**CS 6222: Introduction to Cryptography** **Spring 2020**

# Take-Home Midterm

**Due:** March 27, 2020 at 5pm (Submit on Gradescope) — **No Late Days!** **Instructor:** David Wu

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

https://www.cs.virginia.edu/dwu4/courses/sp20/static/homework.tex

You must submit your problem set via Gradescope. Please use course code **MB84NW** to sign up. The exam is divided into two sections: one on symmetric cryptography and one on public-key cryptography. Each section contains three problems. You only need to answer **two** of the problems from each section (for a total of **four** problems). If you answer more than two problems from any section, only the first two will be graded. You may cite any result from lecture or the course lecture notes without proof.

**Collaboration Policy.** This is an *individual* assignment. You are not allowed to collaborate with anyone on this problems and you are not permitted to search online for solutions to these problems. If you do consult external sources (that cannot include solutions), you must cite them in your submission.

## 1 Symmetric Cryptography

**Instructions.** Answer any **two** of the three problems in this section. If you answer more than two problems, only the first two will be graded.

**Problem 1-1. Ciphertext Expansion [25 points].** In all of the CPA-secure encryption schemes we have discussed, the length of the ciphertext is greater than the length of the plaintext length. In this problem, we will show that this is necessary. Let $(\mathsf{Encrypt}, \mathsf{Decrypt})$ be a symmetric encryption scheme with message space $\{0,1\}^n$ and ciphertext space $\{0,1\}^m$.

(a) Suppose that $n = m$. Show that $(\mathsf{Encrypt}, \mathsf{Decrypt})$ cannot be CPA-secure.

(b) Suppose that $m = n + \ell$ for some $\ell < n/2$. Describe a CPA adversary that makes $O(2^{\ell/2})$ queries in the CPA-security game and distinguishes with *constant* probability. For simplicity (though not necessary), your may assume that for any choice of key $k$ and message $m$, the output distribution of $\mathsf{Encrypt}(k, m)$ is uniform over a collection of up to $2^\ell$ possible ciphertexts, where the distribution is over the encryption randomness. Be sure to fully describe your attack and give a *precise* analysis of the advantage (note that it suffices to lower bound the advantage by a constant).

**Problem 1-2. CBC-MAC [25 points].** Let $F \colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a secure block cipher, and let $F_{\mathsf{CBC}} \colon \mathcal{K} \times (\{0,1\}^n)^{\le L} \to \{0,1\}^n$ be the *raw-CBC MAC* from lecture. In lecture, we said that raw-CBC is a secure PRF (and thus a secure MAC) for fixed-length messages (and more generally, *prefix-free messages*).

(a) Recall that raw-CBC uses a *fixed* IV (the all-zeroes string). Consider a *randomized* construction where the signing algorithm samples a random $\mathsf{IV} \xleftarrow{\text{R}} \{0,1\}^n$ and computes the MAC on a message $m = (m_1, \dots, m_\ell) \in (\{0,1\}^n)^\ell$ as $t \leftarrow F_{\mathsf{CBC}}(k, (m_1 \oplus \mathsf{IV}, m_2, \dots, m_\ell))$. The tag is the pair $(\mathsf{IV}, t)$. Show that randomized raw-CBC is insecure, even for signing *fixed-length* messages.

(b) Suppose we apply the randomized construction from Part (a) to *encrypted CBC-MAC*; that is, the MAC on $m = (m_1, \ldots, m_t) \in (\{0,1\}^n)^\ell$ is $(\mathsf{IV}, t)$ where $\mathsf{IV} \xleftarrow{\text{R}} \{0,1\}^n$, $t \leftarrow F(k_2, F_{\mathsf{CBC}}(k_1, (m_1 \oplus \mathsf{IV}, m_2, \ldots, m_\ell)))$, and $k_1, k_2$ are independent keys. Is this construction a secure MAC? Give either a proof or an attack.

