Two-Party Semi-Honest Security
for deterministic functionalities

Let f be a function. We say that a protocol Π securely computes f in the presence of a semi-honest adversary if for each party $i \in \{0,1\}$ there exists a polynomial time simulator $\mathcal{S}_i$ *such that for all inputs x_0, x_1:*

$$\text{View}_i^{\Pi}(x_0, x_1) \stackrel{c}{=} \mathcal{S}_i(x_i, f(x_0, x_1))$$

Two-Party Semi-Honest Security
for deterministic functionalities

Let f be a function. We say that a protocol Π securely computes f in the presence of a semi-honest adversary if for each party $i \in \{0,1\}$ there exists a polynomial time simulator $\mathcal{S}_i$ *such that for all inputs x_0, x_1:*

$$\text{View}_i^{\Pi}(x_0, x_1) \stackrel{c}{=} \mathcal{S}_i(x_i, f(x_0, x_1))$$

Pseudorandom Function (PRF)

A function family F is considered pseudorandom *if the following indistinguishability holds*

Real:

private $k \stackrel{\$}{\leftarrow} \{0,1\}^\lambda$

public lookup(m):
  return $F(k, m)$

$$\stackrel{c}{=}$$

Ideal:

private $T \leftarrow \text{EmptyDictionary}$

public lookup(m):
  if $m \notin T$:
    $T[m] \stackrel{\$}{\leftarrow} \{0,1\}^{\text{out}}$
  return $T[m]$

"F looks random"

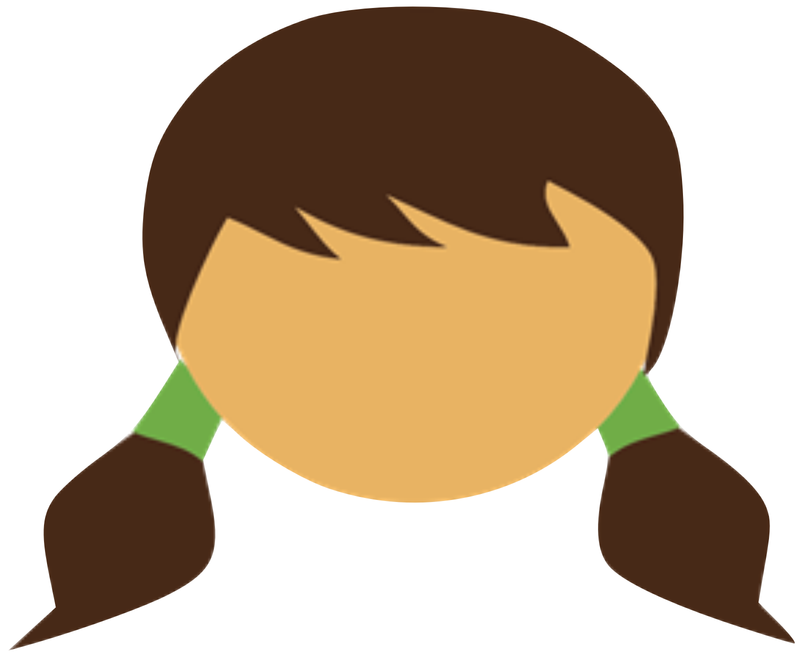
Let's "securely" implement the following functionality

