

# COL872

## Problem Set 2

Mallika Prabhakar (2019CS50440)  
Sayam Sethi (2019CS10399)  
Satwik Jain (2019CS10398)

### Contents

<b>1</b>	<b>Question 1</b>	<b>2</b>
<b>2</b>	<b>Question 2</b>	<b>3</b>
2.1	Question 2.1 . . . . .	3
2.2	Question 2.2 . . . . .	5
2.3	Question 2.3 . . . . .	6
<b>3</b>	<b>Question 3</b>	<b>8</b>

# 1 Question 1

## Question 1: Doubly Efficient Interactive Proofs

**Question.** Give a doubly-efficient protocol for the disjoint sets problem where the verifier runs in time  $\tilde{O}(n)$ , and the prover runs in time  $\tilde{O}(n^2)$ ,

*Proof.* We have two set of sets:  $S = \{S_i\}_{i \in [n]}$  and  $T = \{T_i\}_{i \in [n]}$ , and each  $S_i, T_i \subseteq [d]$ . Numbers till  $n$  can be represented using **log n** bits. So we will try to represent the problem as a polynomial in  $2 \log n$  bits (log n bits each for S and T). To do this, For set of sets S, for each  $j \in [d]$  which can exist in a set, we can create a Boolean algebra expression which takes as input the log n bits which create a number  $i$  and outputs 1 if  $j \in S_i$  and 0 otherwise. Let us name this expression as  $S^j$ . Similarly we can create boolean algebra expression  $T^j$ .

Now taking a **NAND** of these two expression  $(1 - S^j T^j)$ , for the input bits  $I = i_1 i_2 \dots i_{\log n}$  and  $K = k_1 k_2 \dots k_{\log n}$ , we will get an output of 0 if the element  $j$  is present in both the sets  $S_i$  and  $T_k$  and 1 otherwise. In this way we can check for one element. Now we can multiply all the NAND expressions and the resulting expression

$$f(I, K) = \prod_{j=1}^d (1 - S^j(I) T^j(K))$$

will return 0 if any one of the numbers in set  $[d]$  is present in both of  $S_i$  and  $T_k$  i.e.,  $S_i \cap T_k \neq \emptyset$  or the intersection of the sets  $S_i$  and  $T_k$  is empty and return 1 otherwise.

Now to get the answer,  $\Delta$ , we can simply sum up the outputs of  $f(I, K)$  over all possible values of  $I$  and  $K$ .

$$\Delta = \sum_{I, K \in \{0,1\}^{\log n}} f(I, K)$$

Now we can apply the sum check protocol over this. Suppose the value of  $\Delta$  is  $p$ . Now the prover will prove that  $\delta = p$  to the verifier. as the total number of variables is  $2 \log n$ , We will have  $2 \log n$  rounds in the sum check protocol. In each iteration of the protocol, the verifier has to evaluate a univariate polynomial of degree atmost  $d$  at each iteration. which will take  $O(d)$  iterations via the Horner's rule. So the expected running time of the Verifier is  $O(d * 2 \log n) = O(\log^3 n)$  as  $d = \log^2 n$  which can be represented as  $\tilde{O}(1)$  (as poly logarithmic factors are omitted).

Now we will find the running time of the prover. In the  $j^{th}$  sum, we have  $(1 + \deg(g))2^{v-j}$  terms where  $g$  is the univariate polynomial formed. As there are  $2 \log n$  rounds, for a  $v$  round protocol, the prover has to do  $\sum_{j=1}^{2 \log n} (1 + \deg(g))2^{v-j}$  evaluations. As each univariate polynomial will have a degree of atmost  $d$ , this comes out to be  $O(2^v d)$  evaluations. For this problem we have  $v = 2 \log n$ , So we will have  $O(n^2 \log^2 n)$  evaluations and we have know that each evaluation takes  $O(\log^2 n)$  time for a  $O(\log n)$  variate polynomial, so the total time for the prover will be  $O(n^2 \log^4 n)$  which is  $\tilde{O}(n^2)$ .  $\square$

## 2 Question 2

Zero-knowledge protocols for group-theoretic problems

### 2.1 Question 2.1

HVZK PoK for DDH

**Question.** Consider the following language:

$$\mathcal{L}_{DDH} = \{(g, h, u, v) : \exists a \in \mathbb{Z}_q \text{ s.t. } u = g^a, v = h^a\} \quad (1)$$

Construct an honest-verifier zero-knowledge proof-of-knowledge protocol for  $\mathcal{L}_{DDH}$ . The protocol must have perfect completeness, the knowledge error must be  $1/q$ , and it should satisfy the honest-verifier zero-knowledge property.

