CS 6501: Advanced Topics in Cryptography **Spring 2019**

Problem Set 2

Due: February 22, 2019 at 5pm (submit via Gradescope) **Instructor:** David Wu

Instructions: You **must** typeset your solution in LaTeX using the provided template:

https://www.cs.virginia.edu/dwu4/courses/sp19/static/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use course code **9YD875** to sign up. Note that Gradescope requires that the solution to each problem starts on a **new page**.

Problem 1: Commitment Schemes [20 points]. In this problem, we will study the notion of a cryptographic commitment scheme At a high level, a commitment scheme enables a user to commit to a message m in a way that the commitment hides m. Later on, the user can open the commitment and convince another party that the committed message is indeed m. The commitment scheme is binding if the user cannot open the commitment to two distinct values. In this problem, we will work with the following definition:

Definition (Non-Interactive Commitment). A non-interactive commitment scheme with message space $\mathcal{M}$ consists of three algorithms $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ with the following properties:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input the security parameter λ, the setup algorithm outputs public parameters pp.
- $\mathsf{Commit}(\mathsf{pp}, m) \to (c, r)$: On input the public parameters pp, a message $m \in \mathcal{M}$, the commit algorithm outputs a commitment c and an opening r.
- $\mathsf{Verify}(\mathsf{pp}, m, c, r) \to b$: On input the public parameters pp, a message $m \in \mathcal{M}$, a commitment c, and an opening r, the verification algorithm outputs a bit $b \in \{0,1\}$.

The commitment scheme should satisfy the following properties:

- **Correctness:** For all messages $m \in \mathcal{M}$,

$$\Pr[\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (c,r) \leftarrow \mathsf{Commit}(\mathsf{pp}, m) : \mathsf{Verify}(\mathsf{pp}, m, c, r) = 1] = 1.$$

- **Perfect Hiding:** For all messages $m_0, m_1 \in \mathcal{M}$,

$$\left\{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (c,r) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_0) : c\right\} \equiv \left\{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (c,r) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_1) : c\right\}$$

- **Computational Binding:** For all efficient adversaries $\mathcal{A}$, if we sample $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(c, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$,

$$\Pr\left[m_0 \neq m_1 \text{ and } \mathsf{Verify}(\mathsf{pp}, m_0, c, r_0) = 1 = \mathsf{Verify}(\mathsf{pp}, m_1, c, r_1)\right] = \mathrm{negl}(\lambda).$$

Consider the following commitment scheme $\Pi_{\mathsf{com}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: First, sample a group $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$, which outputs the description of a group $\mathbb{G}$ of prime order p and generator g. Then, sample $x \xleftarrow{R} \mathbb{Z}_p$, compute $h \leftarrow g^x$, and output the public parameters $\mathsf{pp} = (\mathbb{G}, p, g, h)$. The message space for the commitment scheme is $\mathbb{Z}_p$.
- $\mathsf{Commit}(\mathsf{pp}, m) \to (c, r)$: Parse $\mathsf{pp} = (\mathbb{G}, p, g, h)$. Sample $r \xleftarrow{R} \mathbb{Z}_p$ and compute $c \leftarrow g^r h^m$. Output (c, r).
- $\mathsf{Verify}(\mathsf{pp}, m, c, r) \to b$. Parse $\mathsf{pp} = (\mathbb{G}, p, g, h)$. Output 1 if $c = g^r h^m$ and 0 otherwise.

Now, show the following:

(a) Show that Π_{com} is correct.

(b) Show that Π_{com} is perfectly hiding.

(c) Show that under the discrete log assumption with respect to $\mathsf{GroupGen}$, Π_{com} is computationally binding.

(d) Show how to extend the above commitment scheme to a *vector* commitment scheme $(\mathsf{Setup}_n, \mathsf{Commit}_n, \mathsf{Verify}_n)$ where the message space is $\mathcal{M} = \mathbb{Z}_p^n$ for some $n = \mathrm{poly}(\lambda)$. Your new commitment algorithm Commit_n should have the same structure as the commit algorithm in Π_{com}. Namely, $\mathsf{Commit}_n(\mathsf{pp}, m)$ should take as input the public parameters output by Setup_n and a vector $m \in \mathbb{Z}_p^n$, and output a commitment $c \in \mathbb{G}$ and opening $r \in \mathbb{Z}_p$. In your description, you should specify how the Setup_n, Commit_n, and Verify_n algorithms work. Finally, show that your vector commitment scheme is correct, perfectly hiding, and computationally binding.