Input: P_0, P_1 input nothing

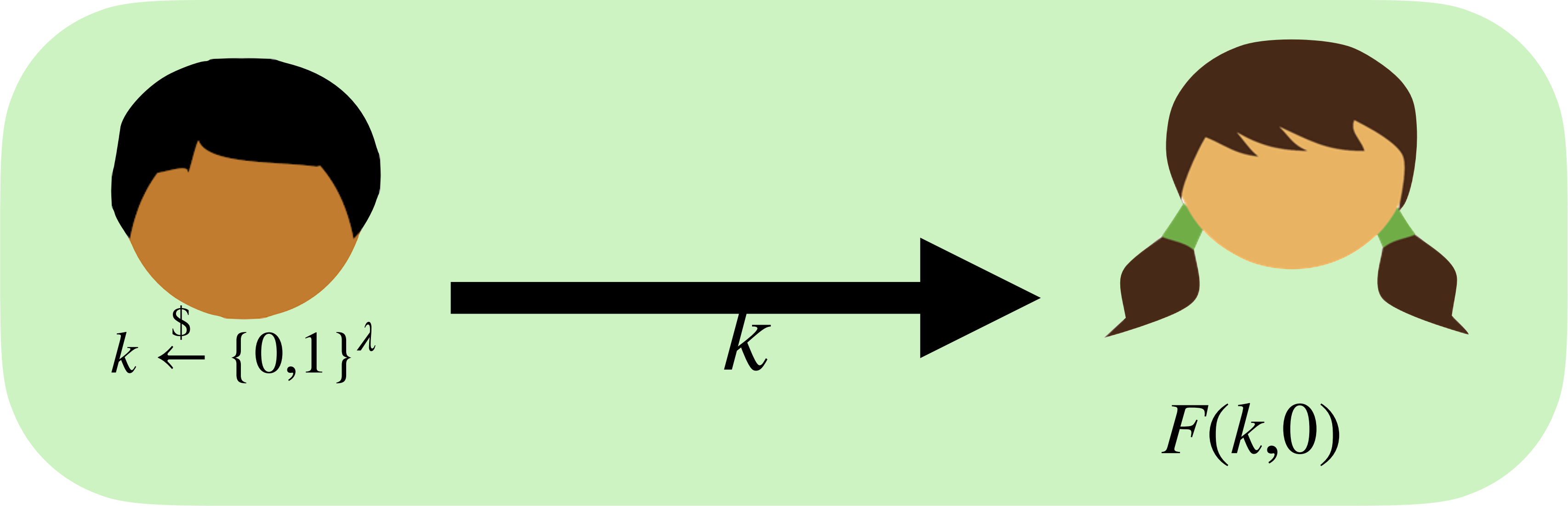
Output: P_0 outputs an encryption key k, P_1 outputs $F(k,0)$

$k \xleftarrow{\$} \{0,1\}^\lambda$

$$\xrightarrow{\quad k \quad}$$



$F(k,0)$



$\text{View}_0(\ )$:
$k \xleftarrow{\$} \{0,1\}^\lambda$
return k

$=$

$\mathcal{S}_0(k)$:
$k' \xleftarrow{\$} \{0,1\}^\lambda$
return k'

The simulated view is not consistent with the output!

$\text{View}_1(\ )$:
$k \xleftarrow{\$} \{0,1\}^\lambda$
return k

$=$

$\mathcal{S}_1(F(k,0))$:
$k' \xleftarrow{\$} \{0,1\}^\lambda$
return k'

Two-Party Semi-Honest Security **for deterministic functionalities**

Let f be a ***deterministic*** *functionality. We say that a protocol Π securely computes f in the presence of a semi-honest adversary if for each party $i \in \{0,1\}$ there exists a polynomial time* simulator $\mathcal{S}_i$ *such that for all inputs x_0, x_1:*

