CS 6222: Introduction to Cryptography **Spring 2021**

Take-Home Final Exam

Due: May 14, 2021 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/sp21/static/homework.tex

You must submit your problem set via Gradescope. The exam is divided into three sections. Each section contains two problems. You only need to answer **one** problem from each section (for a total of **three** problems). If you answer more than one problem from any section, only the first one 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 (these cannot include solutions), you must cite them in your submission.

1 Symmetric Cryptography

Instructions. Answer **one** of the two problems in this section. If you answer both problems, only the first one will be graded.

Problem 1-1: Cryptographic Combiners [25 points]. Suppose we have two candidate constructions Π_1, Π_2 of a cryptographic primitive, but we are not sure which of them is secure. A cryptographic combiner provides a way to use Π_1 and Π_2 to obtain a new construction Π such that Π is secure if at least one of Π_1, Π_2 is secure (*without* needing to know which of Π_1 or Π_2 is secure). Combiners can be used to "hedge our bets" in the sense that a future compromise of one of Π_1 or Π_2 would not compromise the security of Π. In this problem, we will study candidate combiners for different cryptographic primitives.

(a) Let $G_1, G_2 \colon \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be arbitrary PRG candidates. Define the function $G(s_1, s_2) := G_1(s_1) \oplus G_2(s_2)$. Prove or disprove: if at least one of G_1 or G_2 is a secure PRG, then G is a secure PRG.

(b) Let $H_1, H_2 \colon \{0,1\}^* \to \{0,1\}^\lambda$ be arbitrary collision-resistant hash function candidates. Define the function $H(x) := H_1(H_2(x))$. Prove or disprove: if at least one of H_1 or H_2 is collision-resistant, then H is collision-resistant.

(c) Let $(\mathsf{Sign}_1, \mathsf{Verify}_1)$ and $(\mathsf{Sign}_2, \mathsf{Verify}_2)$ be arbitrary MAC candidates. Define $(\mathsf{Sign}, \mathsf{Verify})$ as following:

- $\mathsf{Sign}((k_1, k_2), m)$: Output (t_1, t_2) where $t_1 \leftarrow \mathsf{Sign}_1(k_1, m)$ and $t_2 \leftarrow \mathsf{Sign}_2(k_2, m)$.
- $\mathsf{Verify}((k_1, k_2), (t_1, t_2))$: Output 1 if $\mathsf{Verify}_1(k_1, m, t_1) = 1 = \mathsf{Verify}_2(k_2, m, t_2)$ and 0 otherwise.

Prove or disprove: if at least one of $(\mathsf{Sign}_1, \mathsf{Verify}_1)$ or $(\mathsf{Sign}_2, \mathsf{Verify}_2)$ is a secure MAC, then $(\mathsf{Sign}, \mathsf{Verify})$ is a secure MAC.

Problem 1-2: Homomorphism and Security [25 points]. Let $F\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a candidate PRF construction.

(a) Suppose that for all $k, x, c \in \{0,1\}^\lambda$, $F(k, x \oplus c) = F(k, x) \oplus c$. Show that F cannot be a secure PRF.

(b) Suppose that for all $k, x, c \in \{0,1\}^\lambda$, $F(k \oplus c, x) = F(k, x) \oplus c$. Show that F cannot be a secure PRF.

Let $(\mathsf{Encrypt}, \mathsf{Decrypt})$ be a symmetric encryption scheme with message space $\{0,1\}^n$.

(c) Suppose there is an efficient algorithm $\mathsf{Combine}(\mathsf{ct}_1, \mathsf{ct}_2)$ that takes ciphertexts $\mathsf{ct}_1 \leftarrow \mathsf{Encrypt}(k, x_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{Encrypt}(k, x_2)$ and outputs a new ciphertext ct' where $\mathsf{Decrypt}(k, \mathsf{ct}') = x_1 \oplus x_2$. Note that Combine is a *public* algorithm (it does not require knowledge of the secret key). Show that $(\mathsf{Encrypt}, \mathsf{Decrypt})$ cannot be CCA-secure.

