CS 355: Topics in Cryptography **Spring 2021**

Problem Set 3

Due: May 10, 2021 at 11:59pm

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

https://crypto.stanford.edu/cs355/21sp/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Note that Gradescope requires that the solution to each problem starts on a **new page**.

Bugs: We make mistakes! If it looks like there might be a mistake in the statement of a problem, please ask a clarifying question on Piazza.

Problem 1: Conceptual Questions [10 points]. For each of the following statements, say whether it is TRUE or FALSE. Write *at most one sentence* to justify your answer.

(a) Let $\langle P, V \rangle$ be a zero-knowledge interactive protocol for some language. The protocol has perfect completeness and soundness error 1/3. Which of the following are true:

i A malicious verifier interacting with an honest prover will always accept a true statement.

ii An honest verifier interacting with a malicious prover will "learn nothing" besides the statements validity.

(b) Consider a modified version of Schnorr's signature in which the signing nonce r is computed as $r \leftarrow H(m)$, where $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a hash function (modeled as a random oracle), m is the message to be signed, and q is the order of the group used for the signature scheme. This deterministic version of Schnorr's signature scheme is secure.

(c) The security of the Fiat-Shamir transform implies that a sigma protocol with a random challenge and soundness 1/2 can be *directly* converted to a NIZK by replacing the challenge message with a hash, so long as the hash function is modeled as a random oracle.

(d) Recall the SNARG constructed in class from the PCP theorem and a merkle tree. If the PCP theorem is instantiated with soundness error ϵ, then the SNARG also has soundness error ϵ.

Problem 2: Understanding Interactive Proofs [15 points]. *(Problems from "The Foundations of Cryptography - Volume 1, Basic Techniques" by Oded Goldreich)*

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

(b) *The role of prover randomness:* Let L be a language with an interactive proof system. Show that there exists an interactive proof system for L for which the prover P is deterministic.
[**Hint:** Use the fact that P is unbounded.]

(c) *The role of errors:* Let L be a language with an interactive proof system with perfect soundness, that is if $x \notin L$, the verifier *never* accepts (not even with negligible probability). Show that $L \in \mathsf{NP}$.

Problem 3: Sigma Protocol for Circuit Satisfiability [10 points]. Let circuit-SAT be the language of satisfiable Boolean circuits[1] :

$$\text{circuit-SAT} = \left\{C\colon \{0,1\}^n \to \{0,1\} \mid n \in \mathbb{N},\ \exists (x_1,\ldots,x_n) \in \{0,1\}^n \text{ such that } C(x_1,\ldots,x_n) = 1\right\}.$$

Let $\mathsf{Commit}\colon \{0,1\} \times \mathcal{R} \to \mathcal{C}$ be a perfectly-binding and computationally-hiding commitment scheme with message space $\{0,1\}$, randomness space $\mathcal{R}$, and commitment space $\mathcal{C}$. Suppose that there exist Sigma protocols $\langle P_{\text{XOR}}, V_{\text{XOR}}\rangle$ and $\langle P_{\text{AND}}, V_{\text{AND}}\rangle$ for languages $\mathcal{L}_{\text{XOR}}$ and $\mathcal{L}_{\text{AND}}$, respectively, where:

$$\mathcal{L}_{\text{XOR}} = \left\{(c_1,c_2,c_3) \in \mathcal{C}^3 \;\middle|\; \begin{array}{c} \exists (m_1,m_2,m_3) \in \{0,1\}^3, (r_1,r_2,r_3) \in \mathcal{R}^3 \text{ such that} \\ \forall i \in \{1,2,3\}\ \ c_i = \mathsf{Commit}(m_i; r_i) \text{ and } m_1 \oplus m_2 = m_3 \end{array}\right\}$$

$$\mathcal{L}_{\text{AND}} = \left\{(c_1,c_2,c_3) \in \mathcal{C}^3 \;\middle|\; \begin{array}{c} \exists (m_1,m_2,m_3) \in \{0,1\}^3, (r_1,r_2,r_3) \in \mathcal{R}^3 \text{ such that} \\ \forall i \in \{1,2,3\}\ \ c_i = \mathsf{Commit}(m_i; r_i) \text{ and } m_1 \wedge m_2 = m_3 \end{array}\right\}.$$

Give a Sigma protocol for circuit-SAT. In addition to describing a protocol, you will also need to show that your protocol satisfies completeness, soundness, and honest-verifier zero-knowledge. [**Hint:** When showing that your protocol is honest-verifier zero-knowledge, you may want to use a hybrid argument. Some of your hybrids might rely on the commitment scheme being computationally hiding, and the other hybrid might rely on the underlying Sigma protocols being honest-verifier zero-knowledge.]

Problem 4: Using Polynomial Commitments [12 pts].

(a) **Aggregation** One interesting property of the KZG polynomial commitment scheme is that it permits *aggregation*: a single proof can establish openings for multiple polynomials. Aggregation for a polynomial commitment is expressed through two algorithms:

- $\mathsf{Open2}(pp, c_0, c_1, f_0, f_1, x) \to \pi$: Creates an opening proof for $f_0(x)$ **and** $f_1(x)$ (which have commitments c_0 and c_1 respectively).
- $\mathsf{Check2}(pp, c_0, c_1, y_0, y_1, x, \pi) \to \{0,1\}$: Checks that π proves $y_0 = f_0(x)$ **and** $y_1 = f_1(x)$.

With aggregation, we need a few extra security properties to capture the fact that legitimate proofs are always accepted, and illegitimate proofs are almost always rejected.

