CS 355: Topics in Cryptography **Spring 2026**

Problem Set 5

Due: Friday, 5 June 2026 (submit via Gradescope)

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

https://cs355.stanford.edu/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use course code **X85KN6** to sign up. 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 Ed.

Problem 1: True/False [4 points].

1. For any pair of points $(x_1, y_1), (x_2, y_2) \in \mathbb{Z}_6^2$, where $x_1 \neq x_2$, there is a polynomial p of degree at most 1 such that $p(x_1) = y_1$ and $p(x_2) = y_2$. (Note: "6" is not a prime.)

2. You and your friends want to determine which one of you has the lowest salary. You design and run a protocol, at the end of which all your friends learn that their Big 4 salaries$^{\text{TM}}$ are higher than yours. This blatant invasion of your privacy could have been avoided if you had used a proper maliciously-secure MPC protocol.

3. In Yao's protocol for secure two-party computation of a function $f(\cdot,\cdot)$ (as described in lecture), the two parties must exchange a number of bits that is at least as large as a Boolean circuit computing f.

4. Say that Alice, with input $x \in \{0,1\}$, and Bob, with input $y \in \{0,1\}$, use Yao's protocol to compute $f(x,y) \in \{0,1\}$, for some function $f : \{0,1\} \times \{0,1\} \to \{0,1\}$. Then, under reasonable computational assumptions, the protocol must hide y from Alice—that is, Alice's probability of guessing Bob's bit after running the protocol is at most $1/2 + \text{negl}(\lambda)$, for security parameter λ.

Problem 2: Generating Beaver Multiplication Triples [15 points]. Recall from lecture that Beaver multiplication triples enables general multiparty computation on secret-shared data. In this problem, we will explore two methods that can be used to generate Beaver multiplication triples. For simplicity, we will just consider the two-party setting and we will generate Beaver multiplication triples over the binary field $\mathbb{Z}_2$ (where addition corresponds to xor). To be precise, we first describe an "idealized process" for generating a single multiplication triple. In this "idealized process", a trusted party generates the triple and then distributes the shares of the triple to the two parties Alice and Bob.

1. The trusted party chooses $a, b \xleftarrow{R} \mathbb{Z}_2$ and computes $c = ab \in \mathbb{Z}_2$.

2. The trusted party distributes a 2-out-of-2 secret sharing of a, b, and c to Alice and Bob. Specifically, the trusted party samples $r_a, r_b, r_c \xleftarrow{R} \mathbb{Z}_2$ and gives r_a, r_b, r_c to Alice. The trusted party then computes $s_a = a \oplus r_a$, $s_b = b \oplus r_b$, and $s_c = c \oplus r_c$, and gives s_a, s_b, s_c to Bob.

By construction $[a] = (r_a, s_a)$ is an additive secret-sharing of a, $[b] = (r_b, s_b)$ is an additive secret-sharing of b, and $[c] = (r_c, s_c)$ is an additive secret-sharing of c. Moreover, $c = ab$, so $([a], [b], [c])$ is a valid Beaver multiplication triple.

We will show how Alice and Bob can generate these Beaver triples without relying on a trusted party. Throughout this problem, you may assume that Alice and Bob are "honest-but-curious" (namely, they follow the protocol exactly as described, but may try to infer additional information from the protocol transcript—this is the model that we considered in lecture).

(a) Show how Alice and Bob can generate a Beaver multiplication triple using Yao's protocol.[1] Your construction should not make any modifications to the internal details of Yao's protocol (in fact, any secure two-party computation protocol can be used here). Then, give an *informal* argument why your protocol is correct and secure. [**Hint:** To apply Yao's protocol, you will need to come up with a two-party functionality f that Alice and Bob will jointly compute. Try letting Alice's inputs to f be her shares (r_a, r_b, r_c), which she samples uniformly at random at the beginning of the protocol.]

(b) Show how Alice and Bob can use a *single invocation* of an 1-out-of-4 oblivious transfer (OT) protocol (on 1-bit messages) to generate a Beaver multiplication triple. Give an *informal* argument why your protocol is correct and secure. (In a 1-out-of-n OT, the sender has n messages $m_1, \ldots, m_n$, while the receiver has a single index $i \in [n]$. At the end of the protocol execution, the sender learns nothing while the receiver learns m_i (and nothing else). The formal definitions of sender and receiver privacy are the analogs of those presented in lecture.) [**Hint:** Try using OT to directly evaluate the functionality f you constructed from Part (a).]