(c) Suppose we use "encrypt-then-MAC" to construct an authenticated encryption scheme for a *fixed-length* message space $\{0,1\}^n$ (i.e., one-block messages) by combining randomized counter-mode encryption with raw-CBC MAC,[1] except we use the *same* key for both the encryption scheme and the MAC. Namely, an encryption of $m \in \{0,1\}^n$ consists of the tuple $(\mathsf{IV}, c, t)$ where $\mathsf{IV} \xleftarrow{\text{R}} \{0,1\}^n$, $c \leftarrow F(k, \mathsf{IV}) \oplus m$, and $t \leftarrow F_{\mathsf{CBC}}(k, (\mathsf{IV}, c))$. Show that the resulting scheme is neither CPA-secure *nor* provides ciphertext integrity (i.e., construct two separate adversaries). *Remark:* This shows that reusing the same key for different cryptographic primitives can have severe consequences!

(d) Does the "encrypt-then-MAC" construction from Part (c) provide authenticated encryption for the *fixed-length* message space $\{0,1\}^n$ if we use independent and uniformly random keys for the randomized counter-mode encryption and raw-CBC MAC? Briefly justify your answer.

**Problem 1-3. Encrypting Twice, Revisited [25 points].** Let $(\mathsf{Encrypt}, \mathsf{Decrypt})$ be a symmetric authenticated encryption scheme with key-space $\mathcal{K} = \{0,1\}^\lambda$. Consider the encrypt-twice cipher $(\mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$ with *independent* keys where $\mathsf{Encrypt}_2((k_1, k_2), m) := \mathsf{Encrypt}(k_2, \mathsf{Encrypt}(k_1, m))$ and

$$\mathsf{Decrypt}_2((k_1, k_2), c) := \begin{cases} \mathsf{Decrypt}(k_1, \mathsf{Decrypt}(k_2, c)) & \mathsf{Decrypt}(k_2, c) \neq \bot \\ \bot & \text{otherwise.} \end{cases}$$

(a) Show that $(\mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$ is still an authenticated encryption scheme even if the adversary learns $k_1$ (but has no information about $k_2 \xleftarrow{\text{R}} \mathcal{K}$). Remember to show both CPA-security *and* ciphertext integrity. To model knowledge of $k_1$, you can assume that the adversary is given $k_1$ at the beginning of the CPA-security and ciphertext integrity games.

(b) Show that $(\mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$ is no longer an authenticated encryption scheme if the adversary learns $k_2$ (but has no information about $k_1 \xleftarrow{\text{R}} \mathcal{K}$). To model knowledge of $k_2$, you can assume that the adversary is given $k_2$ at the beginning of the CPA-security and ciphertext integrity games.

(c) State how to use $(\mathsf{Encrypt}, \mathsf{Decrypt})$ to construct an authenticated encryption where keys are $(k_1, k_2) \in \mathcal{K}^2$ such that the cipher remains an authenticated encryption scheme even if the adversary learns *any* one of the keys (but has no information about the the other). Your construction should not rely on any primitive other than $(\mathsf{Encrypt}, \mathsf{Decrypt})$. *For this part only, you do **not** need to provide a proof of security for your construction, though you are welcome to do so (it is not difficult, but a bit tedious).*

## 2 Public-Key Cryptography

**Instructions.** Answer any **two** of the three problems in this section. If you answer more than two problems, only the first two will be graded. Throughout this section, please remember to justify the existence of any inverses you compute (and in cases where there is ambiguity, say what algebraic structure you are working over).

[1] A variant where we combine counter-mode encryption with *encrypted CBC-MAC* yields the CCM mode of operation—which provides authenticated encryption.

**Problem 2-1. Signatures from Discrete Log [25 points].** Let $\mathbb{G}$ be a group of prime order $p$ with generator $g$. Consider the following digital signature algorithm where the message space is $\mathbb{Z}_p$:

- $\mathsf{Setup}$ : Sample $x, y \xleftarrow{\text{R}} \mathbb{Z}_p^*$ and compute $h \leftarrow g^x$, $z \leftarrow g^y$. Output the verification key $\mathsf{vk} = (g, h, z)$ and the signing key $\mathsf{sk} = (g, h, z, x, y)$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: On input the signing key $\mathsf{sk} = (g, h, z, x, y)$ and a message $m \in \mathbb{Z}_p$, compute and output the signature $\sigma \in \mathbb{Z}_p$ such that $z = g^m h^\sigma$.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: On input the verification key $\mathsf{vk} = (g, h, z)$, a message $m \in \mathbb{Z}_p$ and a signature $\sigma \in \mathbb{Z}_p$, output 1 if $z = g^m h^\sigma$ and 0 otherwise.