- **(Perfect) Aggregate Correctness:** For all d, all polynomials f_0 and f_1 in the field, of degree at most d, and all x in the field,

$$\Pr\left[\begin{array}{ll} pp \leftarrow \mathsf{Setup}(d) & \\ c_0 \leftarrow \mathsf{Commit}(pp, f_0) & \\ c_1 \leftarrow \mathsf{Commit}(pp, f_1) & : \ \mathsf{Check2}(pp, c_0, c_1, f_0(x), \\ \pi \leftarrow \mathsf{Open2}(pp, c_0, c_1, f_0, f_1, x) & \qquad f_1(x), x, \pi) = 1 \end{array}\right] = 1$$

  (where Commit is from the base polynomial commitment scheme).

- **Aggregate Evaluation Binding:** For all efficient adversaries $\mathcal{A}$, if we set $pp \leftarrow \mathsf{Setup}(d)$ and $(x, c_0, c_1, \pi, \pi', y_0, y_1, y_0', y_1') \leftarrow \mathcal{A}(pp, d)$, then

$$\Pr\left[\begin{array}{c} \mathsf{Check2}(pp, c_0, c_1, y_0, y_1, x, \pi) = 1 \\ \wedge \mathsf{Check2}(pp, c_0, c_1, y_0', y_1', x, \pi') = 1 \\ \wedge (y_0, y_1) \neq (y_0', y_1') \end{array}\right] = \text{negl}(\lambda).$$

[1] You can assume without loss of generality that a Boolean circuit consists of only XOR and AND gates.

For this problem, construct Open2 and Check2 and show that your construction satisfies aggregate correctness. Then, give an informal argument (2 or 3 sentences) that it satisfies aggregate evaluation binding (assuming t-SDH is hard) in the random oracle model.

[Extra Credit (2 pts)] Give a formal proof that your scheme is aggregate evaluation binding (assuming t-SDH is ard) in the random oracle model.

You may use a random oracle over whichever domain you like.

Hint: you may want to revisit aggregation for BLS signatures.

(b) **Accumulators** Cryptographic accumulators represent a data structure as a small digest in such a way that operations over the data structure can be verified with access to only the digest. A *set accumulator* over universe $\mathcal{U}$ comprises five algorithms:

- $\mathsf{Create}(S \subset \mathcal{U}) \to d$: Creates a digest representing the set $S \subset \mathcal{U}$.
- $\mathsf{ProveMem}(S \subset \mathcal{U}, d, x \in S) \to \pi$: Creates a proof that $x \in S$.
- $\mathsf{VerifyMem}(d, x, \pi) \to \{0,1\}$: Verifies a proof that $x \in S$.
- $\mathsf{ProveNonMem}(S \subset \mathcal{U}, d, x \notin S) \to \bar{\pi}$: Creates a proof that $x \notin S$.
- $\mathsf{VerifyNonMem}(d, x, \bar{\pi}) \to \{0,1\}$: Verifies a proof that $x \notin S$.

A set accumulator is *membership-secure* if forging membership proofs for elements not in the set is hard. Similarly, a set accumulator is *non-membership-secure* if forging non-membership proofs for elements in the set is hard.[2] That is, if for all efficient adversaries $\mathcal{A}$, the following probabilities are negligible in λ:

$$\Pr\left[\begin{array}{l|c} (S,x,\pi) \leftarrow \mathcal{A}(\mathcal{U}) & x \notin S \wedge \\ d \leftarrow \mathsf{Create}(S) & \mathsf{VerifyMem}(d,x,\pi) = 1 \end{array}\right] \quad \Pr\left[\begin{array}{l|c} (S,x,\bar{\pi}) \leftarrow \mathcal{A}(\mathcal{U}) & x \in S \wedge \\ d \leftarrow \mathsf{Create}(S) & \mathsf{VerifyNonMem}(d,x,\bar{\pi}) = 1 \end{array}\right]$$

Using an evaluation-binding polynomial commitment scheme (which may, or may not be the KZG scheme) for polynomials over $\mathbb{F}$, build a set accumulator for $\mathcal{U} = \mathbb{F}$. Prove that your accumulator is membership and non-membership secure, assuming that the underlying polynomial commitment scheme is binding.

Hint: What set of values is naturally associated with a polynomial?

(c) **Extra Credit [3 points].** Now, assume that your accumulator construction is instantiated with the KZG polynomial commitment scheme.

Show how one can compute the digest d' of $S \cup \{y\}$ given the digest d for S, and the secret exponent of the polynomial commitment scheme, α. Your algorithm,

- $\mathsf{Update}(S, d, y) \to d'$: returns a digest d' equal to $\mathsf{Create}(S \cup \{y\})$

should run in time independent of $|S|$, and you should show it is correct.

Furthermore, show how one can compute a single update token, u, that allows any membership proof π for some x which is valid with respect to d to be updated into a new membership proof π' which is valid with respect to d'. That is, given two algorithms:

[2] In the literature, both properties are called "collision resistance". We avoid that name here to avoid a confusion with hash functions.

- $\mathsf{MakeToken}(S, d, y) \to u$: creates update token u for membership proofs with respect to d
- $\mathsf{UpdateProof}(d, x, \pi, u) \to \pi'$: creates a new proof π' which is valid with respect to d'

and show that these algorithms produce a valid π'.[3]

Problem 5: Time Spent [3 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 questions to help us design future problem sets. You do not need to answer these questions, and if you would prefer to answer anonymously, please use this form. 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?

[3] Note: Some accumulator constructions to not admit the efficient update algorithms you build in this sub-problem. We suspect yours will, but if you're worried that your construction for part (b) does not allow for efficient updates, feel free to consult with the teaching staff.