CS255: Cryptography and Computer Security **Winter 2017**

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 serious violation of the honor code.
- Students are bound by the Stanford honor code.
- You have two hours.

Problem 1. Questions from all over.

a. Explain what goes wrong if we use the same key k in deterministic counter mode to encrypt two different messages $m_0 \neq m_1$.

b. Suppose $h : \mathcal{X} \to \mathcal{Y}$ is a collision resistant hash function, where $\mathcal{Y} \subseteq \mathcal{X}$. Is the function $h^2(x) = h(h(x))$ collision resistant? Give an attack on h^2, or prove that h^2 is collision resistant by showing that an attack on h^2 gives an attack h.

c. Let (S, V) be a MAC where the tag space is $\mathcal{T} = \{0,1\}^{16}$. Show that (S, V) cannot be a secure MAC.

d. Let (S, V) be a secure MAC defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ where $\mathcal{T} = \{0,1\}^n$. Define a new MAC (S', V') as follows: $S'(k, m)$ is the same as $S(k, m)$, except that the last eight bits of the output tag t are truncated. That is, S' outputs tags in $\{0,1\}^{n-8}$. Algorithm $V'(k, m, t')$ accepts if there is some $b \in \{0,1\}^8$ for which $V(k,\ m,\ t' \parallel b)$ accepts. Is (S', V') a secure MAC? Give an attack or argue security.

e. Let (G, S, V) be a secure signature scheme and let $(\text{pk}, \text{sk}) \xleftarrow{R} G()$. One concern is that the signer Bob, who has sk, can find two messages $m_0 \neq m_1$ that have the same signature σ. Suppose Bob gives Alice the signature σ on message m_0. Bob can later claim that he actually signed m_1, not m_0. A judge would have difficulty deciding what Bob actually signed. We say that the signature is non-binding. Give an example of a secure, but non-binding signature scheme.

Hint: try using the hash function from Problem 3 of homework 3, in combination with a secure signature scheme (G', S', V'). Make sure to explain how algorithms (G, S, V) in your signature scheme work.

Problem 2. Authenticated encryption. In class we said that when using encrypt-then-MAC it is important to use independent keys for the CPA-secure cipher and for the MAC. Let us see an example of a CPA-secure cipher and a secure MAC that are insecure when used in encrypt-then-MAC with the same secret key k.

Let (E, D) be a block cipher defined over $(\mathcal{K}, \mathcal{X})$ where $\mathcal{X} = \{0,1\}^n$ and $|\mathcal{X}|$ is large. Consider randomized CBC mode encryption built from (E, D) as the CPA-secure cipher for single block

messages: an encryption of $m \in \mathcal{X}$ is the pair $c := (r,\ E(k,\ r \oplus m))$ where r is the random IV. As the secure MAC, let's use RawCBC built from (E, D). This MAC is secure in this context because it is only being applied to fixed length messages (messages in $\mathcal{X}^2$): the tag on a ciphertext $c \in \mathcal{X}^2$ is $t := E\big(k,\ E(k, c[0]) \oplus c[1]\big)$. The complete encrypt-then-MAC ciphertext is $(c, t) \in \mathcal{X}^3$.

a. Show that using the same key k for both this cipher and this MAC in encrypt-then-MAC results in a cipher that does not provide authenticated encryption.

b. Suppose the cipher and MAC keys are not equal, but are chosen so that they have a known relation, say $k_{\text{mac}} = k_{\text{enc}} \oplus \Delta$ for a known $\Delta \in \mathcal{X}$. Give an example of a CPA-secure cipher and a secure MAC that when combined using encrypt-then-MAC do not provide authenticated encryption.
Hint: try to adjust the construction from part (a).

