CS255: Cryptography and Computer Security **Winter 2015**

Final Exam

Instructions:

- **Answer all five questions.**
- The exam is open book and open notes. Laptops are allowed with the network card turned off. Connecting to a network during the exam is a violation of the honor code.
- Students are bound by the Stanford honor code.
- You have two hours.

Problem 1. Questions from all over.

a. Both signatures and MACs are used for data integrity, but have different properties. Why is an RSA signature (2048 bits) so much longer than a MAC tag (80-128 bits)?

b. Let $F : K \times X \to Y$ be a secure PRF with $K = X = Y = \{0,1\}^n$. For each of the following three, briefly explain why it is secure or describe an attack:

$$F_1(k,x) := F\big(F(k,0^n),x\big)$$
$$F_2(k,x) := F\big(F(k,0^n),x\big) \,\|\, F(k,x) \quad \text{(here } \| \text{ denotes concatenation)}$$
$$F_3(k,x) := F(k,x) \,\|\, F\big(k,F(k,x)\big)$$

c. In class we showed that truncating the output of a PRF does not hurt security. That is, if $F : K \times X \to Y$ is a secure PRF with $Y = \{0,1\}^n$ then $F'(k,x) := F(k,x)[0,\ldots,\ell]$ is also a secure PRF for every $0 \le \ell \le n$. Here $F'(k,x)$ outputs the first ℓ bits of the output of $F(k,x)$. Can truncating the output of a one-way function $f : X \to Y$ damage the security of the one-way function? Justify your answer.

d. Can truncating the output of a collision resistant hash function $H : M \to T$ damage its collision resistance? Justify your answer.

e. Let p be a prime with $p \equiv 2 \bmod 3$. Show an efficient algorithm that takes $\alpha \in \mathbb{Z}_p^*$ as input and outputs the cube root of α in $\mathbb{Z}_p^*$. That is, show how to efficiently solve the equation $x^3 - \alpha = 0$ in $\mathbb{Z}_p$. **Hint:** recall how RSA decryption works.

f. Is your algorithm from part (e) able to compute cube roots modulo a composite $N = pq$ when the factorization of N is unknown? If so explain why, if not explain why not.

Problem 2. Signcryption. In secure email systems the sender often needs to sign an outgoing email with her secret key and encrypt the email with the recipient's public key for confidentiality. The combination of public-key encryption and signing is called *Signcryption*. One signcryption proposal uses a public-key encryption system (Gen_{enc}, E, D) and a signature scheme (Gen_{sig}, S, V). Every user generates a signing key pair $(\text{sk}_{\text{sig}}, \text{pk}_{\text{sig}})$ and an encryption key pair $(\text{sk}_{\text{enc}}, \text{pk}_{\text{enc}})$.

When Alice wants to send a signed and encrypted (signcrypted) message m to Bob she obtains Bob's public encryption key $\text{pk}_{\text{enc}}^{(\text{bob})}$ and computes

$$\text{ct} \leftarrow E(\text{pk}_{\text{enc}}^{(\text{bob})},\ m) \quad \text{and} \quad \sigma \leftarrow S(\text{sk}_{\text{sig}}^{(\text{alice})},\ \text{ct})\ .$$

Alice then sends $(\text{ct}, \sigma, \text{cert}_{\text{alice}})$ to Bob, where $\text{cert}_{\text{alice}}$ is Alice's certificate on her signing key. Bob checks Alice's certificate and signature σ and then decrypts ct to recover m. If the signature on ct is valid Bob believes the message m came from Alice.

Unfortunately, this simple scheme is problematic. An attacker, Charlie, could intercept the transmitted message and re-sign the ciphertext ct with his own signing key to obtain

$$\sigma' \leftarrow S(\text{sk}_{\text{sig}}^{(\text{charlie})},\ \text{ct}) \qquad (*)$$

Charlie then forwards $(\text{ct}, \sigma', \text{cert}_{\text{charlie}})$ to Bob. When Bob receives this signcrypted message he will incorrectly think that Alice's plaintext m is from Charlie. This can cause problems, for example, when Alice submits her signcrypted homework to the professor, Charlie intercepts Alice's email and re-signs it with his own key as in (*). Charlie then submits Alice's homework as his own and gets credit for Alice's work.

a. Propose a way to prevent the attack above while keeping the same signcryption structure.

b. Another difficulty with the proposal above is that Bob does not obtain Alice's signature on the message m. As a result he cannot easily show a 3rd party (e.g. a judge) that Alice signed m. Propose a different signcryption method by which Bob obtains Alice's signature on the message m. Please be specific about how your proposal works. When Alice and Bob use your proposal, can Charlie mount the attack described above?

