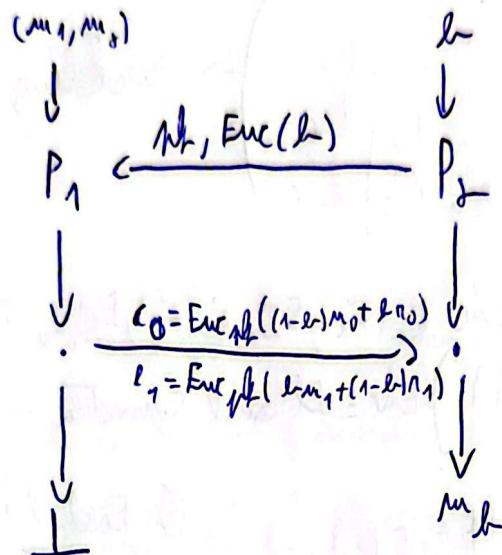


Privacy:

TP.4.0

TP4:

(Q1)



$$\begin{aligned}
 r_0, r_1 &\in \text{rand}(2^{\delta}) \\
 m_0, m_1 &\leftarrow 2^{\delta} \\
 r &= \text{Enc}_{\text{pk}}(h) \\
 y &= \text{Enc}_{\text{pk}}(1-h) \\
 &= \text{Enc}_{\text{pk}}(1) \cdot h^{-1} \\
 r_0 &= y^{m_0} \cdot x^{r_0} \\
 r_1 &= x^{m_1} \cdot y^{r_1}
 \end{aligned}$$

$$nk, \text{Enc}_{\text{pk}}(h)$$

$$P_2 \quad h \in \{0, 1\}$$

$$rk \leftarrow 2^{\delta} \quad 1/rk = p^{100\%}$$

If $h=0$: $x=r_0$ else: $x=r_1$

$$\text{Dec}_{\text{pk}}(x) = \log_g(y^{m_x}) = m_x$$

(check pattern(m_x))

~~fairly~~ Feasible because
 m_x is small enough!!

(Q2) GARBLING PHASE

| | | Inputs | Outputs |
|---|---|--------|---------------------|
| A | B | C | C |
| 0 | 0 | 0 | $E_{L_0, R_0}(l^0)$ |
| 0 | 1 | 0 | $E_{L_0, R_1}(l^0)$ |
| 1 | 0 | 0 | $E_{L_1, R_0}(l^1)$ |
| 1 | 1 | 1 | $E_{L_1, R_1}(l^1)$ |

Garbled Table = shuffle $\left[E_{L_0, R_0}(l^0), E_{L_0, R_1}(l^0), E_{L_1, R_0}(l^1), E_{L_1, R_1}(l^1) \right]$

$(E_{A,B}(C) = Enc_A(Enc_B(C || 0^M)) \wedge Enc \equiv ElGamal \text{ Symmetric Encryption})$

2) DATA TRANSFER PHASE (S)

Each bits generation has its own collection $(l_0^0, l_1^0, l_2^0, l_0^1, l_1^1, l_2^1)$!

Stream bits on P_0 (Garbler)

For each bit i *

$a_i \in \mathbb{R}$:

map: $a_i \rightarrow (l_0^0, l_1^0, l_2^0, l_0^1, l_1^1, l_2^1)$

compute garbled table T_i

map: $a_i \rightarrow l_0^{a_i}$

Compute challenges r_0, r_1
for 0+ where $m_0 = l_0^0$
 $\wedge m_1 = l_0^1$.

map: $l_0^{a_i} \wedge r_i \rightarrow r_i = a_i \wedge l_0^{a_i}$

P_1 (Executor)

Stream bits b_0, b_1, b_2
Take next bit b_i

Init oblivious Transfer to get safely $l_0^{b_i}$

$$Dec_{P_0}(l_0^{b_i}) = log_g(l_0^{a_i \wedge b_i}) = l_0^{b_i}$$

$$Dec_{P_0, P_1}(T_i) =$$

$[m_0, m_1, l_0^{a_i \wedge b_i}, m]$ (order could be different.)

