Practice Problem Solutions

Question	Parts	Points
1: True or False	10	20
2: Authentication schemes	6	20
3: Short Answer Questions	3	15
4: Generating Randomness	1	5
5: Pseudorandom functions	2	10
6: Protocol Analysis	2	10
7: MAC1	1	5
8: Variants on Diffie-Hellman (DH)	4	20
9: Course Survey	5	5
Total:		110

Name: ______________________________

The actual midterm will be for a total of 120 points.

Problem 1. [20 points] **True or False** (10 parts)
Please circle **T** or **F** for the following. *No justification is needed (nor will be considered).*

(a) [2 points] A CPA-secure encryption scheme must hide the length of encrypted messages.

Solution: False.

(b) [2 points] Let H be a standard collision-resistant hash function (e.g., SHA256), and let p be a random 32-bit string. An attacker who learns $H(p)$ can recover p.

Solution: True by brute force.

(c) [2 points] Time-based one-time passwords require a client and a server to share a secret.

Solution: True.

(d) [2 points] The shortest digital signatures commonly used in practice today are 2048 bits long.

Solution: False.

(e) [2 points] HTTPS uses a “trust on first use” design for distribution of public keys.

Solution: False.

(f) [2 points] Let (E, D) be an arbitrary CPA-secure symmetric encryption scheme. This scheme remains secure if an attacker learns only one bit of the secret key.

Solution: False.

(g) [2 points] A CPA-secure encryption scheme on n-bit messages (with perfect correctness) must have ciphertexts that are more than n bit long.

Solution: True.

(h) [2 points] Large-scale quantum computers can efficiently break all known digital-signature schemes.

Solution: False.

(i) [2 points] In the lab 2 web of trust, Alice adds Bob as a trusted contact by signing Bob's public key. Cedric also adds Bob as a trusted contact by signing Bob's public key. Assuming lab 2 has been succesfully completed, Alice can now retreive Cedric's public key in an authenticated manner.

Solution: False.

(j) [2 points] Is it the case that for *every possible* CPA secure encryption scheme, its ciphertext looks random?

Solution: False. We can always append a 0 to a CPA secure encryption scheme and it will remain CPA secure.

Problem 2. [20 points] **Authentication schemes** (6 parts)
We consider the problem of a client authenticating to a database server over a network. After authenticating, the client will send a variable-length query command to the server.

(a) [3 points] Give two advantages of password-based authentication over MAC-based authentication.

Solution:

- A password is easier to remember than a MAC secret key.
- A human can run a password-based authentication protocol without a computer (e.g., over the phone).

(b) [3 points] Give two advantages of MAC-based authentication over password-based authentication.

Solution:

- A MAC allows authenticating a request (not just a principal).
- An attacker who sees many MACd messages cannot produce new ones. (In contrast, an attacker who observes a password login learns your password.)

(c) [3 points] Give two advantages of MAC-based authentication over signature-based authentication.

Solution:

- MACs are faster to compute.
- MACs are shorter (128 bits versus 384 bits or more).

(d) [2 points] Give one advantage of signature-based authentication over MAC-based authentication.

Solution:

- Signatures do not require a shared secret.
- Many parties can verify a single signature.

(e) [5 points] To provide strong security, the database administrator decides to use MAC-based authentication. During account registration, each client will generate a MAC key k and send it to the server. To authenticate, the client will send the message $t = \text{MAC}(k, \texttt{user\_name})$ to the server over the open network. The server, will check that $t == \text{MAC}(k, \texttt{user\_name})$ and will authenticate the client if so.

Notice that if an attacker can observe the communications between Alice and the server, explain how the attacker can impersonate Alice without knowing her key k. Explain how to modify the authentication protocol to eliminate this replay attack (*without* using additional cryptographic tools).

Solution: Attack: The attacker observes Alice's authentication attempt. Then the attacker sends the same tag t to the server later on. Essentially, the MAC tag t here acts as a password.

Fix: At authentication time, the server chooses a random 256-bit nonce and sends it to Alice. Alice must apply the MAC to the nonce and her username.

(f) [4 points] The scheme from the prior part does not authenticate the client's query. Assume that the MAC takes as input 128-bit messages and the client's query can be of arbitrary length. Your friend proposes using a hash function $H\colon \{0,1\}^* \to \{0,1\}^{128}$, to hash the username and query and then feed the hashed values to the MAC. Explain why the resulting authentication scheme cannot have 128-bit security.

