Paxson
Spring 2017

CS 161
Computer Security

Midterm 2

PRINT your name: ____________________________, ____________________________
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct will be reported to the Center for Student Conduct, and may result in partial or complete loss of credit.

SIGN your name: ____

PRINT your class account login: `cs161-`______ and SID: _________________________

Your TA's name: ____

Your section time: ____

Exam # for person sitting to your left: _______________ Exam # for person sitting to your right: _______________

You may consult one sheet of paper (double-sided) of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted.

You have 80 minutes. There are 5 questions, of varying credit (300 points total). The questions are of varying difficulty, so avoid spending too long on any one question. Parts of the exam will be graded automatically by scanning the **bubbles you fill in**, so please do your best to fill them in somewhat completely. Don't worry—if something goes wrong with the scanning, you'll have a chance to correct it during the regrade period.

If you have a question, raise your hand, and when an instructor motions to you, come to them to ask the question.

Do not turn this page until your instructor tells you to do so.

Question:	1	2	3	4	5	Total
Points:	80	60	40	60	60	300
Score:						

Problem 1 ***True/False*** **(80 points)**

For each of the following, FILL IN THE BUBBLE next to **True** if the statement is correct, or next to **False** if it is not. Each correct answer is worth 4 points. Incorrect answers are worth 0 points. Answers left blank are worth 1 point.

(a) Public key encryption is usually slower and has more overhead than symmetric key encryption.

[✓ **True**] **False**

(b) If Alice comes up with a new idea that she wants to patent, to prove that she was the one who came up with the idea she can generate a digital signature of a description of the idea.

True [✓ **False**]

Solution: Digital signatures provide *non-repudiation*, which means that Alice cannot *deny* that she indeed signed some statement (except by arguing that her private key has been stolen). They do not provide "provenance," i.e., establishing that Alice is the *original* creator of some message that she signs, since she could have gotten the message from somewhere else.

(c) A cryptographic hash function is deterministic: given the same input, it always produces the same output.

[✓ **True**] **False**

(d) A block cipher is deterministic: given the same input and key, it always produces the same output.

[✓ **True**] **False**

(e) Alice and Bob share a symmetric key K, known to nobody else. Alice sends Bob a message $m =$ "I owe you $100" and attaches a MAC tag t computed using K and m. Bob can use m and t to prove to a third party that Alice sent m.

True [✓ **False**]

Solution: Either Alice or Bob could have computed t, because they both know K. Furthermore, a third party cannot verify the tag without also knowing K.

(f) A Diffie–Hellman exchange is secure against a passive attacker (Eve) but not against an active attacker (Mallory).

[✓ True] ○ False

(g) Suppose Alice has two secret keys, k_1 and k_2. If she encrypts messages using AES-CBC encryption with k_1, then she can generate secure IV's by computing $IV_i = \text{HMAC}(m_i, k_2)$, i.e., the IV is the HMAC of the message (m_i) using secret key k_2.

○ True [✓ False]

Solution: HMAC is deterministic, so the same message will always produce the same IV, leading to deterministic encryption.

(h) For AES encryption, Alice should not use the same symmetric key to encrypt more than one message.

○ True [✓ False]

Solution: Our keyed cryptographic solutions (whether symmetric or public-key) are designed to allow repeated reuse of keys. This is important because keys take extra effort to create or share.

However, during the exam it became apparent that some students observed that the problem simply stated "AES" without specifying a block cipher mode. If one were to use AES without any accompanying mode, then to use it securely *would* require changing keys for each new message (where presumably each message is a single block). Given this ambiguity, we decided to remove the question, so it is worth 0 points. (Doing so also makes the point count for the True/False questions match what we originally intended, i.e., 20 questions totalling 80 points.)

(i) Given a random key, Alice can use AES-ECB to create a PRNG.

[✓ True] ○ False

Solution: Alice can just encrypt monotonically increasing counters (0, 1, 2...) to generate pseudorandom outputs. Since the counters never repeat, the output of AES with a random key is a series of pseudorandom blocks.

(j) To communicate confidentially in the presence of an eavesdropper, Alice and Bob can use the Diffie-Hellman protocol to exchange RSA public keys that they then use for public key encryption.