(d) Show how to construct a CPA-secure symmetric encryption scheme that supports the above Combine algorithm. Namely, describe the key-space, Encrypt, Decrypt, and the Combine algorithms for your construction. Your construction can use a secure PRF $F\colon \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ (but no other primitives or assumptions). Note that ciphertexts output by Combine can be *longer* than those output by Encrypt (and correspondingly, your Decrypt algorithm may behave differently depending on the length of the ciphertext). Prove that your construction is CPA-secure (**Hint:** You can cite a theorem from class here).

2 Public-Key Cryptography

Instructions. Answer **one** of the two problems in this section. If you answer both problems, only the first one will be graded.

Problem 2-1: Hash Functions from Discrete Log [25 points]. Let $\mathbb{G}$ be a group of prime order p with generator g. Sample $h_1, \dots, h_n \xleftarrow{R} \mathbb{G}$ and define the hash function $H_{h_1,\dots,h_n}\colon \mathbb{Z}_p^n \to \mathbb{G}$ as follows:

$$H_{h_1,\dots,h_n}(x_1, \dots, x_n) := h_1^{x_1} h_2^{x_2} \cdots h_n^{x_n} \in \mathbb{G}.$$

(a) Show that $H_{h_1,\dots,h_n}$ is collision resistant under the discrete log assumption in $\mathbb{G}$. Specifically, in the (keyed) collision-resistant hashing security game, the adversary is first given $h_1, \dots, h_n \leftarrow \mathbb{Z}_p$ and it succeeds if it outputs $(x_1, \dots, x_n) \neq (x_1', \dots, x_n')$ such that $H_{h_1,\dots,h_n}(x_1, \dots, x_n) = H_{h_1,\dots,h_n}(x_1', \dots, x_n')$. In the discrete log security game, the adversary is given $h \xleftarrow{R} \mathbb{G}$, and it wins if it outputs $x \in \mathbb{Z}_p$ such that $h = g^x$. **Hint:** Consider a reduction algorithm that starts by guessing the index $i^* \in [n]$ (uniformly at random) where $x_{i^*} \neq x_{i^*}'$. Show that your reduction algorithm succeeds whenever the guess is correct. Remember to compute the advantage of your reduction algorithm (for breaking the discrete log assumption).

(b) Show that the function $H_{h_1,\dots,h_n}$ has a *trapdoor* that can be used to sample pre-images. Specifically, show that if someone knew the discrete logs of $h_1, \dots, h_n$ (i.e., $z_i \in \mathbb{Z}_p$ where $h_i = g^{z_i}$ for each $i \in [n]$), then for any $t \in \mathbb{Z}_p$, they can find a pre-image $(x_1, \dots, x_n) \in \mathbb{Z}_p^n$ such that $H_{h_1,\dots,h_n}(x_1, \dots, x_n) = g^t$.

Problem 2-2: Hardness of RSA [25 points]. Recall that an RSA challenge consists of a modulus $N = pq$, an exponent e, and an input $y \in \mathbb{Z}_N^*$. The goal is to compute $x \in \mathbb{Z}_N^*$ such that $x^e = y \pmod{N}$.

(a) Consider an RSA challenge (N, e, y), and suppose you have an efficient algorithm that takes as input (N, e, y) and outputs values $a \in \mathbb{Z}_N^*$ and $b \in \mathbb{Z}$ such that $a^e = y^b \pmod{N}$ and $\gcd(b, e) = 1$. Show how to use (a, b) to efficiently compute a solution $x \in \mathbb{Z}_N^*$ to the RSA challenge (N, e, y). **Hint:** Recall that $\gcd(b, e) = 1$ means there exist integers $s, t \in \mathbb{Z}$ such that $bs + et = 1$.

