Homework 1: Crypto Party

Due: Thursday, February 11 @ 11:59 pm ET

Do not include any identifying information (name, login, Banner ID, etc.) on your handin. When you're done, please submit a PDF file to Gradescope where your answer to each problem (not each individual question) is on a separate page. Make sure to assign each problem on Gradescope to the correct pages in your writeup (or you may receive a deduction).

Unless otherwise stated, you should justify and explain all of your proposed attacks, schemes, protocols, defenses, etc., referencing specific properties when necessary. Precise, effective communication and presentation of your ideas and solutions in your writeups is something we highly value when grading. Similarly, be concise, but make sure that you don't elide intermediate details, take subtleties for granted, or make any assumptions without justification.

While we encourage everyone to discuss and work through the problems on the homeworks, please remember that your homework solutions must be written up *entirely independently*, in your own words. You may not share your solutions with anyone (or read solutions written by others). You should not write your solutions while working with other students, and when you're writing your solutions, you should ensure that you independently understand and can reproduce your answers without referring to notes from collaboration sessions and consulting with other students.

Reminder: At the top of every problem (not question), you should write a *collaboration statement* that states who you discussed the problem with, who contributed major intellectual insights to your understanding of the problem, and any external resources you referenced while solving the problem. See the *Collaboration Policy* for examples. If you did not collaborate with anyone and did not refer to any sources, your collaboration statement should state as much.

Homework 1 Instructions

Answer Problems 1–5. If you're a CS162 student, you must also answer Problems 6 and 7.

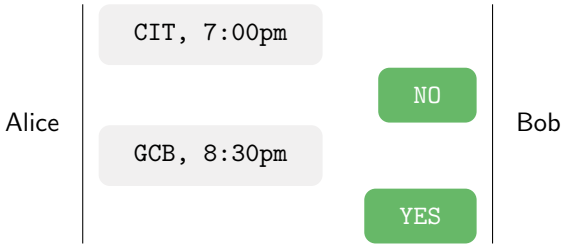
When you're ready to submit, you should hand in your answers to Problems 1–5 to the "CS166: Problems 1–5" drop on Gradescope. CS162 students should also hand in their answers to Problems 6–7 to the "CS162: Problems 6–7" drop on Gradescope in a separate PDF. (Answers submitted by CS166-only students to the CS162 questions will not be graded.)

Questions start on Page 2.

Problem 1: Dating with Public Keys

Alice and Bob, both Brown CS students, are secretly dating. In order to set up a meeting, they exchange encrypted messages using a *deterministic* public key encryption scheme (deterministic meaning that multiple encryptions of a given plaintext always produce the same ciphertext).

Alice and Bob both have their own public-private key pair, (PK_A, SK_A) and (PK_B, SK_B) respectively. When Alice wants to send a message, m, to Bob, she encrypts it using his public key and sends the resulting ciphertext, $c = \mathsf{Enc}(m, PK_B)$, to him. Similarly, Bob's messages to Alice are computed from $\mathsf{Enc}(m, PK_A)$. In plaintext, the messages Alice and Bob exchange are always of the following form:



Assume that they always plan to meet at a Brown building, and when referring to a specific Brown building, the name they use is consistent (i.e. they won't alternate between "SciLi" and "Sciences Library").

For **all parts of this problem**, limit each of your answers to **150 words maximum**—longer responses will receive no credit.

Question a) Sierra is Alice's curious roommate who wants to find out about the secret dates between Alice and Bob. She knows both of their public keys, the form of their messages (including the name they use to refer to all of Brown's buildings), and she can eavesdrop on the ciphertexts being exchanged. Describe how Sierra can find out when and where the next meeting is going to be, even though she is unable to learn the secret keys.

Question b) Alice and Bob found out that Sierra can learn about their meetings. Propose a *simple* modification to the protocol that prevents the attack from part (a) and explain why the protocol will prevent the attack, and explain why it works. By "simple", your new protocol cannot rely on sending additional messages or require changing any of the cryptographic functions involved or adding new cryptographic functions.

Question c) TRUE or FALSE: Sierra is an IND-CPA adversary. Explain.