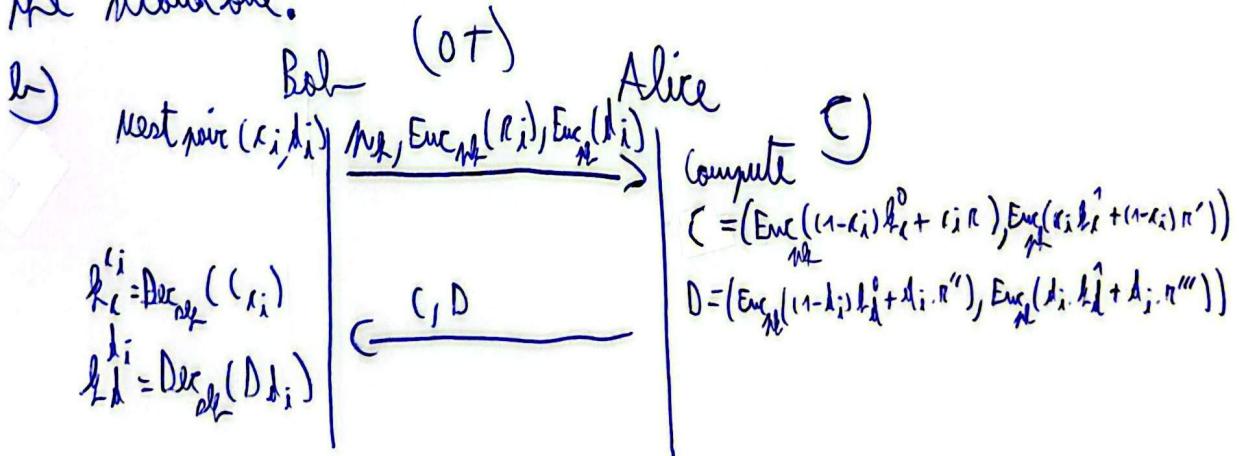
Get valid decryption $l_0^{a_i \wedge b_i}$ with m as ending zeros, all others are invalid.

3) A R C D A&R A&C A&D B&C B&D C&D E F T P.7.2

| A | R | C | D | A&R | A&C | A&D | B&C | B&D | C&D | E | F | T | P.7.2 |
|---|---|---|---|-----|-----|-----|-----|-----|-----|---|---|---|-------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

a) Alice for each pair (x_i, b_i) of her stream input create a mapping collection $(l_x^0, l_x^1, l_{x \wedge r}^0, l_{x \wedge r}^1, l_x^0, l_x^1, l_d^0, l_d^1, l_e^0, l_e^1, l_f^0, l_f^1)$ where each l_x^B means label for variable x equaling boolean B .

There are 2 garbled tables containing $2^4 = 16$ entries resulting from the shuffle of the list of all possible combination (x_i, b_i, l_i, d_i) and each entry is encoded as $\text{Enc}_{\text{pk}_i}(\text{Enc}_{\text{pk}_i}(\text{Enc}_{\text{pk}_i}(\text{Enc}_{\text{pk}_i}(f_x^{x_i, b_i, l_i, d_i}) || 0^n)))$ where n is a security parameter and $i = 1$ for the first garbled list and f for the second one.



c) $D_{T_E i} = \text{Dec}_{\text{pk}_i}(\text{Dec}_{\text{pk}_i}(\text{Dec}_{\text{pk}_i}(\text{Dec}_{\text{pk}_i}(T_{F_i}))))$ Then gets the only decrypted value valid with n ending zeros. \uparrow Garbled Table for E. This value is l_x^0 .

Some logic for extracting l_x^0 with T_{F_i} .

- 4) When integers are > 54 bits we need big-int, virtual integers, using several registers. It impacts drastically the performance.
- The exponents get huge and then generate very big cipher texts that complexity the workload and overload the network.

TP. 4.3

Lower classical keys can be broken now using brute force (ex for 80 bits). On the other hand 128 bits key are robust enough (2^{80} keys, very hard to brute force).