

Introduction to Cryptography Exercise Set

Submit your solutions to exactly 6 out of the following 12 questions by March 2nd 2008.

1. (**Limitations of Information-Theoretic Security**) Perfect Secrecy is quite hard to achieve, as we saw in the class. Are there relaxations of perfect secrecy that are meaningful and easy to achieve? The following problems consider two possible relaxations of the definitions of encryption and perfect secrecy.

- (a) (**Statistical Shannon Secrecy**) Recall the definition of perfect secrecy which requires that given a ciphertext, all messages are equally likely. A relaxation of this – statistical secrecy – only requires that all messages are *almost* equally likely. Formally, perfect secrecy can be shown to be equivalent to the following condition (which says that, given a ciphertext, any message is equally likely to have been sent).

$$\forall c, \forall m, \Pr[\mathbf{E}(\mathbf{K}, m) = c] = \frac{1}{|\mathbf{M}|}$$

We relax this requirement by saying that an encryption scheme is ϵ -statistically secure, if

$$\forall c, \sum_{m \in \mathbf{M}} \left| \Pr[\mathbf{E}(\mathbf{K}, m) = c] - \frac{1}{|\mathbf{M}|} \right| \leq \epsilon.$$

Show that, for an encryption scheme to be ϵ -statistically secure, it is necessary that $|\mathbf{K}| \geq (1 - \epsilon)|\mathbf{M}|$.

- (b) (**Shannon Theorem for Interactive Encryption**) In the classical setting, to encrypt a message m to Bob, Alice computes a ciphertext $c = \mathbf{E}(k, m)$ and sends it to Bob (Here, k is the secret-key that Alice and Bob share).

We modify this setting by allowing \mathbf{E} to be an *interactive protocol*. That is, we allow Alice and Bob to *interact*, such that given the *transcript* of the interaction, Bob can use his secret-key to decrypt and recover the message.

More formally, we define an interactive encryption scheme as a triple of algorithms $(\mathbf{G}, \mathbf{E}, \mathbf{D})$. The algorithms \mathbf{G} and \mathbf{D} work just as before. The encryption algorithm \mathbf{E} is defined by a pair (A, B) of *interacting randomized* Turing machines, that work as follows. The conversation between A and B , where A and B share a common secret key k , and A has an additional input m , is defined by a sequence of messages \mathbf{c} . We define $\mathbf{c} = (c_1, c_2, \dots)$ inductively.

$$c_i = \begin{cases} A(k, m, \{c_j\}_{j=1}^{i-1}) & \text{if } i \text{ is odd} \\ B(k, \{c_j\}_{j=1}^{i-1}) & \text{if } i \text{ is even} \end{cases}$$

The final ciphertext is defined to be c , generated as above.

(For intuition, think of A as Alice and B as Bob. Alice computes the first message $c_1 = A(k, m)$ depending on the key k and message m and sends c_1 to Bob. Bob computes c_2 as a function of the shared key k and c_1 , and sends c_2 to Alice. And so on. The final ciphertext is the sequence of all c_i 's, that Bob sees during the conversation. At the end, Bob computes $D(k, c)$ which gives him the message m that Alice intended to send.)

The analogue of the perfect secrecy definition for interactive encryption states that

$$\forall m_1, m_2, \forall c, \Pr[\mathbf{E}(\mathbf{K}, m_1) = c] = \Pr[\mathbf{E}(\mathbf{K}, m_2) = c],$$

where the probability is taken over the coin-tosses of the Turing machines A and B .

Show that for an interactive encryption scheme to be perfectly secret, it is necessary that $|\mathbf{K}| \geq |\mathbf{M}|$.

