Homework 2

1 Problem 1 (30 points)

In this problem, we will formalize the notion of a stateful encryption scheme. A stateful encryption scheme operates on four spaces: a key space $\mathcal{K}$, a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, and a *state* space $\mathcal{S}$. We will take the message space to be $\{0,1\}^{\leq n}$ for some integer n. The scheme consists of three functions $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. Init takes no input, and produces a state state_0. Enc takes as input a key $k \in \mathcal{K}$, message $m \in \mathcal{M}$, and state $\text{state} \in \mathcal{S}$, and produces a ciphertext $c \in \mathcal{C}$ as well as a new state state'. Decryption takes as input a key k, ciphertext c, and state state, and produces a plaintext $m \in \mathcal{M}$ and a new state state'.

For this problem, we will only consider one-way communication from Alice to Bob, and assume all ciphertexts are received in the order they were sent. Alice and Bob share a secret key k. To begin, Alice runs $\text{state}_0 \leftarrow \mathsf{Init}()$ and Bob separately runs $\text{state}_0 \leftarrow \mathsf{Init}()$ (we will assume $\mathsf{Init}()$ is deterministic so that the output is always identical). They each set their internal state to state_0.

To encrypt the first message $m^{(1)}$, Alice will run $(c^{(1)}, \text{state}_1) \leftarrow \mathsf{Enc}(k, m^{(1)}, \text{state}_0)$, and then send $c^{(1)}$ to Bob. Alice will also update her internal state to state_1. Bob will run $(m'^{(1)}, \text{state}'_1) \leftarrow \mathsf{Dec}(k, c^{(1)}, \text{state}_0)$, and set his internal state to state'_1. To encrypt the second message $m^{(2)}$, Alice will run $(c^{(2)}, \text{state}_2) \leftarrow \mathsf{Enc}(k, m^{(2)}, \text{state}_1)$, and then send $c^{(2)}$ to Bob. Alice will also update her internal state to state_2. Bob will run $(m'^{(2)}, \text{state}'_2) \leftarrow \mathsf{Dec}(k, c^{(2)}, \text{state}_1)$, and set his internal state to state'_2. This process continues until Alice stops sending messages to Bob.

We will only ask for correctness and security as long as at most n bits are ever encrypted.

(a) Devise a precise definition for correctness. Informally justify why you defined things the way you do. There are multiple possible definitions, any are fine as long as they handle any sequence of messages (provided the sum of the lengths of messages never exceeds n bits) and are compatible with part (c) below.

(b) Devise a notion for perfect secrecy. Again, your definition should handle any sequence of messages provided the sum of the lengths of messages never exceeds n bits.

(c) Show how to modify the one-time pad so that it satisfies the correctness and secrecy notion above.

2 Problem 2 (10 points)

Suppose you are given a stateless encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ with key space $\mathcal{K}_\lambda$ and message space $\mathcal{M}$. Suppose further that the scheme is statistically secure for up to ℓ messages.

Using $(\mathsf{Enc}, \mathsf{Dec})$, devise a new encryption scheme $(\mathsf{Enc}', \mathsf{Dec}')$ with key space $\mathcal{K}_\lambda$ and message space $\mathcal{M}^\ell$ (that is, the messages for the new scheme consist of tuples of ℓ messages for the original scheme) that is statistically secure for a *single* message. Prove the scheme secure, assuming the statistical security of $(\mathsf{Enc}, \mathsf{Dec})$.

3 Problem 3 (15 points)

In the following, $g(\lambda)$ and $h(\lambda)$ are negligible functions, and p is a polynomially-bounded function. For each function below, decide if the function is guaranteed to be negligible. If so, prove it. If not, show an example of negligible g and h and polynomially bounded p such that the function below is not negligible.

(a) $f_a(\lambda) = \max\big(g(\lambda), h(\lambda)\big)$.

(b) $f_b(\lambda) = \max\big(g(\lambda), \frac{1}{p(\lambda)}\big)$.