*Proof.* In an honest verifier setting, the zero-knowledge proof-of-knowledge can be defined as follows:

- Consider an honest verifier  $\mathcal{V}$  and a prover  $\mathcal{P}$ .
- **Common input:**  $g, u, h, v$  where all of them are group elements.
- **Prover's private input:**  $a$  s.t.  $u = g^a, v = h^a$
- **Claim to prove:** Prover knows  $a$

**The protocol:**

1. **P:** prover picks random  $t_1, t_2, t_3$  and sends  $x = g^{t_1}$ ,  $y = h^{t_2}$  and  $z = (g \cdot h)^{t_3}$  to the verifier
2. **V:** verifier sends uniformly random  $c_1, c_2, c_3$  with respect to the pairs  $(g, u), (h, v), (g \cdot h, u \cdot v)$
3. **P:** prover calculates  $w_1 = t_1 + c_1 \cdot a$ ,  $w_2 = t_2 + c_2 \cdot a$  and  $w_3 = t_3 + c_3 \cdot a$  and sends  $w_1, w_2, w_3$  to the verifier
4. **V:** verifier checks if  $x \cdot u^{c_1} = g^{w_1}$ ,  $y \cdot v^{c_2} = h^{w_2}$  and  $z \cdot (u \cdot v)^{c_3} = (g \cdot h)^{w_3}$

**Completeness:**

For every tuple  $(w_i, t_i, c_i)$  given prover knows the value  $a$

$$\begin{aligned} x \cdot u^{c_1} &= g^{t_1} \cdot g^{a \cdot c_1} = g^{t_1 + a \cdot c_1} = g^{w_1} \\ y \cdot v^{c_2} &= h^{t_2} \cdot h^{a \cdot c_2} = h^{t_2 + a \cdot c_2} = h^{w_2} \\ z \cdot (u \cdot v)^{c_3} &= (g \cdot h)^{t_3} \cdot (g \cdot h)^{a \cdot c_3} = (g \cdot h)^{t_3 + a \cdot c_3} = (g \cdot h)^{w_3} \end{aligned} \quad (2)$$

Hence an honest verifier is always convinced for an honest prover, implying perfect completeness.

**Knowledge error:**

We define the extractor's run as follows:

**Extractor for  $\mathcal{L}_{DDH}$** 

1. Extractor receives  $x, y, z$
2. It then sends uniformly random  $c_1, c_2, c_3$  to the prover
3. Then it receives  $w_1, w_2, w_3$  in response
4. Extractor rewinds upto after point 1
5. It sends uniformly random  $c'_1, c'_2, c'_3$  and receives  $w'_1, w'_2, w'_3$
6. The extractor outputs  $(w'_1 - w_1)/(c'_1 - c_1)$  which is equal to  $(w'_2 - w_2)/(c'_2 - c_2)$  and  $(w'_3 - w_3)/(c'_3 - c_3)$  assuming prover has the correct witness

Figure 1: Extractor that extract the witness from a prover of  $\mathcal{L}_{DDH}$

Let us assume that the prover succeeds in convincing the verifier with probability  $\epsilon$ , where  $\epsilon$  is non-negligible. Now, our extractor would fail if  $c'_i = c_i$  and this will happen with probability  $1/q$ . Therefore, the knowledge error will be at least  $1/q$ . For the case when  $c'_i \neq c_i$ , the extractor's success probability can be computed by,

$$\begin{aligned}
 & \Pr [\text{extractor succeeds} \mid c'_i \neq c_i] \\
 &= \Pr [\text{prover convinces verifier on receiving } c_i] \cdot \Pr [\text{prover convinces verifier on receiving } c'_i] \\
 &= \epsilon^2
 \end{aligned} \tag{3}$$

Therefore, the success probability of extractor can be given by:

$$\Pr [\text{extractor succeeds}] = \left(1 - \frac{1}{q}\right) \cdot \epsilon^2 = \epsilon^2 - \frac{\epsilon^2}{q} \tag{4}$$

Which is non-negligible if  $\epsilon$  is non-negligible. Hence it is a PoK.

