Practice Problem Solutions

Question	Parts	Points
1: Collision resistance	3	15
2: Hash-and-Sign	1	5
3: MAC	1	5
4: Public-Key Encryption	1	10
5: Encryption in practice	3	15
6: Discrete-logarithm problem	3	15
7: Certificate revocation	2	10
8: Metadata-hiding messaging	4	20
9: LPN Error Correction	2	10
10: iOS Security	3	15
11: Software security	2	10
12: Privilege separation	1	15
13: Runtime defenses	2	10
14: Differential Privacy	2	10
Total:		165

Name: _____

The actual final will be for a total of 180 points.

Problem 1. [15 points] **Collision resistance** (3 parts)
Let $H : \{0,1\}^* \to \{0,1\}^{256}$ be a collision-resistant hash function.

Alice wants to convince Bob that she knows whether the stock market will go up or down tomorrow. But, she doesn't want to give this information to Bob.

Alice expresses her prediction as a bitstring $m \in \{0,1\}^*$ and gives $c \leftarrow H(m)$ to Bob.

After the stock market closes tomorrow, Alice will send m' to Bob, who checks that $c = H(m')$. If this check passes, Bob will accept m' as Alice's prediction. Otherwise, Bob will reject.

(a) [5 points] Show that c *commits* Alice to her prediction. That is, explain why Alice can never trick Bob into accepting a prediction $m' \neq m$.

Solution: If Alice could find (m, m') such that $H(m) = H(m')$ and $m \neq m'$, Alice would have broken the collision resistance of H.

(b) [5 points] Explain why this scheme might leak some information about Alice's prediction to Bob.

Solution: A collision-resistant hash function could leak the first few bits of its input to the output.

(c) [5 points] Say that instead we model $H : \{0,1\}^* \to \{0,1\}^{256}$ as a random oracle. Explain how Alice and Bob could patch their scheme to (a) hide all information about Alice's prediction while (b) still preventing Alice from changing her prediction after the fact. (***Hint***: Remember that all CPA-secure encryption schemes use randomness.)

Solution: Compute $c = H(m\|r)$ for random $r \in \{0,1\}^{256}$, where $\|$ is string concatenation.

Problem 2. [5 points] **Hash-and-Sign** (1 part)
Construct a signature scheme for messages in $\{0,1\}^{512}$ given a signature scheme for messages in $\{0,1\}^{256}$. You can use a hash function H but must state the precise properties you assume H satisfies (as well as its domain and range).

Solution: To sign a message $M \in \{0,1\}^{512}$, take a collision resistant hash function $H : \{0,1\}^{512} \to \{0,1\}^{256}$, and output a signature of $H(M)$ (with respect to the underlying signature scheme for messages in $\{0,1\}^{256}$).

Problem 3. [5 points] **MAC** (1 part)
Show how to use a PRF F that outputs a *single* bit, to design a secure MAC scheme for messages in $\{0,1\}^{128}$. You can assume that F takes as input a key K and a message in $\{0,1\}^*$ and outputs a bit.

Solution: The MAC will have a key K associated with it, and the output will be $\{F(K,(M,b_1,\ldots,b_k))\}_{b_1,\ldots,b_k\in\{0,1\}}$, for a constant k, depending on the desired security.

Problem 4. [10 points] **Public-Key Encryption** (1 part)
Construct a CPA secure (also known as semantically secure) public-key encryption scheme from an arbitrary 2-message key exchange protocol, in which party 1 chooses randomness r_1 and sends a message m_1, which is a function of r_1 (i.e., $m_1 = m_1(r_1)$), party 2 chooses randomness r_2 and replies with a message m_2, which is a function of m_1 and r_2 (i.e., $m_2 = m_2(r_2, m_1)$), and the shared secret s can be efficiently computed from (m_i, r_{1-i}), for any $i \in \{1, 2\}$, and is indistinguishable from uniform given only (m_1, m_2).

You can assume (for simplicity) that the secret key s is a binary string of length k (for some k), and construct an encryption scheme for messages in $\{0, 1\}^k$.

Solution: The KeyGen algorithm samples randomness r_1, computes $m_1 = m_1(r_1)$, and outputs $(sk, pk) = (r_1, m_1)$. The encryption algorithm is given $pk = m_1$ and a message m to encrypt, chooses randomness r_2, computes s from (m_1, m_2, r_2) and outputs $(m_2, m \oplus s)$ as the encryption of m. The decryption algorithm given $sk = r_1$ and a ciphertext $(m_2, m \oplus s)$, uses m_2, r_1 to compute s, and uses this to unmask $m \oplus s$, to obtain the decrypted message m.