○ **True** [✓ ● **False**]

> **Solution:** The Diffie-Hellman protocol allows Alice and Bob to agree on a shared key for *symmetric* encryption. It does not provide keys to use for public key encryption.

(k) ISPs are obligated to verify the IP source address on any traffic entering their network.

○ **True** [✓ ● **False**]

(l) A useful property of fiber optic cables is that the technology fundamentally eliminates the possibility of eavesdropping.

○ **True** [✓ ● **False**]

(m) It's difficult for an off-path attacker sending IP packets with a spoofed source to view the responses to those packets.

[✓ ● **True**] ○ **False**

(n) In the event where the domain-name-to-IP-address binding changes, the DNS server responsible for the given domain name sends invalidation messages to clients in order to flush their mappings.

○ **True** [✓ ● **False**]

> **Solution:** DNS responses have expiration dates. It is the client's responsibility to flush expired mappings.

(o) Randomizing the DNS query identifier prevents an on-path attacker from spoofing DNS responses.

○ **True** [✓ ● **False**]

(p) If a laptop joining a WiFi network uses both DHCP and DNS, it will first use DHCP before using DNS.

[✓ True] False

(q) When establishing a TCP connection, the client and the server engage in a three-way handshake to determine the shared Initial Sequence Number they will both use for that connection.

True [✓ False]

Solution: In TCP, the client and the server each select their own ISN; they do not share an ISN.

(r) Hosts that use DHCP on a wired networking technology such as Ethernet are protected against possible DHCP spoofing attacks.

True [✓ False]

Solution: DHCP requests are sent using broadcast, regardless of networking technology.

(s) Source port randomization helps defend against an off-path attacker performing the Kaminsky DNS cache poisoning attack.

[✓ True] False

Solution: An off-path attacker must resort to blind DNS response spoofing. By adding source port randomization, the likelihood of a correctly spoofed DNS response is significantly lower.

(t) "Bailiwick" checks in modern DNS resolvers will prevent a malicious name server responsible for `foo.com` from using the Additional fields in its DNS responses to poison cache entries for `bar.com`.

[✓ True] False

(u) If a WiFi network `MyCoolWifi` uses WPA2-Personal, and the administrator of the network chooses the network's password carefully so that it cannot be guessed, that will prevent an attacker who does not know the `MyCoolWifi` password from successfully eavesdropping on messages sent by users who know the password and have joined `MyCoolWifi`.

[✓ **True**] ○ **False**

Problem 2 ***Short questions*** **(60 points)**

(a) (8 points) Let M be a message of size 64 bytes. Let $C = \text{AES-CBC}_K(M)$; C does not include the IV. For which of the following values can a one-bit error in the value the receiver has for it potentially cause EVERY BLOCK to be decrypted incorrectly? **Mark ALL that apply.**

○ C ○ IV [✓ K] ○ **None of these**

Solution: First, observe that when using AES, a 64-byte message will result in 4 plaintext blocks, since AES is a 128-bit (16 byte) block cipher.

A single-bit error in the ciphertext can only affect one of the 4 blocks of ciphertext. A corrupted ciphertext block will always cause its corresponding plaintext block to fail to decrypt properly. With CBC, the ciphertext block is also used to decrypt the *next* plaintext block (via XOR'ing with the AES decryption of the next ciphertext block). However, the effects of the corrupted ciphertext block will not extend further, given that the other ciphertext blocks do not have errors. Thus, at most 2 of 4 blocks will decrypt incorrectly.

If the transmitted IV has a single-bit error, then that will result in an incorrect decryption of the first block of plaintext, since it's computed as the XOR of the IV with the AES decryption of the first cipherblock. However, neither the IV nor the first block of plaintext is used in subsequent decryptions, so in this scenario only 1 of the 4 blocks will decrypt incorrectly.

If the key has a single-bit error, then when decrypting all of the AES operations will fail to produce the correct value, so every block will decrypt incorrectly.

(b) (8 points) Let M be a message of size 64 bytes. Let $C = \text{AES-CTR}_K(M)$; C does not include the nonce. For which of the following values can a one-bit error in the value the receiver has for it potentially cause EVERY BLOCK to be decrypted incorrectly? **Mark ALL that apply.**

○ C [✓ Nonce] [✓ K] ○ **None of these**

Solution: In this scenario, a single corrupted block of ciphertext will only affect the decryption of the corresponding plaintext block. For both encryption and decryption, the ciphertext/plaintext block pairs are isolated (this is why it's easy to parallelize both encryption and decryption for CTR mode). So only 1 of the 4 blocks will decrypt incorrectly.