(e) Suppose you are given a hash function $H\colon \mathbb{Z}_p \to \mathbb{G}$ (modeled as a random oracle). Show how you can modify your vector commitment scheme from Part (d) so that the public parameters only need to consist of the group description $(\mathbb{G}, p, g)$ and H (and nothing else). You only need to prove correctness of your modified scheme. Note that your modified scheme should still be perfectly hiding and computationally hiding, and a proof of such should exist (even though we are not asking you to provide it).

Problem 2: Constructing PRFs from DDH in the Random Oracle Model [20 points].

(a) For a PRF $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, and an adversary $\mathcal{A}$, we define two experiments, Experiment 0 and Experiment 1. For $b \in \{0,1\}$, we define:

> **Experiment b:**
> (1) At the start of the experiment, the challenger samples a random key $k \xleftarrow{R} \mathcal{K}$.
> (2) The adversary submits a *challenge query* $x^* \in \mathcal{X}$ to the challenger.
> (3) If $b = 0$, the challenger replies with $y^* = F(k, x^*)$. If $b = 1$, the challenger replies with $y^* \xleftarrow{R} \mathcal{Y}$.
> (4) The adversary can then makes any number of (adaptive) *evaluation queries*, each consisting of a value $x \in \mathcal{X}$, where $x \neq x^*$.
> (5) For each evaluation query $x \neq x^*$, the challenger computes $y = F(k, x)$ and gives y to the adversary.
> (6) At the end of the experiment, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

For $b \in \{0,1\}$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define $\mathcal{A}$'s advantage in the *single-challenge* security game as

$$\mathsf{SC\text{-}PRFAdv}[\mathcal{A}, F] := |\Pr[W_0] - \Pr[W_1]|$$

We say that a F is single-challenge secure if for all efficient efficient adversaries $\mathcal{A}$, the value $\mathsf{SC\text{-}PRFAdv}[\mathcal{A}, F]$ is negligible.

Show that if F is single-challenge secure, then F is a secure PRF. In particular, show that if there is a PRF adversary $\mathcal{A}$, then there is a single-challenge PRF adversary $\mathcal{B}$ such that

$$\mathsf{PRFAdv}[\mathcal{A}, F] \leq Q \cdot \mathsf{SC\text{-}PRFAdv}[\mathcal{B}, F],$$

where Q is the number of queries $\mathcal{A}$ makes in the PRF security game. [**Hint:** Try using a similar structure as that used in the security proof of the Blum-Micali construction from lecture.]

(b) Let $\mathbb{G}$ be a group of prime order p, and let $H : \{0,1\}^n \to \mathbb{G}$ be a hash function that is modeled as a random oracle. Define a candidate PRF $F : \mathbb{Z}_p \times \{0,1\}^n \to \mathbb{G}$ as follows:

$$F(k, x) := H(x)^k.$$

Show that if the decisional Diffie-Hellman (DDH) assumption holds in $\mathbb{G}$ and we model H as a random oracle, then F is single-challenge secure. In particular, show that if there is a single-challenge PRF adversary $\mathcal{A}$, then there exists a DDH distinguisher $\mathcal{B}$ such that

$$\mathsf{SC\text{-}PRFAdv}[\mathcal{A}, F] \leq Q_{\mathrm{RO}} \cdot \mathsf{DDHAdv}[\mathcal{B}, \mathbb{G}],$$

where Q_{RO} is a bound on the number of random oracle queries $\mathcal{A}$ makes in the single-challenge PRF security game. Combined with the result from Part (a), this shows that F is a secure PRF under the DDH assumption. More precisely, if there exists a PRF adversary $\mathcal{A}$ (that makes Q queries and Q_{RO} random oracle queries), then there is a DDH distinguisher $\mathcal{B}$ such that

$$\mathsf{DDHAdv}[\mathcal{B}, \mathbb{G}] \geq \frac{1}{Q} \cdot \frac{1}{Q_{\mathrm{RO}}} \cdot \mathsf{PRFAdv}[\mathcal{A}, F].$$

We often refer to the $1/(Q \cdot Q_{\mathrm{RO}})$ factor as the "security loss" in the reduction. Intuitively, this statement says that the more queries $\mathcal{A}$ makes in the PRF security game, the smaller the distinguishing advantage of the DDH adversary $\mathcal{B}$. We say that such security reductions are *non-tight*.