Problem 5. [15 points] **Encryption in practice** (3 parts)
To install the Rustup utility, the documentation instructs Linux users to run the shell command:

```
curl https://sh.rustup.rs | sh
```

This shell command downloads the document at the specified URL and pipes it to the shell.

(a) [5 points] Say that you run the command above and that your ISP is malicious. Explain how the ISP could trick your computer into executing a shell script that is different from the one at `https://sh.rustup.rs`.

> **Solution:** The adversary could drop packets after the initial data transfer begins. Curl will output EOF and your shell will execute a prefix of the full shell script.

(b) [5 points] Explain what you (as the user) could do to prevent the attack of part (a), *without* changing anything on the server side.

> **Solution:** First download the entire file, making sure that the TLS connection closed cleanly. Then execute the file.

(c) [5 points] Explain what the Rustup developers could do to prevent the attack of part (a), *without* changing the way that users invoke the script.

Hint: It is possible to define a function in a shell script. The shell will not execute any code in the body of a function definition until the function definition ends and the script explicitly calls the function.

> **Solution:** Code the file in such a way that a prefix of the file will not execute any code. One way to do this is to wrap the entire script in a function definition and then execute the function at the very end.

Problem 6. [15 points] **Discrete-logarithm problem** (3 parts)
The discrete-logarithm problem modulo p is defined as follows:

- **Input:**
  - A large prime p,
  - a generator $g \in \{0, \ldots, p-1\}$, and
  - a value $h \leftarrow g^x \bmod p$ for $x \leftarrow_R \{0, \ldots, p-1\}$.
- **Output:** The unique value x such $h = g^x \bmod p$.

For this problem, assume that multiplying two numbers modulo p takes time $O(\log^2 p)$.

(a) [5 points] How much time does it take to compute $g^x \mod p$ for a random $x \in \{0, \ldots, p-1\}$?

Solution: $O(\log^3 p)$ time

(b) [5 points] The following simple algorithm computes discrete logs modulo p:

Algorithm.

- For $i = 1, 2, 3, \ldots$:
  - Compute $h' \leftarrow g^i \bmod p$.
  - If $h = h'$, output i.

What is the running time of the algorithm?

Solution: $O(p \cdot \log^3 p)$ time

(c) [5 points] Say that you are given B discrete-log problem instances $(h_1, \ldots, h_B)$. If $B = \sqrt{p}$, explain how you can solve at least one of the problems in time $O(\sqrt{p} \cdot \mathsf{polylog}(p))$.

Solution: With good probability, at least one of the discrete logs will have size $O(\sqrt{p})$. The algorithm above will find it in time $O(\sqrt{p})$.

Problem 7. [10 points] **Certificate revocation** (2 parts)
This problem deals with certificate revocation in the public-key infrastructure.

(a) [5 points] The *online certificate status protocol* (OCSP) works as follows: when your browser gets a public-key certificate from a server, the certificate contains the URL of an OCSP server, typically run by the certificate authority who issued the certificate. The client then asks the OCSP server "Is this certificate `<hash of cert>` still valid?" The OCSP server responds with a signed message indicating YES or NO.

Many people worry that OCSP violates client privacy. Why would this be?

Solution: The OCSP server learns which websites the client is visiting.

(b) [5 points] A more recent technology, called *OCSP stapling* has the web server (e.g., `nytimes.com`) fetch a signed OCSP response from its certificate authority indicating that its certificate is still valid.

This eliminates the privacy concern from part (a), but requires the web server to periodically fetch a new signed attestation from the certificate authority. Why would the web server need to refresh its OCSP response?

Solution: The whole point of OCSP is to check whether a certificate is still valid. So the OCSP response includes a timestamp and clients will reject the response if it is too old.

Problem 8. [20 points] **Metadata-hiding messaging** (4 parts)
There are N students, each of whom wants to submit their course evaluations to the registrar anonymously.

To do so, the students propose using a two-server mix net. That is, Server A has a keypair $(\mathsf{pk}_A, \mathsf{sk}_A)$ for a CCA-secure public-key encryption system $(\mathsf{Enc}, \mathsf{Dec})$ and Server B has a similar keypair $(\mathsf{pk}_B, \mathsf{sk}_B)$.

Each student $i \in \{1, \ldots, N\}$ will take her course evaluation m_i and encrypt it as

$$\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_A, \mathsf{Enc}(\mathsf{pk}_B, m_i)).$$

Each student i sends her ciphertext ct_i to server A, who shuffles and decrypts them. Server A sends the resulting ciphertexts to server B who shuffles and decrypts them and sends the resulting plaintext messages to the registrar.