Problem 2: Commitments (in the Midst of a Breakup)

After Bob didn't show up at the Graduate Center Bar at 8:30pm (even though he previously agreed to go there in Problem 1), Alice decided that it was time for them to break up. They no longer trust each other, so much in fact that they refuse to be together in the same room. As part of the breakup, Bob wants to take one of the n houseplants he and Alice previously shared. Alice agrees, but she likes some houseplants more than others, and so she doesn't want Bob to just be able to pick any particular one.

To settle this, they send texts to each other over a communication channel (which guarantees message delivery, but the time to deliver a given message is unpredictable) and assign unique numbers $\{0, 1, ..., n-1\}$ to each plant. Then, they agree that Bob should roll an n-sided die, and Bob will receive the plant that corresponds to the number Bob rolled. However, suppose Bob really wants a particular houseplant with associated number b. All Bob needs to do is simply notify Alice that he rolled b, regardless of what his n-sided die actually rolled.

For **parts (b) and (c)**, limit each of your answers to **50 words maximum**—longer responses will receive no credit. Part (a) has no word limit, *but your entire answer to this specific problem has to fit on a single page in your writeup*, or part (a) receives no credit.

Question a) Using cryptography, design a protocol between Alice and Bob that emulates the rolling of an unbiased n-sided die (that is, the outcome is uniformly distributed in $\{0, 1, ..., n-1\}$), and does it in such a way that neither Alice nor Bob can "cheat" during the protocol; that is, neither Alice nor Bob can affect the outcome of the die roll without the other person knowing. (Recall that Alice and Bob don't want to meet in person and communicate only via text messages. Your solution should *not* involve the use of a third-party.)

If a player detects cheating during the course of the protocol, your protocol should specify that the player should *abort* from the protocol; that is, they stop participating and no outcome is reached.

Then, explain why your protocol prevents Alice and Bob from cheating. Specifically, explain why the outcome is uniformly distributed in $\{0, 1, ..., n-1\}$ as long as at least one player follows the protocol honestly and no player aborts during the protocol. Your justification should reference all of the relevant properties of the cryptographic methods you have employed in the protocol.

Question b) In your protocol from part (a), is it possible for Alice or Bob to discover the outcome of the die roll before their ex? If so, what could they do to prevent their ex from knowing the outcome?

Question c) JC is a trustworthy friend of Alice and Bob. JC is known for keeping their friends' secrets safe and would never be bribed. JC can communicate via text messages with Alice and Bob and keep track of messages previously exchanged. However, JC cannot perform any computations. In particular, JC cannot execute arithmetic operations, generate random numbers, run programs, or apply cryptographic methods to the content of the messages transmitted by Alice and Bob.

Can JC help Alice and Bob with their protocol such that they learn the outcome of the die roll at the same time (if at all)? If yes, explain how and why your proposed protocol ensures both Alice and Bob receive the outcome at the same time; if no, explain what parts of the protocol prevent such a system from existing.

Problem 3: Hybrid Cryptosystems

Zachary says: "This was a midterm problem from a previous offering of the course! You might want to work on it independently before discussing with others to get a sense of how the midterm works."

Consider Alice and Bob, two former lovers who exchange confidential messages over an insecure channel. Alice and Bob both have their own public key (PK_A, PK_B) and private key (SK_A, SK_B), and Alice and Bob each know each other's public keys. When Alice and Bob want to communicate, they keep their communication confidential by first exchanging symmetric keys k_A and k_B in the following "setup phase":

1. Alice randomly generates a symmetric key, k_A, encrypts it with Bob's public key using an IND-CPA-secure public key cryptosystem, and sends the resulting ciphertext as a payload to Bob.
2. Bob decrypts the ciphertext using his private key, and learns k_A.
3. Bob randomly generates a symmetric key, k_B, encrypts it with Alice's public key using an IND-CPA-secure public key cryptosystem, and sends the resulting ciphertext as a payload to Alice.
4. Alice decrypts the ciphertext using her private key, and learns k_B.

After doing this, Alice and Bob can send symmetrically encrypted messages to each other in the "chat phase": whenever Alice wants to send a chat message to Bob, she encrypts it with k_B using some arbitrary symmetric cryptosystem and sends the resulting ciphertext; when Bob wants to send a message to Alice, he does the same except using k_A. (It's also worth noting that Alice and Bob's chat messages are arbitrary, sometimes seemingly random and unpredictable plaintexts.)

