CS255: Introduction to Cryptography **Winter 2026**

Final Exam

Instructions:

- **Please answer all five questions. You have three hours.**
- During the exam you may consult any static resource on your laptop such as your notes or a book. You may not access a search engine, an AI, or a friend during the exam. You are expected to do the exam on your own. **You may not interact, collaborate, or discuss the exam with another person or an AI chat bot during the exam day. That would be a gross violation of the honor code.**
- To submit your answers please either (i) use the provided LaTeX template, or (ii) use the provided PDF with a tablet and write your answers in the provided spaces, or (iii) write your answers on a sheet of paper, starting every question on a new page. When done, please upload your solutions to Gradescope (course code `8DE6NR`). **We added fifteen minutes to the exam to give you time to upload your answers to Gradescope.**
- The **LaTeX template** for the final is available at here. Please do not share this link with others.
- Students are bound by the Stanford honor code. In particular, you are expected to do the exam on your own.

Problem 1. (*Questions from all over*) **(20 points)**

a. Let $F : \mathcal{K} \times \mathcal{X} \to \{0,1\}^n$ be a secure PRF where $1 \in \mathcal{X}$. Define

$$F'(k, x) \coloneqq F(k, x) \oplus F(k, 1)$$

Is F' a secure PRF? If so, give a reduction. If not, describe a distinguishing attack.

Your answer:

b. Let p be a large prime, and consider the function $G : \mathbb{Z}_p \to \{0,1\}^n$ defined by

$$G(s) \coloneqq \big(\text{chop}(s^3),\ \text{chop}(s^4),\ \text{chop}(s^5)\big).$$

Here s^3, s^4, s^5 are all in $\mathbb{Z}_p$ and $\text{chop}(x)$ treats the input x as an integer in $\{0, 1, \ldots, p-1\}$ and outputs all the bits of this integer *except* for the two most significant bits. Is this function G a secure PRG?

Your answer:

c. Let $f, h : \mathcal{X} \to \mathcal{Y}$ be efficiently computable functions, where h is collision resistant (but f need not be). Is the function $w(x) := \big(f(x), h(x)\big)$ collision resistant? Justify your answer.

Your answer:

d. Let p be a prime where $2p-1$ is divisible by 5. Show an efficient algorithm that takes $u \in \mathbb{Z}_p$ as input and outputs the fifth root of u in $\mathbb{Z}_p$. That is, show how to efficiently solve the equation $x^5 - u = 0$ in $\mathbb{Z}_p$. Remember to explain why your algorithm computes the correct answer.

Your answer:

e. Explain why a quantum computer puts present day encrypted HTTPS traffic at risk, even though we do not currently have a working quantum computer that can break discrete log or factor large integers.

Your answer:

Problem 2. (*Expanding the message space of a cipher and MAC*) **(20 points)**

a. Let (E, D) be a CPA-secure encryption scheme that encrypts messages in some space M. Let (E', D') encrypt messages in M^ℓ, for some $\ell > 1$, by encrypting each component independently, but using the same secret key. That is, for $\ell = 3$,

$$E'\big(k, (m_0, m_1, m_2)\big) := \big(\ E(k, m_0),\ E(k, m_1),\ E(k, m_2)\ \big)$$

Is (E', D') CPA secure? If so, explain why. If not, describe an attack.

Your answer:

b. Suppose that (E, D) provides authenticated encryption. Does (E', D') provide authenticated encryption? If so, explain why. If not, describe an attack.

Your answer:

c. Let (S, V) be a secure MAC for messages in M. Let (S', V') tag messages in M^ℓ, for $\ell > 1$, by tagging each component independently, using the same signing key for each component. That is, for $\ell = 3$, we have

$$S'\big(k, (m_0, m_1, m_2)\big) \coloneqq \big(\ S(k, m_0),\ S(k, m_1),\ S(k, m_2)\ \big)$$

Show that (S', V') is not a secure MAC. Your attack should make use of a single message query.

Your answer:

d. Let (S'', V'') be as above, but sign each component using a different randomly selected secret key. That is, for $\ell = 3$, we have

$$S''\big((k_0, k_1, k_2), (m_0, m_1, m_2)\big) \coloneqq \big(\ S(k_0, m_0),\ S(k_1, m_1),\ S(k_2, m_2)\ \big)$$

Is (S'', V'') a secure MAC? If so, explain why. If not, describe an attack.

Your answer:

Problem 3. (*The danger of black-box cryptography*) **(20 points)**

In class we explained that developers should not implement cryptography themselves, but instead use a trusted library. In this question we explore the danger of using a cryptographic library that has been compromised by an attacker. The attacker's goal is to make it so that the output of every function looks perfectly correct, and yet the library is completely insecure.

a. Consider the symmetric encryption function provided by the library: `encrypt(key, msg)`. The function takes as input a 128-bit key buffer and a message buffer, and returns the encrypt-then-mac encryption of `msg` using `key`, where encryption is done using randomized counter mode and the AES-128 block cipher. A trusted library would choose the 128-bit nonce `iv` — used to initialize the counter in counter mode — as a random 128-bit value. However, this malicious library chooses the nonce `iv` as follows:

- First, it splits the key `key` into k_0 and k_1 where k_0 are the low 64 bits and k_1 are the top 64 bits of `key`.
- Next, it chooses a random 64-bit nonce r and a random bit b in $\{0, 1\}$.
- Finally, it sets the nonce `iv` as

$$\texttt{iv} \coloneqq \mathsf{AES}(k',\ r \| k_b)$$

where k' is an AES key that is hardcoded into the library executable code. Recall that $r \| k_b$ denotes the concatenation of r and k_b.

