Popa & Wagner
Spring 2016

CS 161 Computer Security

Homework 3

Due: Friday, March 11, at 11:59pm

Instructions. This homework is due Friday, March 11, at 11:59pm. It *must* be submitted electronically via Gradescope (and not in person, in the drop box, by email, or any other method). This assignment must be done on your own.

Please put your answer to each problem on its own page, in the order that the problems appear. For instance, if your answer to every problem fits on a single page, your solution will be organized as follows:

page 1: your solution to problem 1
page 2: your solution to problem 2
page 3: your solution to problem 3
page 4: your solution to problem 4
page 5: your solution to problem 5
page 6: your optional feedback ("problem 6")

If your solution to problems 3 and 4 both take up two pages, your solution would be organized as follows:

page 1: your solution to problem 1
page 2: your solution to problem 2
page 3: first page of your solution to problem 3
page 4: second page of your solution to problem 3
page 5: first page of your solution to problem 4
page 6: second page of your solution to problem 4
page 7: your solution to problem 5
page 8: your optional feedback ("problem 6")

Scan your solution to a PDF—or, write it electronically and save it as a PDF. Then, upload it to Gradescope.

Problem 1 ***Warmup: Short-Answer Questions*** **(10 points)**

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

(a) Name one encryption scheme mentioned in class that protects against eavesdroppers but is vulnerable to man-in-the-middle attacks.

(b) True or False: If a hash function has collision resistance, the hash function also has second preimage resistance.

(c) 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.

(d) TRUE or FALSE: If the wind speeds over the golden gate bridge are truly random, then a good one-time pad would be the concatenation of wind speeds taken over a given period of time, since a one time pad needs to be random.

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

(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.

Problem 2 *Attacking encryption* (27 points)

FlapChat is a happening web forum for folks who love to fly. FlapChat.com uses passwords and session cookies to authenticate their users. In particular, once a user logs in (by entering their username and password), the FlapChat.com server sets a cookie that contains information about the user. The cookie will be sent with all subsequent requests to FlapChat.com, and the FlapChat.com server will use that cookie to authenticate the user (so the user doesn't have to enter in their password again). Users are only given this cookie once they enter in their password correctly, so if the server sees a request accompanied by such a cookie, then it knows that it comes from an authenticated user.

You've learned that the cookie contains the encryption of a credential string. The credential string contains the user's username, the user's display name, and the account-type (the letter O or A, which indicates whether this is an ordinary user or an administrator); the fields are separated by exclamation marks. For instance, the plaintext credential string for Alice might be

```
alice!Alice X. Jones!O
```

Once she has logged in, her cookie contains an encrypted version of this plaintext credential string.

You want to break into the web site and gain administrator-level access. This question has all the information needed to spot the vulnerabilities, so your attack should use only the information given here; no social engineering of site administrators, no eavesdropping on their traffic, no getting malware on admin's machines, no exploiting XSS bugs in the site, nothing like that.

(a) You haven't been able to find any information about how the encryption works, but looking at the cookie values, you suspect they must be adding enough random bytes of padding to the end of the plaintext credential string to bring it up to a multiple of 16 bytes and then encrypting it somehow. You've been poking around the web site and you immediately notice that FlapChat is a public forum: anyone can create an account. In particular, you can create an ordinary account with any username and display name you want (assuming the username hasn't already been taken), but you can't create an administrator account. Describe an easy attack that will get you administrator-level access.

(Since you don't know how they are doing the encryption, your attack had better work no matter what encryption algorithm they use.)

(b) You show your buddy how to do the attack from part (a). Unfortunately, your buddy has a big mouth and starts bragging about his administrator-level access, and before you know it, the FlapChat developers spot the security hole and fix the problem. In particular, they change their system to require admins to manually approve each new account that is created, and they are careful not to approve any new account that would let you exploit the vulnerability mentioned in part (a). Curses! You've been foiled.

You go sulk for a week, but then by good luck you have a chance to chat up the developer of FlapChat at a party. You get him bragging about his elegant design and pretty soon you've learned how the plaintext credential string is encrypted: it is padded by appending random bytes until it is a multiple of 16 bytes long, then it is encrypted with AES-CTR mode, using a 128-bit AES key K stored on the FlapChat server. Thus, the cookie contains an AES-CTR ciphertext, i.e., $IV \mid\mid C_1 \mid\mid C_2 \mid\mid \cdots \mid\mid C_k$, where IV is a random initialization vector chosen randomly each time a user logs in, and where $C_i = Z_i \oplus M_i$ where $Z_i = \text{AES}_K(IV + i)$ and $M_1 \mid\mid M_2 \mid\mid \cdots \mid\mid M_k$ is the padded plaintext credential string. When the server receives a cookie, it decrypts it using AES-CTR, pulls out the username, display name, and account-type by using the exclamation marks as separators (ignoring all padding bytes), checks that the username is in the database of registered users and that the account-type is either `O` or `A`, and accepts the user as authenticated if all of these checks pass. It then gives access based upon the account-type (for instance, if the HTTP request asks to perform some administrator-only action, the action is performed only if there is a valid cookie and its account-type is `A`).