(c) $f_c(\lambda) = \min\big(g(\lambda), h(\lambda)\big)$.

(d) $f_d(\lambda) = \min\big(g(\lambda), \frac{1}{p(\lambda)}\big)$.

(e) $f_e(\lambda) = g(\lambda) + h(\lambda)$.

(f) $f_f(\lambda) = p(\lambda)h(\lambda)$.

(g) $f_g(\lambda) = g(\lambda)^{1/2}$.

(h) $f_h(\lambda) = g(\lambda)^{\frac{1}{\log \lambda}}$.

4 Problem 4 (25 points)

Suppose $(\mathsf{Enc}, \mathsf{Dec})$ is an encryption scheme that is computationally secure for a single message, with key space $\mathcal{K}_\lambda$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}_\lambda$. Assume $\mathcal{M} = \{0,1\}^n$ for some integer n.

Which of the following encrpytion algorithms are guaranteed to represent computationally secure schemes? For schemes that are guaranteed to be computationally secure for any secure $(\mathsf{Enc}, \mathsf{Dec})$, prove security using a reduction. For schemes that are not secure for any $(\mathsf{Enc}, \mathsf{Dec})$, provide an attack. It may be that for some $(\mathsf{Enc}, \mathsf{Dec})$ the scheme is secure, but for others it is insecure. In this case both (1) give an example of a secure $(\mathsf{Enc}, \mathsf{Dec})$ that yields an insecure scheme, and provide an attack, and also (2) provide an example of a secure $(\mathsf{Enc}, \mathsf{Dec})$ that yields a secure scheme and prove security. Pathological examples are allowed, and if needed you may use a secure PRG as a building block for the examples.

You do not need to explain how to decrypt.

(a) $\mathsf{Enc}_a(k, m) = \big(\mathsf{Enc}(k, m), m_{[1]}\big)$. Here, $m_{[1]}$ is the first of m.

(b) $\mathsf{Enc}_b(k, m) = \big(\mathsf{Enc}(k, r), \mathsf{Enc}(r, m)\big)$. Here, r is chosen randomly from $\mathcal{K}_\lambda$.

(c) $\mathsf{Enc}_c(k, m) = \mathsf{Enc}\,(k, (m, r))$. Here, the message space for Enc_c is $\{0, 1\}^{n/2}$ and r is a random $n/2$-bit message.

(d) $\mathsf{Enc}_d(k, m) = \mathsf{Enc}(k, m) \oplus \mathsf{Enc}(k, 0^n)$.

5 Problem 5 (20 points)

Suppose PRG is a secure PRG with input length λ and output length 3λ. Which of the following are secure PRGs. If it is a secure PRG, prove it using a reduction. If not, demonstrate why it is not a secure PRG (this could either be because it is not secure, or because it does not satisfy the semantics of a PRG). If there are more than one reason why it is not a secure PRG, demonstrate both.

(a) $\mathsf{PRG}_a(s) = \mathsf{PRG}(s)_{[1,2\lambda]}$. That is, run PRG, delete the last λ bits, and output the first 2λ.

(b) $\mathsf{PRG}_b(r, s) = (r, \mathsf{PRG}(s))$. Here, r, s are λ bits, and PRG has output length 4λ.

(c) $\mathsf{PRG}_c(s) = (r, \mathsf{PRG}(s))$. Here, r, s are λ bits, and PRG is a probabilistic algorithm that choose a fresh r for each invocation.

(d) $\mathsf{PRG}_d(s) = (s, \mathsf{PRG}(s))$.

(e) **Bonus (10 points).** $\mathsf{PRG}_e(s) = \mathsf{PRG}(\mathsf{PRG}_0(s)), \mathsf{PRG}(\mathsf{PRG}_1(s)), \mathsf{PRG}(\mathsf{PRG}_2(s))$. Here, $\mathsf{PRG}_0(s)$ represents the first λ bits of the output of PRG_s, $\mathsf{PRG}_1(s)$ the second λ bits, and $\mathsf{PRG}_2(s)$ the final λ bits.