POIS End Semester Question Bank

April 2024

1 One Time Pad

Problem 1. Recall that the one-time pad is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ where $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$ and $\mathsf{Enc}(k, m) = k \oplus m$. Notice that when the key $k = 0^n$ is used, then $\mathsf{Enc}(k, m) = m$ and this does not seem secure. Suppose we improve the one-time pad by setting the key space to $\mathcal{K} := \{0,1\}^n \backslash 0^n$. That is, we take 0^n out of the key space so that it will never be chosen as a key. Does the resulting cipher have perfect secrecy? Justify your answer.

2 PRG and PRF

Problem 2. Let $\mathcal{G}$ be a group of prime order p with generator $g \in \mathcal{G}$. Assume that the discrete log problem is hard in $\mathcal{G}$. Consider the following PRG defined over $(\mathcal{Z}_p, \mathcal{G}^2)$: given an input $x \in \mathrm{Z}_p$, the PRG outputs $G(x) := (g^{4x}, g^{5x}) \in \mathcal{G}^2$. Is this a secure PRG? If so, explain why. If not, describe an attack.

Problem 3.

(a) Construct a PRF F_a such that, if the first bit of the key is leaked, then it is insecure – that is, it is possible to distinguish the PRF from a random function with non-negligible advantage, given the first bit of the key. You may use as a starting point any secure PRF F'. Your F must be a secure PRF if the key is completely hidden.

(b) Construct a PRF F_b such that, if the XOR of all bits of the key is leaked, then it is insecure — that is, it is possible to distinguish the PRF from a random function with non-negligible advantage, given the XOR of the key bits. You may use as a starting point any secure PRF F'. Your F must be a secure PRF if the key is completely hidden.

(c) Construct a PRF F_c that is still secure if *any* single bit of the key is leaked. You may use as a starting point a secure PRF F', but you may not assume anything about F except its security: F' may be insecure if any bit of key is leaked. As a hint, if F' has key space K, try having F_c have key space K^2.

(d) Is your PRF from part (c) guaranteed to be secure if the adversary is allowed to learn the XOR of all of the key bits? (What if the starting PRF F' is actually F_b?).

(e) Let F be a PRF. Suppose the adversary is allowed to choose an efficiently computable function $f : K \to \{0,1\}$ that outputs a single bit, and the adversary learns $f(k)$ where k is the PRF key. Show how, no matter what F is, the adversary can devise an efficiently computable f such that he can distinguish $F'(k, \cdot)$ from random given $f(k)$.

3 Modes of Operation

Problem 4. Does counter mode encryption require a PRP, or is a PRF sufficient? Justify your answer.

4 MACs, CCA Security and Hash Functions

Problem 5. You are given a secure MAC scheme $(\mathcal{S}, \mathcal{V})$ that can be used to sign a sequence of bytes whose length is a multiple of eight. That is, its length can be 8 bytes, 16 bytes, 24 bytes, etc. Explain how to use this MAC scheme to sign a sequence of bytes of arbitrary length (i.e., including messages whose length may not be a multiple of eight).

Problem 6. Show that if H_1 and H_2 are distinct collision resistant functions with range $\mathcal{T} := \{0,1\}^n$, then $H(x) := H_1(x) \oplus H_2(x)$ need not be collision resistant.

Hint: Let $\mathcal{F}$ be a collision resistant hash function with range $\mathcal{T}$. Use $\mathcal{F}$ to construct two collision resistant functions H_1, H_2 such that H is not collision resistant.

Problem 7. The $UNIXcrypt$ function is a hash function that only looks at the first eight bytes of the input message. For example, $crypt(helloworld)$ returns the same value as $crypt(hellowor)$. Consider the following MAC system $(\mathcal{S}, \mathcal{V})$ whose key space $\mathcal{K}$ is the set of eight character strings: $\mathcal{S}(k,m) := crypt(m||k)$; $\mathcal{V}(k,m,t) := \{\text{output yes if } t = crypt(m||k)\}$

Here || denotes string concatenation. Show that this MAC is vulnerable to a chosen Plaintext attack. In particular, show that an adversary can recover k with 8×256 chosen message queries.