For **all parts of this problem**, limit each of your answers to **200 words maximum**—longer responses will receive no credit.

Question a) Consider an adversary, Erica, who knows PK_A and PK_B, is able to intercept payloads sent between Alice and Bob, and is able to *modify or replace them entirely* with payloads of her own.

Show how Erica can perform an attack that allows her to read the contents of messages Alice sends to Bob (and vice versa). (*Hint:* There's no requirement that Alice and Bob have to be able to read each other's messages after Erica's attack.)

Question b) What is something *simple* that Alice and Bob can do to ensure they can detect if the attack described in part (a) is happening (but not necessarily prevent it)? Explain. By "simple", your answer should not involve any applications of cryptography outside of what's already defined in the protocol, and should not require Alice and Bob to agree beforehand on some predetermined shared secret or phrase (since then they might just agree on a symmetric key).

Question c) We'd like to modify the "setup phase" of the chat protocol to fix the problems described in part (a). Alice and Bob would like to avoid simply defaulting to public/private key encryption for all messages—as we've previously discussed, this is much slower than symmetric key encryption, and if they are chatting frequently then the performance overhead might be too great.

Explain how to modify the "setup phase" of the protocol such that the attack from part (a) is impossible, but still preserves usage of symmetric-key-only encryption in the chat phase.

Question d) For this question, ignore the changes you proposed in part (b) and (c). Consider a slight modification to the scenario where Alice and Bob do not know each other's public keys prior to the "setup phase". They quickly resolve this issue by sending their public keys to each other as payloads. At this point, they now know each other's public keys, and therefore can proceed to the "setup phase" as normal.

Does Erica have less, the same, or more capabilities in this scenario than she did in the attack in part (a)? Explain your answer. (To be specific, her "capabilities" refer to her ability to read the contents of Alice and Bob's chat messages.)

Problem 4: Exceptional Access

Law enforcement agencies have been lobbying for exceptional access to encrypted communications. However, many cybersecurity experts have argued that enabling exceptional access would have untenable security consequences. Two cryptographers from GCHQ (the UK's equivalent of the NSA) have proposed a method of exceptional access that they believe does not compromise the integrity of encryption in the following article: `https://www.lawfareblog.com/principles-more-informed-exceptional-access-debate`. Please reference the article in your answers.

The second article describes a Cisco Webex vulnerability which unintentionally allows "ghost access" to meetings. Feel free to refer to this article in your answers (though this article is optional): `https://www.securityweek.com/cisco-webex-vulnerability-allows-ghost-access-meetings`

Question a) The first article discusses several principles that can be used to evaluate exceptional access methods. What standards would need to be followed for you to find an exceptional access reasonable (3 standards max)? If you believe that exceptional access is never acceptable, please justify why. How does your answer differ from the GCHQ principles, if at all?

Question b) Do you believe that the GCHQ authors' fifth principle—that "any exceptional access solution should not fundamentally change the trust relationship between a service provider and its users"—is met by their proposed exceptional access mechanism? Why or why not? (4–6 sentences)

Question c) One argument when it comes to exceptional access is that if companies don't provide a mechanism for the government to access communications between users, then the only option left for government agencies is hacking. Resorting to hacking negatively impacts government agencies' incentives to disclose vulnerabilities. Do you find this argument in favor of exceptional access convincing? Why or why not? (4–6 sentences)

These are open questions and will be graded for thought and justification. Please reference and/or quote specific arguments from the readings in your answers.

Problem 5: CIT-Branded Block Ciphers

Charles has ignored all of the warnings about hand-rolling your own cryptography (see the Lecture 3 readings) and has created a new block cipher mode called CIT mode. Figure 1 provides an overview of how CIT mode works—you can assume that the block cipher used is a secure, deterministic block cipher with a block size and key size of 256 bits.