Describe how to attack the crypto and get administrator-level access to the site.

(Your attack has to be something you can do on your own by attacking the crypto.)

(c) You get drunk one evening and use your administrator-level access to post a really funny prank message on the site. The site administrators are not amused, investigate a little further, and realize that maybe using CTR mode was not such a good idea. So, they beef up their security with a one-line fix: instead of using AES-CTR mode, they use AES-CBC mode. For good measure, they change the AES key so that people can't use old cookies; everyone will have to log in again and get a new cookie encrypted with the new scheme.

Describe how you can still break their crypto and get administrator-level access to the site.

(d) OK, so maybe AES-CBC mode wasn't such a great idea, either. Describe a better cryptographic scheme for computing what to put in the cookie, as a function of the plaintext credential. If the plaintext credential is given by $M_1 \mid\mid M_2 \mid\mid \cdots \mid\mid M_k$, what value should they store in the cookie? Be specific.

Problem 3 ***Outwitting Eve and Mallory*** **(15 points)**

Alice and Bob would like to use what they learned in CS 161 to communicate by email without letting anyone else read their letters.

(a) Alice and Bob decide to use public-key cryptography. Alice emails Bob her public key, and vice versa. Now they can communicate by encrypting messages using the respective public keys. Is Alice and Bob's communication secure against Eve, the passive eavesdropper? How about Mallory, the MITM attacker?

(b) Alice and Bob decide instead to use the Diffie-Hellman Key Exchange. Bob emails Alice a value g^x and Alice emails Bob a value g^y. They then use these to establish a secret key g^{xy} that they use to encrypt all of their subsequent emails. Is Alice and Bob's communication secure against Eve? How about Mallory?

(c) Alice and Bob decide not to use email for sending each other public keys. Instead they meet at Rude Awakening, the local tea house, and tell each other their public keys. Subsequently, they communicate by encrypting their message (with the other's encryption public key) and then signing it (using their signing private key).

Unfortunately, Prof. Evil overhears their conversation at Rude Awakening, and, being evil, will communicate everything he heard to all evil-doers in the world, including Eve and Mallory.

Will Alice and Bob's subsequent communication be secure against Eve? How about Mallory?

Problem 4 ***The security of Diffie-Hellman*** **(20 points)**

In this question, you will examine the security of a few ways of using Diffie-Hellman. Let p be a large prime chosen as the first prime larger than 2^{2048} and g be a generator modulo p. As usual in Diffie-Hellman, p, g are publicly known. As always, Alice will pick a secret random number a from the range $0 \ldots p-2$, and Bob will pick his own secret b from the same range. They will derive a shared secret s as $s = H(g^{ab} \bmod p)$, where H is a cryptographic hash function.

Notation: w is a password known to Alice and Bob. E represents a secure symmetric-key encryption algorithm and F is a secure message authentication code (MAC). K_A is Alice's public key, K_A^{-1} is her private key, and Bob knows K_A; similarly for K_B, K_B^{-1}.

Consider the following four protocols, based upon Diffie-Hellman:

- P1: Alice sends $g^a \bmod p$ to Bob. Bob sends $g^b \bmod p$ to Alice.
- P2: Alice sends $E_w(g^a \bmod p)$ to Bob. Bob sends $E_w(g^b \bmod p)$ to Alice.
- P3: Alice sends $g^a \bmod p$. Bob sends $g^b \bmod p$ and $F_s(g^a \bmod p \mid\mid g^b \bmod p)$. Alice sends $F_s(g^b \bmod p \mid\mid g^a \bmod p)$.

- P4: Alice sends $g^a \bmod p$ and $\text{Sign}_{K_A^{-1}}(g^a \bmod p)$. Bob sends $g^b \bmod p$ and $\text{Sign}_{K_B^{-1}}(g^b \bmod p)$.

We'd like to know whether each protocol is (1) secure against passive attacks (i.e., eavesdroppers who can observe all messages but don't send anything) and (2) secure against man-in-the-middle attacks.

By secure, we mean that at the end of the protocol, if Alice derives the shared secret s, no one other than Alice and Bob know s; and if Bob derives a shared secret s', then no one other than Alice and Bob know s'. (It'd be nice if $s = s'$, but if there is a man-in-the-middle attack, that might not be guaranteed, and that is not necessarily required for the protocol to be secure.)

(a) Write the following table on your solution, and fill in each of the 8 cells in the table with "Yes" or "No". You do not need to justify any of your answers.

	Secure against passive attacks?	Secure against active attacks?
P1		
P2		
P3		
P4		

(b) Explain why you said that P2 is/isn't secure against active attacks.

(c) Explain why you said that P3 is/isn't secure against active attacks.

(d) Explain why you said that P4 is/isn't secure against active attacks.