$$\{\text{View}_i^{\Pi}(x_0, x_1)\}$$

$$\overset{c}{=}$$

$$\{\mathcal{S}_i(x_i, y_i) \mid (y_0, y_1) \leftarrow f(x_0, x_1)\}$$

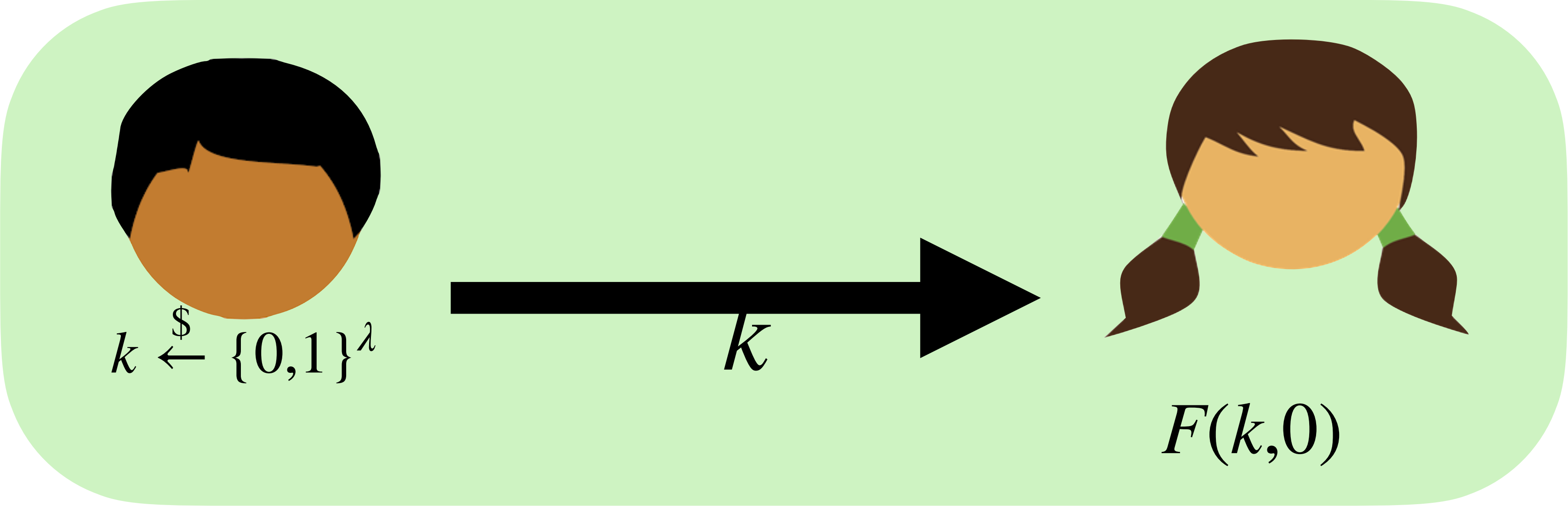
Two-Party Semi-Honest Security

Let f be a functionality. We say that a protocol Π securely computes f in the presence of a semi-honest adversary if for each party $i \in \{0,1\}$ there exists a polynomial time simulator $\mathcal{S}_i$ *such that for all inputs x_0, x_1:*