(b) Fix an RSA modulus $N = pq$ and exponent e. Suppose you have found an algorithm $\mathcal{A}$ that solves the RSA problem on all inputs in some set $S \subseteq \mathbb{Z}_N^*$ where $|S|/|\mathbb{Z}_N^*| = \varepsilon$ and runs in time $\text{poly}(\log N)$. Namely, for every $y \in S$, $\mathcal{A}(y, e, N)$ outputs $x \in \mathbb{Z}_N^*$ where $x^e = y \bmod N$. Show how to use $\mathcal{A}$ to construct an efficient algorithm that runs in time $\text{poly}(\log N, 1/\varepsilon)$ and on *any* input $y \in \mathbb{Z}_N^*$, successfully outputs x such that $x^e = y \bmod N$ with probability at least $1/2$. Your algorithm must *always* terminate in $\text{poly}(\log N, 1/\varepsilon)$ time. **Hint:** A random value $z \xleftarrow{\text{R}} \mathbb{Z}_N^*$ will satisfy $z \in S$ with probability ε. You can also use the fact that $(1 - \varepsilon)^n \leq e^{-\varepsilon n}$ for all positive integers n.

3 Cryptographic Protocols and Lattice-Based Cryptography

Instructions. Answer **one** of the two problems in this section. If you answer both problems, only the first one will be graded.

Problem 3-1: Cryptographic Protocols and Definitions [25 points].

(a) Consider the following protocol for proving knowledge of the factorization of an RSA modulus $N = pq$. Both the prover and the verifier know N. The proof consists of a single message where the prover sends the factorization (p, q) to the verifier and the verifier accepts if p, q are prime and $N = pq$. Show that this protocol is zero-knowledge *if and only if* there exists a polynomial time algorithm for factoring.

(b) Recall the Schnorr signature scheme from class. We work over a group $\mathbb{G}$ of prime order p and with generator g and a hash function H (modeled as a random oracle). The verification key is a pair (g, h) and the signing key is $x \in \mathbb{Z}_p$ where $h = g^x$. A signature on a message m is a pair $\sigma = (u, z)$ where $z = r + cx$ such that $g^z = uh^c$ and $c \leftarrow H(g, h, u, m)$. Consider a variant of this signature scheme where we forget to hash u when computing c (i.e., we set $c \leftarrow H(g, h, m)$). Show that this signature scheme is insecure.

(c) Fix a modulus q and let χ be a distribution over $\mathbb{Z}_q$ where $\Pr[x \leftarrow \chi : |x| < B] = 1$. Show that hardness of $\mathsf{LWE}_{n,m,q,\chi}$ implies hardness of $\mathsf{SIS}_{n,m,q,\beta}$ whenever $q > 4\beta B m$.

Problem 3-2: CCA-Attack on Regev Encryption [25 points]. In this problem, we will show that the basic Regev encryption scheme from lecture is *not* CCA-secure. In fact, we will show the stronger property that a CCA adversary is able to recover the secret key itself. Recall that in Regev encryption, the secret key is a vector $\mathbf{s} \in \mathbb{Z}_q^n$ and the public key is a pair $(\mathbf{A}, \mathbf{b})$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{b} \in \mathbb{Z}_q^m$. An encryption of a bit $b \in \{0, 1\}$ consists of a pair $(\mathbf{Ar}, \mathbf{b}^\mathsf{T}\mathbf{r} + b \cdot \lfloor q/2 \rfloor)$. To decrypt a ciphertext $(\mathbf{u}, v)$, the decryption algorithm computes $z = v - \mathbf{s}^\mathsf{T}\mathbf{u} \bmod q$ and outputs 1 if $q/4 \leq z \leq 3q/4$ and 0 otherwise.

(a) For $z \in \mathbb{Z}_q$, define the function $f_z \colon \mathbb{Z}_q \to \{0,1\}$, where $f_z(x) = 1$ if $q/4 \leq (z + x \bmod q) \leq 3q/4$ and 0 otherwise. Construct an algorithm that given $O(\log q)$ queries to f_z recovers the value of $z \in \mathbb{Z}_q$. Your algorithm is allowed to make arbitrary queries to f_z. Prove the correctness of your algorithm.

(b) Using your algorithm from Part (a), show that an adversary playing the CCA-security game for Regev encryption is able to *recover* the secret key using at most $O(n \log q)$ decryption queries.