(a) Show how to *efficiently* implement the signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ (i.e., give an efficient algorithm to compute $\sigma$ from $\mathsf{sk} = (g, h, z, x, y)$ and $m$).

(b) Show that under the discrete log assumption in $\mathbb{G}$, this signature scheme satisfies *selective one-time* security. In the selective one-time security game, the adversary is allowed to make a *single* signing query on an arbitrary message $m \in \mathbb{Z}_p$ *before* seeing the verification key $\mathsf{vk}$. The adversary wins if it outputs a pair $(m^*, \sigma^*)$ such that $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$ and $m^* \neq m$. Specifically, the selective one-time signature security game proceeds as follows:

$$\begin{tikzcd}
\text{Adversary} & & \text{Challenger} \\
{} \arrow[rr, "m \in \mathbb{Z}_p"] & & {} \\
{} & & {} \arrow[ll, "\mathsf{vk},\ \sigma"'] \\
{} \arrow[rr, "m^*,\ \sigma^*"] & & {}
\end{tikzcd}$$

Here, the challenger samples $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, and the adversary wins if $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$ and $m \neq m^*$. To show this, you should show how to use an efficient adversary $\mathcal{A}$ for the selective one-time signature security game to construct an efficient adversary $\mathcal{B}$ for the discrete log game (for $\mathbb{G}$).

(c) Show that an adversary who sees two message-signature pairs $(m_1, \sigma_1)$ and $(m_2, \sigma_2)$ under $\mathsf{vk}$ where $m_1 \neq m_2$ can forge signatures on arbitrary messages $m \in \mathbb{Z}_p$. This shows that this scheme can only be used to sign a *single* message.

Later on in this class, we will show how to construct signature schemes based on the discrete log assumption that can be used to sign *multiple* messages.

**Problem 2-2. Computing on Encrypted Data [25 points].** Let $N = pq$ be an RSA modulus where $\gcd(N, \varphi(N)) = 1$. Consider the following public-key encryption scheme with message space $\mathbb{Z}_N$. The public key $\mathsf{pk} = N$ is the RSA modulus $N = pq$ and the secret key $\mathsf{sk}$ is the factorization $\mathsf{sk} = (p, q)$. Let $g = 1 + N \in \mathbb{Z}_{N^2}^*$. To encrypt a message $m \in \mathbb{Z}_N$, sample $h \xleftarrow{\text{R}} \mathbb{Z}_{N^2}^*$ and compute $c \leftarrow g^m h^N \in \mathbb{Z}_{N^2}^*$.

(a) Show that the discrete logarithm assumption base $g$ in $\mathbb{Z}_{N^2}$ is easy. Namely, give an efficient algorithm that takes as input $(g, h)$ where $h = g^x$ for some $x \in \mathbb{Z}_N$, and outputs $x$. **Hint:** Use the binomial theorem: $(a + b)^k = \sum_{i=0}^{k} \binom{k}{i} a^i b^{k-i}$.

(b) Show how to *efficiently* implement the decryption algorithm $\mathsf{Decrypt}(\mathsf{sk}, c)$. Namely, describe an efficient algorithm that given the secret key $\mathsf{sk} = (p, q)$ and a ciphertext $c = g^m h^N$, outputs the message $m \in \mathbb{Z}_N$. You may use the fact that $\varphi(N^2) = N\varphi(N)$. **Hint:** Remember that the decrypter can compute $\varphi(N) = (p-1)(q-1)$ from the secret key $\mathsf{sk} = (p, q)$.

(c) Show that this public-key encryption scheme is semantically secure assuming that no efficient adversary is able to distinguish the following two distributions:

$$(N, u) \quad \text{and} \quad (N, v),$$

where $N = pq$ is an RSA modulus, $u \xleftarrow{\text{R}} \mathbb{Z}_{N^2}^*$ and $v \xleftarrow{\text{R}} \{h \in \mathbb{Z}_{N^2}^* : h^N\}$. Namely, show that the above encryption scheme is semantically secure assuming that it is hard to distinguish random values in $\mathbb{Z}_{N^2}^*$ from random $N^{\text{th}}$ powers in $\mathbb{Z}_{N^2}^*$.