5 Public Key Encryption

Problem 8. What is the smallest possible positive value of e that can be used to define the RSA trapdoor function? Explain why a smaller value of e cannot be used.

Problem 9. In the ElGamal public key encryption system, is it safe to fix the group $\mathcal{G}$ and generator $g \in \mathcal{G}$, so that all users in the world use the same $(\mathcal{G}, g)$?

Problem 10. Here, we will see how to shorten the public keys for signature schemes. Let $(\text{Gen}, \text{Sign}, \text{Ver})$ be a many-time secure signature scheme, with public keys in some set X. Let $H : K \times X \to Y$ be a collision resistant hash function. Show how to construct a many-time secure signature scheme $(\text{Gen}', \text{Sign}', \text{Ver}')$ with public keys in $K \times Y$. Note that $K \times Y$ will in general be much smaller than X. Prove the security of your scheme.

Hint: Gen' will work as follows. To generate a public and secret key, run $(\text{sk}, \text{pk}) \leftarrow \text{Gen}()$ and $k \leftarrow K$, and then output $\text{sk}' = \text{sk}$ and $\text{pk}' = (k, H(k, \text{pk}))$. Your job is to describe how to sign and verify, and to prove security.

Problem 11. Assume that Alice signs two messages using El Gamal, and for both messages she uses the same k. Show how an attacker can totally break the signature scheme (recover the signing key), without solving an instance of the Discrete Log Problem.

Problem 12. Consider a key exchange protocol where the client has the public keys of a server, chooses a key $k \xleftarrow{\$} \{0,1\}^n$ for a private key scheme, interacts with the server, and at the end decides whether or not to accept the key as valid. For simplicity we restrict ourselves to two-message protocols (one message from client to server and one message from server to client). Consider the following attack on such protocols:

- Client sends the first message to the adversary.
- Adversary gets a polynomial number of interactions with the server.
- Adversary sends a message to the client.
- The client chooses $b \xleftarrow{\$} \{1,2\}$. If it accepts the message and obtained a key k *and* $b = 1$ then it sends k to the adversary. Otherwise, (either $b = 2$ or it did not accept the message) it sends a random string $k' \xleftarrow{\$} \{0,1\}^n$ to the adversary.
- The adversary outputs $b' \in \{1,2\}$. We say the adversary is successful if $b' = b$.

We say the protocol is *secure* if the probability the adversary succeeds in this attack is at most $\frac{1}{2} + n^{-\omega(1)}$.

For each of the following protocols, either prove that it is secure or give an example showing it is insecure. **Notation:** We denote by $(\mathsf{Sign}, \mathsf{Ver})$ a secure signature scheme. We denote by $\mathsf{Enc}^{\mathsf{pub,cca}}$ a CCA secure public key encryption scheme, by $\mathsf{Enc}^{\mathsf{pub,cpa}}$ a CPA secure public key encryption scheme, and by $\mathsf{Enc}^{\mathsf{priv,cpa}}$ a CPA secure private key encryption scheme. The protocol is secure if it is secure for any suitable choice of the underlying schemes. In all cases we denote by e and by v the public encryption key and verification key of the server, and assume that the client knows them.

(a) Protocol 1:

- Client chooses $k \xleftarrow{\$} \{0,1\}$ and $m \xleftarrow{\$} \{0,1\}^n$ and sends to server $\mathsf{Enc}_e^{\mathsf{pub,cpa}}(k \circ m)$.
- Server decrypts ciphertext to get k, m and sends to client $m, \mathsf{Sign}_s(m)$ (if ciphertext is invalid then server sends "invalid".
- Clients verifies signature and if it passes verification, it considers the key k as valid. (It will use m as a label of the key in future conversations with the server)

(b) Protocol 2: Same as Protocol 1 but with $\mathsf{Enc}^{\mathsf{pub,cca}}$ instead of $\mathsf{Enc}^{\mathsf{pub,cpa}}$.