This nonce `iv` looks like a random 128-bit value. The library outputs the resulting ciphertext obtained by encrypting `msg` using `key` and this `iv`. It uses (randomized counter mode)-then-MAC to generate the ciphertext.

Show that an attacker who knows the hardcoded k', but does not know `key`, can decrypt every ciphertext generated by this library using `key`, with high probability, after observing some ten ciphertexts generated by this library using `key`.

Your answer:

b. Suppose the developer chooses to use nonce-based counter, that is, the developer uses the function `encrypt(key, iv, msg)` so that they specify the `iv` explicitly as an argument. Now encryption is a deterministic function of the arguments. Does the attack from part (a) still apply? Justify your answer.

Your answer:

c. Let's show that a similar attack to the one in part (a) can also be used to leak the secret signing key in the Schnorr signature scheme from Lecture 18. The scheme uses a group $\mathbb{G}$ of prime order q with generator $g \in \mathbb{G}$. Recall that an honest signer begins by choosing a random $\rho \leftarrow\!\!\$\, \mathbb{Z}_q$ and computing $R \leftarrow g^\rho$. Instead, the malicious library chooses ρ and R as follows:

- Split the secret 256-bit signing key into 32 bytes $b_0, b_1, \ldots, b_{31} \in \{0,1\}^8$.
- Compute $r \leftarrow H(m)$ where m is the message to be signed and $H : \mathcal{M} \to \{0,1\}^{128}$ is a public hash function.
- Choose a random $s \xleftarrow{R} \{0, 1, \ldots, 31\}$ and set $t \leftarrow (\langle s \rangle_5 \| b_s) \oplus \mathsf{AES}(k', r)[0 \ldots 12]$.
  Here $\langle s \rangle_5$ is the 5-bit binary representation of s, and $\mathsf{AES}(k', r)[0 \ldots 12]$ refers to the least significant 13 bits of $\mathsf{AES}(k', r)$. As before, k' is a secret key embedded in the library executable code.
- repeat the following two steps: $\rho \xleftarrow{R} \mathbb{Z}_q$, $R \leftarrow g^\rho \in \mathbb{G}$ until $R[0 \ldots 12] = t$.

The rest of the signing process is as in the Schnorr scheme using the ρ and R generated above. Note that R is a uniform group element, as with an honest Schnorr signer. It is easy to calculate this R from the final Schnorr signature.

Show that an attacker who knows k' can learn the secret 256-bit signing key after seeing about 100 message-signature pairs generated by this library.

Your answer:

Problem 4. (*Signcryption*) **(20 points)**

In a secure email system the sender often needs to sign an outgoing email with her secret key and encrypt the email with the recipient's public key for confidentiality. The combination of public-key encryption and signing is called *Signcryption*. One signcryption proposal uses a public-key encryption system (Gen_{enc}, E, D) and a signature scheme (Gen_{sig}, S, V). Every user generates a signing key pair $(\text{sk}_{\text{sig}}, \text{pk}_{\text{sig}})$ and an encryption key pair $(\text{sk}_{\text{enc}}, \text{pk}_{\text{enc}})$. When Alice wants to send a signed and encrypted (signcrypted) message m to Bob she obtains Bob's public encryption key $\text{pk}_{\text{enc}}^{(\text{bob})}$ and computes

$$ct \leftarrow E(\text{pk}_{\text{enc}}^{(\text{bob})},\ m) \quad \text{and} \quad \sigma \leftarrow S(\text{sk}_{\text{sig}}^{(\text{alice})},\ ct)\ .$$

Alice then sends $(ct, \sigma, \text{cert}_{\text{alice}})$ to Bob, where $\text{cert}_{\text{alice}}$ is Alice's certificate on her signing key. Bob checks Alice's certificate and signature σ and then decrypts ct to recover m. If the signature on ct is valid Bob believes the message m came from Alice. This method is called *encrypt-then-sign*.

Unfortunately, this simple scheme is problematic. An attacker, Charlie, could intercept the transmitted message and re-sign the ciphertext ct with his own signing key to obtain

$$\sigma' \leftarrow S(\text{sk}_{\text{sig}}^{(\text{charlie})},\ ct) \qquad (*)$$

Charlie then forwards $(ct, \sigma', \text{cert}_{\text{charlie}})$ to Bob. When Bob receives this signcrypted message he will incorrectly think that Alice's plaintext m is from Charlie. This can cause problems, for example, when Alice submits her signcrypted homework to the professor, Charlie intercepts Alice's email and re-signs it with his own key as in (*). Charlie then submits Alice's homework as his own and gets credit for Alice's work, even though Charlie never saw Alice's solution in the clear.

a. Propose a way to prevent the attack above while keeping the same encrypt-then-sign signcryption structure. You may assume that the encryption scheme is CCA secure.