Solution: The attacker can find a collision in H in roughly 2^{64} time. Therefore the attacker can find two messages that have the same MAC tag in roughly this much time.

Problem 3. [15 points] **Short Answer Questions** (3 parts)

(a) [5 points] Consider the Diffie-Hellman key exchange protocol working modulo a large prime p with generator g. (In the rest of this problem, we leave the "$\mathrm{mod} p$" operator implicit.) In the usually Diffie-Hellman protocol, Alice chooses a independently and uniformly at random from $\{1, \ldots, p\}$ and sends g^a to Bob. Bob chooses b similarly and sends g^b to Alice.

Now assume that there exists an active adversary who can replace g^a sent by Alice with a new value of the adversary's choosing. Similarly, the attacker can replace the value that Bob sends to Alice. What might the Adversary do to learn the secret key on which Alice and Bob agree? Explain.

Solution: The adversary replaces the g^a with a g^{evil} where it knows $evil$. Bob sees g^{evil} and sends g^b to Alice, which the adversary replaces with g^{bad}, where it knows bad. Alice computes $(g^{evil})^a = g^{evil \cdot a}$, and the adversary can compute this as well, since it knows $evil$ and sees g^a. Similarly, Bob computes $g^{b \cdot bad}$, which the adversary can compute as well, since it knows bad. The adversary will need to continue to intercept traffic between Alice and Bob, decrypt and re-encrypt, since Alice is going to send traffic encrypted with $g^{a \cdot evil}$ to Bob, and Bob expects traffic encrypted with $g^{b \cdot bad}$. But since the adversary knows both keys, this is easy.

(b) [5 points] We are given a random oracle $H : \{0,1\}^* \to \{0,1\}^{256}$. Is $H(x) = x \parallel H(x)$ collision resistant? Is it one way? $\parallel$ denotes concatenation.

Solution: $H(x) = x \parallel H(x)$ is collision resistant because for any $x_1 \neq x_2$, the hash will be different. It is not one-way, since the hash gives away x.

(c) [5 points] Show how to turn any one-way hash function into a hash function that is one-way but not collision-resistant (CR).

Solution: Let g be a one-way hash function. Apply g to all but the last bit of the input to obtain the output for f. Then f is one-way but not CR, since twiddling the last bit of the input doesn't change the output of f.

Problem 4. [5 points] **Generating Randomness** (1 part)
As we have seen in class, cryptography relies on the fact that users can generate randomness (which is needed for encryption, key exchange, etc). However, perfect randomness can be hard to generate. Suppose a user works hard to generate a perfectly random (secret) seed $K \in \{0,1\}^{256}$, but doesn't have the ability to generate more randomness. How can the user use K for all his cryptographic purposes, which require generating many random strings?

Solution: Use K as a seed for a PRF $F : \{0,1\}^{256} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$. The User will keep a counter $c \in \{0,1\}^{128}$ initialized to $c = 0^{128}$. Every time she needs a random string she uses $F(K, c)$ and updates the counter to be $c \leftarrow c + 1$, so that next time she uses the PRF, it is used with an updated counter.

Problem 5. [10 points] **Pseudorandom functions** (2 parts)
Let $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudorandom function, where the first argument is the key and the second argument is the input.

(a) [5 points] Say that, for all $x \in \{0,1\}^n$, it holds that $F(0^n, x) = 0^n$. Can F still be a secure PRF? Why or why not?

Solution: Yes! A PRF only requires $F(K, \cdot)$ to look random for a *random* key K. In particular, any PRF that is changed so that $F(0^n, x) = 0^n$ for every $x \in \{0,1\}^n$ remains a PRF.

(b) [5 points] Let $R(x)$ be a random function from $\{0,1\}^n$ to $\{0,1\}^n$. At how many distinct points $x_1, \ldots, x_k$ must you evaluate R before expecting to find a collision in the values $\{R(x_1), \ldots, R(x_k)\}$ with a significant probability (say, 50%).

Solution: We expect to find collisions after evaluating R on $2^{n/2}$ points. This is precisely the birthday paradox.

Problem 6. [10 points] **Protocol Analysis** (2 parts)
You are given are two protocols in which the sender's party performs the following operation:

Protocol A

$$y = e_{k_1}(x \parallel H(k_2 \parallel x))$$

where x is the message, H is a hash function such as SHA256, e is a symmetric-key encryption algorithm, $\parallel$ denotes concatenation, and k_1, k_2 are secret keys which are only known to the sender and the receiver.