(c) Give a *tight* reduction of the security of F to the DDH assumption in the random oracle model. In particular, show that if there is a PRF adversary $\mathcal{A}$ for F, then there exists a DDH distinguisher $\mathcal{B}$ such that

$$\mathsf{DDHAdv}[\mathcal{B}, \mathbb{G}] \geq \mathsf{PRFAdv}[\mathcal{A}, F] - \mathrm{negl}(\lambda).$$

[**Hint:** Use the random self-reducibility of DDH.]

Problem 3: Coppersmith Attacks on RSA [15 points]. In this problem, we will explore what are known as "Coppersmith" attacks on RSA-style cryptosystems. As you will see, these attacks are very powerful and very general. We will use the following theorem:

Theorem (Coppersmith, Howgrave-Graham, May). Let N be an integer of unknown factorization. Let p be a divisor of N such that $p \geq N^{\beta}$ for some constant $0 < \beta \leq 1$. Let $f \in \mathbb{Z}_N[x]$ be a monic polynomial of degree δ. Then there is an efficient algorithm that outputs all integers x such that

$$f(x) = 0 \bmod p \qquad \text{and} \qquad |x| \leq N^{\beta^2/\delta}.$$

In the statement of the theorem, when we write $f \in \mathbb{Z}_N[x]$, we mean that f is a polynomial in an indeterminate x with coefficients in $\mathbb{Z}_N$. A *monic* polynomial is one whose leading coefficient is 1.

When $N = pq$ is an RSA modulus (where p, q are identically-distributed primes), the interesting instantiations of the theorem have either $\beta = 1/2$ (i.e., we are looking for solutions modulo a prime factor of N) or $\beta = 1$ (i.e., we are looking for small solutions modulo N).

For this problem, let N be an RSA modulus with $\gcd(\phi(N), 3) = 1$ and let $F_{\text{RSA}}(m) := m^3 \pmod N$ be the RSA one-way function.

(a) Let $n = \lceil \log_2 N \rceil$. Show that you can factor an RSA modulus $N = pq$ if you are given:

- the low-order $\lceil n/3 \rceil$ bits of p,
- the high-order $\lceil n/3 \rceil$ bits of p, or
- the high-end $\lceil n/6 \rceil$ bits of p *and* the low-end $\lceil n/6 \rceil$ bits of p.

(b) In the dark ages of cryptography, people would encrypt messages directly using F_{RSA}. That is, they would encrypt an arbitrary bitstring $m \in \{0,1\}^{\lfloor \log_2 N \rfloor / 5}$ by

- setting $M \leftarrow 2^{\ell} + m$ for some integer ℓ to make $N/2 \leq M < N$, and
- computing the ciphertext as $c \leftarrow F_{\text{RSA}}(M)$.

Note that the first step corresponds to padding the message M by prepending it with a binary string "$10000\cdots000$."

Show that this public-key encryption scheme (known as "textbook RSA") is very broken. In particular, give an efficient algorithm that takes as input (N, c) and outputs m.

(c) To avoid the problem with the padding scheme above, your friend proposes instead encrypting the short message $m \in \{0,1\}^{\lfloor \log_2 N \rfloor / 5}$ by setting $M \leftarrow (m \| m \| m \| m \| m) \in \{0,1\}^{\lfloor \log_2 N \rfloor}$ and outputting $c \leftarrow F_{\text{RSA}}(M)$. Show that this "fix" is still broken.

Problem 4: Time Spent [5 points for answering]. How long did you spend on this problem set? This is for calibration purposes, and the response you provide will not affect your score.

Optional Feedback [0 points]. Please answer the following *optional* questions to help us design future problem sets. You do not need to answer these questions. However, we do encourage you to provide us feedback on how to improve the course experience.

(a) What was your favorite problem on this problem set? Why?

(b) What was your least favorite problem on this problem set? Why?

(c) Do you have any other feedback for this problem set?

(d) Do you have any other feedback on the course so far?