Problem 3. Pitfalls in symmetric encryption.

a. Recall that in randomized CBC encryption using AES-128 the encryptor chooses a random 128-bit IV, encrypts the given plaintext using a 128-bit AES key k in CBC mode using the chosen IV, and outputs IV as the first block of the ciphertext.

Most often CBC is implemented in a cryptographic library and developers hardly look at the inner workings of the library. Suppose an attacker changes the way in which the library chooses the IV. Instead of choosing the IV at random the attacker modifies the library to choose the IV as $\text{IV} \leftarrow AES(k_{\text{attacker}},\ k)$ where k_{attacker} is a key known to the attacker and to no one else.

What is the security impact of this library modification on the confidentiality of plaintexts encrypted with this library against an attacker who knows k_{attacker}?

b. Fortunately, the attack from part (a) can be detected by black-box testing (i.e. without looking at source code). Please explain how.

c. Show how the attack from part (a) can be made undetectable to black-box testing (assuming at most 2^{40} plaintexts are encrypted using a given key). Your attack illustrates the dangers of trusting black-box crypto implementations.
Hint: a solution where the attacker's work is increased by a factor of 2^{40} compared to part (a) is acceptable.

d. Recall that in our description of nonce-based CBC encryption we used two keys k_0 and k_1. One key was used for "randomizing" the nonce and the other key was used for the CBC chain. Show that if one incorrectly sets $k_0 = k_1$ then the resulting encryption system is not CPA secure.

Hint: Recall that in the nonce-based CPA security game the challenger has a bit $b \in \{0,1\}$. The attacker sends to the challenger pairs of equal-length messages $(m_0^{(i)}, m_1^{(i)})$ along with unique nonces n_i for $i = 1, 2, \ldots$. The challenger responds with the encryption of $m_b^{(i)}$, encrypted using nonce n_i. The attacker's goal is to determine the bit b. The attack in question requires two queries to the challenger to determine b. The query messages contain no more than two blocks.

Problem 4. Expanding the message space of a cipher and MAC.

a. Let (E, D) be a CPA-secure encryption scheme that encrypts messages in some space M. Let (E', D') encrypt messages in M^ℓ, for some $\ell > 1$, by encrypting each component independently, but using the same secret key. That is, for $\ell = 3$,

$$E'\big(k, (m_0, m_1, m_2)\big) = \big(\ E(k, m_0),\ E(k, m_1),\ E(k, m_2)\ \big)$$

Is (E', D') CPA secure? If so, explain why. If not, describe an attack.

b. Suppose that (E, D) provides authenticated encryption. Does (E', D') provide authenticated encryption? If so, explain why. If not, describe an attack.

c. Let (S, V) be a secure MAC for messages in M. Let (S', V') tag messages in M^ℓ, for $\ell > 1$, by tagging each component independently, using the same signing key for each component. That is, for $\ell = 3$, we have

$$S'\big(k, (m_0, m_1, m_2)\big) = \big(\ S(k, m_0),\ S(k, m_1),\ S(k, m_2)\ \big)$$

Show that (S', V') is not a secure MAC. Your attack should make use of a single message query.

d. Let (S', V') be as above, but sign each component using a different randomly selected secret key. That is, for $\ell = 3$, we have

$$S'\big((k_0, k_1, k_2), (m_0, m_1, m_2)\big) = \big(\ S(k_0, m_0),\ S(k_1, m_1),\ S(k_2, m_2)\ \big)$$

Is (S', V') a secure MAC? If so, explain why. If not, describe an attack.