2. (Strong vs. Weak One-way Functions)

- (a) Let $\tilde{n} = (n, a_1, a_2, \dots, a_n)$, where $n \in \mathbb{N}$ and $a_i \in \{0, 1\}^n$ for each i , let $x_i \in \{0, 1\}$ for each i , and $c \in \{0, 1\}$. We define a collection of one-way functions $\{f_{\tilde{n}} : D_{\tilde{n}} \mapsto \{0, 1\}^*\}_{\tilde{n}}$ as follows. Define a'_i inductively as $a'_1 = a_1$ and for $i > 1$, $a'_i = a_i + \sum_{j < i} a'_j$. Given this $f_{\tilde{n}}$ is defined as follows:

$$f_{\tilde{n}}(x_1, x_2, \dots, x_n, c) = \begin{cases} \sum_i a_i x_i & \text{if } c = 0 \\ \sum_i a'_i x_i & \text{if } c = 1 \end{cases}$$

$f_{\tilde{n}}$ is conjectured to be a weak one-way function. Exhibit an attack that shows that $f_{\tilde{n}}$ is *not* a strong one-way function.

3. (Number Theory) Let p be some prime. Recall that \mathbb{Z}_p^* is the multiplicative group of integers modulo p .

- (a) Show that exactly half the elements in the multiplicative group \mathbb{Z}_p^* are quadratic residues. That is, the set of quadratic residues \mathbf{QR}_p is defined to be $\{x \in \mathbb{Z}_p^* \mid \exists y \in \mathbb{Z}_p^*, y^2 \equiv x \pmod{p}\}$. We want you to show that $|\mathbf{QR}_p| = \frac{1}{2}|\mathbb{Z}_p^*|$.
- (b) Show that *each* quadratic residue in \mathbb{Z}_p^* has exactly two square roots.
- (c) Show that, to check that $g \in \mathbb{Z}_p^*$ is a generator of the multiplicative group \mathbb{Z}_p^* , it is sufficient to check that $g^{\frac{p-1}{q}} \not\equiv 1 \pmod{p}$ for every q that is a prime divisor of $p-1$.
- (d) Assume that $g = p^k$ for some prime p and integer k . Show how to solve the equation $z \equiv x^2 \pmod{q}$, given z .

4. (Pseudorandom Generators)

Recall the construction of a 1 bit stretching pseudorandom bit generator from a one-way permutation. If f is a one-way permutation, define $G(x) = f(x) \cdot B(x)$, where B was a hard-core bit of f . Why doesn't this construction work if f is just a one-way function? Can you think of a sufficient condition (weaker than asking for it to be a one-way permutation) on f that lets it work? Try (just try) to construct pseudorandom generators from one-way functions.

5. (Pseudorandom Functions)

A pseudorandom family of functions is a family of functions with the property that the input-output behaviour of a random member of the family is computationally indistinguishable from that of a random function. That is, a PPT with only black box access to a random member of the family will behave almost the same as if the function was chosen uniformly at random from the space of all functions.

Formally, if f is a function, denote A^f the PPT Machine A with oracle access to f . We say a deterministic function

$$F_s^n(x) = F^n(x, s) : \{0, 1\}^n \times \{0, 1\}^{s(n)} \rightarrow \{0, 1\}^{l(n)}$$

is a pseudorandom function generator (here s is the seed) if \forall polynomial Q, \forall PPT $A, \exists n_0$ s.t. $\forall n > n_0$, letting $F = F^n$,

$$\left| \Pr_{s \in \{0, 1\}^{s(n)}} [A^{F_s}(1^n) = 1] - \Pr_{g: \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}} [A^g(1^n) = 1] \right| < \frac{1}{Q(n)},$$

where g was chosen uniformly at random among all functions from $\{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$. Here, for a random seed s , F_s is supposed to be indistinguishable from a random g (for PPTs given oracle access to them).

Suppose $F_s(x)$ is a pseudorandom function generator (in particular, we are assuming pseudorandom function generators exist).

1. Are the following functions G necessarily pseudorandom function generators?

- $G_s(x) = F_x(s)$ (assume $s(n) = n$).
- $G_s(x) = F_s(x) \cdot F_s(\bar{x})$ (where \bar{x} denotes the string obtained by flipping all the bits of x).

If so prove it. If not, give a counterexample.

2. For the previous question, what would your answer be if F was the pseudorandom function generator constructed in class?
3. Let $s(n) = n^2, l(n) = n$. Denote M_s the matrix obtained by treating the bits of s as entries of an $n \times n$ matrix. Consider the following candidate construction for a PRFG: $F_s(x) = M_s x$, where the matrix multiplication is done over \mathbb{Z}_2 . Why isn't this a PRFG?

