Privacy :

TP7:

Q1)

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

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

$P_1 \xrightarrow{c_0 = Enc_{pk}((1-b)m_0 + b r_0),\ c_1 = Enc_{pk}(b m_1 + (1-b) r_1)} P_2$

P_1: $\perp$ $\qquad$ P_2: m_b

P_1: $r_0, r_1 \in rand(\mathbb{Z}_q^{2})$, $m_0, m_1 \in \mathbb{Z}_q$

P_2: $b \in \{0;1\}$, $sk \in \mathbb{Z}_q \wedge pk = g^{sk} \% p$

$P_1 \xleftarrow{pk,\ Enc_{pk}(b)} P_2$

$x = Enc_{pk}(b)$

$y = Enc_{pk}(1-b)$

$= Enc_{pk}(1) \cdot x^{-1} \% p$

$c_0 = y^{m_0} \cdot x^{r_0} \% p$

$c_1 = x^{m_1} \cdot y^{r_1} \% p$

$P_1 \xrightarrow{c_0,\ c_1} P_2$

If $b=0$: $c = c_0$ else : $c = c_1$

$Dec_{sk}(c) = \log_g^*(g^{m_b}) = m_b$

Check pattern(m_b)

~~Easibl~~ Feasable because m_b is small enough!!

Q2) GARBLING PHASE

1)

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

MAP (=)

a	b	c
k_a^0	k_b^0	k_c^0
k_a^0	k_b^1	k_c^0
k_a^1	k_b^0	k_c^0
k_a^1	k_b^1	k_c^1

Inputs labels (a, b); Output label (c) → Garbling Keys

Garbled Table = shuffle $([E_{k_a^0,k_b^0}(k_c^0), E_{k_a^0,k_b^1}(k_c^0), E_{k_a^1,k_b^0}(k_c^0), E_{k_a^1,k_b^1}(k_c^1)])$

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

2) DATA TRANSFER PHASE(S)

⚠ Each bits generation has its own collection $(k_a^0, k_a^1, k_b^0, k_b^1, k_c^0, k_c^1)$!

Stream bits a — P_0 (Garbler) | P_1 (Executor)

For each bit i *

$a_i \in a$:

map: $i \to (k_a^0, k_a^1, k_b^0, k_b^1, k_c^0, k_c^1)$

compute garbled table T_i

map: $a_i \to k_a^{a_i}$

$\xrightarrow{k_a^{a_i},\, T_i}$

Stream bits b, sk, pk

Take next bit b_i

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

$\xleftarrow{pk,\, Enc_{pk}(b_i)}$

Compute challenges c_0, c_1 for OT where $m_0 = k_b^0$ $\wedge$ $m_1 = k_b^1$.

$\xrightarrow{c_0,\, c_1}$

$Dec_{sk}(c_{b_i}) = \log_g(g^{k_b^{b_i}}) = k_b^{b_i}$

$Dec_{k_a^{a_i}, k_b^{b_i}}(T_i) =$

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

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

$\xleftarrow{k_c^{a_i \wedge b_i}}$

map: $k_c^{a_i \wedge b_i} \to c_i = a_i \wedge b_i$

3)

A	B	C	D	A∧B	A∧C	A∧D	B∧C	B∧D	C∧D	E	F
0	0	0	0	0							
0	0	0	1	0							
0	0	1	0	0							
0	0	1	1	0							
0	1	0	0	0							
0	1	0	1	0							
0	1	1	0	0							
0	1	1	1	0							
1	0	0	0	0							
1	0	0	1	0							
1	0	1	0	0							
1	0	1	1	0							
1	1	0	0	1							
1	1	0	1	1							
1	1	1	0	1							
1	1	1	1	1							

TP.7.7

a) Alice for each pair (a_i, b_i) of her stream input create a mapping collection $(k_a^0, k_a^1, k_b^0, k_b^1, k_c^0, k_c^1, k_d^0, k_d^1, k_e^0, k_e^1, k_f^0, k_f^1)$ where each k_x^B means label for variable x equaling boolean B.

There are 2 garbled tables T_E^i and T_F^i containing $2^4 = 16$ entries resulting from the shuffle of the list of all possible combination (a_i, b_i, c_i, d_i) and each entry is encoded as

$$Enc_{k_a^{a_i}}\left(Enc_{k_b^{b_i}}\left(Enc_{k_c^{c_i}}\left(Enc_{k_d^{d_i}}\left(k_x^{g_x(a_i,b_i,c_i,d_i)} \,\|\, 0^n\right)\right)\right)\right)$$

where n is a security parameter and $x = e$ for the first garbled list and f for the second one.

b)

Bob	(OT)	Alice
Next pair (c_i, d_i)	$pk, Enc_{pk}(c_i), Enc_{pk}(d_i)$ $\longrightarrow$	Compute C
		$C = (Enc_{pk}((1-c_i)k_c^0 + c_i r), Enc_{pk}(c_i k_c^1 + (1-c_i) r'))$
$k_c^{c_i} = Dec_{sk}(C_{c_i})$	$\xleftarrow{C,\ D}$	$D = (Enc_{pk}((1-d_i)k_d^0 + d_i \cdot r''), Enc_{pk}(d_i \cdot k_d^1 + d_i \cdot r'''))$
$k_d^{d_i} = Dec_{sk}(D_{d_i})$		

d)

$DT_{Ei} = Dec_{k_a^{a_i}}(Dec_{k_b^{b_i}}(Dec_{k_c^{c_i}}(Dec_{k_d^{d_i}}(T_{Ei}))))$ then gets the only decrypted value valid with n ending zeros.

$\Downarrow$ Garbled Table for E. This value is $k_e^{e_i}$.

Same logic for extracting $k_f^{b_i}$ with T_{F_i}.

TP.7.3

~~4) → when integers are > 64 bits we need big-ints, virtual integers using several registers. It impacts drastically the performance.~~

~~→ The exponents get huge and then generate very big cypher texts that complexify the workload and overload the network.~~

Lower classical keys can be broken now using brute force (ex for 80 bits keys). On the other hand 128 bits key are robust enough (287474 billions times harder to brute force).