(c) Protocol 3:

- Client chooses $k \xleftarrow{\$} \{0,1\}^n$, $k' \xleftarrow{\$} \{0,1\}^n$ and $m \xleftarrow{\$} \{0,1\}^n$ and sends to server $\mathsf{Enc}_e^{\mathsf{pub,cpa}}(k \circ k' \circ m)$.
- Server decrypts ciphertext to get k, k', m and sends to client $y = \mathsf{Enc}_{k'}^{\mathsf{priv,cpa}}(m)$ and $\mathsf{Sign}_s(y)$ (if ciphertext is invalid then server sends "invalid".
- Clients verifies signature and if it passes verification and y decrypts with k' to the value m, it considers the key k as valid.

(d) Protocol 4:

- Client chooses $k \xleftarrow{\$} \{0,1\}^n$ and sends to server $y = \mathsf{Enc}_e^{\mathsf{pub,cpa}}(k)$.
- Server decrypts ciphertext to get k, chooses $m \xleftarrow{\$} \{0,1\}^n$ at random and sends to client y,m and $\mathsf{Sign}_s(y \circ m)$ (if ciphertext is invalid then server sends "invalid".
- Clients verifies signature and if it passes verification, it considers the key k as valid.

6 Oblivious Transfer and Multiparty Computation

Problem 13. A *secure coin-tossing protocol* is defined as follows: it is a two-party protocol with the two parties named Alice and Bob. There is a polynomial-time function *result* that takes as input the transcript of the protocol (i.e., the sequence of all messages exchanged between the two parties), and outputs a bit $b \in \{0,1\}$. Note that if the two parties are probabilistic, the transcript $trans = \mathsf{trans}\langle A(1^n), B(1^n)\rangle$ of the execution of the protocol (where both Alice and Bob are given as input the security parameter n) is a random variable. We require that as long as at least one party follows the protocol, for every $b \in \{0,1\}$, the probability that $result(trans) = b$ is between $\frac{1}{2} - \epsilon(n)$ and $\frac{1}{2} + \epsilon(n)$ where n is the security parameter for the protocol, and $\epsilon(\cdot)$ is a function such that $\epsilon(n) = n^{-\omega(1)}$.

An example for a simple protocol attempting to solve this problem would be for Alice to choose b at random and to send it to Bob, and for the result function to simply output b. This would work in the case that both Alice and Bob are honest (i.e., follow the protocol) and also if just Alice is honest, but not in the case that just Bob is honest. Hence this is not a secure coin-tossing protocol.

Construct a secure coin-tossing protocol.

In a simplified scenario, suppose that you have two machines but one of them might be faulty — can you run an interaction between them that results in an unbiased coin toss?

Problem 14. A *private coin tossing* protocol is the following variant of a coin tossing protocol. Intuitively, it supposed to be a protocol where only Alice actually knows the result of the coin toss, but Bob is still guaranteed that this result is random. (You can think that Alice learns the output b while the transcript only contains a commitment to b.) More formally, again, there are two parties Alice and Bob and a function *result* on the transcript of the protocol, but this time *result* is *not* a polynomial-time function. We require the following properties of the protocol:

(a) As before, if at least one of the parties is honest, for every $b \in \{0,1\}$,

$$\tfrac{1}{2} - n^{-\omega(1)} < \Pr[result(trans) = b] < \tfrac{1}{2} + n^{-\omega(1)}$$

(b) If Alice is honest, then she knows the result: there is a polynomial-time algorithm RES that on input the private randomness and view that Alice used, outputs the same output as *result*. That is, no matter what algorithm B^* Bob uses $RES(\mathsf{view}_A\langle A, B^*\rangle) = result(\mathsf{trans}\langle A, B^*\rangle)$ where $\langle A, B^*\rangle$ in both sides of the equation denotes an execution between the honest Alice algorithm A and a possibly cheating (but polynomial-time Bob algorithm B^*).

(c) Bob does not know the result. That is, consider the following attack: Bob participates in the protocol using an arbitrary algorithm B^* where Alice uses the honest algorithm A, and at the end Bob outputs a guess b' for $result(trans)$, where $trans$ is the transcript of the execution. Then the probability that $b' = result(trans)$ is at most $1/2 + n^{-\omega(1)}$.

