# CS 171: Problem Set 10

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

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

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

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

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

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 $(1^\lambda, x)$. $x = (\mathsf{pp}, g^{\mathsf{a}}, g^{\mathsf{b}}, g^{\mathsf{c}})$, and $w \in \mathbb{Z}_q$.
- $P$ samples $\mathsf{x} \leftarrow \mathbb{Z}_q$, computes $g^{\mathsf{t}} = (g^{\mathsf{a}})^{\mathsf{x}}$, and sends $(g^{\mathsf{x}}, g^{\mathsf{t}})$ to $V$. Note that $\mathsf{t} = \mathsf{a} \cdot \mathsf{x} \mod q$.
- $V$ samples $\mathsf{y} \leftarrow \mathbb{Z}_q$ and sends $\mathsf{y}$ to $P$.
- $P$ computes $\mathsf{z} = w \cdot \mathsf{y} + \mathsf{x}$ and sends $\mathsf{z}$ to $V$.
- V checks that:
  1. $g^{\mathsf{z}} = (g^{\mathsf{b}})^{\mathsf{y}} \cdot g^{\mathsf{x}}$, and
  2. $(g^{\mathsf{a}})^{\mathsf{z}} = (g^{\mathsf{c}})^{\mathsf{y}} \cdot g^{\mathsf{t}}$

  If both checks pass, then the verifier accepts the proof. Otherwise, they reject.

**Questions:**

1. Show that this proof system satisfies completeness and soundness.
2. 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.

# 2 Hiding and Binding For KZG Commitments (15 Points)

In discussion 11, we showed that the basic KZG commitment protocol is not hiding because the $\mathsf{Commit}$ function is deterministic. In section 2.1 below, we give a modified version of the scheme in which the $\mathsf{Commit}$ function is randomized.

**Question:** Prove that the commitment scheme given in section 2.1 satisfies the notions of hiding and polynomial binding given in section 2.2, assuming that the $d$-discrete log problem is hard.

## 2.1 A Randomized Polynomial Commitment Scheme

1. $\mathsf{Gen}(1^n)$:
   (a) Let $d$ be polynomial in $n$.
   (b) Set up a bilinear map by sampling
   $$\mathsf{pp} = (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathcal{G}(1^n)$$
   (c) Sample $h \leftarrow \mathbb{G}$ and $\tau \leftarrow \mathbb{Z}_q^*$.
   (d) Finally, output
   $$\mathsf{params} = \left(\mathsf{pp}, g^{\tau}, g^{(\tau^2)}, \ldots, g^{(\tau^d)}, h, h^{\tau}, h^{(\tau^2)}, \ldots, h^{(\tau^d)}\right)$$
2. $\mathsf{Commit}(\mathsf{params}, f)$:
   (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 $\beta_0, \ldots, \beta_d \leftarrow \mathbb{Z}_q$ independently and uniformly at random, and let
   $$r(X) = \sum_{i=0}^{d} \beta_i \cdot X^i$$
   (c) Compute and output the commitment:
   $$\begin{aligned}\mathsf{com} &= \prod_{i=0}^{d} \left(g^{(\tau^i)}\right)^{\beta_i} \cdot \prod_{i=0}^{d} \left(h^{(\tau^i)}\right)^{\alpha_i} \\ &= g^{r(\tau)} \cdot h^{f(\tau)}\end{aligned}$$

   *Note*: We also define $\mathsf{Commit}(\mathsf{params}, f; r)$ to take the random polynomial $r$ as input, rather than sampling $r$ internally.

## 2.2 Definitions

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

**Definition 2.1 (Hiding)**

$\underline{\mathsf{Hiding\text{-}Game}(n, \mathcal{A})}$*:*

1. *The challenger samples* $\mathsf{params} \leftarrow \mathsf{Gen}(1^n)$ *and sends* $\mathsf{params}$ *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:* $\mathsf{com}^* = \mathsf{Commit}(\mathsf{params}, f_b)$. *They send* $\mathsf{com}^*$ *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* $\mathcal{A}$*,*

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

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

**Definition 2.2 (Polynomial Binding)**

$\underline{\mathsf{Polynomial\text{-}Binding\text{-}Game}(n, \mathcal{A})}$*:*

1. *The challenger samples* $\mathsf{params} \leftarrow \mathsf{Gen}(1^n)$ *and sends* $\mathsf{params}$ *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*

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

*Otherwise, the output of the game is* $0$.

*The commitment scheme satisfies **polynomial binding** if*

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

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

**Definition 2.3 (A Variant of Discrete Log)**
$\underline{d\text{-}\mathsf{Discrete\text{-}Log}(n, \mathcal{A})}$*:*

1. *Let* $d$ *be polynomial in* $n$.

2. *The challenger samples* $\mathsf{pp} = (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathcal{G}(1^n)$ *as well as* $\tau \leftarrow \mathbb{Z}_q$*. Then they send the adversary:* $\left(\mathsf{pp}, g^{\tau}, g^{(\tau^2)}, \ldots, g^{(\tau^d)}\right)$

3. *The adversary* $\mathcal{A}$ *outputs a guess* $\tau'$ *for* $\tau$*. 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* $\mathcal{A}$*,*

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

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

## 3 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.