(c) Let $\ell \in \mathbb{N}$ be a constant. Show how to build a 1-out-of-2^ℓ OT protocol (on 1-bit messages) using ℓ invocations of an 1-out-of-2 OT protocol (on λ-bit messages) together with a PRF $F\colon \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}$. Here, $\{0,1\}^\lambda$ is the key-space of the PRF and $\{0,1\}^\ell$ is the domain of the PRF. Then, give an *informal* argument for why your protocol satisfies correctness, sender privacy, and receiver privacy. [**Hint:** Start by having the sender sample 2ℓ independent PRF keys. The sender will use these keys to blind each of its messages $m_1, \ldots, m_{2^\ell}$.]

Problem 3: Key-Exchange from LWE [18 points]. In this problem, we will formalize the concept of a *non-interactive key exchange* (NIKE) protocol, and then construct it from LWE. NIKE protocols are a core component of Internet protocols like TLS, and the lattice-based NIKE that we develop in this problem is a simplified variant of some of the leading candidates in the NIST competition for standardizing post-quantum key-exchange.

[1] You may use the variant of Yao's protocol where only one party receives output (and the other party learns nothing).

A *non-interactive key exchange* (NIKE) protocol for a key space $\mathcal{K}$ consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input the security parameter λ, the setup algorithm outputs the public parameters pp.
- $\mathsf{ClientPublish}(\mathsf{pp}) \to (\mathsf{priv}, \mathsf{pub})$: On input the public parameters pp, the client-publish algorithm outputs a secret value priv, and a public message pub.
- $\mathsf{ServerPublish}(\mathsf{pp}) \to (\mathsf{priv}, \mathsf{pub})$: On input the public parameters pp, the server-publish algorithm outputs a secret value priv, and a public message pub.
- $\mathsf{KeyGen}(\mathsf{priv}, \mathsf{pub}) \to \mathsf{key}$: On input a secret value priv, and a public message pub, the key generation algorithm outputs a key $\mathsf{key} \in \mathcal{K}$.

Correctness. We require that when $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pub}_0, \mathsf{priv}_0) \leftarrow \mathsf{ClientPublish}(\mathsf{pp})$, $(\mathsf{pub}_1, \mathsf{priv}_1) \leftarrow \mathsf{ServerPublish}(\mathsf{pp})$, we have

$$\Pr\left[\mathsf{KeyGen}(\mathsf{priv}_0, \mathsf{pub}_1) = \mathsf{KeyGen}(\mathsf{priv}_1, \mathsf{pub}_0)\right] = 1 - \mathrm{negl}(\lambda)$$

where the probability is taken over the randomness of all procedures.

Security. For a NIKE protocol $(\mathsf{Setup}, \mathsf{ClientPublish}, \mathsf{ServerPublish}, \mathsf{KeyGen})$, we define the following two experiments:

Experiment b $\quad (b = 0, 1)$:

- The challenger computes the following:
  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$,
  $(\mathsf{priv}_0, \mathsf{pub}_0) \leftarrow \mathsf{ClientPublish}(\mathsf{pp})$,
  $(\mathsf{priv}_1, \mathsf{pub}_1) \leftarrow \mathsf{ServerPublish}(\mathsf{pp})$,
  $\mathsf{key}_0 \leftarrow \mathsf{KeyGen}(\mathsf{priv}_0, \mathsf{pub}_1)$,
  $\mathsf{key}_1 \xleftarrow{R} \mathcal{K}$.
  It provides $(\mathsf{pp}, \mathsf{pub}_0, \mathsf{pub}_1, \mathsf{key}_b)$ to the adversary.
- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.

Let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. Then, we say that a NIKE protocol is secure if

$$\left|\Pr[W_0] - \Pr[W_1]\right| = \mathrm{negl}(\lambda).$$

(a) Explain in words why the security definition above captures our intuitive notion of security for key-exchange.

