Secure Computation

Proof of honest But Curious :

Perfect world caracterizations :

Look @ case where 1 Adversary 1 Honest otherwise it is trivial

→ $\mathcal{F}$ knows who is corrupted

→ For P_i corrupted P_j not:

1) $\mathcal{F}$ recieves x_i' for P_i
2) $\mathcal{F}$ recieves x_j from P_j
3) returns $f_i(x_i', x_j)$ to P_i
4) if P_i HALT then stop
5) else return $f_j(x_i', x_j)$ to P_j

We say that a protocol Π securely emulates $\mathcal{F}$ if

$\forall A$ against Π, $\exists S$ against $\mathcal{F}$ st :

$$\mathrm{Real}_{\pi,A} \approx \mathrm{Ideal}_{\mathcal{F},S}$$

A rend le protocole non secure s'il rend pas le m̂ resultat

wait for proof thing for exemple

Oblivious Transfer

<u>Goal</u> : $\begin{cases} f_1((m_0, m_1), b) = \perp \\ f_2((m_0, m_1), b) = m_b \end{cases}$

⇒ " P_2 asks for m_b to P_1 that doesn't know which one P_1 is sending "

⇒ Based on Homomorphic encryption's properties:

$$\begin{cases} c_1 \cdot c_2 = \mathrm{Enc}_{pk}(m_1 + m_2) \\ a \cdot c_1 = [\mathrm{Enc}_{pk}(m_1)]^a \end{cases}$$

<u>Protocol</u> :

$(m_0, m_1) \rightsquigarrow P_1$ $\qquad$ $b \rightsquigarrow P_2$

$P_1 \xleftarrow{pk,\ \mathrm{Enc}_{pk}(b)} P_2$

$P_1 \xrightarrow[c_1 = \mathrm{Enc}_{pk}(b m_1 + r_1(1-b))]{c_0 = \mathrm{Enc}_{pk}((1-b) m_0 + r_0 b)} P_2$

$P_1 \to \perp$ $\qquad$ $P_2 \to m_b$

<u>Note</u> :

→ c_0 and c_1 are computed without knowing b

→ $m_b = \mathrm{Dec}_{sk}(c_b)$

→ is honest but curious security (proof by simulation)

Used for

Yao's Grabled Circuits

<u>Goal</u>: → $\begin{cases} P_1 \text{ has } x_1 \\ P_2 \text{ has } x_2 \end{cases}$

→ P_2 need to compute $f_2(x_1, x_2)$ st

- P_1 learns nothing about x_2
- P_2 learns nothing about x_1

→ P_1 has $f_1(x_1, x_2) = \perp$

<u>Note</u> : The fact that $f_1 = \perp$ is not an issue and can be converted using the following technique

- $f_1'((x_1, r), x_2) = \perp$ for a random r and
- $f_2'((x_1, r), x_2) = (f_1(x_1, x_2) \oplus r, f_2(x_1, x_2))$

→ and send back $f_1(x_1, x_2) \oplus r$ to P_1 at the end

<u>Procedure</u>:

P_1 $\qquad\qquad$ P_2

| <u>Obfuscation</u> | : From f_2 that is public do

i) input wire → k_w^0, k_w^1 (output wire = 0, 1 to be able to read)

ii) gate ⇒ $\mathrm{Enc}_{k_{w_1}^0} \mathrm{Enc}_{k_{w_2}^1}(g(0,1) \| 0^n)$ 4 things

⇒ only one pair of key can decrypt

iii) shuffle !

$\xrightarrow{\text{Send obfuscated}}$ f_2

| <u>Send key</u> | : from x_1 compute the k_i (either 0 or 1) $\xrightarrow{k_{i,1}}$

P_2 needs to input the keys at the spots of his input... Can't just ask P_1 otherwise he compromises his key... ⇒ OT

for each bit b_i of the key x_2:

$\xleftarrow{\mathrm{Enc}_{pk}(b_i),\ pk}$

$\xrightarrow[m_1 = k_i^1]{c_0, c_1 \text{ with } m_0 = k_i^0}$ has k_i^b without compromizing

Can compute solution of $f_2(x_1, x_2)$