6. (A new scheme based of QRA)

In this problem, we propose a candidate encryption scheme whose security (we claim) is based on the Quadratic Residuosity Assumption (QRA). Your goal is to prove that it is indeed a secure encryption scheme, under the QRA. One has to show that the scheme is correct, and that it satisfies the computational indistinguishability definition from the class. The encryption scheme is as given below.

- **Key-generation Algorithm** $G(1^k)$: Choose random primes p and q of length k each such that $p \equiv q \equiv 3 \pmod{4}$. Let $N = pq$. Choose a random element $a \in \mathbb{Z}_N^*$. The public key is (N, a^2) and the secret-key is a .
 - **Encryption Algorithm** $E(pk, b)$: To encrypt a bit b , choose a random $t \in \mathbb{Z}_N^*$ such that the Jacobi symbol $(\frac{t}{N}) = b$. Output $t + \frac{a^2}{t} \pmod{N}$ as the encryption of b .
 - **Decryption Algorithm** $D(sk, c)$: Output the Jacobi symbol $(\frac{c+2*sk}{N})$ as the decryption of c .
- (a) Prove that the above scheme is indeed correct. That is, the decryption algorithm outputs b , when given as input a correctly generated ciphertext of b .
 - (b) Prove that the above scheme is secure under the QRA. In particular, show that if there is a ppt adversary A that breaks the computational indistinguishability of the encryption scheme, there is another ppt adversary A' that breaks the quadratic residuosity assumption.

7. (Equivalence of notations)

A good definition of security of an encryption scheme is supposed to capture the types of attacks that one expects the scheme to resist, in the real world. There could potentially be multiple mathematical formulations of this security property. When all these formulations are proven equivalent, it is considered as strong evidence that any one of them is indeed *the right* definition to look at.

There are (at least) three distinct definitions of the basic security of an encryption scheme – Semantic Security (of Goldwasser and Micali), Yao Security and Computational Indistinguishability. These notions have been shown to be equivalent. In this problem, we will show parts of this equivalence result.

(a) **Yao Security**

We consider a definition of security for a public-key cryptosystem proposed by Yao. The idea is that Alice has a polynomial number (n^k) of strings that she wants to send to Bob using as few bits as possible. These strings are selected from a probability distribution known to both Alice and Bob and Alice wants to send enough bits to Bob so that he can (with high probability) reconstruct all of the messages. Note that Alice and Bob are *not* trying to keep any of these messages secret; Alice is just trying to deliver them to Bob as efficiently as possible.

Now suppose that Bob gets encryptions of the messages “for free,” in addition to the bits that Alice sends him (however, Bob doesn’t know the secret key to decrypt those ciphertexts). We say that a cryptosystem is Yao-Secure if the average number of bits which Alice must send to Bob is the same regardless of whether or not Bob possesses a copy of the ciphertexts. (That is, sharing the ciphertexts does not help Alice compress the messages.)

In the definitions below, we let $M = \{M_n\}$ be a sequence of probability distributions over $\{0, 1\}^*$ where M_n only assigns positive probability to n -bit strings. We denote by $\{A_n\}$ a family of probabilistic polynomial-size encoding circuits, and by $\{B_n\}$ a family of probabilistic polynomial-size decoding circuits.

– **The cost of communicating M :**

We say that *the cost of communicating M is less than or equal to $f(n)$* (in symbols, $C(M) \leq f(n)$) if there exist $\{A_n\}$ and $\{B_n\}$ such that the following two properties are satisfied: (for some constant k , for all c , and for all sufficiently large $n \dots$)

1. “ B_n understands A_n ”

$$\Pr [m_1 \leftarrow M_n; \dots m_{n^k} \leftarrow M_n; \beta \leftarrow A_n(\vec{m}); \vec{y} \leftarrow B_n(\beta) : \vec{m} = \vec{y}] > 1 - n^{-c}$$

2. “ A_n transmits at most $f(n)$ bits per message”

$$E \left[m_1 \leftarrow M_n; \dots m_{n^k} \leftarrow M_n; \beta \leftarrow A_n(\vec{m}) : \frac{|\beta|}{n^k} \right] \leq f(n)$$