$$\{\text{View}_i^{\Pi}(x_0, x_1), \text{Output}^{\Pi}(x_0, x_1)\}$$

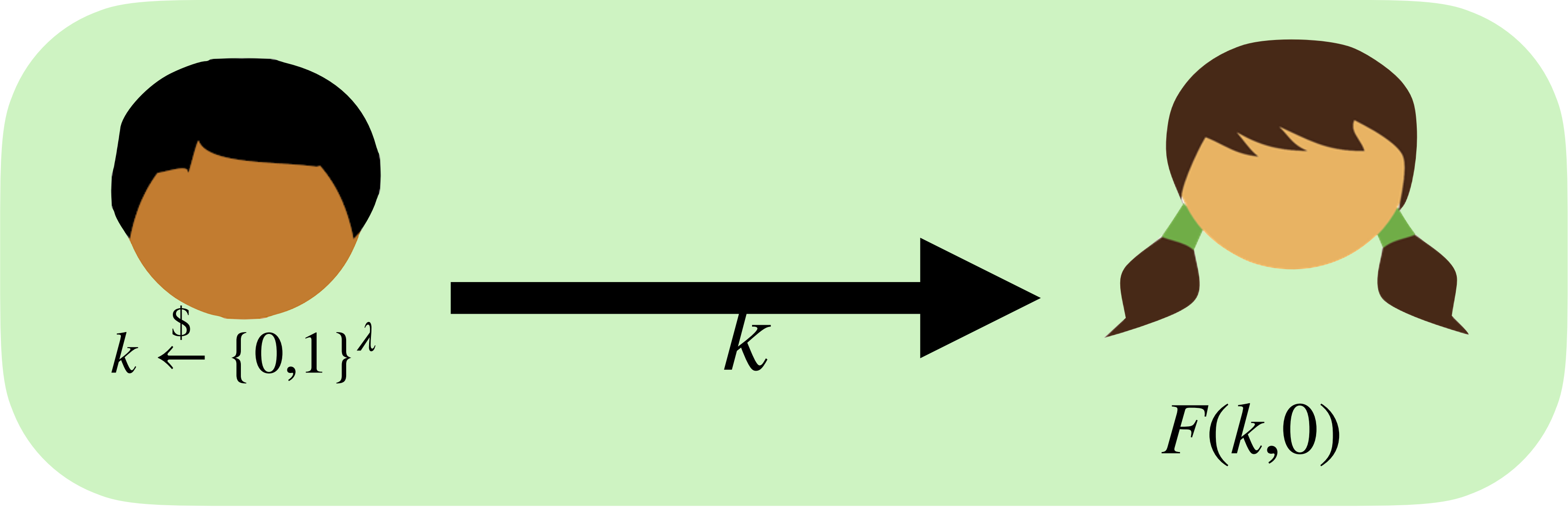
$$\overset{c}{=}$$

$$\{\mathcal{S}_i(x_i, y_i), (y_0, y_1) \mid (y_0, y_1) \leftarrow f(x_0, x_1)\}$$



$$\{\mathrm{View}_i^{\Pi}(x_0, x_1), \mathrm{Output}^{\Pi}(x_0, x_1)\} \stackrel{c}{=} \{\mathcal{S}_i(x_i, y_i), (y_0, y_1) \mid (y_0, y_1) \leftarrow f(x_0, x_1)\}$$