Problem 5. Password attacks.

a. Suppose an attacker obtains the hash h of some user's password. The password is hashed using SHA256 and no salts are used. The attacker assumes that the password is exactly 8 characters long and each character can be one of 100 possible values. Suppose that on a single machine the attacker can compute 10,000,000 SHA256 hashes per second. Moreover, the attacker rents 1000 machines on some cloud service.

The attacker tries to find the user's password by exhaustively trying all passwords of length 8 until it finds one that hashes to h. Approximately how many seconds will it take the attacker to find the user's password in the worst case?

b. Will public salts make the attack harder? You may assume that if the attacker obtains the user's hashed password it also obtains the user's public salt.

c. Will a slow hash function like PBKDF2 make the attack harder?

d. Now, suppose an attacker obtains the hashed password of ten thousand users. As in part (a) each password is hashed using SHA256 and no salts are used. The attacker assumes that passwords are 8 characters long and each character can be one of 100 possible values. Assuming the attacker has the same computing resources as in part (a), approximately how many seconds will it take to recover all ten thousand passwords in the worst case? Explain why. You may assume that the computation is dominated by the repeated computation of SHA256.

e. Will public salts slow down this attack?

f. Again, an attacker obtains one hash h of some user's password. The password is hashed using SHA256, no salts are used, and the password is 8 characters. Let us show that with pre-processing the attacker can recover the user's password much faster than in part (a).

Recall that SHA256 is a function that outputs hashes in $\{0,1\}^{256}$. Let R be a function that maps elements of $\{0,1\}^{256}$ to eight character passwords. For example, R might map the bottom 56 bits of its given input to an eight character string and ignore the remaining bits of the input.

Pre-processing: Before obtaining h the attacker builds the following data structure:

- Let T be a table indexed by 8-character passwords. Initially $T[\text{pw}] = \perp$ for all pw.
- Let $s = 10^9$. The attacker chooses s random 8-character passwords $\text{pw}_{\mathbf{0}}, \ldots, \text{pw}_{\mathbf{s-1}}$.
- Let $\ell = 10^8$. The attack constructs the following s hash chains, each of length ℓ:

$$\begin{array}{ccccccc}
\text{pw}_{\mathbf{0}} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} \text{pw}_{\mathbf{0}}^{(\mathbf{1})} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} & \cdots & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} \text{pw}_{\mathbf{0}}^{(\ell-\mathbf{1})} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} & \text{pw}_{\mathbf{0}}^{(\ell)} \\
\text{pw}_{\mathbf{1}} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} \text{pw}_{\mathbf{1}}^{(\mathbf{1})} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} & \cdots & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} \text{pw}_{\mathbf{1}}^{(\ell-\mathbf{1})} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} & \text{pw}_{\mathbf{1}}^{(\ell)} \\
 & \vdots & \vdots & & \vdots & & \\
\text{pw}_{\mathbf{s-1}} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} \text{pw}_{\mathbf{s-1}}^{(\mathbf{1})} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} & \cdots & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} \text{pw}_{\mathbf{s-1}}^{(\ell-\mathbf{1})} & \xrightarrow{\text{SHA256}} \cdot \xrightarrow{R} & \text{pw}_{\mathbf{s-1}}^{(\ell)}
\end{array}$$

- The attacker records the beginning and ending points of these hash chains. That is, for $i = 0, \ldots, s-1$ the attacker sets $T[\text{pw}_{\mathbf{i}}^{(\ell)}] = \text{pw}_{\mathbf{i}}$.

Note that the table T is indexed by the tail of s hash chains where each hash chain is of length ℓ. We chose ℓ and s so that $\ell \times s$ is bigger than the number of 8-character passwords (10^{16}). Let us assume that *every* 8-character password appears somewhere in at least one of these s hash chains. The table T contains only 10^9 entries total.

Explain how the attacker can use the data structure T to recover the password using ℓ computations of SHA256. How many seconds will the attack take on a single machine? You may assume that the time for all table lookups in T and evaluation of R is negligible compared to SHA256 computations and can be ignored. We only care about the total time to compute all the required SHA256 throughout the algorithm.

g. Will public salts prevent the attack from part (f)? Explain why. Here we assume that pre-processing work is free; we only count the online time needed to recover a password once its hash becomes known to the attacker.