– **The cost of communicating M , given encryptions:**

Let (G, E, D) be a cryptosystem. We say that *the cost of communicating M , given encryptions using E , is less than or equal to $f(n)$* (in symbols, $C(M|E(M)) \leq f(n)$) if there exist $\{A_n\}$ and $\{B_n\}$ such that the following two properties are satisfied: (for some constant k , for all c , and for all sufficiently large $n \dots$)

1. “ B_n understands A_n ”

$$\Pr [\left[(PK, SK) \leftarrow G(1^n); m_1 \leftarrow M_n; \dots m_{n^k} \leftarrow M_n; c_1 \leftarrow E_{PK}(m_1), \dots, c_{n^k} \leftarrow E_{PK}(m_{n^k}); \right. \\ \left. \beta \leftarrow A_n(\vec{m}, PK, \vec{c}); \vec{y} \leftarrow B_n(\beta, PK, \vec{c}) : \vec{m} = \vec{y} \right] > 1 - n^{-c}$$

2. “ A_n transmits at most $f(n)$ bits per message”

$$E\left[(PK, SK) \leftarrow G(1^n); m_1 \leftarrow M_n; \dots m_{n^k} \leftarrow M_n; c_1 \leftarrow E_{PK}(m_1), \dots, c_{n^k} \leftarrow E_{PK}(m_{n^k});\right.$$

$$\left. \beta \leftarrow A_n(\vec{m}, PK, \vec{c}) : \frac{|\beta|}{n^k} \right] \leq f(n)$$

– **Yao-Security**

We say that a cryptosystem is Yao-secure if for all M , for all c and for all sufficiently large n ,

$$C(M|E(M)) \leq f(n) \Rightarrow C(M) \leq f(n) + \frac{1}{n^c}$$

Prove that Yao-security implies the computational indistinguishability of encryption schemes.

(b) **Semantic Security** Informally, semantic security is supposed to capture the following intuitive notion of security.

An Encryption scheme is called semantically secure if, for every probability distribution of messages, everything that can be **efficiently computed** given the encrypted message, can be **efficiently computed** *without it*.

Formally, an encryption scheme (G, E, D) is called semantically secure if, for every probability distribution π_n of messages, every function f , and every ppt algorithm \mathcal{A} , there exists a ppt algorithm \mathcal{B} such that, for every polynomial p , and sufficiently large n (which is the size of the messages),

$$\Pr[\mathcal{A}(E(\text{pk}, m), \text{pk}, 1^n) = f(m)] \leq \Pr[\mathcal{B}(\text{pk}, 1^n) = f(m)] + \frac{1}{p(n)}$$

where the left probability is taken over the coin-tosses of the algorithm \mathcal{A} , E and G , and the probability distribution of messages π_n , and the right probability is taken over the coin-tosses of \mathcal{A} and G and the probability distribution of messages π_n .

Show that semantic security implies computationally indistinguishability of encryption schemes.

8. (Chosen Ciphertext Security)

Recall the definition of chosen-ciphertext security of public-key encryption schemes from class. In this problem, we will define a corresponding notion for private-key encryption schemes, and propose a candidate construction of an encryption scheme secure against chosen-ciphertext attacks (CCA-secure). Your goal would be to prove that it is indeed CCA-secure.

In the CCA-game, the adversary \mathcal{A} gets access to both an encryption and a decryption oracle. In particular, \mathcal{A} can ask the oracles to encrypt a message m of its choice, or decrypt a ciphertext c of its choice. At some point, \mathcal{A} outputs a pair of messages (m_0, m_1) , and gets $C_{\text{challenge}} = E(k, m_b)$, where b is a random bit. That is, the adversary gets the encryption of either m_0 or m_1 , chosen at random. Its goal, now, is to predict which message was encrypted – m_0 or m_1 . The adversary continues to have access to the encryption and decryption oracles, *except that it cannot ask the decryption oracle to decrypt $C_{\text{challenge}}$* . At the end, it outputs a bit b' . An encryption scheme is CCA-secure, if for any ppt adversary \mathcal{A} , any polynomial $p(\cdot)$, and for sufficiently large n ,

$$\Pr[b' = b] < \frac{1}{2} + \frac{1}{p(n)}$$

The encryption scheme works as follows. Fix a family of pseudorandom functions

$$\mathcal{F} = \{f_s \mid f_s : \{0, 1\}^n \mapsto \{0, 1\}^n\}_{s \in \{0, 1\}^n}$$