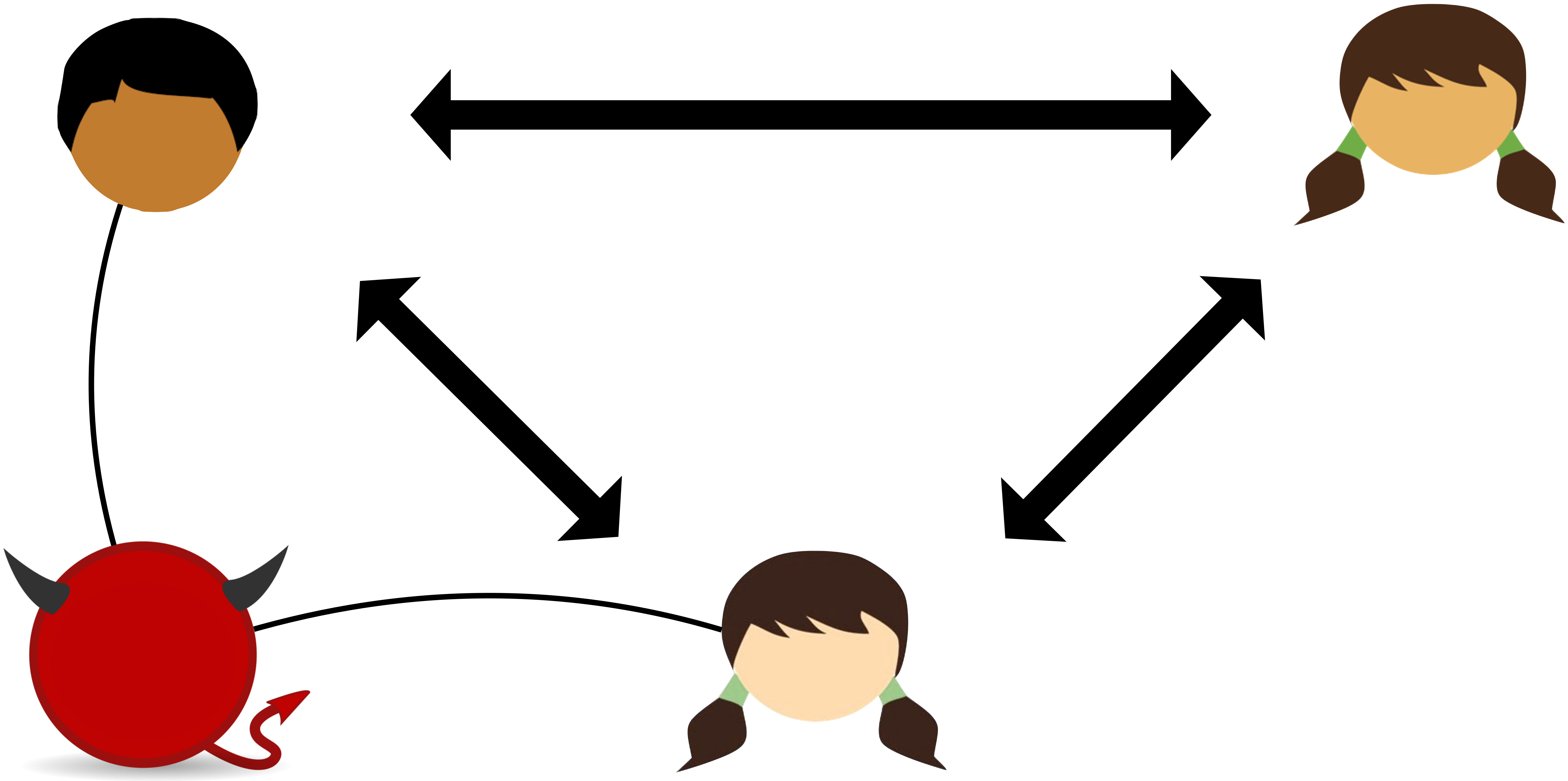
$$\{k, (k, F(k,0))\} \stackrel{c}{=} \{\mathcal{S}_1(F(k,0)), (k, F(k,0)) \mid k \leftarrow \{0,1\}^\lambda\}$$

