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

Problem Set 4

Due: April 12, 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: Conceptual Questions [30 points]. For each of the following questions, a brief explanation (i.e., 2-5 sentences) suffices.

(a) Let S be the set of languages that have perfectly sound NIZK proofs in the plain model (i.e., without random oracles or a common reference string) with a *deterministic* zero-knowledge simulator. What is the class of languages S?

(b) Consider a variant of Schnorr's Σ-protocol for proving knowledge of discrete log (i.e., $x \in \mathbb{Z}_p : h = g^x$) where the verifier samples the challenge uniformly from a set $S \subseteq \mathbb{Z}_p$ where $|S| = \text{poly}(\lambda)$. Show that an efficient prover who does *not* know the discrete log x can nevertheless convince the verifier to accept with non-negligible probability. [**Remark:** This shows that the challenge space in Schnorr's protocol (and more generally, Σ-protocols) must be super-polynomial to provide negligible soundness error.]

(c) Consider a 4-round variant of the Σ-protocol for showing "knowledge of common discrete log" (i.e., showing that (g_1, g_2, h_1, h_2) satisfy $h_1 = g_1^x$ and $h_2 = g_2^x$ for some $x \in \mathbb{Z}_p$), where the verifier first commits to its challenge using a computationally-hiding commitment scheme. Is this *proof system* sound?

(d) Consider a variant of Schnorr's signature scheme where the challenge is derived by computing $c \leftarrow H(g, h, m)$, where (g, h) is the public key, m is the message, and H is the hash function (modeled as a random oracle). In particular, the signer does not include its commitment $u = g^r$ as input to H. Show that this variant is insecure (i.e., describe an attack that breaks unforgeability).

(e) In class, we saw that using the same randomness (i.e., the same value $u = g^r$) with Schnorr's signature scheme to sign two different messages leaks the signing key. Suppose instead the signer uses different, but correlated, randomness r_1 and r_2. Namely, the signer samples $r_1 \xleftarrow{R} \mathbb{Z}_p$ to sign the first message m_1 and then computes $r_2 \leftarrow a \cdot r_1 + b$ to sign the second message $m_2 \neq m_1$. Suppose that $a, b \in \mathbb{Z}_p$ are publicly known. Show that given m_1, m_2, their associated signatures σ_1, σ_2, and a, b, one can recover the secret signing key. [**Remark:** This demonstrate that it is critical to *derandomize* signature schemes based on Σ-protocols. Otherwise, security depends critically on the signer's entropy source!]

(f) Suppose Alice (with secret input $x \in \{0,1\}$) and Bob (with secret input $y \in \{0,1\}$) use Yao's protocol to compute a function $f(x, y) \in \{0,1\}$ for some function $f : \{0,1\} \times \{0,1\} \to \{0,1\}$. You may assume that all parties are semi-honest and that both parties learn the output of the computation. Is it true that under reasonable computational assumptions, Alice does not learn y at the end of the protocol (i.e., Alice's probability of guessing Bob's bit after running the protocol is at most $1/2 + \text{negl}(\lambda)$)?

Problem 2: Proofs on Committed Values [30 points]. Recall from Problem Set 2 that a Pedersen commitment to a value $x \in \mathbb{Z}_p$ consists of a single group element $\sigma = g^x h^r$, where $r \xleftarrow{R} \mathbb{Z}_p$ is the commitment randomness and (g, h) is part of the public parameters of the commitment scheme. In this problem, we will develop Σ-protocols for proving various properties on *committed* values. For each problem, you should argue completeness, honest-verifier zero-knowledge, and proof of knowledge for each of your Σ-protocols (or cite the relevant analysis from the lecture notes). When analyzing the proof of knowledge property, feel free to assume that the prover convinces the honest verifier with probability 1. In all cases, you may assume that the prover knows the committed message and randomness. The prover should be an efficient algorithm in all of your constructions.

(a) Give a Σ-protocol for showing that a commitment is to a binary value. In other words, describe a Σ-protocol for the following language:

$$\mathcal{L} = \left\{\sigma \in \mathbb{G} \mid \exists r \in \mathbb{Z}_p : \sigma = h^r \text{ or } \sigma = g h^r\right\}.$$

(b) Give a Σ-protocol for proving knowledge of an opening to a Pedersen commitment. Namely, describe a Σ-protocol for the following language:

$$\mathcal{L} = \left\{\sigma \in \mathbb{G} \mid \exists x, r \in \mathbb{Z}_p : \sigma = g^x h^r\right\}.$$

(c) Give a Σ-protocol for proving that two Pedersen commitments are commitments to the same value. Namely, describe a Σ-protocol for the following language:

$$\mathcal{L} = \left\{\sigma_1, \sigma_2 \in \mathbb{G} \mid \exists x, r_1, r_2 \in \mathbb{Z}_p : \sigma_1 = g^x h^{r_1} \text{ and } \sigma_2 = g^x h^{r_2}\right\}.$$

(d) Using the protocols from Part (a) and Part (c), construct a Σ-protocol for showing that a committed value is less than 2^d (for a fixed and publicly-known parameter d). Namely, describe a Σ-protocol for the following language:

$$\mathcal{L} = \left\{\sigma \in \mathbb{G} \mid \exists x, r \in \mathbb{Z}_p : \sigma = g^x h^r \text{ and } x < 2^d.\right\}$$

The complexity of your Σ-protocol should be *polynomial* in d (and $\log p$). When arguing zero-knowledge, you may use (without proof) that the Pedersen commitment scheme is perfectly hiding (shown in Problem Set 2). You only need to show knowledge against computationally-bounded provers and moreover, you may assume that the discrete logarithm problem is hard in $\mathbb{G}$. [**Hint:** Have the prover start by constructing $n = \lceil \log p \rceil$ fresh Pedersen commitments to the bits of x.]

Problem 3: Precomputing Oblivious Transfers [15 points]. In this question, we will explore how to precompute oblivious transfers (OT).

(a) Suppose that in a separate offline phase, a trusted party samples $r_0, r_1 \xleftarrow{R} \{0,1\}^\ell$ and $b \xleftarrow{R} \{0,1\}$. It gives (r_0, r_1) to the sender and (b, r_b) to the receiver. The pair of value (r_0, r_1) and (b, r_b) is referred to as an "OT correlation." Show that the sender and the receiver can use their OT correlation to implement a two-round oblivious transfer protocol on ℓ-bit messages. Show that your scheme is correct and provides *perfect sender security* as well as *perfect receiver security*.

(b) Instead of relying on a trusted party to generate the OT correlations in the offline phase, show how the sender and receiver can generate an OT correlation using an OT protocol. [**Remark:** Observe that this OT protocol is *input-independent*, and thus can be done before any protocol execution. Moreover, using a technique called *OT extension*, the sender and the receiver can precompute a large number of OTs with very modest cost.]

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?