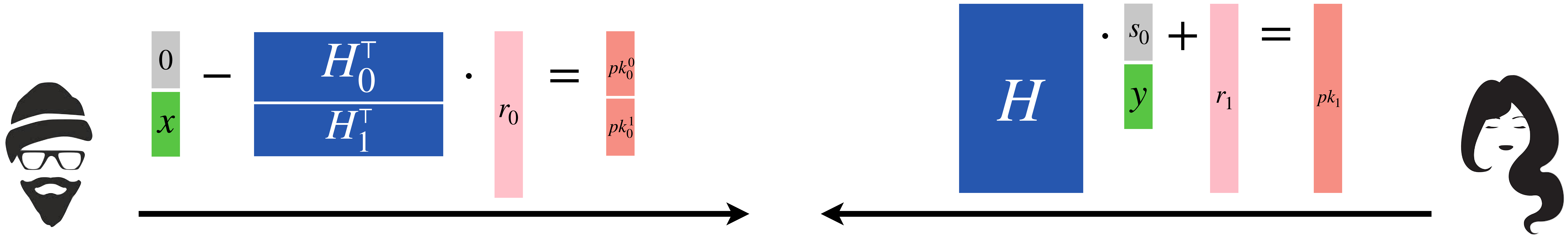
Malicious Security

In the malicious setting, Alice and Bob must prove that pk_0, pk_1 are well-formed



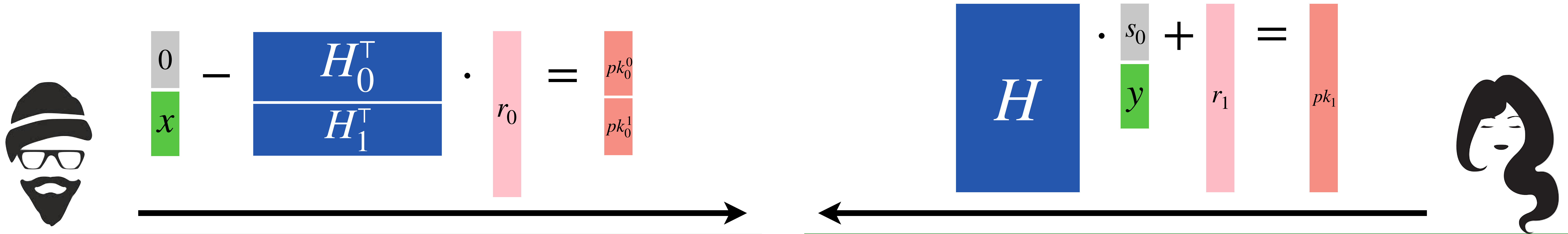
Malicious Security

In the malicious setting, Alice and Bob must prove that pk_0, pk_1 are well-formed



Malicious Security

In the malicious setting, Alice and Bob must prove that pk_0, pk_1 are well-formed

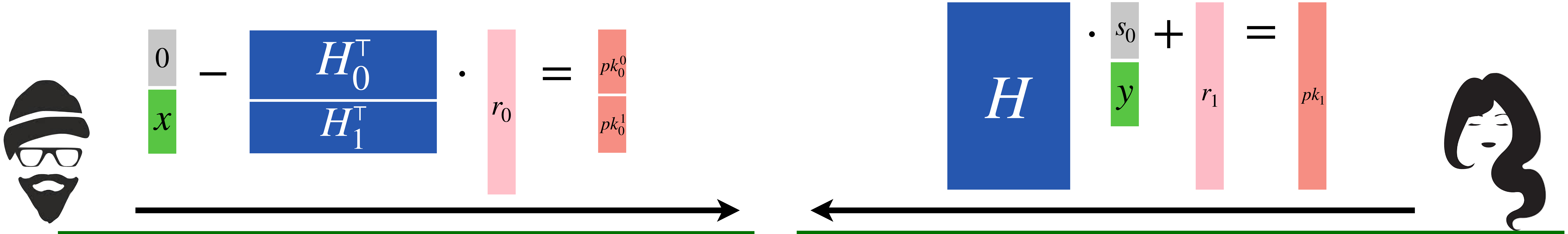


ZK: knows a sparse r_0 such that $H_0^\top \cdot r_0 = \mathsf{pk}_0^0$

ZK: knows sparse r_1 such that $\exists s, H \cdot s + r_1 = \mathsf{pk}_1$

Malicious Security

In the malicious setting, Alice and Bob must prove that pk_0, pk_1 are well-formed



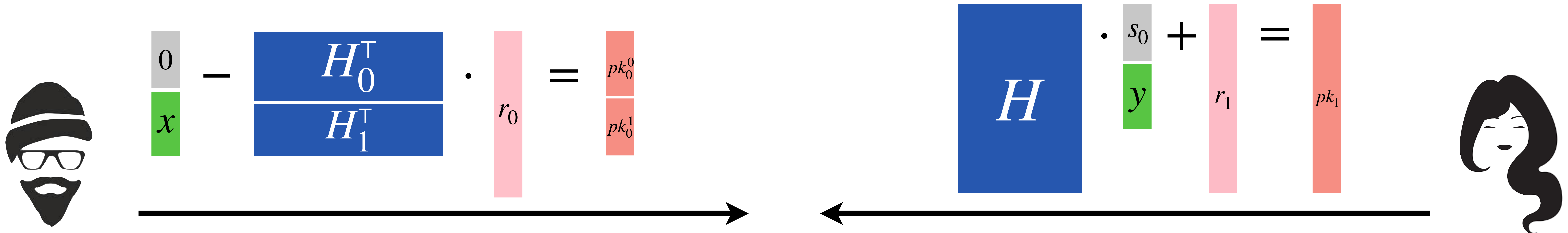
ZK: knows a sparse r_0 such that $H_0^\top \cdot r_0 = \mathsf{pk}_0^0$

ZK: knows sparse r_1 such that $\exists s, H \cdot s + r_1 = \mathsf{pk}_1$

We introduce a new efficient ZKPoK for LPN relations, with communication $O(t \cdot \log n)$

Malicious Security

In the malicious setting, Alice and Bob must prove that $\mathsf{pk}_0, \mathsf{pk}_1$ are well-formed



ZK: knows a sparse r_0 such that $H_0^\top \cdot r_0 = \mathsf{pk}_0^0$

ZK: knows sparse r_1 such that $\exists s, H \cdot s + r_1 = \mathsf{pk}_1$

We introduce a new efficient ZKPoK for LPN relations, with communication $O(t \cdot \log n)$

Idea

Recent works on *pseudorandom correlation generator* show how to distribute shares of $\Delta \cdot r$ where $r \in \mathbb{F}_p^n$ is t-sparse, and $\Delta \in \mathbb{F}_{p^k}$, via *function secret sharing* (which exist under OWF)

$\Longrightarrow$ we use these techniques to *authenticate* the noise vectors with a MAC Δ

$\Longrightarrow$ we use the MAC to check the correct computation of the pk$'s$, *al la* SPDZ.

Thank you for your attention!

Questions?