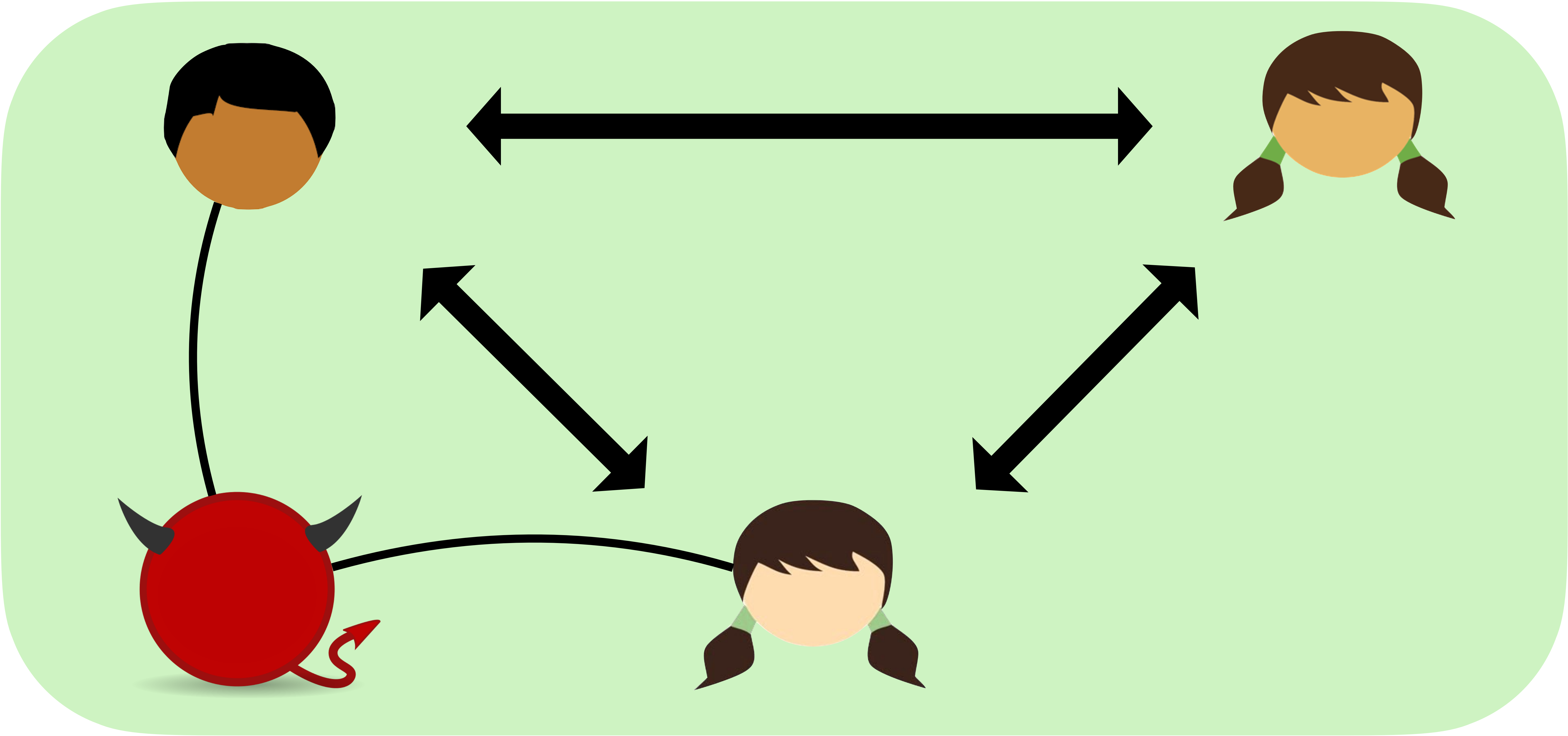


Fact: there does not exist $\mathcal{S}_1$ proving this protocol secure

Proof: By using the existence of $\mathcal{S}_1$ to construct a distinguisher for the PRF

$$\{k, (k, F(k,0))\} \overset{c}{=} \{\mathcal{S}_1(F(k,0)), (k, F(k,0)) \mid k \leftarrow \{0,1\}^\lambda\}$$





We consider a single global adversary who corrupts a subset of the parties

Two-Party Semi-Honest Security

Let f be a functionality. We say that a protocol Π securely computes f in the presence of a semi-honest adversary if for each party $i \in \{0,1\}$ there exists a polynomial time simulator $\mathcal{S}_i$ *such that for all inputs x_0, x_1:*

$$\{\mathrm{View}_i^{\Pi}(x_0, x_1), \mathrm{Output}^{\Pi}(x_0, x_1)\}$$

$$\approx$$

$$\{\mathcal{S}_i(x_i, y_i), (y_0, y_1) \mid (y_0, y_1) \leftarrow f(x_0, x_1)\}$$

Semi-Honest Security

Let $P_0, ..., P_{n-1}$ be n parties. Let f be a functionality. We say that a protocol Π securely computes f in the presence of a semi-honest adversary if for each subset $c \subseteq \{0,...,n-1\}$ of corrupted parties there exists a polynomial time simulator $\mathcal{S}_c$ *such that for all inputs $x_0, ..., x_{n-1}$:*

$$\left\{\left(\bigcup_{i\in c} \text{View}_i^{\Pi}(x_0, ..., x_{n-1})\right), \text{Output}^{\Pi}(x_0, ..., x_{n-1})\right\}$$

$$\approx$$

$$\left\{\mathcal{S}_c\left(\bigcup_{i\in c}\{x_i, y_i\}\right), (y_0, ... y_{n-1}) \mid (y_0, ... y_{n-1}) \leftarrow f(x_0, ..., x_{n-1})\right\}$$