**HVZK:**

We can define the honest-verifier zero-knowledge simulator as follows:

**HVZK Simulator for  $\mathcal{L}_{DDH}$** 

1. Simulator picks  $c_1, c_2, c_3 \leftarrow \mathbb{Z}_q$  and  $w_1, w_2, w_3 \leftarrow \mathbb{Z}_q$
2. it sets  $x = g_1^w / u_1^c$ ,  $y = h_2^w / v_2^c$ ,  $z = (g \cdot h)_3^w / (u \cdot v)_3^c$
3. It then outputs the tuples  $(x, c_1, w_1), (y, c_2, w_2)$  and  $(z, c_3, w_3)$

Figure 2: Honest-Verifier Zero Knowledge Simulator for  $\mathcal{L}_{DDH}$

This transcript is identical to the real-world transcript. Hence we have a simulator.  $\square$

## 2.2 Question 2.2

### ZK protocol for DDH

**Question.** Next, consider the complement of  $\mathcal{L}_{DDH}$ , denoted by  $\mathcal{L}_{nDDH}$  (defined below).

$$\mathcal{L}_{nDDH} = \{(g, h, u, v) : \forall a \in \mathbb{Z}_q \text{ either } u \neq g^a \text{ or } v \neq h^a\} \quad (5)$$

Construct a protocol for  $\mathcal{L}_{nDDH}$ . The protocol must have perfect completeness, constant soundness and it should satisfy zero-knowledge w.r.t. auxiliary information.

*Proof.* In a zero-knowledge with auxiliary information setting, the common input is the same as before. We have to show that the given prover can distinguish between the groups created by  $u \cdot v$  and  $g \cdot h$ . The protocol for  $\mathcal{L}_{nDDH}$  is defined as follows:

1. **V:** Verifier chooses a bit  $b$  and  $x \leftarrow \mathbb{Z}_q$  and a message  $m = g \cdot h$  when  $b = 0$  and  $m = u \cdot v$  when  $b = 1$ . Where  $u$  and  $v$  are defined as  $g^x$  and  $h^x$  respectively.
2. **V:** It sends a zero-knowledge proof to the prover that it has chosen a bit  $b$  and  $x$ . This is possible since verifying  $\exists b : \exists x : m_b^x$  is possible in polynomial time ( $O(1)$  time) and zero-knowledge reductions exist to NP-complete problem. This is taken as a preventive measure towards avoiding the verifier from using the auxiliary information.
3. **V:** Verifier then sends  $y = m_b^x$  to the prover
4. **P:** Prover then sends a bit  $b'$  by distinguishing if  $m_b^x$  was a group element of  $m_0$  or  $m_1$

#### Completeness:

Since prover is unbounded, prover can find all the group elements for  $u \cdot v$  and  $g \cdot h$ . It is a yes instance for  $\mathcal{L}_{nDDH}$ ; therefore, both groups will be disjoint, and hence it can identify and send the appropriate bit  $b' = b$ . Hence it has perfect completeness.

#### Soundness:

In case the prover is not able to identify the bit  $b$  for the message  $y$ , it can output the correct answer bit with the probability  $\frac{1}{2} + \epsilon$  where  $\epsilon$  is negligible information the prover can extract from the zero-knowledge proof provided by the verifier.

#### Simulator for ZK with auxiliary info:

##### Simulator $\mathcal{S}$

1. Verifier sends a zk proof of choosing a bit  $b$  and  $x$  and message  $y = m_b^x$  to the simulator
2. Simulator checks the zero-knowledge proof and sends a random bit  $b'$
3. If the verifier accepts the proof, its work is done, else simulator rewinds to the start and repeat the interaction
4. This repeats for atmost  $n$  iterations until the verifier accepts simulator's guess or simulator outputs  $\perp$

Figure 3: Simulator for zero-knowledge protocol with auxiliary information

Probability of success of simulator (let probability of verifier choosing the bit  $b = 0$  be  $p$ ):

$$\begin{aligned} P_{succ} &= \Pr [b' = 0 | b = 0] + \Pr [b' = 1 | b = 1] \\ &= \frac{1}{2}p + \frac{1}{2}(1 - p) = \frac{1}{2} \end{aligned} \quad (6)$$

After repeating (rewinding) for  $n$  iterations,

$$\begin{aligned} P_{succ} &= 1 - \Pr [\text{simulator outputs } \perp] \\ &= 1 - \frac{1}{2^n} \end{aligned} \quad (7)$$

The transcript outputted by the simulator looks identical to a valid prover's transcript. Hence we have a protocol for zero knowledge with auxiliary information.  $\square$