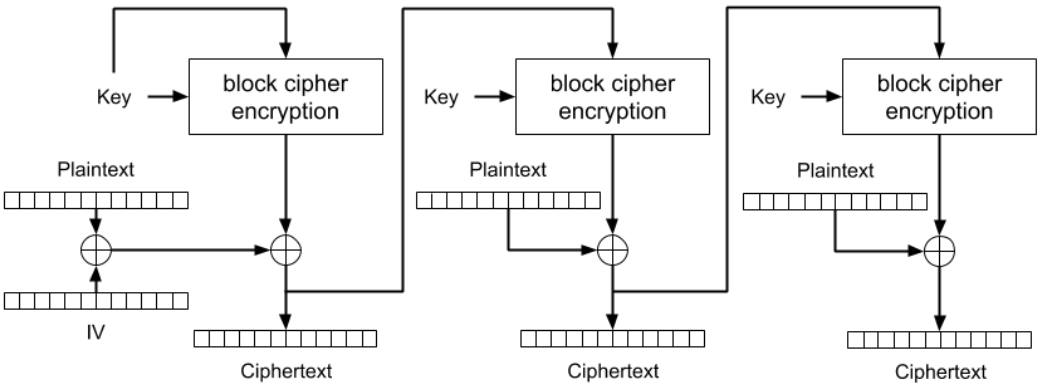


Figure 1: The encryption scheme for CIT mode.

(And yes, you're seeing that correctly—the CIT block cipher mode involves encrypting the encryption key with itself...)

For **parts (a) through (e)**, limit each of your answers to **100 words maximum**—longer responses will receive no credit. Part (f) has no word limit.

Question a) Draw a diagram of the decryption scheme for CIT mode.

Question b) Suppose Charles encrypts two equal-length plaintext messages, m_1 and m_2, using the same key *and the same initialization vector* under CIT mode, to produce two ciphertexts, c_1 and c_2. m_1 and m_2 differ in at least one bit at some index i.

Consider an adversary, Marcus, who can eavesdrop on the ciphertexts produced by Charles; that is, Marcus can see c_1 and c_2. What, if anything, can Marcus learn about the underlying plaintexts m_1 and m_2?

Question c) Suppose that exactly one bit at index j of a given ciphertext, c, which was encrypted in CIT mode, is corrupted in transmission (that is, after the encryption is complete). When c is decrypted using CIT decryption, which bits of the plaintext will be corrupted? Explain.

Question d) TRUE or FALSE: Encryption in CIT mode is parallelizable.[1] Explain.

Question e) TRUE or FALSE: Decryption in CIT mode is parallelizable. Explain.

Question f) TRUE or FALSE: CIT mode is IND-CPA-secure. Explain.

(*Hint:* In a IND-CPA game, the key remains fixed throughout the duration of the game, but the IV used will be different for every encryption.)

[1] "Parallelizable" in the context of block ciphers comes from our discussion of ECB mode in Lecture 3: that is, for a given input plaintext (split into fixed-length blocks), we should be able to compute ciphertext blocks without having to wait for previous ciphertext blocks (and thus we can parallelize the computation of the ciphertext blocks), and vice versa for decryption of ciphertext blocks to plaintext blocks. Thus, we say encryption and decryption in ECB mode is parallelizable.

Problem 6: Hash Functions, RSA Edition?

Only CS162 students are required to complete this problem.

In the *Cryptography II* lecture, we discussed how public-key digital signature cryptosystems generally form signatures *over a cryptographic hash* of the intended message. Consider the standard RSA public-key cryptosystem described below, which we'll denote as $\mathsf{RSA}_{\text{std}}$ (pronounced as "RSA standard"):

- A standard RSA key-pair is formed via public key $PK = \langle n, e \rangle$, where n is the "RSA modulus" (which is a large prime number that defines the key length) and e is some fixed, small, prime number, and private key $SK = d$, where d has the property $(x^e)^d = x \mod n$ for all x. (For the purpose of this problem, exactly how these numbers are generated doesn't matter.) In this problem, let's assume $e = 3$.
- In $\mathsf{RSA}_{\text{std}}$, the signing operation $\mathsf{Sign}_{SK}(M)$ works by computing $S = \mathsf{hash}(M)^d \mod n$, where S is the signature on M. To verify the signature S, a third-party can execute $\mathsf{Verify}(M, S, PK)$, which results in a correct verification if $S^3 = \mathsf{hash}(M) \mod n$ holds true. (Observe that the verification algorithm does not require knowledge of d, only $PK = \langle n, 3 \rangle$.)

You can assume that the hash function satisfies all of the properties of cryptographic hash functions.