Protocol B

$$y = x,\ E_{PK}(H(x))$$

where SK is a private key of the sender (not shared with the receiver), PK is a public key of the receiver, and E is a public-key encryption algorithm.

(a) [5 points] Provide a step-by-step description of what the receiver does upon receipt of y.

Solution: The receiver does the following in protocol A:

- Decrypt using k_1 and obtain $x \parallel H(k_2 \parallel x)$. Extract x.
- Hash $k_2 \parallel x$ and compare to $H(k_2 \parallel x)$.

In case of protocol B (which is completely flawed):

- Decrypt using SK and obtain $H(x)$. Extract x.
- Hash x and compare to $H(x)$.

(b) [5 points] State whether confidentiality and integrity is achieved for each of the two protocols given in the previous problem. Briefly justify each answer.

Solution:
In the case of protocol A:

- confidentiality: YES through encryption.
- integrity: YES through hashing.

In case of protocol B:

(c) confidentiality: NO, no encryption.

(d) integrity: NO, as anybody can replace the message and compute the hash.

Problem 7. [5 points] **MAC1** (1 part)
Let MAC be a message authentication code for messages in $\{0,1\}^{256}$ that is secure against adaptive chosen message attacks. Let $F : \{0,1\}^{512} \to \{0,1\}^{256}$ be a one-way function (i.e., easy to compute and hard to invert). Consider the new message authentication code for messages in $\{0,1\}^{512}$:

$$\mathsf{MAC}'(K, M) = \mathsf{MAC}(K, F(M)).$$

Is MAC' secure against adaptive chosen message attacks?

Solution:

No. For example consider F that ignores the 256 most significant bits and simply applies a OWF on the 256 least significant bits. Such F is one-way, and yet MAC' is not secure against adaptive chosen message attacks since the tag is the same for all messages that agree on the 256 least significant bits.

Problem 8. [20 points] **Variants on Diffie-Hellman (DH)** (4 parts)
In class, we saw the Diffie-Hellman key-exchange protocol. The protocol works with the integers modulo a large prime $p \approx 2^{2048}$ and uses a carefully chosen generator $g \in \{1, \ldots, p\}$.

The protocols we consider use the following form:

- Alice samples $a \leftarrow_R \{1, \ldots, p\}$ at random and sends Bob $A = g^a \bmod p$.
- Bob samples $b \leftarrow_R \{1, \ldots, p\}$ at random and sends Alice $B = g^b \bmod p$.

(a) [5 points] Assume that one addition or one multiplication modulo p takes time $O(\log p)$ operations. What is the asymptotic complexity of computing $g^a \bmod p$?

Solution: By repeated squaring, we need $O(\log p)$ multiplications. The total number of operations is then $O(\log^2 p)$.

(b) [5 points] Say that Alice and Bob plan to use $g^{a+b} \bmod p$ as their shared secret. What is wrong with this plan?

Solution: A network eavesdropper can recover the shared secret as $A \cdot B \bmod p = g^{a+b} \bmod p$.

(c) [5 points] Say that Alice and Bob plan to use $g^{(a^b)} \bmod p$ as their shared secret. What is wrong with this plan?

Solution: Neither Alice nor Bob can compute this shared secret efficiently using known algorithms. (The usual Diffie-Hellman shared secret is $g^{ab} \bmod p$.)

(d) [5 points] Say that Alice and Bob plan to use $(g^{ab} + g^a + g^b + 127) \bmod p$ as their shared secret. Why is this variant as secure against a passive eavesdropper as the original Diffie-Hellman protocol?

Solution: Assume that there is a passive eavesdropper that can compute the shared secret $S = (g^{ab} + g^a + g^b + 127) \bmod p$. Then, the passive eavesdropper can compute the normal Diffie-Hellman shared secret as $g^{ab} = (S - A - B - 127) \bmod p$.

So, if a passive eavesdropper can break this new protocol, it can also break the usual Diffie-Hellman protocol. (Or, put another way: if the original protocol is secure against a passive eavesdropper, then so is this one.)

Problem 9. [5 points] **Course Survey** (5 parts)

(a) [1 point] What was your favorite topic in the class so far?

(b) [1 point] Are there any topics that you think we should remove next time?

(c) [1 point] Are there any topics that you were hoping to see covered in this class or lab assignments, which aren't currently covered?

(d) [1 point] What was your favorite part of the lab assignments so far?

(e) [1 point] What was the most tedious or least interesting part of the lab assignments?