CS 388H: Cryptography **Fall 2021**

Homework 5: Cryptographic Protocols

Due: December 2, 2021 at 11:59pm (Submit on Gradescope) **Instructor:** David Wu

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

`https://www.cs.utexas.edu/~dwu4/courses/fa21/static/homework.tex`

You must submit your problem set via Gradescope (accessible through Canvas).

Collaboration Policy. You may discuss your general *high-level* strategy with other students, but you may not share any written documents or code. You should not search online for solutions to these problems. If you do consult external sources, you must cite them in your submission. You must include the names of all of your collaborators with your submission. Refer to the official course policies for the full details.

Problem 1: Correlated Primes and Factoring [16 points]. When generating an RSA modulus $N = pq$, it is important to sample p, q *independently*. Suppose however we have a limited entropy pool, and the primes p, q we sample happen to lie in a narrow interval: $|p-q| < N^{1/4}$. We will show that even though $N^{1/4}$ is *exponentially* large, factoring is easy in this setting.

(a) Given $N = pq$ where $|p-q| < N^{1/4}$, construct an efficient algorithm that recovers p, q. The running time of your algorithm should be polynomial in the length of N (i.e., $\mathrm{poly}(\log N)$). **Hint:** First show that the arithmetic mean $\mu = (p+q)/2$ is very close to $\sqrt{N}$ (i.e., $0 < \mu - \sqrt{N} < 1$). One of the inequalities ($\mu > \sqrt{N}$) follows by the arithmetic-mean geometric-mean (AM-GM) inequality.

(b) Why is this attack not an issue if we sample p, q independently? *(No need for a precise calculation.)*

Problem 2: Basics of Interactive Proofs [18 points]. Recall that the class of languages with an interactive proof system is the class of languages that can be decided with polynomial space: $\mathsf{IP} = \mathsf{PSPACE}$. Here, we highlight several important aspects of interactive proofs:

(a) **Prover randomness:** Let $\mathcal{L}$ be a language with an interactive proof system with a randomized prover. Show that there exists an interactive proof system for $\mathcal{L}$ with a deterministic prover (with the same completeness guarantee).

(b) **Verifier randomness:** Let $\mathcal{L}$ be a language with an interactive proof system where the verifier is *deterministic*. Show that $\mathcal{L} \in \mathsf{NP}$.

Remark: The above two properties show that allowing the prover to be randomized does not increase the power of interactive proofs, whereas allowing the verifier to be randomized is critical.

(c) **Imperfect soundness:** Let $\mathcal{L}$ be a language with a perfectly sound interactive proof (i.e., an unbounded prover will *never* convince the honest verifier of a statement $x \notin \mathcal{L}$). Show that $\mathcal{L} \in \mathsf{NP}$.

Problem 3: Bit Commitments from PRGs [16 points]. In class, we constructed a zero-knowledge proof for NP using a cryptographic commitment scheme. In this problem, we will show how to construct a commitment scheme from PRGs. Let $G\colon \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be a PRG. In this setting the commitment scheme is an interactive protocol between a "committer" and a "verifier." The protocol works as follows:

- **Initial message:** The verifier samples a random string $r \xleftarrow{\text{R}} \{0,1\}^{3\lambda}$ and sends it to the committer.
- **Commitment:** To commit to a bit $b \in \{0,1\}$, the committer samples a seed $s \xleftarrow{\text{R}} \{0,1\}^\lambda$. If $b = 0$, it sends $c \leftarrow G(s)$ as its commitment and if $b = 1$, it sends $c \leftarrow G(s) \oplus r$ as its commitment.
- **Opening:** To open up a commitment $c \in \{0,1\}^{3\lambda}$, the committer sends (b, s) where $b \in \{0,1\}$ is the bit and $s \in \{0,1\}^\lambda$ is the seed used to construct the commitment. The verifier accepts if either $(b = 0 \wedge G(s) = c)$ or $(b = 1 \wedge G(s) = r \oplus c)$.

(a) Show that if G is secure, then this commitment scheme is computationally hiding, even against a malicious verifier. Namely, show that if G is a secure PRG, then for *any* choice of initial message $r \in \{0,1\}^{3\lambda}$, a commitment to the bit 0 is computationally indistinguishable from a commitment to the bit 1.

