Homework 1: Crypto Party

Due: Thursday, February 23 @ 11:59 pm EST

Overview and instructions

This homework has 6 problems:

- Problems 1–4 are required for all students
- Problems 5–6 are required for **CS1620/CS2660 students only**

Note on collaboration

You are welcome (and encouraged!) to collaborate with your peers, but the solutions you write down must be **your own work** (ie, written by you). You are responsible for independently understanding all work that you submit—after discussing a problem as a group, you should ensure that you are able to produce your own answers independently to ensure that you understand the problem. For more information, please see the course Collaboration Policy.

In your submission, we ask that you include a brief *collaboration statement* describing how you collaborated with others on each problem—see the next section for details.

How to submit

You will submit your work in PDF form on Gradesope. Your PDF should conform to the following requirements:

- **Do not** include any identifying information (name, CS username, Banner ID, etc.) in your PDF, since all homeworks are graded anonymously
- Each problem (where "problem" is one of the Problems 1–4 or 1–6) should start on a separate page. When you submit on Gradescope, you will be asked to mark which pages correspond to which problem
- At the start of each problem, write a brief *collaboration statement* that lists the names and CS usernames of anyone you collaborated with and what ideas you discussed together
- If you consulted any outside resources while answering any question, you should cite them with your answer

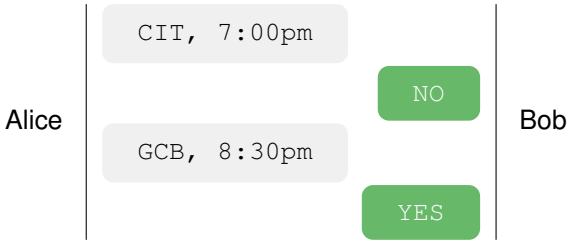
There are two separate Gradescope submissions for this assignment:

- All students should submit Problems 1–4 to the assignment labeled **"Homework 1: Problems 1–4"**
- CS1620/CS2660 students must also submit Problems 5–6 to the assignment labeled **"Homework 1: Problems 5–6"**. For this part, you can either make a separate PDF with problems 5–6, or just have one PDF and then mark the pages for these problems. Submissions for Problems 5–6 from CS1660-only students will not be graded (ie, there is no extra credit).

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 Bob's 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 building on Brown's campus, and that,when referring to a specific Brown building, the names they use are consistent (i.e. they won't alternate between "SciLi" and "Sciences Library").

Answer the following questions based on this scenario. Your responses to each question should be no more than one paragraph (around 150 words) each.

Question a) Eve wants to find out about the secret dates between Alice and Bob and 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 can eavesdrop on the ciphertexts being exchanged. Describe how Eve can find out when and where the next meeting is going to be, even though Eve is unable to learn the secret keys. (*Hint:* The encryption scheme is *deterministic*).

Question b) TRUE or FALSE: Eve is an IND-CPA adversary. Explain.

Question c) Alice and Bob found out that Eve 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.

Problem 2: 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 read over this article and then consider the following questions.

For another example, this (optional) article describes a Cisco Webex vulnerability which unintentionally allows "ghost access" to meetings:
`https://www.securityweek.com/cisco-webex-vulnerability-allows-ghost-access-meetings`

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

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, ie. finding or exploiting known vulnerabilities in a system. However, relying on such vulnerabilities can incentivize governments to avoid disclosing them to the public so they remain useful to law enforcement. Do you find hacking a compelling mechanism for providing exceptional access? Why or why not? (4–6 sentences)

Problem 3: CIT-Branded Block Ciphers

Someone 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.