If the transmitted nonce has an error, however, that will mar the decryption of every plaintext block, since for all of them, the nonce (along with the varying counter) is input to AES encryption to produce the pseudo-random value needed to XOR with the ciphertext to recover the plaintext. Thus, every block will decrypt incorrectly.

As in the first problem, if the key has a single-bit error, then every stage of decryption will fail.

(c) (12 points) Let $M = M_1||M_2$ be a message of size exactly 2 blocks (where M_i are the blocks of M). Assume that Alice shares a secret key k with Bob. Alice encrypts M using AES-ECB, and computes $C = C_1||C_2$. She then computes a tag $t = F_k(C)$, and sends (C, t) to Bob. Which of the following choices for F would allow Bob to detect if an attacker tampers with C? **Mark ALL that apply.**

- ○ $F_k(C) = \text{MAC}_k(C_1) \; || \; \text{MAC}_k(C_2)$
- [✓ ● $F_k(C) = \text{MAC}_k(C_1 \; || \; C_2)$]
- [✓ ● $F_k(C) = \text{MAC}_k(C_2 \; || \; C_1)$]
- ○ $F_k(C) = \text{MAC}_k(\text{MAC}_k(C_1))$
- ○ $F_k(C) = \text{MAC}_k(C_1 \oplus C_2)$
- ○ None of these

Solution: $F_k(C) = \text{MAC}_k(C_1) \; || \; \text{MAC}_k(C_2)$ fails because an attacker can extract the separate values $\text{MAC}_k(C_1)$ and $\text{MAC}_k(C_2)$. The attacker can then construct an alternative message $M' = M_2||M_1$, with a ciphertext of $C' = C_2||C_1$, and provide a corresponding tag of $\text{MAC}_k(C_2) \; || \; \text{MAC}_k(C_1)$.

$F_k(C) = \text{MAC}_k(C_1 \; || \; C_2)$ is sound because the MAC covers the entire ciphertext, with each element distinctly represented.

$F_k(C) = \text{MAC}_k(C_2 \; || \; C_1)$ is also sound because the MAC covers the entire ciphertext, with each element distinctly represented, even though it is computed in a manner that seems a bit quirky.

$F_k(C) = \text{MAC}_k(\text{MAC}_k(C_1))$ fails because it does not protect the M_2 portion of the original message. The attacker can provide the ciphertext $C' = C_1||C_2'$, for some $C_2' \neq C_2$, plus the original tag, and Bob will have no way to detect the alteration. (In this scenario, the attacker may have difficulty knowing what value of C_2' will decrypt to a useful alternative message; but the problem simply asks for whether the attacker can alter the ciphertext without Bob discerning that fact.)

$F_k(C) = \text{MAC}_k(C_1 \oplus C_2)$ has a problem similar to the first scheme. The attacker can construct $M' = M_2||M_1$, providing the ciphertext $C' = C_2||C_1$, and the same tag will validate M' since $C_1 \oplus C_2 = C_2 \oplus C_1$.

(d) (12 points) Alice wants to encrypt messages using the CTR mode of operation, but instead of using AES, she wants to use some other keyed function F. Which of the following properties must F *necessarily* have to allow Alice and Bob to exchange messages securely? **Mark ALL that apply.**

- ○ The size of the input and output strings must be equal
- ● F must be deterministic, i.e., the same input must always map to the same output
- ● Unless one has knowledge of the key, the output of F must appear indistinguishable from a random string of the same length
- ○ F must be invertible, i.e., given the output along with the key, it must be possible to obtain the corresponding input.

Solution:

1. For security we need the output of F to appear to be a random string (per the next item). That property does not require that F produces output strings of the same size as its input. For example, a seeded pseudo-random number generator takes a small input (the seed) and produces a potentially very large pseudo-random output from it.
2. We need F to appear indistinguishable from a random string if one does not know the key; otherwise an attacker can learn something about the original message upon observing the ciphertext.
3. If F is not deterministic, then for CTR mode it will not be decryptable, since CTR decryption uses the *encryption* of F to recover the pseudo-random keystream to XOR with the ciphertext.
4. While we usually think of our block cipher functions as needing to be invertible, for CTR mode this is not necessary, because as noted above CTR mode only uses the *encryption* functionality of F; it does not ever use F's decryption functionality.

To receive any credit for the problem, answers needed to select both item 2 and item 3. Each is fundamental to understanding the viability of using CTR mode, with item 2 providing security and item 3 providing correctness.

(e) (8 points) RANK the following types of attackers according to which is strongest (rank=1). An attacker of type A is stronger than an attacker of type B if A can succeed at any attacks that B can succeed at, plus A has additional attack capabilities that B does not. If two types of attackers have equal strength, then give them the same rank: either both as 1 (with the third, inferior type of attacker ranked as 2), or both as 2 (if the third type of attacker is superior to them, and thus ranked 1). If none of the attackers is strictly stronger than any of the others, give them all the same rank of 1.

1. Passive on-path attacker

   ○ 1 [✓ ● 2] ○ 3

2. Off-path attacker

   ○ 1 ○ 2 [✓ ● 3]

3. MITM attacker

   [✓ ● 1] ○ 2 ○ 3

Solution:

1. MITM attackers can not only observe traffic like a passive attacker can, but can also modify communication-in-flight.
2. Passive on-path attackers can observe all communication, but cannot modify it.
3. Off-path attackers cannot observe communication.

When writing the problem, we meant "on-path" in its usual sense of "cannot intercept, but can observe," plus no further restrictions (so, able to spoof). This is the usual meaning of "on-path attacker." We added the term "passive" since in beta-testing we found a potential for confusion between on-path attackers and MITMs (or *in-path* attackers). In general, "passive" attackers do not only observe. Consider Eve-the-eavesdropper from our crypto discussions: she can't prevent messages from being delivered, but she can fake up her own messages, or replay earlier messages.

However, in grading we realized that the unfamiliar combination "passive on-path attacker" might be interpreted as an attacker who only watches. Such an attacker is still fairly strong. For example, they can steal any HTTP cookies, and if they collude with another party they can provide sequence numbers or identifiers for spoofing. On the other hand, an off-path attacker is weak: they can't see anything, and since they can't, they will have great difficulty launching a successful spoofing attack. We decided that students might view these two types of attackers as not fully comparable: each of them has a capability the other lacks. So we allowed for solutions that ranked them the same. However, we don't view it as defensible that an off-path attacker is stronger than a purely passive attacker, because the attacker's opportunities are severely limited.

(f) (12 points) Tyrion is trying to download the latest version of the Bearship web browser from its official website, caltopia.edu. However, the download from caltopia.edu is taking too long, so Tyrion logs onto FastTorrent, a file sharing service where anyone can upload files for others to download. He searches for Bearship on FastTorrent and downloads the first file that's returned in the search results. Which of the following could caltopia.edu publish on their website to help users like Tyrion efficiently verify that the file they downloaded from a file-sharing service really is the Bearship browser, and not some malicious program? Efficiency means that if the browser is n bytes, the user does not have to download more than $n + O(1)$ bytes. Assume that the caltopia.edu website is secure. **Mark ALL that apply.**

- ○ Publish a screenshot of Bearship's user interface
- [✓ Publish a secure hash (e.g., SHA-256) of the Bearship binary]
- [✓ Publish a digital signature of the Bearship binary]
- [✓ Publish a MAC, along with an associated key, of the Bearship binary]
- ○ Publish a version of the Bearship binary encrypted using caltopia.edu's public key
- ○ Publish a version of the Bearship binary encrypted using AES, along with an associated key.

Solution: An attacker's malicious browser can provide whatever user interface the attacker desires, so it does not help Tyrion to know what to expect the UI to look like.

A digital signature, a secure hash, or a MAC along with the associated key all provide ways for Tyrion to verify he has downloaded the correct binary, given that the caltopia.edu website is secure (and thus the hash has not been tampered with, nor, for the MAC case, has the associated key).

Encrypting an n byte input produces an $O(n)$-byte output, whether done using symmetric key cryptography or public-key cryptography. Thus, for the user to confirm that they have a true copy of the browser, these schemes will require the user to download $n + O(n)$ bytes, violating the efficiency requirement.

Problem 3 ***Just How Brutal Does The Force Need To Be?*** **(40 points)**

Consider a secure hash function H that produces a 60-bit hash.

(a) Suppose that $H(1)$ happens to hash to 0 (i.e., 60 zero bits). If you don't know anything further about H other than that fact and that it's a secure hash function, what is the probability that $H(2)$ also hashes to 0?

Solution: Given that H is indeed a secure hash function, then we expect all outputs from it to be equally probable. Thus, the probability that any particular input hashes to a specific output is 2^{-60}.

(b) What is the probability that H has at least one collision? Explain in 1 sentence.

Solution: The probability is 1 (i.e., it is certain).

Secure hash functions take arbitrarily large inputs and reduce them to a fixed-sized output. Thus, the existence of collisions is guaranteed (in fact, an infinite number of collisions). Security comes from the fact that the output space is still so large that these collisions are hard to find.

(c) Suppose that commodity hardware can compute a single computation of H in 10 nanoseconds ($= 10^{-8}$ sec). Within an order of magnitude, how many years will it take for an attacker using a single system to find an x such that $H(x) = y$ for a specific y? You can approximate one year as $3 \cdot 10^7$ sec.

Solution: The attacker can try 10^8 values of x every second. There are about $3 \cdot 10^7$ seconds in a year, so the attacker can try about $3 \cdot 10^{15}$ values of x per year.

Since $2^{10} \approx 10^3$, to fully explore a 60-bit space requires $\approx 10^{18}$ guesses. (It could be a bit more, since some guesses may produce collisions for values other than y.) If the attacker can try $3 \cdot 10^{15}$ values per year, then to try 10^{18} values will take about $3 \cdot 10^2$ years. So **about three centuries**.

Full credit required providing a concrete value for the number of years, rather than simply an expression to evaluate it. The point of the problem was to be able to formulate a concrete estimate analogous to the lecture presentation of how to determine how long it would take to brute-force a 128-bit AES key. Writing down an expression for it is simple; being able to evaluate that in one's head (or with a bit of scratch paper, as available for the exam) to get an actual final number is a valuable skill. The importance of that distinction is why we spent class time walking through how to do it.

On the other hand, because such estimates are rough, we did not require an on-the-dot answer (indeed, above we only give a rough answer), and we did not concern ourselves with the distinction between 50% probability of success vs. near-certain probability of success. All of that was encompassed by asking for "within an order of magnitude," and any answer in the range of 100 to 1,000 years received full credit.

(d) Suppose now that a sustained form of Moore's Law means that after every year, H can be computed twice as quickly as for the previous year. (For this problem, assume that this acceleration happens discretely year-by-year, rather than being spread across a given year, as is actually more realistic.) Given this change, now about how many years will it take the attacker to find such an x?

Solution: The previous problem developed that it will take about 300 years using a fixed rate of computations-per-second. Call that fixed rate Y_0, i.e., the rate of computations in Year 0. In a subsequent year i, we can compute $2^i \cdot Y_0$ computations. Thus, we need to determine for what value of j does $\sum_{i=0}^{j} 2^i \approx 300$? Since in general we have $\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$, we get either $j = 7$ ($2^8 - 1 = 255$) or $j = 8$ ($2^9 - 1 = 511$). We started our numbering at 0, so this means **eight or nine years**. Suddenly, the attack starts to become feasible!

Full credit required an answer in the range of 7.5 to 11 years *or* that was expressed as essentially $\log_2$ of the answer for the previous part.

Problem 4 ***RST injection*** **(60 points)**

This problem concerns RST injection attacks.

(a) In ONE SENTENCE, explain the purpose of a RST injection attack: that is, what goal does an attacker try to accomplish by launching such an attack?

> **Solution:** In a RST injection attack, the attacker aims to disrupt an existing TCP connection, causing it to prematurely terminate.

(b) What information is needed for an attacker to carry out a successful RST injection attack?

> **Solution:** The attacker needs to know the target connection's 4-tuple (source and destination addresses and ports), and the current sequence number that the target expects to see from the spoofed source.
>
> A common error was to indicate that the attacker needs to know *both* sequence numbers (used in each direction).

(c) Explain under what circumstances an off-path attacker can conduct a successful RST injection attack. If it is impossible for them to do so, explain why they cannot.

> **Solution:** An off-path attacker can conduct a successful attack if they can somehow gather or infer the information specified in the previous question.
>
> Alternatively, a solution that states that randomized ISNs make the attack infeasible is also acceptable.

(d) Suppose an attacker launches a RST injection attack against Alice. Are there situations in which Alice can detect that the attack has occurred? If **YES**, explain how she might do so. If **NO**, explain why it's not possible for her to do so.

> **Solution:** The answer we had intended was: **YES**. When the attacker injects their spoofed traffic, they cannot prevent any traffic sent by Alice's legitimate peer (Bob) that's already in flight from also arriving. Thus, Alice can observe both the receipt of a RST purportedly from Bob (the attack), as well as additional traffic arriving from Bob. Such a pattern does not make sense for the benign situation that Bob's own system sent the RST.
>
> However, in grading the problem we realized that some students interpreted

it as whether Alice could detect the attack absent any additional activity; or whether there were other scenarios (such as Alice and Bob sharing information) that would enable Alice to detect the attack. Thus, we allowed full credit for either a **YES** or a **NO** answer if it indicated some sort of correct scenario or interpretation in support.

Simply answering **YES** or **NO** without any sort of viable explanation did not receive any credit, as such answers do not convey an understanding of how the attack manifests. Likewise, simply stating that Alice notices that the connection has terminated did not receive any credit, since by itself that does not enable Alice to distinguish between an attack and a benign failure.

(e) If Alice connects to a WPA2-Enterprise WiFi network, can an attacker successfully launch a RST injection attack against her? Explain why or why not.

Solution: Yes, Alice is still vulnerable. WPA2-Enterprise will prevent a *local* attacker from observing Alice's traffic. However, there can still be an eavesdropper elsewhere along the network path between Alice and Bob who can observe all of the information necessary to conduct a successful RST-injection attack against Alice.

Problem 5 ***How do you keep a secret?*** **(60 points)**

Frodo is trying to send a series of secret messages to Galadriel. For this question, we will refer to Galadriel as G and Frodo as F.

(a) (30 points) Assume that G and F share a secret key k. Let THE-ALL-SEEING-EYE be a passive attacker who sees all messages sent by F to G.

For each encryption scheme below, select whether the encryption scheme is **Secure** or **Insecure**, and justify your answer in one or two sentences. A scheme is secure if, after observing all of the ciphertexts sent by F, THE-ALL-SEEING-EYE learns nothing about the contents of any plaintext message beyond the message's length.

i. Encryption scheme:

1. F sets $IV = H(\text{the current timestamp})$, where H is a cryptographically strong hash function, and the timestamp is produced with enough precision such that two separate encryptions will not use the same timestamp value.
2. To send a new message m, F encrypts m using AES-CTR mode with IV and sends the resulting ciphertext.

Is this protocol secure against THE-ALL-SEEING-EYE?

[✓ **Secure**] **Insecure**

Solution: The IV will never repeat, and the use of H ensures that successive IV's do not have any apparent relationship with one another. THE-ALL-SEEING-EYE has no control over the IV, and we're using CTR mode, so confidentiality is assured if we assume (as we usually do) that AES is a secure block cipher.

ii. Encryption scheme:

1. F computes $p = H(k)$, where H is a cryptographically strong hash function
2. F stores a long-term counter x, which starts at 0 (assume no risk of overflow)
3. Let n be the number of bits in k, and assume that for this question, all messages are exactly n bits long and p is also n bits long. To send a new message m, F increments x by one and computes $s = \text{PRNG}(x)[1 : n]$, where s is the first n bits of a secure PRNG with a seed of x.
4. F then computes $k_x = p \oplus s$ and sends the ciphertext: $c = m \oplus k_x$.

Is this protocol secure against THE-ALL-SEEING-EYE?

Secure [✓ **Insecure**]

Solution: PRNGs are deterministically computable, so they only provide pseudorandomness if the seed is unknown. Here, we're just seeding the PRNG with an incrementing counter, so THE-ALL-SEEING-EYE can easily compute that value and XOR it away from the ciphertext. That means that the ciphertext is just equal to $c = m \oplus p$, where p is a deterministic hash of the secret key; i.e., we have a many-time-pad usage.

(b) (30 points) Now assume that instead of sharing a secret key, G and F have shared their public keys, K_G and K_F. They use these keys to exchange a new symmetric key using the following protocol:

1. $F \longrightarrow G \quad : \quad \{F, N_1\}_{K_G}$
2. $G \longrightarrow F \quad : \quad \{N_1, N_2\}_{K_F}$
3. $F \longrightarrow G \quad : \quad \{N_2\}_{K_G}$

In step 1, F generates a random nonce N_1 and sends it to G along with his identity F (to identify himself to G), after encrypting these with her public key. G responds in step 2 by generating a random nonce N_2, and sends it to F along with N_1 (to prove that she possesses the private key K_G^{-1}) after encrypting these with F's public key. F finally returns N_2 encrypted with G's public key, to prove that he possesses the private key K_F^{-1}.

F and G now compute a new symmetric key as $K = N_1 \oplus N_2$, and encrypt all subsequent messages using K.

Frodo meets Mallory, and unfortunately doesn't know that Mallory is evil. Frodo decides to start a conversation with Mallory after obtaining her public key K_M. Mallory realizes that if Frodo initiates the same key exchange protocol with her, then she can do so with Galadriel while pretending to be Frodo. (Assume Mallory has obtained Galadriel's public key.)

i. **Fill in the blanks** such that at the end of these steps, Galadriel thinks that she's exchanged a symmetric key K' with Frodo. Write down the value of K'.

(i) $F \longrightarrow M \quad : \quad \{F, N_1\}_{K_M}$ (ii) $M \longrightarrow G \quad : \quad$ __________

(iii) $G \longrightarrow M \quad : \quad$ __________ (iv) $M \longrightarrow F \quad : \quad$ __________

(v) $F \longrightarrow M \quad : \quad$ __________ (vi) $M \longrightarrow G \quad : \quad$ __________

$K' =$ _______________

Solution:

(i) $F \longrightarrow M \quad : \quad \{F, N_1\}_{K_M}$ (ii) $M \longrightarrow G \quad : \quad \{F, N_1\}_{K_G}$

(iii) $G \longrightarrow M \;:\; \{N_1, N_2\}_{K_F}$ (iv) $M \longrightarrow F \;:\; \{N_1, N_2\}_{K_F}$

(v) $F \longrightarrow M \;:\; \{N_2\}_{K_M}$ (vi) $M \longrightarrow G \;:\; \{N_2\}_{K_G}$

$K' = \; N_1 \oplus N_2$

Each step somewhat inevitably follows from the previous, given Mallory's goal plus the constraints of what information Mallory has available (such as messages sent to Mallory by F or G) and what messages Mallory can construct (such as repackaging information Mallory has received that was encrypted using K_M, for which she can recover the original message).

ii. Which of the following changes to **step 1** ("$F \longrightarrow G : \{F, N_1\}_{K_G}$") would prevent Mallory from impersonating Frodo? **Mark ALL that apply.**
The notation $[M]_{K^{-1}}$ represents a signature over M using a private key K^{-1}.

● $F \longrightarrow G \;:\; (C, [C]_{K_F^{-1}})$,
where $C = \{F, N_1\}_{K_G}$
(i.e., include a signature computed over the ciphertext)

○ $F \longrightarrow G \;:\; (\{F, N_1\}_{K_G}, [F, N_1]_{K_F^{-1}})$
(i.e., include a signature computed over the plaintext)

● $F \longrightarrow G \;:\; (C, [C]_{K_F^{-1}})$,
where $C = (F, \{N_1\}_{K_G})$
(i.e., only encrypt the nonce in the ciphertext; include a signature computed over this ciphertext)

○ None of these

Solution: The upper left and upper right options would prevent Mallory from impersonating Frodo. To impersonate Frodo, Mallory would need to re-encrypt the ciphertexts in these two options with Galadriel's public key. However, she would then be unable to forge Frodo's signature on the modified ciphertexts. The lower left option doesn't work because it doesn't require Mallory to compute a new signature after she re-encrypts the ciphertext with Galadriel's public key.