Observe that $\mathsf{RSA}_{\text{std}}$'s Sign operation is over the hash of the message M. We discussed in lecture how signing a cryptographic hash has performance benefits over signing the full message; specifically, by signing a smaller message (the hash), the signing operation is faster overall. However, it's worth asking whether or not the cryptographic hash has anything more to do with the security of the cryptosystem as well. For instance, if we're in a situation where all of our messages are smaller than the output length of hash, perhaps removing the hash function will result in a more performant cryptosystem!

Thus, consider the following simplified scheme $\mathsf{RSA}_{\text{simple}}$. In $\mathsf{RSA}_{\text{simple}}$, $\mathsf{Sign}_{SK}(M)$ is exactly the same as the corresponding operation from $\mathsf{RSA}_{\text{std}}$, except without the application of the hash function. That is, in $\mathsf{RSA}_{\text{simple}}$, $\mathsf{Sign}_{SK}(M)$ produces the signature $S = M^d \mod n$ and $\mathsf{Verify}(M, S, PK)$ checks if $S^3 = M \mod n$ holds true.

Question a) Zachary and Lilika are using $\mathsf{RSA}_{\text{simple}}$ to send messages with integrity guarantees to each other. William is a network attacker who can inject, modify, and drop messages sent between Zachary and Lilika, and William wants to trick Lilika into believing that Zachary sent a message that Zachary did not.

Explain how William, who knows Zachary's public key PK, can find a (M, S) pair (and give a specific example of such a pair) such that S is a valid signature on M, even without knowledge of Zachary's secret key SK. (For part (a), the message M does not have to be chosen in advance and can be gibberish.)

Question b) Bernardo is holding an auction for his personal collection of balloon animals. In this auction, Zachary and Lilika submit bids to Bernardo, where their bids are signed using $\mathsf{RSA}_{\text{simple}}$. Each bid M is an integer (in dollars), and they will send their $\mathsf{RSA}_{\text{simple}}$ signature on M; that is, they send $S = M^d \mod n$. Bernardo will accept whichever bid is highest (and will expect that person to pay up however much they bid!).

Suppose Zachary sends a bid M (and its corresponding signature S) to Bernardo. Is it possible for William to tamper with S in such a way that he can find a new signature, S', that corresponds to a bid that is some x times as much as Zachary's original bid? Explain. (You can choose any value of $x > 0$, and you can assume that $xM < n$ for your chosen x.)

Question c) Explain why the use of a cryptographic hash function in $\mathsf{RSA}_{\text{std}}$ prevents the forgery attacks described in parts (a) and (b). (*Hint:* How do the properties—and which properties—of a cryptographic hash function prevent these kind of attacks?)

Problem 7: Counting for Cryptographers is Hard

Only CS162 students are required to complete this problem.

In the *Cryptography II* lecture, we described how one can convert a block cipher into a stream cipher using a "counter" technique. This technique is actually more formally known as CTR block cipher mode encryption. Specifically, $\mathsf{CTR}_{\text{lecture}}$ (to denote the scheme specifically shown on Slide 17 in Lecture 3) works in the following way:

- The encrypting user knows the secret key $\langle k, t \rangle$, where k is a random bitstring and t is a "counter" with a number of bits equal to the block size.
- Given a plaintext $M = m_0||m_1||\cdots||m_{n-1}$, the ciphertext generated by $\mathsf{CTR}_{\text{lecture}}$ mode is

  $$C = c_0||c_1||\cdots||c_{n-1}$$

  where each $c_i = E_k(t+i) \oplus m_i$.
- After encrypting a message M (not after each individual m_i), the user updates their secret key $\langle k, t \rangle$ to $\langle k, t+1 \rangle$.

For this question, assume that the block size of the block cipher used is 32 bits.

Question a) TRUE or FALSE: $\mathsf{CTR}_{\text{lecture}}$ mode is IND-CPA-secure against an attacker with polynomially bounded resources. Explain. (*Hint:* The answer is FALSE.)

Question b) Is there a way to easily fix the security issues described in part (a)? Under your proposed mitigation, are there any limitations on the number of times the encrypting user can use a given secret key k under your new version of CTR mode? Explain. (*Hint:* How many times can you use the secret key k?)