Note that if Alice and Bob participated in an execution of a private coin-tossing protocol with transcript $trans$, and Alice knows $b = result(trans) = RES(view)$ (where $view$ is her view including the random tape she used) then Alice can *prove* to Bob that $result(trans) = b$ by simply sending to Bob the random tape she used (using this random tape and $trans$ Bob can reconstruct Alice's view $view$ in the execution and then compute $result(trans) = RES(view)$).

Construct a private coin-tossing protocol and prove its security under the assumptions we learned in class.

Problem 15. Recall that in class, we learned about 1-out-of-2 Oblivious Transfer (OT). In an OT protocol, a sender has two message bits $m_0, m_1 \in \{0,1\}$, and a receiver has a choice bit $b \in \{0,1\}$. The sender wants to send m_b to the receiver while satisfying correctness (the receiver obtains m_b), sender's privacy (the receiver gains no knowledge about the message m_{1-b}), and receiver's privacy (the sender gains no knowledge about the choice bit b).

In this problem, we focus on achieving security against *honest-but-curious* senders and receivers.

(a) Show how you can use any 1-bit OT scheme to build an ℓ-bit OT scheme for transferring ℓ-bit messages $m_0, m_1 \in \{0,1\}^\ell$. Here $\ell = \ell(\lambda)$ is a (possibly large) polynomial in security parameter λ. Your scheme can only invoke the given 1-bit OT scheme at most $\lambda \ll \ell$ times. You can assume the existence of a pseudorandom generator.

(b) A 1-out-of-n secret sharing scheme is one where the sender has n messages $m_0, \ldots, m_{n-1} \in \{0,1\}^\ell$ and the receiver wants the i^{th} message m_i.

You are given a 1-out-of-2 OT scheme with ℓ-bit messages. Show how to construct a 1-out-of-n OT scheme for any integer $n \geq 2$. You can assume the existence of a PRF family. For full credit, your scheme must invoke the 1-out-of-2 OT scheme at most $O(\log n)$ many times.

Problem 16. The following questions rely on the definition of two-party secure computation. Recall that we think of computing a function $f : \{0,1\}^n \times \{0,1\}^n \to bits^n \times \{0,1\}^n$, where if Alice has input x and Bob has input y, at the end of the protocol Alice gets w and Bob gets z, where $(w,z) = f(x,y)$, and we also use the notation $w = f_1(x,y)$, $z = f_2(x,y)$. Neither Alice nor Bob should get anything else.

(a) Suppose we have a protocol to evaluate every polynomial-time computable $f()$ whose two outputs are the same: $f_1(x,y) = f_2(x,y)$ for every x, y. Use that to construct a protocol to evaluate every polynomial-time computable function.

(b) Suppose we have a protocol to evaluate every poly-time function $f()$, use that to construct a protocol to evaluate every *poly-time probabilistic process*, where a probabilistic process is a function $g : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$ on *three* inputs, and if Alice has input x and Bob has input y, then Alice gets w and Bob gets z where $(w,z) = f(x,y,r)$ for a uniformly random string r (note that neither Alice nor Bob can control or know r).

7 Secret Sharing Schemes

Problem 17.

In class, we saw the notion of threshold secret sharing. Given n users (numbered 1 through n), define the (t,n)-threshold access structure to be the collection of sets

$$\mathcal{A}_{\mathsf{threshold}} = \{S \subseteq [n] : |S| \geq t\}$$

The property of a secret sharing scheme, then, is that when a set $T \subseteq [n]$ of users come together, they can reconstruct the secret from their shares if and only if $T \in \mathcal{A}_{\mathsf{threshold}}$.

In this problem, we will generalize secret sharing to other access structures. In general, an access structure $\mathcal{A}$ is a monotone collection of subsets of $[n]$. Here, monotonicity refers to the condition that if $\mathcal{A}$ contains a set T, it also contains all supersets of T.