(a) [5 points] If students' evaluations can be arbitrary bitstrings in $\{0,1\}^*$, explain why CCA security may not be enough to hide which student is sending which evaluation.

Solution: CCA-secure encryption schemes do not hide message lengths.

(b) [5 points] Consider an attacker who can see everything that the registrar sees and also can see everything that some number of mix servers sees. How many mix servers does the attacker need to compromise to learn which honest student sent a particular course evaluation?

Solution: Two – all of them.

(c) [5 points] Say that the first mix server is *actively* malicious (i.e., can deviate from the protocol) and colludes with the registrar. What can the first mix server do to learn which honest student is sending which evaluation?

Solution: There are many options. One is: first server drops all messages except that one from student i.

(d) [5 points] To save computation, the sysadmin running the first server proposes an optimization: rather than shuffle all messages according to a random permutation, just shuffle the first half of the ciphertexts and separately shuffle the second half. Explain why this optimized protocol provides a weaker notion of security against an adversarial second server and registrar.

Solution: If the second server and registrar collude, they can learn whether a message came from students $\{1, \ldots, n/2\}$ or $\{n/2 + 1, \ldots n\}$.

Problem 9. [10 points] **LPN Error Correction** (2 parts)
Recall the LPN based error correction scheme (Lecture 17). n refers to the number of bits in the secret key s, m is the number of ring oscillator pairs and the number of bits in the helper data b and noise vector e.

The **Gen** step determines the b vector and exposes it. The **Rep** step chooses the n most stable bits (choosing the n out of m counter values that are the largest absolute values regardless of sign), and uses the corresponding n equations to solve for s.

Consider a modified $(\mathbf{Gen}', \mathbf{Rep}')$, where $\mathbf{Gen}'$, in addition to outputting $b \in \{0,1\}^m$ (as is outputted by **Gen**), also outputs $z \in \{0,1\}^m$ that specifies the n stable locations. $\mathbf{Rep}'$ takes as input (b, z) and a vector e', with the guarantee that e and e' agree on the stable locations (i.e., $e_i = e'_i$ for every $i \in \{0,1\}^m$ s.t. $z_i = 1$), and outputs s.

(a) [5 points] Briefly explain whether or not this scheme has, roughly speaking, equivalent error correction capability to the original scheme.

Solution: Yes. In both schemes, we are assuming that counter values that are large will not change across **Gen** and **Rep** invocations.

(b) [5 points] Briefly explain whether or not this scheme has equivalent security in hiding s and e to the original scheme.

Solution: No. The adversary gets more information about the biometric bits (which of them are the most stable), so the problem faced by the adversary is not exactly LPN.

Problem 10. [15 points] **iOS Security** (3 parts)
The iOS secure boot mechanism operates in three steps:

1. The Boot ROM reads the low-level bootloader code and checks that it is correctly signed using Apple's signing key. The Boot ROM then executes the bootloader.
2. The bootloader code reads the relevant parts of the kernel and checks that they are correctly signed using Apple's signing key. The bootloader then executes the kernel.
3. The kernel then boots up the phone.

(a) [5 points] A much simpler design would just have the entire kernel built into the Boot ROM. Then there would be no need for signature verification or multi-step boot. Explain why this would be a bad idea.

Solution: There would be no way to update the kernel in case of bugs. Also, the ROM is small and a kernel is big.

(b) [5 points] When the device boots, it reads Apple's signature-verification key from ROM into a buffer in memory and uses this key for signature verification. An attacker finds a bug in the boot ROM: the bug enables the attacker to overwrite the buffer in memory that stores Apple's public key. The overwrite happens early in the boot process—before the device has verified any signatures. Explain how the attacker can now execute any code on the phone she wants.

Solution: The attacker replaces Apple's public key with her own. Then she replaces Apple's bootloader with her own bootloader that runs arbitrary code.

(c) [5 points] Say that an attacker uses the attack of part (b) to cause your phone to boot a malicious version of iOS. Later on, you reboot your phone. Explain why, on reboot, you will either have a clean OS or you phone will not boot.

Solution: The boot ROM is read only, so on boot your phone will load Apple's true signature-verification key into memory. The secure boot process will then refuse to load the evil OS.

Problem 11. [10 points] **Software security** (2 parts)
Alyssa P. Hacker is designing a large software system and wants your advice on security.

(a) [5 points] Alyssa needs to communicate securely over the Internet between computers located in different data centers. Her requirements are relatively simple: just encrypting the TCP connections between a small number of computers. She is considering either using a large existing software library that supports many features (e.g., the OpenSSL library that implements TLS), or implementing a much smaller library that supports just the small number of features she requires. What advice would you give her, and why?