1. The **key-generation** algorithm chooses a random seed $s \in \{0, 1\}^n$. This is the shared secret-key.
2. To **encrypt** a message m , choose a random r , and output the pair $[r, f_s(r) \oplus m, f_s[r || (f_s(r) \oplus m)]]$. (Here, $||$ is the string concatenation operator)

3. To **decrypt** a ciphertext $[c_1, c_2, c_3]$, first check if $f_s(c_1 || c_2) \stackrel{?}{=} c_3$. If the check fails, output \perp . Else, output $c_2 \oplus f_s(c_1)$ as the decryption.

Show that this scheme is correct, and is CCA-secure.

9. (“Breaking” the DSS)

The DSA (Digital Signature Algorithm) was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS). In this problem, we try to break the DSA under various attacks. For starters, the DSA works as follows:

- **Key-Generation:** $G(1^k)$. Pick primes p, q such that $p = 2q + 1$. Pick a generator h of the multiplicative group \mathbf{Z}_p^* , and set $g = h^2 \pmod{p}$. (Thus, g is a generator of a subgroup of order q of \mathbf{Z}_p^*) Pick a random element $x \in \mathbf{Z}_p^*$. The public (verification) key is (p, g, g^x) , and the secret (signing) key is x .
- **To sign a message m :** Choose k at random in \mathbf{Z}_q^* . Let $r = g^k \pmod{p}$. Solve the equation $rx - ks \equiv m \pmod{q}$ to find (the unique) solution s . Output (r, s) . (**Note:** k is kept secret.)
- **To verify a signature (r, s) of a message m :** Check if $(g^x)^r r^{-s} \stackrel{?}{=} g^m \pmod{p}$. If this equation checks, then output “signature verified” else output “fail”.

Before starting on the problems, convince yourself that this is indeed a correct signature scheme.

- Why does the k need to be secret in the signing algorithm? Find a way to recover the secret signing key, given a single signature (r, s) of a message m , along with the randomness k used to generate the signature.
- Usually (in practice), the k used for signing is not generated completely at random, but by using a pseudorandom generator. Again, in practice, people use a simple “pseudorandom” generator such as the linear congruential generator. A linear congruential generator LCG, on input a seed s_0 outputs a sequence (s_1, s_2, \dots, s_n) such that $s_i = as_{i-1} + b \pmod{q}$. (where a and b are assumed to be public parameters)

Show that if the k 's are generated using an LCG, then given *two* signatures (of two messages), one can recover the secret signing key. (“**Lesson**”: Beware of using unproven, patched-up pseudorandom generators).
- Finally, prove that if the k 's are generated using a cryptographic pseudorandom generator (one that fools all polynomial-time statistical tests, as defined in the class), then the DSA remains secure. (In other words, show that if there is an adversary A that breaks the DSA that uses a PRG, then there is an adversary B that uses the original DSA – one where k 's are generated randomly)

10. (Commitment)

Bob is performing a magic trick. Alice thought of a number and Bob is going to guess it. Of course, Bob doesn't want Alice to change her number if he actually does manage to guess it. So he wants to see her write down a “commitment” to her number. On the other hand, Alice does not want Bob to learn the number from her commitment. Quite the conundrum.

Let us solve their problem.

A *bit commitment* protocol consists of two stages:

- The *commit* stage: Alice has a bit b to which she wishes to commit to Bob. She and Bob exchange messages. At the end of the stage Bob has some information that represents b .
- The *decommit* stage: At the end of which Bob knows b .

The protocol must obey the following:

For all polynomials Q , for large enough¹ n ,

¹In a bit commitment scheme, since the size of the input is technically 1, n serves the purpose of a security parameter, i.e. we measure the “polynomial time” in PPT as polynomial in n : it is a way of measuring the amount of security.

