

CS 171: Problem Set 10

Due Date: April 25, 2024 at 8.59pm via Gradescope

1 Course Evaluation (Extra Credit: 2 Points)

Complete your course evaluation for this course. You can write as much or as little as you want. Include a screenshot of the submission receipt when you submit this assignment to Gradescope to prove that you've finished your evaluation.

2 Proof of Decryption (10 Points)

We will construct a zero-knowledge proof system for DDH triples. This can be used to prove that a given El Gamal ciphertext was decrypted correctly without revealing the secret decryption key.

Let $\text{pp} = (\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$ be a group in which DDH is hard. Let \mathcal{L} be the language of DDH triples for this group:

$$\mathcal{L} = \{(\text{pp}, g^a, g^b, g^c) : c = a \cdot b \pmod{q}\}$$

Given an instance $x = (\text{pp}, g^a, g^b, g^c) \in \mathcal{L}$, let the corresponding witness be $w = b$. The witness provides a simple way to verify that $x \in \mathcal{L}$:

$$R(x, w) = \begin{cases} 1 & \text{if } g^w = g^b \text{ and } (g^a)^w = g^c \\ 0 & \text{else} \end{cases}$$

We can also prove that $x \in \mathcal{L}$ without revealing the witness to the verifier. To do so, we will construct a zero-knowledge proof below.

A Zero-Knowledge Protocol for \mathcal{L} :

- **Inputs:** The prover P takes inputs $(1^\lambda, x, w)$ and the verifier V takes inputs $(1^\lambda, x)$. $x = (\text{pp}, g^a, g^b, g^c)$, and $w \in \mathbb{Z}_q$.
- P samples $x \leftarrow \mathbb{Z}_q$, computes $t = a \cdot x \pmod{q}$, and sends (g^x, g^t) to V .
- V samples $y \leftarrow \mathbb{Z}_q$ and sends y to P .
- P computes $z = w \cdot y + x$ and sends z to V .
- Final step: TBD

Questions:

1. Fill in the final step to show how V should verify the proof (g^x, g^t, y, z) .
2. Show that this proof system satisfies completeness and soundness.
3. Show that this proof system satisfies honest-verifier zero-knowledge.

The definitions of completeness, soundness, and honest-verifier zero-knowledge are given in Discussion 11.

3 Hiding and Binding For KZG Commitments (15 Points)

In discussion 11, we showed that the basic KZG commitment protocol is not hiding because the `Commit` function is deterministic. In section 3.1 below, we give a modified version of the scheme in which the `Commit` function is randomized.

Question: Prove that the commitment scheme given in section 3.1 satisfies the notions of hiding and polynomial binding given in section 3.2, assuming that the d -discrete log problem is hard.

3.1 A Randomized Polynomial Commitment Scheme

1. `Gen`(1^n):

- (a) Let d be polynomial in n .
- (b) Set up a bilinear map by sampling

$$\text{pp} = (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathcal{G}(1^n)$$

- (c) Sample $h \leftarrow \mathbb{G}$ and $\tau \leftarrow \mathbb{Z}_q^*$.
- (d) Finally, output

$$\text{params} = \left(\text{pp}, g^\tau, g^{(\tau^2)}, \dots, g^{(\tau^d)}, h, h^\tau, h^{(\tau^2)}, \dots, h^{(\tau^d)} \right)$$

2. `Commit`(`params`, f):

- (a) Let f be a polynomial $\in \mathbb{Z}_q[X]$ of degree $\leq d$:

$$f(X) = \sum_{i=0}^d \alpha_i \cdot X^i$$

where every $\alpha_i \in \mathbb{Z}_q$.

- (b) Sample a polynomial $r \in \mathbb{Z}_q[X]$ of degree $\leq d$ uniformly at random. In other words, sample $\beta_0, \dots, \beta_d \leftarrow \mathbb{Z}_q$ independently and uniformly at random, and let

$$r(X) = \sum_{i=0}^d \beta_i \cdot X^i$$

- (c) Compute and output the commitment:

$$\begin{aligned} \text{com} &= \prod_{i=0}^d \left(g^{(\tau^i)} \right)^{\beta_i} \cdot \prod_{i=0}^d \left(h^{(\tau^i)} \right)^{\alpha_i} \\ &= g^{r(\tau)} \cdot h^{f(\tau)} \end{aligned}$$

Note: We also define `Commit`(`params`, f ; r) to take the random polynomial r as input, rather than sampling r internally.

3.2 Definitions

Hiding basically says that $\text{Commit}(f, \text{params})$ doesn't reveal any information about f . The definition of hiding resembles the definition of CPA security.

Definition 3.1 (Hiding)

Hiding-Game(n, \mathcal{A}):

1. The challenger samples $\text{params} \leftarrow \text{Gen}(1^n)$ and sends params to the adversary \mathcal{A} .
2. \mathcal{A} outputs two polynomials $f_0, f_1 \in \mathbb{Z}_q[X]$ of degree $\leq d$.
3. The challenger samples $b \leftarrow \{0, 1\}$ and computes: $\text{com}^* = \text{Commit}(\text{params}, f_b)$. They send com^* to \mathcal{A} .
4. \mathcal{A} outputs a guess b' for b . The output of the game is 1 if $b' = b$ and 0 otherwise.

The commitment scheme is **hiding** if for any PPT adversary \mathcal{A} ,

$$\Pr[\text{Hiding-Game}(n, \mathcal{A}) \rightarrow 1] \leq \frac{1}{2} + \text{negl}(n)$$

Next, we'll consider a notion called polynomial binding, which says that the adversary cannot find two inputs to Commit that produce the same commitment. This resembles the definition of collision-resistance.

Definition 3.2 (Polynomial Binding)

Polynomial-Binding-Game(n, \mathcal{A}):

1. The challenger samples $\text{params} \leftarrow \text{Gen}(1^n)$ and sends params to the adversary \mathcal{A} .
2. \mathcal{A} outputs two pairs (f_0, r_0) and (f_1, r_1) , where f_0, r_0, f_1, r_1 are polynomials $\in \mathbb{Z}_q[X]$ of degree $\leq d$.
3. The output of the game is 1 if $f_0 \neq f_1$, and

$$\text{Commit}(\text{params}, f_0; r_0) = \text{Commit}(\text{params}, f_1; r_1)$$

Otherwise, the output of the game is 0.

The commitment scheme satisfies **polynomial binding** if

$$\Pr[\text{Polynomial-Binding-Game}(n, \mathcal{A}) \rightarrow 1] \leq \text{negl}(n)$$

Finally, we will prove polynomial binding using the hardness of the following problem.

Definition 3.3 (A Variant of Discrete Log)

d -Discrete-Log(n, \mathcal{A}):

1. Let d be polynomial in n .

2. The challenger samples $\text{pp} = (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathcal{G}(1^n)$ as well as $\tau \leftarrow \mathbb{Z}_q$. Then they send the adversary: $(\text{pp}, g^\tau, g^{(\tau^2)}, \dots, g^{(\tau^d)})$
3. The adversary \mathcal{A} outputs a guess τ' for τ . The output of the game is 1 if $\tau' = \tau$ and 0 otherwise.

The d -discrete-log problem is hard if for any PPT adversary \mathcal{A} ,

$$\Pr[d\text{-Discrete-Log}(n, \mathcal{A}) \rightarrow 1] \leq \text{negl}(n)$$

Note that if the d -discrete-log problem is hard, then in addition, the regular discrete log problem is hard for \mathbb{G} .