### 2.3 Question 2.3

HVZK PoK for  $k$ -out-of- $t$  DDH

**Question.** Let  $k, t$  be two integers such that  $k \leq t$ . Consider the following language:

$$\mathcal{L}_{DDH,k,t} = \left\{ (g, h, u_1, v_1, \dots, u_t, v_t) : \begin{array}{l} \exists \text{ indices } i_1, i_2, \dots, i_k \in [t], \\ \exists \text{ exponents } a_1, a_2, \dots, a_k \in \mathbb{Z}_q, \\ \text{s.t. } \forall j \leq k, u_{i_j} = g^{a_j} \text{ and } v_{i_j} = h^{a_j} \end{array} \right\} \quad (8)$$

*Proof.* We propose the following protocol:

#### HVZK Protocol

1. Prover samples  $\alpha \leftarrow \mathbb{Z}_q$ ,  $\pi \leftarrow \text{Perm}([t], [t])$
2. Prover computes  $g' = g^\alpha$ ,  $h' = h^\alpha$ ,  $\mathcal{U} = \pi(\{u_i^\alpha\}_{i \in [t]})$ ,  $\mathcal{V} = \pi(\{v_i^\alpha\}_{i \in [t]})$ ,  $c = \text{Commit}((\pi, \alpha); r)$  and sends them to the verifier
3. Verifier picks a bit  $b \leftarrow \{0, 1\}$  and sends it to the prover
4. If  $b = 0$ , prover sends  $r$  to the verifier and the verifier then checks if the prover did indeed use the committed  $\pi$  and  $\alpha$  for transforming the input. The interaction ends.
5. If  $b = 1$ , prover sends  $\pi(i_1, i_2, \dots, i_k)$
6. For each  $j \in \{1, 2, \dots, k\}$ , prover sends ZKP for  $\mathcal{L}_{DDH}(g', h', \mathcal{U}_{\pi(j)}, \mathcal{V}_{\pi(j)})$  using the protocol given in Question 2.1

Figure 4: HVZK Protocol for  $\mathcal{L}_{DDH,k,t}$

**Completeness:** This protocol has perfect completeness since the interaction by the prover having a witness to an instance in  $\mathcal{L}_{DDH,k,t}$  will be accepted by the verifier.

**Proof of Knowledge:** We define the extractor as follows:

1. Extractor receives  $g', h', \mathcal{U}, \mathcal{V}$  from the prover

2. Extractor sends  $b = 0$  and extracts  $\pi, \alpha$
3. Extractor rewinds the prover and now sends  $b = 1$  and receives  $\pi(i_1, \dots, i_k)$
4. For each  $j \in \{i_1, \dots, i_k\}$ , extractors follows the steps followed by the extractor in Question 2.1 to obtain all  $a_1, a_2, \dots, a_k$

Let us assume that the prover's success probability is  $\epsilon_0$  when  $b = 0$  and  $\epsilon_1$  when  $b = 1$ . Since prover succeeds in convincing the verifier,  $\epsilon_0/2 + \epsilon_1/2 = \epsilon$  is non-negligible. Additionally, since the prover can convince any verifier, both  $\epsilon_0$  and  $\epsilon_1$  should also be non-negligible. Therefore, the extractor's success probability is given as,

$$\begin{aligned}
\Pr[\text{extractor succeeds}] &= \Pr[\text{prover succeeds} \mid b = 0] \cdot \Pr[\text{extractor succeeds} \mid b = 1] \\
&= \epsilon_0 \cdot \Pr[\text{extractor succeeds} \mid b = 1] \\
&= \epsilon_0 \cdot \prod_{j \in \{i_1, \dots, i_k\}} \Pr[\text{extractor for } j \text{ succeeds}] \\
&= \epsilon_0 \cdot \prod_{j \in \{i_1, \dots, i_k\}} \left( \epsilon_{\mathcal{L}_{DDH}}^2 - \frac{\epsilon_{\mathcal{L}_{DDH}}^2}{q} \right) \\
&= \epsilon_0 \cdot \left( \epsilon_{\mathcal{L}_{DDH}}^2 - \frac{\epsilon_{\mathcal{L}_{DDH}}^2}{q} \right)^k \\
&\geq \epsilon_0 \cdot \epsilon_{\mathcal{L}_{DDH}}^{2k} - k \cdot \epsilon_0 \cdot \frac{\epsilon_{\mathcal{L}_{DDH}}^2}{q}
\end{aligned} \tag{9}$$

Therefore, the success probability is non-negligible if  $\epsilon_0, \epsilon_{\mathcal{L}_{DDH}}$  are non-negligible and  $k$  is a constant. Thus, it is a PoK.