- *Hiding*: After the commit stage Bob cannot guess b with probability greater than $\frac{1}{2} + \frac{1}{Q(n)}$.
A commitment scheme is called:
 - *Computational Hiding*: If the protocol is hiding for all PPT Bobs
 - *Perfect Hiding*: If the protocol is hiding for all Bobs.
- *Binding*: Alice can decommit only one possible value. If she tries to decommit a different value, she is caught with probability at least $1 - \frac{1}{Q(n)}$.
A commitment scheme is called:
 - *Computational Binding*: If the protocol is binding for all PPT Alices.
 - *Perfect Binding*: If the protocol is binding for all Alices.

A commitment scheme is called *non-interactive* if the protocol does not ask that Bob send any messages, i.e., Alice does all the message sending.

1. Describe a non-interactive commitment scheme assuming that one-way permutations exist.
2. Does your construction work using a one-way function rather than a one-way permutation?
3. We will now describe a commitment scheme that only assumes the existence of one-way functions. Take $G : \{0, 1\}^n \rightarrow \{0, 1\}^{3n}$, a PRG
 - **Commit**: The receiver first sends to the committer a uniformly at random $r \in \{0, 1\}^{3n}$ to the committer
 - The committer picks $s \in \{0, 1\}^n$ uniformly at random.
 - If $b = 0$ the committer sends $G(s) = message_1$ as the commitment.
 - If $b = 1$ the committer sends $G(s) \oplus r = message_1$ as the commitment.
 - **Decommit**: The committer reveals $s = message_2$ to the receiver.
 - The receiver concludes that $b = 0$ was committed to if $G(message_2) = message_1$.
 - The receiver concludes that $b = 1$ was committed to if $G(message_2) \oplus r = message_1$.

Prove that this is indeed a commitment scheme. In particular, prove that the scheme is *computationally hiding* and *perfectly binding*.

11. (Interactive proof)

Bob is color-blind. His sister always teases him before going to school by telling him “Your socks are mismatched! Quick! Go change them before the bus comes.” Bob is thus always chasing after the bus in the morning. At some point, however, he realizes that his sister is more interested in making him run than in keeping his feet properly matched. Help Bob design a protocol he can use with his sister to tell whether his sister is teasing him or is telling him the truth about his mismatched socks.

12. (Zero Knowledge Proof)

Consider the following zero-knowledge protocol for proving that a value a is a quadratic residue modulo n . The idea of this construction is to simply parallelize the simpler ZKP for quadratic residuosity.

Input: n and a . Let α such that $\alpha^2 \equiv a \pmod n$ be given to P .

1. P chooses a random prime $p = 2q + 1$ where q is a random k -bit prime, a random g of order q in \mathbb{Z}_p^* , a random $\beta \leftarrow \mathbb{Z}_q^*$, and sets $h = g^\beta \pmod p$. P sends p, q, g, h to V .
2. V computes a random string c of length k and a random $r \leftarrow \mathbb{Z}_q^*$, and sends $x = g^r h^c$ to P .
3. P chooses k random values $r_1, \dots, r_k \pmod n$, computes $s_i = r_i^2 \pmod n$, and sends s_1, \dots, s_k to V .
4. V sends r and c to P .
5. P checks that $x = g^r h^c$. If so, P sends to V , for each i , the value $t_i = r_i \alpha^{c_i}$, where c_i is the i th bit of c . P also sends β to V .
6. V checks that $t_i^2 \equiv s_i a^{c_i}$ for each i , and that $h = g^\beta$. If so, V accepts, otherwise V rejects.

This defines a P, V which you will prove is a zero-knowledge proof system. It should be clear that P, V has completeness 1. In order to achieve the zero-knowledge property, we added the commitment x into the protocol, as you will see when proving this.

1. Prove that P, V has negligible soundness (ie, V can only be convinced with negligible probability if a is a quadratic non-residue modulo n).
2. Prove that if the discrete logarithm problem is hard, then P, V is computational zero-knowledge (i.e., the simulated transcript is *indistinguishable* from a real one).