- We will associate to a subset $T \subseteq [n]$ its characteristic vector $x_T \in \{0,1\}^n$. For example, letting $n = 4$ and $T = \{1,3\}$, we have $x_T = (1010)$.
- We will associate to an access structure $\mathcal{A}$ a (monotone) Boolean function $f_{\mathcal{A}} : \{0,1\}^n \to \{0,1\}$ where
$$T \in \mathcal{A} \text{ if and only if } f_{\mathcal{A}}(x_T) = 1 \ .$$

For example, in the case of the threshold access structure, $f_{\mathcal{A}}$ is simply the threshold function which outputs 1 if and only if the input has at least t ones.

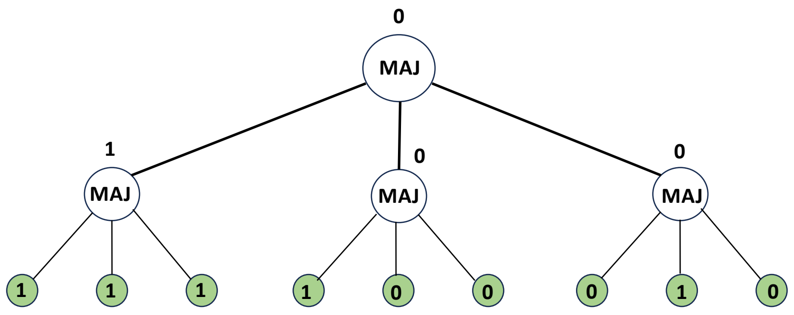

Figure 1: The function f representing a hierarchical access structure on 9 users. A set of users corresponds to a Boolean labeling of the leaves. For example, the labeling of the leaves above corresponds to the set of users $T = \{1, 2, 3, 4, 8\}$. An internal node is labeled 1 if and only if the majority of its children are labeled 1. A given set is in the access structure if and only if the root is labeled 1. (In this case, since the root has label 0, T is not in the access structure.)

- **(a)** Let $n = 3^m$ be the number of parties, and let the hierarchical access structure be defined by the majority-of-majorities function (see Figure 1 for an example with $m = 2$ and $n = 9$). Construct a secret sharing scheme for the hierarchical access structure on $n = 3^m$ users for arbitrary depth m. The shares in your scheme should have bit-length polynomial in n.
- **(b)** Let the conjunctive normal form (CNF) access structure be defined by a monotone CNF formula Ψ with n variables and m clauses (monotone in the sense that the formula has no negations). Construct a secret sharing scheme for the CNF access structure. The shares in your scheme should have bit-length polynomial in n and m.

Problem 18. In this problem, we will consider a voting system for an election with two candidates. The proposed system will use Shamir's secret-sharing. The system consists of a committee of size n which collects and aggregates the votes. In particular, each voter chooses a number $s^{(v)}$ which is 0 if she wants to vote for candidate 0, and 1 if she wants to vote for candidate 1. Voter v submits its vote $s^{(v)} \in \{0, 1\}$ by picking a random t-degree polynomial f_v over a finite field $\mathbb{F}$ with free coefficient $s^{(v)}$, and gives committee member i the value $s_i^{(v)} = f_v(i)$.

After the voting deadline is past, each committee member i computes

$$s_i^* = \sum_v s_i^{(v)}$$

Observe that $f^*(\cdot) = \sum_v f_v(\cdot)$ is a t-degree polynomial over $\mathbb{F}$ with free coefficient $s^* = \sum_v s^{(v)}$. Now, to reconstruct the secret, i.e. to compute s^* which is the total number of voters for party 1, any set of $t+1$ or more committee members can use their shares to interpolate the polynomial $f^*(\cdot)$, thus obtaining s^*.

For this problem, you may assume that there exists a public-ledger, and all parties have read/write access to this ledger. You can use any cryptographic tool learned in the class such as: a public-private-key encryption scheme, a digital signature scheme, a message authentication code (MAC), and so on.

(a) In the voting system described above, *cheating committee members* can prevent the reconstruction of the secret by contributing bad shares $s_i^{*'} \neq s_i^*$. Suggest a modification to this voting system which overcomes this attack. Argue why in your modified voting system this attack is not possible.