**Honest-Verifier Zero-Knowledge:** We define the simulator as follows:

1. Simulator picks a random  $\alpha$  and a random  $\pi$  and sends  $g', h', \mathcal{U}, \text{cal}V$  to the verifier. It also samples a random set  $\{p_1, p_2, \dots, p_k\}$ .
2. If the simulator receives  $b = 0$ , it sends the opening to the verifier and outputs the transcript.
3. Else, it sends  $\pi(p_1, p_2, \dots, p_k)$  to the verifier.
4. The simulator then simulates the proof using the simulator defined in Question 2.1 for each  $j \in \{p_1, p_2, \dots, p_k\}$  and outputs the combined transcript.

This transcript is identical to the real world transcript since  $\pi(p_1, p_2, \dots, p_k)$  is indistinguishable from  $\pi(i_1, i_2, \dots, i_k)$  and the remaining interaction is either identical to the prover or to the simulator of Question 2.1. Hence, it is a HVZK protocol.  $\square$

### 3 Question 3

#### Question 3: Impossibility of two-round zero-knowledge protocols with auxiliary information

**Question.** Prove that two-round protocols (where the verifier sends the first message and prover sends the second message) cannot satisfy the zero-knowledge property in the presence of auxiliary information. Describe the argument in full detail.

*Proof.* (Proof Idea: We use the auxiliary input as the string that is sent to the prover. Thus, the simulator (even if it is non-black box) cannot "generate" a valid response unless  $L$  itself is in BPP)

The default interaction between the prover and the verifier is defined as:

1.  $V_1(x, r) = m_1$  is sent to the prover
2.  $P_2(x, m_1) = m_2$  is sent to the verifier
3.  $V_3(m_1, m_2) = m_3 \in \{0, 1\}$  is the result of the interaction (rejected or accepted)

Now consider the following verifier  $V^* = (V_1^*, V_3^*)$  that takes in auxiliary information  $z$  and interacts with the prover as follows:

$$V_1^*(z)(x, r) = z, V_3^*(z)(x, r, m_2) = V_3(m_1, m_2) \quad (10)$$

Since by our assumption, the given protocol is two-round zero-knowledge in the presence of auxiliary information, we will have a simulator for  $V^*$ . Let any such simulator be  $S^*$ . We now propose the following algorithm  $\mathcal{A}$  for checking if  $x \in \mathcal{L}_{yes}$ ,

#### Algorithm $\mathcal{A}$

1. Sample a random  $r$  and compute  $m_1 = V_1(x, r)$
2. Obtain the transcript  $(m_1, m_2, m_3)$  on running  $S^*(x, m_1)$ .
3. If  $S^*$  fails to simulate, output 0. Else output  $m_3$ .

Figure 5: Algorithm for checking  $x \in \mathcal{L}_{yes}$

We first note that  $\mathcal{A}$  is a ppt algorithm since it uses  $V_1, S^*$  in sequence which are both ppt algorithms.

Now, consider the probability of  $\mathcal{A}$  in deciding  $x \in \mathcal{L}_{yes}$  when  $x$  is a yes instance (assuming that completeness of the protocol is  $c$  and soundness error is  $s$ ),

$$\begin{aligned} \Pr[\mathcal{A} \text{ outputs } 1] &= \Pr[S^* \text{ does not output } \perp] \cdot \Pr[m_3 = 1 | x \in \mathcal{L}_{yes}] \\ &= (1 - \mu(n)) \cdot \Pr[m_3 = 1 | x \in \mathcal{L}_{yes}] \\ &= (1 - \mu(n)) \cdot c \end{aligned} \quad (11)$$

Therefore, we can see that  $\mathcal{A}$  can decide  $x \in \mathcal{L}_{yes}$  with probability  $> 2/3$  in ppt.

Again, consider the probability of  $\mathcal{A}$  in deciding  $x \in \mathcal{L}_{no}$  when  $x$  is a no instance,

$$\begin{aligned} \Pr[\mathcal{A} \text{ outputs } 0] &= \Pr[S^* \text{ does not output } \perp] \cdot \Pr[m_3 = 0 | x \in \mathcal{L}_{no}] \\ &= (1 - \mu(n)) \cdot \Pr[m_3 = 0 | x \in \mathcal{L}_{no}] + \mu(n) \\ &= (1 - \mu(n)) \cdot (1 - s) + \mu(n) \end{aligned} \quad (12)$$



Therefore, we can see that  $\mathcal{A}$  can decide  $x \in \mathcal{L}_{no}$  with probability  $> 2/3$  in ppt.  
Therefore,  $\mathcal{L}$  is in BPP. □