(b) Consider the following NIKE protocol:[2]

> Let $n = \text{poly}(\lambda), q, \chi_B$ be parameters for which $\mathsf{LWE}_{\mathsf{HNF}}(n, n, q, \chi_B)$ and $\mathsf{LWE}_{\mathsf{HNF}}(n, n+1, q, \chi_B)$ is hard. Recall from lecture that in practice, for $\lambda = 128$, we use $n \approx 800$.
>
> Define the key space $\mathcal{K} = \{0,1\}$ and consider the following algorithms.
>
> - $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Sample a matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$ and set $\mathsf{pp} = \mathbf{A}$.
> - $\mathsf{ClientPublish}(\mathsf{pp}) \to (\mathsf{priv}, \mathsf{pub})$: Sample vectors $\mathbf{s} \leftarrow \chi_B^n$, $\mathbf{e} \leftarrow \chi_B^n$. Then, set $\mathsf{priv} = \mathbf{s}$, and $\mathsf{pub} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$.
> - $\mathsf{ServerPublish}(\mathsf{pp}) \to (\mathsf{priv}, \mathsf{pub})$: Sample vectors $\mathbf{s} \leftarrow \chi_B^n$, $\mathbf{e} \leftarrow \chi_B^n$. Then, set $\mathsf{priv} = \mathbf{s}$, and $\mathsf{pub} = \mathbf{A}\mathbf{s} + \mathbf{e}$.
> - $\mathsf{KeyGen}(\mathsf{priv}, \mathsf{pub}) \to \mathsf{key}$: Let $\mathsf{priv} = \mathbf{s} \in \mathbb{Z}_q^n$ and $\mathsf{pub} = \mathbf{b} \in \mathbb{Z}_q^n$. The key generation algorithm first samples a small noise term $e \leftarrow \chi_B$. Then, if $\|\langle \mathbf{s}, \mathbf{b} \rangle + e\|_\infty \le \lfloor q/4 \rceil$, set $\mathsf{key} = 0$. Otherwise, set $\mathsf{key} = 1$.

Here, $\lfloor q/4 \rceil$ denotes the integer closest to $q/4$, with ties broken downward. Suppose that q is prime and chosen to satisfy $4nB^2/q = \text{negl}(\lambda)$. Prove that the protocol satisfies correctness. For the proof, feel free to use the following fact (you do not need to prove this fact):

> For any prime q, for $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$ any two non-zero vectors $\mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^n$, and $c \in \mathbb{Z}_q$,
>
> $$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times n}} \left[\mathbf{s}_0^T \mathbf{A} \mathbf{s}_1 = c \right] = 1/q - \text{negl}(\lambda),$$
>
> where the probability is over the random choice of $\mathbf{A}$.

(c) Prove that the protocol above is secure assuming $\mathsf{LWE}_{\mathsf{HNF}}(n, n, q, \chi_B)$ and $\mathsf{LWE}_{\mathsf{HNF}}(n, n+1, q, \chi_B)$. The definition of $\mathsf{LWE}_{\mathsf{HNF}}$ is on the last page of this problem set. [**Hint:** Use a hybrid argument.]

Problem 4: Time Spent [1 point 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 the anonymous feedback form available on the course website. 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?

[2] We restrict the key space to $\mathcal{K} = \{0,1\}$ for simplicity. To get a NIKE protocol for $\mathcal{K} = \{0,1\}^{128}$, we can simply run 128 parallel instances of the protocol using the same public matrix $\mathbf{A}$.

(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?

Appendix: Definition of LWE in Hermite Normal Form.

We review the formal definitions of the Learning with Errors problem in *Hermite Normal Form*. Note that in this variant of the LWE problem, the vector $\mathbf{s}$ is sampled from the B-bounded error distribution χ_B instead of the uniform distribution. This version of the LWE problem is known to be as hard as the standard LWE problem.

$\mathsf{LWE}_{\mathsf{HNF}}(n, m, q, \chi_B)$: Let $n, m, q, B \in \mathbb{N}$ be positive integers, and let χ_B be a B-bounded distribution over $\mathbb{Z}_q$. For a given adversary $\mathcal{A}$, we define the following two experiments:

Experiment b $\quad (b = 0, 1)$:

- The challenger computes

$$\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{m \times n}, \quad \mathbf{s} \leftarrow \chi_B^n, \quad \mathbf{e} \leftarrow \chi_B^m, \quad \mathbf{b}_0 \leftarrow \mathbf{A} \cdot \mathbf{s} + \mathbf{e}, \quad \mathbf{b}_1 \xleftarrow{R} \mathbb{Z}_q^m,$$

  and gives the tuple $(\mathbf{A}, \mathbf{b}_b)$ to the adversary.

- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.

Let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. Then, we define $\mathcal{A}$'s advantage in solving the $\mathsf{LWE}_{\mathsf{HNF}}$ problem for the set of parameters n, m, q, χ_B to be

$$\mathsf{HNF\text{-}LWEAdv}_{n,m,q,\chi_B}[\mathcal{A}] := \Big|\Pr[W_0] - \Pr[W_1]\Big|.$$