(d) Show that given the public key $\mathsf{pk}$ and two ciphertexts $c_1 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_1)$, $c_2 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_2)$, there is an efficient algorithm that outputs a new ciphertext $c$ where $\mathsf{Decrypt}(\mathsf{sk}, c) = m_1 + m_2 \in \mathbb{Z}_N$. Your algorithm should only depend on *public* parameters and *not* the value of the messages $m_1, m_2$. *Remark:* This is an example of an encryption scheme that supports computation on *encrypted* values.

**Problem 2-3. Authenticated Key Exchange [25 points].** Consider the following protocol for authenticated key exchange (AKE) with mutual (i.e., two-sided) authentication. Both the client and the server have a public/private key-pair $(\mathsf{vk}_C, \mathsf{sk}_C)$ and $(\mathsf{vk}_S, \mathsf{sk}_S)$ for a digital signature scheme, respectively. They also have certificates $\mathsf{cert}_C$ and $\mathsf{cert}_S$ that authenticate $\mathsf{vk}_C$ and $\mathsf{vk}_S$, respectively. The AKE protocol operates over a group $\mathbb{G}$ of prime order $p$ and generator $g$. The client samples a fresh $x \xleftarrow{\text{R}} \mathbb{Z}_p$ and the server samples a fresh $y \xleftarrow{\text{R}} \mathbb{Z}_p$ in each invocation of the protocol:

$$\begin{tikzcd}
\text{Client} & & \text{Server} \\
{} \arrow[rr, "{g^x,\ \mathsf{cert}_C}"] & & {} \\
{} & & {} \arrow[ll, "{g^y,\ \mathsf{cert}_S,\ \mathsf{Sign}(\mathsf{sk}_S, (\mathsf{cert}_C, g^x, g^y))}"'] \\
{} \arrow[rr, "{\mathsf{Sign}(\mathsf{sk}_C, (\mathsf{cert}_S, g^x, g^y))}"] & & {}
\end{tikzcd}$$

In the second step, the client validates the signature with respect to the verification key contained in $\mathsf{cert}_S$ before computing its third message. At the end of the protocol, if all of the signatures verify (with respect to the verification keys identified by the certificates), the client and server computes the shared key as $k \leftarrow H(g, g^x, g^y, g^{xy})$. Moreover, the client outputs the party identified by $\mathsf{cert}_S$ as its peer in the connection and the server outputs the party identified by $\mathsf{cert}_C$ as its peer. Throughout this problem, you should consider an *active* network adversary that is allowed to register a certificate of its own (i.e., the adversary has a certificate $\mathsf{cert}_A$ for its identity $A$, which is *different* from both the client's identity $C$ associated with $\mathsf{cert}_C$ and the server's identity $S$ associated with $\mathsf{cert}_S$).

(a) Suppose the server does not sign $\mathsf{cert}_C$ in its reply to the client. Namely, the server computes $\mathsf{Sign}(\mathsf{sk}_S, (g^x, g^y))$ instead of $\mathsf{Sign}(\mathsf{sk}_S, (\mathsf{cert}_C, g^x, g^y))$. Show that there is an identity misbindinng attack on this protocol.

(b) Suppose the client only signed the server's certificate and not the Diffie-Hellman shares in the final message. Namely, the client computes $\mathsf{Sign}(\mathsf{sk}_C, \mathsf{cert}_S)$ instead of $\mathsf{Sign}(\mathsf{sk}_C, (\mathsf{cert}_S, g^x, g^y))$. Show that an adversary is able to establish a session with the server such that the adversary knows the shared key $k$, but the server thinks it is communicating with the party identified by $\mathrm{cert}_C$ (i.e., the client). **Hint:** Remember that an active network adversary is allowed to observe (and tamper with) *multiple* interactions between the client and the server.

(c) Suppose that the client signed its Diffie-Hellman share in its first message, and dropped the third message entirely. Namely, the client's first message is now $(g^x, \mathrm{cert}_C, \mathsf{Sign}(\mathsf{sk}_C, g^x))$ and the overall protocol now completes in two rounds. Show that there is an identity misbinding attack on this protocol.

(d) Suppose that instead of signing the pair $(g^x, g^y)$, the client and server instead signed $g^{xy}$. Explain why this is a bad idea.

This exercise illustrates that designing AKE protocols is very delicate, and simple modifications will lead to broken designs.