(b) Show that this commitment scheme is statistically binding. Namely, a dishonest committer (even one that is computationally unbounded) cannot find a commitment $c \in \{0,1\}^{3\lambda}$ and openings $s_0, s_1 \in \{0,1\}^\lambda$ such that $(0, s_0)$ and $(1, s_1)$ are both valid openings, except with negligible probability over the choice of r. Here, the verifier is honest and will always sample a uniform random string $r \xleftarrow{\text{R}} \{0,1\}^{3\lambda}$.

Remark: In conjunction with the protocol from class, this problem shows that one-way functions (which are sufficient for PRGs) imply (computational) zero-knowledge proofs for all NP languages.

Problem 4: Proving Relations in the Exponent [25 points]. Let $\mathbb{G}$ be a group of prime order p and generator g where the DDH assumption holds. The Chaum-Pedersen protocol we discussed in class allows a prover to convince a verifier that a tuple (g, g^x, g^y, g^z) is a DDH tuple (i.e., that $z = xy \bmod p$). Namely, the Chaum-Pedersen protocol proves membership in the DDH language $\mathcal{L}_{\mathsf{DDH}}$ where

$$\mathcal{L}_{\mathsf{DDH}} = \left\{(g, h, u, v) \in \mathbb{G}^4 \mid \exists x \in \mathbb{Z}_p : (h = g^x) \wedge (v = u^x)\right\}.$$

In this problem, we consider two related algebraic languages. You should *not* use general zero-knowledge proofs for NP languages in this problem.

(a) **Proving validity of the DDH random self-reduction.** A useful building block in various cryptographic protocols is the ability to apply the DDH random self-reduction to a candidate DDH tuple and prove that the self-reduction was applied properly. Specifically, define the following language corresponding to the DDH random self-reduction:

$$\mathcal{L}_{\mathsf{rsr}} = \left\{\big((g, h, u, v), (u', v')\big) \mid \exists \alpha, \beta \in \mathbb{Z}_p : (u' = u^\alpha g^\beta) \wedge \left(v' = v^\alpha h^\beta\right)\right\}.$$

Construct a Σ-protocol for $\mathcal{L}_{\mathsf{rsr}}$. Prove completeness, special soundness, and HVZK of your protocol.

(b) **Proving that a tuple is not a DDH tuple.** Consider the complement of the DDH language: namely, that a tuple (g, g^x, g^y, g^z) is *not* a DDH tuple (i.e., that $z \neq xy \bmod p$). Formally, we can write

$$\mathcal{L}_{\mathsf{nDDH}} = \left\{(g, g^x, g^y, g^z) \mid z \neq xy \bmod p\right\}$$

In this setting, the prover knows the exponents $x, y, z \in \mathbb{Z}_p$, and both the prover and the verifier know the statement (g, g^x, g^y, g^z). Use the Σ-protocol for $\mathcal{L}_{\mathsf{rsr}}$ together with the Chaum-Pedersen protocol to construct a Σ-protocol for $\mathcal{L}_{\mathsf{nDDH}}$. Remember to prove completeness, soundness,[1] and HVZK. **Hint:** Consider the following scenario. Suppose Alice has the numbers 1 through 5 written down on individual pieces of paper. She selects one of the numbers and places each of the remaining four numbers in four different sealed envelopes. Alice wants to convince Bob that she did *not* choose the number 3 without revealing anything more about her selection. Could she do this by opening exactly one of the sealed envelopes?

Problem 5: Time Spent [3 extra credit points]. How long did you spend on this problem set? In addition, please fill out the Course Instructor Survey (`https://utdirect.utexas.edu/ctl/ecis/`) to provide feedback on your course experience this semester. To receive extra credit for this problem, please include a screenshot of your CIS dashboard (accessible via the above link) to indicate you have completed the survey. The CIS dashboard should only say whether you have completed the survey or not and should **not** contain information about your survey responses for any class. Feel free to remove any information other than your name and the name of this course from your proof of survey completion. *In an ideal world, you would prove that you completed the evaluation in zero knowledge. Can you think of a protocol to do this?*

Optional Feedback. 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 overall?

[1] You do *not* need to show special soundness here.