Your answer:

b. Another difficulty with the proposal above is that Bob does not obtain Alice's signature on the message m. As a result he cannot easily show a third party (e.g. a judge) that Alice signed m. Propose a different signcryption method by which Bob obtains Alice's signature on the message m. Please be specific about how your proposal works.

Your answer:

c. Let us consider another Signcryption scheme based on Diffie-Hellman key exchange in a cyclic group $\mathbb{G}$ of prime order q with generator $g \in \mathbb{G}$. Every party in the system generates a secret key $\alpha \xleftarrow{R} \mathbb{Z}_q$ and a public key $\mathsf{pk} := h = g^\alpha \in \mathbb{G}$. In particular, Alice has a certified pk_a and the corresponding α_a, and Bob has a certified pk_b and the corresponding α_b. The signcryption scheme makes use of two more ingredients:

- a symmetric cipher (E, D) defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ that provides authenticated encryption,
- a hash function H that outputs a value in $\mathcal{K}$.

When Alice wants to send a message m to Bob she first obtains Bob's certificate, which includes Bob's public key $\mathsf{pk}_\mathrm{b} = h_\mathrm{b} = g^{\alpha_\mathrm{b}} \in \mathbb{G}$. Next, she computes

$$r \leftarrow h_\mathrm{b}^{\alpha_\mathrm{a}} \in \mathbb{G}, \quad k \leftarrow H(\mathsf{cert}_\mathrm{a}, \mathsf{cert}_\mathrm{b}, r) \in \mathcal{K}, \quad c \leftarrow E(k, m)$$

and sends $ct := (c, \mathsf{cert}_\mathrm{a})$ to Bob. Recall that cert_a contains Alice's public key $h_\mathrm{a} = g^{\alpha_\mathrm{a}} \in \mathbb{G}$.

Explain how Bob decrypts the ciphertext ct. Make sure to explain everything that Bob needs to check. In addition, briefly explain why a successful decryption convinces Bob that the received message came from Alice. An informal explanation is sufficient.

Your answer:

d. Suppose an attacker was able to obtain the sender's (Alice's) secret key, but they do not have the receiver's (Bob's) secret key. Informally, we say that a Signcryption scheme provides forward secrecy if such an attacker cannot decrypt ciphertexts generated by Alice.

- Does the scheme from part (a) provide forward secrecy?
- Does the scheme from part (b) provide forward secrecy?
- Does the scheme from part (c) provide forward secrecy?

Your answer:

Note: real-world signcryption is specified in a standard called HPKE. The authentication modes of HPKE provide various forms of signcryption.

Problem 5. (*Two-time hash-based signatures*) **(20 points)**

a. Explain why the Lamport signature scheme is not secure against an adversary who can request two signatures with respect to the same public key.

Your answer:

b. Our goal in this question is to build a (stateless) signature that remains secure against a 2-time adversary. First, please define the security game for a 2-time secure signature scheme.

Your answer:

c. Let's build a new (stateless) hash-based signature scheme that is secure even if the adversary requests two signatures. We will need two ingredients:

- a one way function $f : \mathcal{X} \to \mathcal{Y}$,
- a hash function $H : \mathcal{M} \to \mathcal{P}(n)$, where $\mathcal{P}(n)$ denotes the set of all subsets of $\{1, 2, \ldots, n\}$. The set $\mathcal{P}(n)$ contains 2^n elements.

We will treat n as a parameter of the scheme. Using these ingredients the signature scheme works as follows:

- *KeyGen*(): choose random $x_1, \ldots, x_n \xleftarrow{R} \mathcal{X}$ and compute $y_i \leftarrow f(x_i)$ for $i = 1, \ldots, n$. Output
$$\mathsf{pk} \coloneqq (y_1, \ldots, y_n) \in \mathcal{Y}^n, \qquad \mathsf{sk} \coloneqq (x_1, \ldots, x_n) \in \mathcal{X}^n$$
- *Sign*(sk, m): First set $S \leftarrow H(m)$ so that $S \subseteq \{1, \ldots, n\}$. Next, output $\sigma \coloneqq \{x_i \mid i \in S\}$.
- *Verify*(pk, m, σ): First set $S \leftarrow H(m)$ so that $S \subseteq \{1, \ldots, n\}$. Let $\sigma = \{\sigma_i \mid i \in S\}$. Accept the signature if and only if $f(\sigma_i) = y_i$ for all $i \in S$.

Suppose that the hash function H satisfies the following strong property: for every three messages $m, m_1, m_2 \in \mathcal{M}$ we have that
$$H(m) \not\subseteq H(m_1) \cup H(m_2).$$
Such a hash function is said to be 2-cover free. Briefly explain why this property is sufficient to ensure that the signature scheme satisfies your definition from part (b), assuming that f is a one-way function.

Your answer:

One can construct the required 2-cover free function, but we will leave that for another day.

d. Explain why the scheme from part (c) may be insecure against an adversary that can request three signatures.

Your answer: