Nick Weaver
Fall 2018

CS 161
Computer Security

Homework 2

Due: Friday, 28 September 2018, at 11:59pm

Instructions. This homework is due **Friday, 28 September 2018, at 11:59pm**. No late homeworks will be accepted unless you have prior accomodations from us. This assignment must be done on your own.

Create an EECS instructional class account if you have not already. To do so, visit https://inst.eecs.berkeley.edu/webacct/, click "Login using your Berkeley CalNet ID," then find the cs161 row and click "Get a new account." Be sure to take note of the account login and password, and log in to your instructional account.

Make sure you have a Gradescope account and are joined in this course. The homework *must* be submitted electronically via Gradescope (not by any other method). Your answer for each question, when submitted on Gradescope, should be a single file with each question's answer on a separate page.

Problem 1 ***True-or-False Questions*** **(45 points)**

Answer each question. You don't need to justify or explain your answer.

(a) TRUE or FALSE: Diffie–Hellman protects against eavesdroppers but is vulnerable to man-in-the-middle attacks.

(b) TRUE or FALSE: Suppose there is a transmission error in a block B of ciphertext using CBC mode. This error propagates to every block in decryption, which means that the block B and every block after B cannot be decrypted correctly.

(c) TRUE or FALSE: The IV for CBC mode must be kept secret.

(d) TRUE or FALSE: The random number r in El Gamal must be kept secret.

(e) TRUE or FALSE: The best way to be confident in the cryptography that you use is to write your own implementation.

(f) TRUE or FALSE: Alice and Bob share a symmetric key k. Alice sends Bob a message encrypted with k stating, "I owe you \$100", using AES-CBC encryption. Assuming AES is secure, we can be confident that an active attacker cannot tamper with this message; its integrity is protected.

(g) TRUE or FALSE: If the daily lottery numbers are truly random, then they can be used as the entropy for a one-time-pad since a one-time-pad needs to be random.

(h) TRUE or FALSE: It is okay if multiple people perform El Gamal encryption with the same modulus p.

(i) TRUE or FALSE: Alice and Bob share a secret symmetric key k which they use for calculating MACs. Alice sends the message $M =$"I (Alice) owe you (Bob) \$100" to Bob along with its message authentication code $\text{MAC}_k(M)$. Bob can present $(M, \text{MAC}_k(M))$ to a judge as proof that Alice owes him \$100 since a MAC provides integrity.

Problem 2 ***Finding Common Patients*** **(40 points)**

Caltopia has two hospitals: Bear Hospital and Tree Hospital, each of which has a database of patient medical records. These records contain highly sensitive patient information that should be kept confidential. For both hospitals, each medical record is a tuple (p_i, m_i), where p_i and m_i are strings that correspond to the patient's full name and medical record respectively; assume that every person in Caltopia has a unique full name. Thus, we can think of Bear Hospital's patient database as a list of tuples $(x_1, m_1), (x_2, m_2), ..., (x_n, m_n)$, where m_i is the medical information that Bear Hospital has for patient x_i. Similarly, we can think of Tree Hospital's database as a list $(y_1, m'_1), (y_2, m'_2), ..., (y_m, m'_m)$, where m'_i is a string that encodes the medical information that Tree Hospital has for the patient named y_i. Note that for a given patient, Tree Hospital and Bear Hospital might have different medical information.

The two hospitals want to collaborate on a way to identify which Caltopia citizens are patients at both hospitals. However, due to privacy laws, the two hospitals cannot share any plaintext information about patients (including their names) unless both hospitals know *a priori* that a patient has used both hospitals.

Thus, the two hospitals decide to build a system that will allow them to identify common patients of both hospitals. They enlist the help of Lady Olenna, who provides them with a trusted, third-party server S, which they will use to discover the names of patients who use both hospitals. Specifically, Bear Hospital will take some information from its patient database and transform it into a list $(x_1^*), (x_2^*), ..., (x_n^*)$ (where (x_i^*) is somehow derived from x_i (the patient's full name) and upload it to S. Similarly, Tree Hospital will take information from its patient database, transform it into a list $(y_1^*), (y_2^*), ..., (y_m^*)$, and upload this transformed list to S. Finally, S will compute a set of tuples $P = (i, j) : x_i = y_j$ of all pairs (i, j) such that $x_i^* = y_j^*$ and send P to both Bear Hospital and Tree Hospital. The two hospitals can then take their respective indices from the tuples in P to identify patients who use both hospitals.

We want to ensure three requirements with the above scheme: (1) if $x_i = y_j$, then $(i, j) \in P$, (2) if $x_i \neq y_j$, then it is very unlikely that $(i, j) \in P$, (3) even if Eve (an attacker) compromises S, she cannot learn the name of any patient at either hospital or the medical information for any patient. For this question, assume that Eve is a passive attacker who cannot conduct Chosen Plaintext Attacks; however, she does know the names of everyone in Caltopia, and there are citizens whose full names are a unique length.

Fill in your solutions below. Your solution can use the cryptographic hash SHA-256 and/or AES with one of the three block cipher encryption modes discussed in class; keep in mind that Eve can also compute SHA-256 hashes and use AES with any block cipher mode. You can assume that Bear Hospital and Tree Hospital share a key k that is not known to anyone else. You *cannot* use public-key cryptography or modular arithmetic.

(a) In the collaboration scheme described above, how should Bear Hospital compute x_i^* (as a function of x_i)? How should Tree Hospital compute y_i^* (as a function of y_i)? Specifically, your solution should define a function F that Bear Hospital will use to

transform x_i into x_i^*, and if relevant, a function G that Tree Hospital will use to transform y_i into y_i^*.

(b) Explain why requirement (1) is met by your solution, i.e., explain why it is guaranteed that if $x_i = y_j$, then $x_i^* = y_j^*$ will hold. Explain your answer in one or two sentences.

(c) Explain why requirement (2) is met by your solution, i.e., if $x_i \neq y_j$, explain why it is unlikely that $x_i^* = y_j^*$. Explain your answer in one or two sentences.

(d) Explain why requirement (3) is met by your solution, i.e., if S is compromised by Eve, then the information known to S does not let Eve learn any patient information (neither the names of patients at a particular hospital nor the medical history for any patient). Explain your answer in one or two sentences.

Problem 3 ***A Bad CTR Implementation*** **(35 points)**

Recall that given a plaintext M separated and padded into sequence $(m_1, \ldots, m_N)$, symmetric key K, and a random nonce, the ciphertext generated via AES-CTR is defined as $C = c_1 \| \cdots \| c_N$, where $c_i = \text{Enc}_K(\text{nonce} \| i) \oplus m_i$. A fundamental requirement of AES-CTR is that the nonce MUST NOT be reused, or else the encryption becomes breakable.

Below is a list of 10 ASCII messages encrypted using $c_i = \text{Enc}_K(\text{nonce}) \oplus m_i$. (no counter is needed for these small messages).

All ciphertexts are hex-encoded strings and all the original messages contain only the characters `[A-Za-z]`, spaces, dashes, periods, and exclamation marks. The output is a hex-encoded string and **the nonce is the same for all these messages**.

(a) Decrypt the target message.

Ciphertext 0: 167c7f4fff941dea1b0b97f49e6d51fea7878c281f8d981e4fb358051eefa c10c140327c77ad640bed157fa1c93717485d777d916be782732095fee4c21ca0ea56f248 a5b8a0398950f38117877c1ac8b17f758af1e0f08e26b5e86d28137b

Ciphertext 1: 177d3349e5cb49a70e4292f0d37c1eb7bdc9b71e26aedd5d1bb35f0408 efba55c04c30747cf93e4497507cb09b251d514a6b7d946ba18f6e2299b0fa8d1defa256f7 0dc9b0ee6cd107f59c538b605cdbbc3c21d9c2cfdc8044f2a8367e1d3a

Ciphertext 2: 11777f48e5950cea1804dce2cb7a56feb8889f3808c79b1249fe4b060da aed51df0f0b717cf91842dd156af5bd21586a406036c32eee95210693b0c6830ba7ea63e9 0dc986a6239d15baa0529d6756daf073678adec7d68e26bde47725413a

Ciphertext 3: 7f387f05aab001ab03428bf49e694cb1ad85992400c7dd5d52e01e3e04 aaed62c548376d398d250bed1872b9c40c1d5441757dc359e48b6d7fbef5ea8c0fefac58f3 48a8bda26dd15dbaa65f8b3279c1be6d74cfc5da85c10cfcc46a355270

Ciphertext 4: 1b777f5ce59249ab1b0edce5d67050b5ee9d902c19c7891552e01e0904 aea35ec9437f7178aa6a49df1870f58b3c1d5c4c7a388731a1a9603adcfff69048a3af5be00 68df1bb229806ba914597624ec1f07b74d396c7d68e1eb3f6383f5534

Ciphertext 5: 16767f51e28e1aea1a0d98f49e394ab6abc991230493941c57b3480f0fb ba2428c66093970aa6a59df0d7bb49d2b1c515632388d6df39e712699f4a39607efa555f5 0980bfee2dd103ff861781741ad8b17074cfc58effc74abdf538365c78

Ciphertext 6: 16757f56e5951bb3572f9de3d77612feac9c8c6d2ea8b32974c0714a3c9 d847eef6a0c4a39b0390bd3133eb4c92a115b49772f8660f5c7673d8ef5f09649efe717d21f 80b7ba039f23ff91429c7b4ed7f04860d3969c959f5ff1b62b7d012d

Ciphertext 7: 1e6c7f4ae48249ba180b92e59e501ea9af9ad82908859c0952fd594a1ba 7a844c44a2d3976ab6a45d5093ea1866e19594b32298b6ba1b24210b1d6c8ae3cefa958f 11199b0bd389050f59c17867748cbf07e74de96e785c60baae338245c7b

Ciphertext 8: 326d3c4daa940ca611428ef4cd695bbdbac7d8000c8c94135cb34c0b02 aba25d8c43306b7cb46a42ca0e6bb8c9270b1d4e7129966fed8b78729db0ef8d1cefa256f 30c8ca3ee389911f4d25880771ad9bf696dce96dacdc704b7a66c325b34

Target Message: 18773041aa8b1ca91c4293ff9e6d56bbee84912919828f101ab36d1e
19abb410c44e2d7d39b8244f9a1971f59e2b14510332248262edc7602099b0e4900daebe
17f21c9cb5ab228503b7df408b324dc7a37421d3d9db85da02b9a67a354060

To help you out, here is a Python 3 snippet that will take bytestrings and output the xor'd bytestring.

```
def bstrxor(a, b):
    return bytes(x ^ y for (x, y) in zip(a, b))

# strxor takes two hex strings and returns the xor'd output as a bytestring
def strxor(a, b):
    return bstrxor(bytes.fromhex(a), bytes.fromhex(b))
```

Hint: What happens when you xor a letter with a space, or a letter with another letter?

(b) Submit the code you used under "Homework 2 Q3b code" in Gradescope.

(c) Can you retrieve K? If so, what is K? If not, why not?

Problem 4 ***Why do RSA signatures need a hash?*** **(40 points)**

This question explores the design of standard RSA signatures in more depth. To generate RSA signatures, Alice first creates a standard RSA key pair: (n, e) is the RSA public key and d is the RSA private key, where n is the RSA modulus. For standard RSA signatures, we typically set e to a small prime value such as 3; for this problem, let $e = 3$.

To generate a standard RSA signature S on a message M, Alice computes $S = H(M)^d \mod n$. If Bob wants to verify whether S is a valid signature on message M, he simply checks whether $S^3 = H(M) \mod n$ holds. Analogous to RSA encryption, d is a private key known only to Alice and $(n, 3)$ is a publicly known verification key that anyone can use to check if a message was signed using Alice's private signing key.

For this question we'll now explore why RSA signatures use a hash function to compute the signatures. Suppose RSA signatures skipped using a hash function and just used M directly, so the signature S on a message M is $S = M^d \mod n$. In other words, if Alice wants to send a signed message to Bob, she will send (M, S) to Bob, where $S = M^d \mod n$ is computed using her private signing key d.

(a) With this simplified RSA scheme, how can Bob verify whether S is a valid signature on message M? In other words, what equation should he check, to confirm whether M, S was validly signed by Alice? You don't need to justify your answer; just show the equation.

(b) Mallory learns that Alice and Bob are using the simplified (hash-less) signature scheme described above and decides to trick Bob. Mallory wants to send some (M, S) to Bob that Bob will think is from Alice, even though Mallory doesn't know the private key. Explain how Mallory can find M, S such that S will be a valid signature on M.

You should assume that Mallory knows Alice's public key n, but not Alice's private key d. She can choose both M and S freely. The message M does not have to be chosen in advance and can be gibberish.

Hint: If Mallory chooses M and then tries to find a corresponding S, she'll be at a dead-end, because finding S requires inverting a one-way function (cubing modulo n), and we know that is hard without knowledge of the trapdoor (the private key d). So instead, she should ...

(c) Sameer is holding an auction. Alice and Bob will submit signed bids to the auctioneer Sameer, signed using this simplified RSA signature scheme. The message M is an integer that is their bid (in dollars), and they will send just their signature on M, signed using this simplified RSA scheme. Sameer will accept whichever bid is highest and expect that person to pay up however much they bid.

Mallory wants to mess with Bob (her rival). So, when Bob forms his bid M and sends Sameer the signed bid $S = M^d \mod n$, Mallory intercepts the message from Bob containing S. Mallory would like to tamper with S to form a new signature S' that corresponds to a bid for $64\times$ as much as Bob's original bid, to force Bob to

win the auction and pay through the nose for it. More precisely, she'd like to find a value S' such that S' is a valid signature on $64M$, so she can replace S with S' and forward the result onto Sameer. Help Mallory out: show how she can compute such an S'.

(Assume that M is small enough that $64M < n$, so $64M$ does not wrap around modulo n.)

(d) Are your attacks in parts (b) and (c) possible against the real RSA signature scheme (the one that includes the cryptographic hash function)? Why or why not?

Problem 5 ***New Block Cipher Mode*** **(35 points)**

Nick decides to invent a new block cipher mode, called NBC. It is defined as follows:

$$C_i = E_k(C_{i-1}) \oplus P_i$$
$$C_0 = \text{IV}$$

Here $(P_1, \ldots, P_n)$ is the plaintext message, E_k is block cipher encryption with key k.

(a) Given $(C_0, C_1, \ldots, C_n)$ and the key k, explain how to recover the original message $(P_1, \ldots, P_n)$.

(b) Is NBC encryption parallelizable? How about decryption? Provide a short justification for each.

(c) As we saw in discussion, CBC mode is vulnerable to a chosen plaintext attack when the IV which will be used to encrypt the message is known in advance. Is NBC vulnerable to the same issue?

(d) Say that Alice means to send the message $(P_1, \ldots, P_n)$ to Bob using NBC mode. By accident, Alice typos and encrypts $(P_1 \oplus 1, \ldots, P_n)$ instead (i.e., she accidentally flips the last bit of the first block).

TRUE or FALSE: after Bob decrypts the resulting ciphertext, every block after the first is incorrect. Explain your answer.

(e) Alice encrypts the message $(P_1, \ldots, P_5)$. Unfortunately, the block C_3 of the ciphertext is lost in transmission, so that Bob recieves $(C_0, C_1, C_2, C_4, C_5)$. Assuming that Bob knows that he is missing C_3, which blocks of the original plaintext can Bob recover?

Problem 6 ***Feedback*** **(0 points)**

Optionally, feel free to include feedback. Whats the single thing we could do to make the class better? Or, what did you find most difficult or confusing from lectures or the rest of class, and what would you like to see explained better? If you have feedback, submit your comments as your answer to Q6.