Problem 5 *Using cryptographic building blocks* (28 points)

This problem tests your understanding of how to use cryptographic algorithms. Alice and Bob conduct business extensively over the Internet, and they would like to be able to enter into binding contracts with other parties located around the world. To save the environment, they'd like to do it entirely electronically, over the Internet. Therefore, we'd like some cryptographic way that each party can agree to the terms of the contract. If Alice and Bob are contemplating a contract, Alice should be able to tell whether Bob has agreed to the contract, and similarly Bob should be able to tell whether Alice has agreed to the contract.

Once Alice and Bob have agreed to the terms of the contract, neither should be able to back out. If one party fails to live up to his/her obligations, or there is any dispute, we assume that the injured party will bring a lawsuit. In case of a lawsuit, it should be possible for Alice to convince the judge that Bob agreed to the terms of the contract (by showing the judge the messages she received from Bob). Similarly, it should be possible for Bob to convince the judge that Alice agreed to the contract. It should never be necessary for any party to give their private key to the judge, but it is OK to give the judge any session keys that are relevant.

The scheme should be secure against attackers with the ability to intercept and/or modify packets sent electronically, and with the ability to inject forged packets. There should be no way for an attacker to fool Alice into thinking that Bob has accepted any contract that he did not actually accept, and no way to fool Bob into thinking that Alice has accepted any contract that she did not actually accept. Also, in the event of a lawsuit, there should be no way for Alice to fool the judge into concluding that Bob accepted the contract when he actually didn't (even if Alice colludes with the attacker), and no way for Bob to fool the judge into concluding that Alice accepted the contract when she actually didn't (even if he colludes with the attacker). Let's look at several possible cryptographic protocols for this.

Assumptions: You can assume that Alice has a public key K_A and a matching private key K_A^{-1}; Bob has a public key K_B and private key K_B^{-1}; and the court has a public key K_C and a private key K_C^{-1}. Assume that everyone knows everyone else's public key, with no possibility of spoofed public keys (e.g., Alice knows Bob's public key and the court's public key, and so on). The parties do not share their private key with anyone. k_1 is a random session key chosen by Alice (a new one is chosen for each contract Alice signs), and k_2 is a random session key chosen by Bob; both k_1 and k_2 are symmetric keys. Each party has his/her own personal computer, which is not shared, is adequately protected from compromise, and can be assumed free of malware. You may assume that all encryption, MAC, and digital signature algorithms are secure against chosen-plaintext attack and are implemented properly, but you shouldn't assume anything about which specific algorithm is used (we want the contract-signing scheme to be secure, as long as we plug in any secure encryption, MAC, and digital signature algorithm). Let T denote the text of the contract. T will include the identities of the parties (Alice and Bob) and all the terms of the contract. Both Alice and Bob already know T. There is no need to prevent the attacker from learning T. At least one of the schemes below is good.

Instructions: For each scheme listed below, write "Good" if the scheme meets the requirements above (no justification is needed in this case), or write "No good" if the scheme does not meet the requirements above and then explain why the scheme is not acceptable (if it is insecure, sketch the attack).

(a) Alice sends $E_{K_B}(T \mid\mid E_{K_C}(T))$ to Bob. Bob sends $E_{K_A}(T \mid\mid E_{K_C}(T))$ to Alice.

(b) Alice sends $E_{K_B}(k_1), \text{MAC}_{k_1}(0 \mid\mid T)$ to Bob. Bob sends $\text{MAC}_{k_1}(1 \mid\mid T)$ to Alice.

(c) Alice sends $\text{Sign}_{K_A^{-1}}(T)$ to Bob. Bob sends $\text{Sign}_{K_B^{-1}}(T)$ to Alice.

(d) Alice sends $\text{Sign}_{K_A^{-1}}(H(T))$ to Bob, where H is a cryptographic hash function. Bob sends $\text{Sign}_{K_B^{-1}}(H(T))$ to Alice.

(e) Alice sends $E_{K_B}(\text{Sign}_{K_A^{-1}}(T))$ to Bob. Bob sends $E_{K_A}(\text{Sign}_{K_B^{-1}}(T))$ to Alice.

(f) Alice sends $k_1, E_{k_1}(T), \text{Sign}_{K_A^{-1}}(k_1), \text{Sign}_{K_A^{-1}}(E_{k_1}(T))$ to Bob. Bob sends $k_2, E_{k_2}(T), \text{Sign}_{K_B^{-1}}(k_2), \text{Sign}_{K_B^{-1}}(E_{k_2}(T))$ to Alice.

(g) Alice sends $E_{K_B}(k_1 \mid\mid U_1), \text{Sign}_{K_A^{-1}}(U_1)$ to Bob where $U_1 = \ell \mid\mid E_{K_C}(k_1) \mid\mid E_{k_1}(T)$ and ℓ is the length of $E_{k_C}(k_1)$. Bob sends $E_{K_A}(k_2 \mid\mid U_2), \text{Sign}_{K_B^{-1}}(U_2)$ to Alice, where $U_2 = \ell \mid\mid E_{K_C}(k_2) \mid\mid E_{k_2}(T)$ (assume ℓ is the length of $E_{k_C}(k_2)$ too).

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

Optionally, feel free to include feedback. What's 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?