Solution: Use OpenSSL. Much better tested. Many possible bugs and considerations in a network and crypto library in particular, which may be difficult to discover for a one-off implementation, despite being in principle simpler. Even though OpenSSL is large, it is likely better-tested than a smaller replacement would be.

(b) [5 points] Alyssa's application will need to send customers records between her servers and applications running on users' smartphones. Alyssa is wondering how she should represent her customer records in network messages, in terms of encoding and decoding them. One of her concerns is that existing data serialization libraries are complex, large pieces of code, and that more code means more bugs. She thinks her customer records can be implemented with a simpler encoder/decoder. Would you suggest that Alyssa use an existing data encoding library, or write a simpler version on her own? What requirements should she consider for the encoding library, whether she use an existing one or write her own? Why?

Solution: She should find an existing message format and associated encoding/decoding library that is designed to handle arbitrary inputs on decoding. It's error-prone to write code that parses arbitrary inputs, so it's good to leverage existing effort by other library developers. However, it's important that the library was designed for this (e.g., not Python's pickle library that was discussed in lecture).

Problem 12. [15 points] **Privilege separation** (1 part)
Ben Bitdiddle is developing a stock trading application for smartphones. Unfortunately, he has gotten a bit carried away with features, and his application has grown to hundreds of thousands of lines of code! He is worried about security.

The main security goal for his application is to ensure that stocks are traded (bought or sold) only when the user authorizes that operation. It would be a problem if an adversary could take advantage of the many potential bugs in Ben's enormous application running on the user's phone to send fake trades on behalf of the user.

(a) [15 points] Succinctly describe how Ben could use privilege separation to revise his design to improve confidence in achieving his security goal. You may need to make some changes to how the backend servers expect to interact with the app running on the user's phone, and you may need to refactor Ben's application into multiple apps that are largely isolated from one another. Describe any interfaces you introduce between your components. Use bullet points to make your design description more compact; if you're writing 10 bullet points, that's probably too verbose.

Solution: Split Ben's smartphone app into two: one handles login and trade approval ("A"), and another does everything else ("B"). Upon login, when the "A" app launches the "B" app, it should give it a reduced form of credentials that allows it to do everything except issue trades on the user's behalf. Adjust the server to support these two kinds of credentials (cookies, presumably). When the giant app "B" wants to make a trade, issue a call to app "A" to perform a trade. "A" should prompt the user, showing what trade is about to take place, and the user must explicitly approve it before "A" sends the request to the server authenticated with "A"'s cookie.

Problem 13. [10 points] **Runtime defenses** (2 parts)

(a) [5 points] Ben Bitdiddle wants to secure his gaming server written in C, by using a "fat pointer" runtime defense. Describe how an adversary could still exploit some memory errors in Ben's C code despite this defense.

Solution: Overwrite the contents of another struct field in a struct that contains an overflowed buffer.

Take advantage of a use-after-free vulnerability to overwrite some new data structure using an old fat pointer that refers to the same memory region.

(b) [5 points] Ben Bitdiddle decides fat pointers aren't a good idea for him, and instead enables control flow integrity when compiling his C code. Is it possible for an adversary to exploit a buffer overflow in Ben's C code despite this defense? What would the adversary require to pull off such an attack, or why is it impossible?

Solution: Possible if adversary can overwrite some important non-control-flow data.

Possible if application uses computed jumps to invoke some sensitive code like `system()` and the adversary finds a buffer overflow that corrupts a function pointer that then gets invoked.

Problem 14. [10 points] **Differential Privacy** (2 parts)
Let f be a function over sensitive data sets. The data sets D and D' both have n records. We will define the global sensitivity of f as:

$$S(f) = \max_{\forall dist(D,D')=1} |f(D) - f(D')|$$

where $dist(D, D') = 1$ if and only if D' can be obtained from D by changing *one* record in D.

Any possible record has a single value that is in between the minimum value a and maximum value b.

(a) [5 points] What is the global sensitivity when f is the (arithmetic) mean? In order to get ϵ-differential privacy, what noise distribution should we add to the outcome $f(D)$?

Solution: $(b-a)/n$. Noise distribution that needs to be added has mean $\frac{b-a}{\epsilon n} Lap(0,1)$.

(b) [5 points] What is the global sensitivity when f is the variance? In order to get ϵ-differential privacy, what noise distribution should we add to the outcome $f(D)$?

Solution: $(b-a)^2/n$. Noise distribution that needs to be added has mean $\frac{(b-a)^2}{\epsilon n} Lap(0,1)$.