COMP 537: Cryptography **Fall 2025**

Problem Set 4

Instructions: You **must** typeset your solution in LaTeX using the provided template. Please submit your problem set via Gradescope. Include your name and the names of any collaborators at the top of your submission.

Problem 1: Key-Homomorphic PRFs [15 points]. In class we saw the PRF $F(k,x) = H(x)^k$ and were told that it has many useful properties. In this problem, we will explore one of these. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PRF defined over groups $(\mathcal{K}, +)$ and $(\mathcal{Y}, \cdot)$, where $+$ and $\cdot$ are the respective group operations in those groups. We say F is *key-homomorphic* if it holds that

$$F(k_1 + k_2, x) = F(k_1, x) \cdot F(k_2, x).$$

(a) Demonstrate with a counterexample that not every PRF is key-homomorphic.

(b) Please prove that the PRF $F(k,x) = H(x)^k$ defined with a random oracle $H : \mathcal{X} \to G$ (where G is a group of prime order p) is a key-homomorphic PRF. This can be proved directly in at most a few lines.

(c) *Key rotation* is a common problem encountered in cloud storage: how to change the key under which data is encrypted without sending the keys to the storage provider? A naive solution is to download the encrypted data, decrypt it, re-encrypt it under a new key, and re-upload the new ciphertext. We will now see how this process can be made more efficient with a key-homomorphic PRF.

Suppose you have a ciphertext c made up of blocks $c_1, \ldots, c_N$ that corresponds to a message $m = (m_1, \ldots, m_N)$ encrypted under a key k_1 using a key-homomorphic PRF F in counter mode, i.e., $c_i = m_i \cdot F(k_1, i)$. Now you want to rotate to a key k_2. It turns out you can send the storage provider a single element $k_{\mathsf{update}} \in \mathcal{K}$ which it can then use to generate c', an encryption of m under k_2. Please tell us how you can compute k_{update} and how the storage provider can use k_{update} and c to compute c'.

Problem 2: Multi-Commitments [10 points]. Let $\mathbb{G}$ be a group of prime order q in which the discrete logarithm problem is hard. Let g and h be generators of $\mathbb{G}$. As we saw in class, the Pedersen commitment scheme commits to a message $m \in \mathbb{Z}_q$ using randomness $r \in \mathbb{Z}_q$ as $\mathsf{Commit}(m; r) := g^m h^r \in \mathbb{G}$. Moreover, we saw that Pedersen commitments are *additively homomorphic*, meaning that given commitments to m_1 and m_2, one can compute a commitment to $m_1 + m_2$. The "public parameters" associated with the Pedersen commitment scheme are the description of the prime-order group $\mathbb{G}$ and the group elements g and h.

(a) Use $\mathbb{G}$ to construct an additively homomorphic commitment scheme $\mathsf{Commit}_n(m_1, \ldots, m_n; r)$ that commits to a length-n vector of messages $(m_1, \ldots, m_n) \in \mathbb{Z}_q^n$ using randomness $r \in \mathbb{Z}_q$. The output of the commitment should be short – only a single group element. You should specify both the public parameters of your scheme (which may be different from that of the basic Pedersen commitment scheme) as well as the description of the Commit_n function.

You do not need to prove (or even argue for) the security of your scheme, but the resulting scheme should be perfectly hiding and computationally binding assuming hardness of discrete log in $\mathbb{G}$.

(b) Show that if you are given a hash function $H\colon \mathbb{Z}_q \to \mathbb{G}$ (modeled as a random oracle), the public parameters for your construction from Part (a) only needs to consist of the description of the group $\mathbb{G}$ and the description of H. Argue *informally* why your modification to the construction is secure. You do *not* need to provide a formal proof. This problem shows that getting rid of public parameters is another reason why random oracles are useful in practice!

Problem 3: ZK Conceptual Questions [12 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 $1/3$.

   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 $H : \{0,1\}^* \to \mathbb{Z}_q$ 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.

Problem 4: Sigma Protocol for Circuit Satisfiability [15 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 argue *informally* that your protocol satisfies completeness, soundness,and honest-verifier zero-knowledge. Please state which assumptions about the underlying tools are needed to prove each property.

[1] A boolean circuit is modeled as a directed acyclic graph where each node represents either a gate or an input to the circuit (if it has in-degree zero). You can assume without loss of generality that a Boolean circuit consists of only XOR and AND gates.

Optional Feedback [5 points]. Please answer the following questions to help design future problem sets. You are not required to answer these questions (the points are free), and if you would prefer to answer anonymously, please use the anonymous feedback form. However, we do encourage you to provide feedback on how to improve the course experience.

(a) Roughly how long did you spend on this problem set?

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

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

(d) Any other feedback for this problem set? Was it too easy/difficult?

(e) Any other feedback on the course so far?