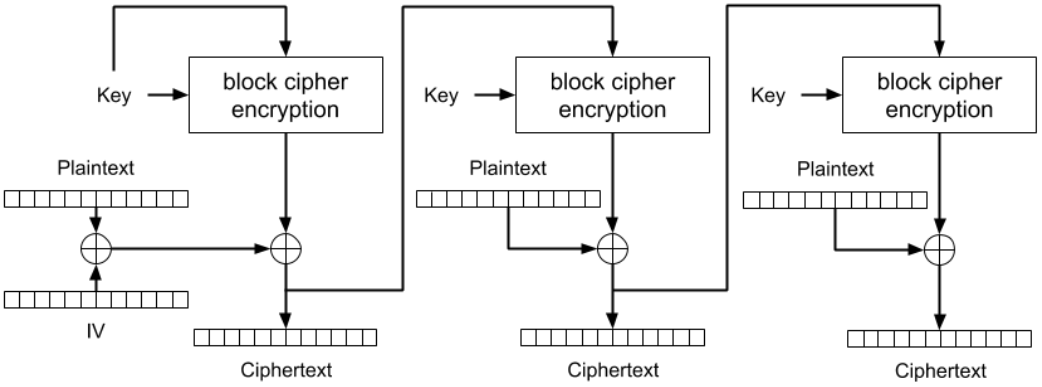


Figure 1: The encryption scheme for CIT mode.

(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**. Part (f) has no word limit.

Question a) Either draw a diagram or write out equations of the decryption scheme for CIT mode using the notation for describing cipher modes from the lectures[1]. Make sure to label all parts of your diagram or variables used.

Question b) Suppose CIT mode is used to encrypt two equal-length plaintext messages, m_1 and m_2, using the same key *and the same initialization vector* 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, Eve, who can eavesdrop on the ciphertexts; that is, Eve can see c_1 and c_2. What, if anything, can Eve 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.[2] 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] You can see examples for writing cipher mode equations in the Cryptography II lecture, slides 12 and 16.

[2] "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 4: TryHackMe Lab: Burp Suite

If you completed HW0, you should have already registered for a TryHackMe account. If so, you have been added to the CS1660 course and have been granted a premium account. If you have not completed HW0, please register for TryHackMe as soon as possible—you may have already received an invite to do so when we approved your account for premium access.

For this problem, you will gain some practice using Burp Suite, a set of tools for testing web applications, which you may find useful for the next project. Depending on when you start this lab, some of the topics about web applications may not have been discussed yet—we will review them in the next few web security lectures.

To begin the lab, log into TryHackMe and go to `https://tryhackme.com/jr/cs1660burpra`, then follow the instructions.

Grading note There is nothing to submit for this part. As you complete each task in the lab, your progress is automatically recorded such that we can view it. TryHackMe rooms are graded based on *completion*, not correctness. As long as you have answered all of the questions, you will get full credit. This lab should not take more than 1 hour to complete—if you are stuck or are dealing with technical issues, make sure to post a question on Edstem.

Problem 5: Hash Functions, RSA Edition? (1620/2660 only)

Note: Only CS1620/CS2660 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. Signing a hash of the message (rather than the message itself) has performance benefits for long messages, since asymmetric cryptography is quite expensive. However, if we are in a situation where the messages are always small (smaller than the output length of hash), does removing the hash function result in a more performant cryptosystem that's just as secure? This problem gives a bit more background on how RSA works, and explores a few scenarios to help us answer this question.

Background: RSA signatures, with hashing Consider the standard RSA public-key cryptosystem described below, which we'll denote as "RSA standard" ($\mathsf{RSA}_{\text{std}}$):

- A standard RSA key-pair is formed via public key $PK = \langle n, e \rangle$, where n is the "RSA modulus" (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 $S^e = \mathsf{hash}(M) \mod n$ holds true. (Note that the verification step does not require knowledge of d, only $PK = \langle n, e \rangle$.)

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

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

Explain how Eve, who knows Alice'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 Alice's secret key SK. (For part (a), the message M does not have to be chosen in advance and can be random.)

Question b) Bernardo is holding an auction for his personal collection of balloon animals. In this auction, Alice and Bob 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 Alice sends a bid M (and its corresponding signature S) to Bernardo. Is it possible for Eve 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 Alice'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 6: Counting for Cryptographers is Hard (1620/1660 only)

Note: Only CS1620/CS2660 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 21 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 an entire 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?)