(b) Additionally, in the voting system described above, *cheating voters* can submit shares which do not correspond to any t-degree polynomial $f_v(\cdot)$, thus causing different groups in the committee to reconstruct different "voting results," i.e. the value of s^*, constructed is different for different groups. Moreover, cheating voters can submit shares of a polynomial with free coefficient $s^{(v)} \notin \{0,1\}$, thus influencing the "voting result" more than they should. Suggest how to further modify the scheme to overcome these problems as well. Argue why your modified voting system does not suffer from these problems.

8 Miscellaneous

Problem 19. For each of the following statements, say whether it is *true* or *false*, and prove your assertion. You can consider as true all the assumption given in class as mentioned in the instructions on the first page. For example, if the statement is that there exists an object with some properties, then either prove that such an object exists by giving a construction based on the assumptions together with a proof that the construction satisfies the properties, or prove that such an object cannot exist.

(a) There exists a pseudorandom generator $G = \{G_n\}$ where for every n, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ such that for every $x \in \{0,1\}^n$, the first $n/3$ bits of $G_n(x)$ are zero.

(b) There exists a pseudorandom generator $G = \{G_n\}$ where for every n, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ such that for every $x \in \{0,1\}^n$, there exist $n/3$ bits of $G_n(x)$ that are zero.

(c) There exists a pseudorandom generator $G = \{G_n\}$ where for every n, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ such that for every $x \in \{0,1\}^n$, if the first $n/3$ bits of x are zero then all the bits of $G_n(x)$ are zero (i.e., $G_n(x) = 0^{2n}$).

(d) There exists a pseudorandom function collection $\{f_s\}_{s\in\{0,1\}^*}$ where, letting $n = |s|$, $f_s : \{0,1\}^n \to \{0,1\}^n$ that satisfies the following: for every $x \in \{0,1\}^n$, $f_{0^n}(x) = 0^n$ (i.e., $f_{0^n}(\cdot)$ is the constant zero function).

(e) There exists a pseudorandom function collection $\{f_s\}_{s\in\{0,1\}^*}$ where, letting $n = |s|$, $f_s : \{0,1\}^n \to \{0,1\}^n$ that satisfies the following: for every $s \in \{0,1\}^n$, $f_s(0^n) = 0^n$.

(f) For $\ell \geq 2$ and a string $x \in \{0,1\}^\ell$, let *cnot* : $\{0,1\}^\ell \to \{0,1\}^\ell$ be the following function: $\mathit{cnot}(x_1,\ldots,x_\ell) = x_1, x_2 \oplus x_1, x_3, \ldots, x_\ell$. (That is, *cnot* flips the second bit of x according to whether or not the first bit is one.) There exists a CPA-secure public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a polynomial time algorithm A, such that for every n, if $(e,d) = \mathsf{Gen}(1^n)$ then for every $x \in \{0,1\}^\ell$ (where ℓ is the message size of the encryption scheme for security parameter n) it holds that

$$\mathsf{Dec}_d\left(A\Big(e, \mathsf{Enc}_e(x)\Big)\right) = \mathit{cnot}(x)$$

(f) For $\ell \geq 2$ and a string $x \in \{0,1\}^\ell$, let *cnot* : $\{0,1\}^\ell \to \{0,1\}^\ell$ be the following function: $\mathit{cnot}(x_1,\ldots,x_\ell) = x_1, x_2 \oplus x_1, x_3, \ldots, x_\ell$. (That is, *cnot* flips the second bit of x according to whether or not the first bit is one.) There exists a CCA-secure public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a polynomial time algorithm A, such that for every n, if $(e,d) = \mathsf{Gen}(1^n)$ then for every $x \in \{0,1\}^\ell$ (where ℓ is the message size of the encryption scheme for security parameter n) it holds that

$$\mathsf{Dec}_d\left(A\Big(e, \mathsf{Enc}_e(x)\Big)\right) = \mathit{cnot}(x)$$