Problem 3. Dictionary attacks.

a. Briefly explain what is a dictionary attack on a password identification scheme.

b. In class we discussed MAC-based challenge-response identification. Suppose the user's secret key sk is a password. Show that a network eavesdropper who listens in one successful execution of the identification protocol can mount a dictionary attack to recover the user's password sk.

c. Suppose the identification protocol from part (b) is executed over a TLS tunnel with one-sided authentication, where the server presents a server-side certificate. The client, however, does not adequately check the server certificate, and in particular, accepts self-signed certificates. Can an active attacker mount a dictionary attack on the user's password? Justify your answer.

d. Let's enhance MAC-based challenge response to protect it from a dictionary attack without using a server certificate. Let $\mathbb{G}$ be a Diffie-Hellman group of prime order q with generator $g \in \mathbb{G}$. Let $h, k \in \mathbb{G}$ be random generators of $\mathbb{G}$ (both are a random power of g) and all of g, h, k are public values. Let $\text{pw} \in Z_q$ be the shared password and let H be a hash function. The ID protocol begins as follows:

- server: $s \xleftarrow{R} \mathbb{Z}_q,\ S \leftarrow g^s h^{\text{pw}}$, send S to user.
- user: $u \xleftarrow{R} \mathbb{Z}_q,\ U_1 \leftarrow g^u k^{\text{pw}},\ U_2 \leftarrow (S/h^{\text{pw}})^u,\ U_3 \leftarrow H(\text{pw}, S, U_1 U_2)$, send (U_1, U_3) to server.

Explain what the server should check when verifying the user's response.
Note: It can be shown that this protocol prevents dictionary attacks against an active adversary, assuming the CDH problem is hard in $\mathbb{G}$.

Problem 4. The iMessage attack.

a. The parity bit b for a binary string m is the XOR of all the bits in m. Consider a cipher where messages are *variable* length bit strings, and encryption is done using randomized counter mode without any padding. No MAC is used, but before the plaintext is encrypted with counter mode, the sender appends a parity bit to the end of the plaintext. When the receiver decrypts, it checks the parity bit and outputs reject if the bit is invalid.

Design a chosen-ciphertext attack that recovers the complete plaintext of every encrypted message. That is, given a target ciphertext c, your goal is to issue a sequence of ciphertexts

to the decryptor. For each ciphertext, the decryptor sends back reject if the ciphertext is rejected. Otherwise it sends accept. Show that the attacker can use these accept/reject responses to decrypt all of c.

This attack, called chop-chop, was used successfully against the 802.11b protocol, where a WiFi access point played the role of the decryptor.

b. Signcryption combines public-key encryption with a signature to provide both message confidentiality and source identification. Signcryption is used in messaging systems like Apple's iMessage. Consider the following simple scheme: First, Alice generates a public-key/secret key for a signature scheme $(\mathrm{pk}_{\mathrm{sig}}, \mathrm{sk}_{\mathrm{sig}})$. Later, when she wants to send a message m to Bob, she obtains Bob's public-key encryption key $\mathrm{pk}_{\mathrm{enc}}$ and uses the following encryption algorithm:

$$E'(\mathrm{sk}_{\mathrm{sig}}, id_{\mathrm{A}}, \mathrm{pk}_{\mathrm{enc}}, id_{\mathrm{B}},\ m) := \left\{ \begin{array}{l} k \xleftarrow{\mathrm{R}} \mathcal{K}, \quad c_0 \xleftarrow{\mathrm{R}} E_{\mathrm{pub}}(\mathrm{pk}_{\mathrm{enc}},\ k), \quad c_1 \xleftarrow{\mathrm{R}} E_{\mathrm{sym}}\big(k,\ (id_{\mathrm{A}}, m)\big) \\ \sigma \xleftarrow{\mathrm{R}} S\big(\mathrm{sk}_{\mathrm{sig}},\ (c_0, c_1, id_{\mathrm{B}})\big) \\ \text{output } (c_0, c_1, \sigma) \end{array} \right\}$$

Here id_{A} and id_{B} are Alice's and Bob's identities, and $(E_{\mathrm{sym}}, D_{\mathrm{sym}})$ is a symmetric cipher with key space $\mathcal{K}$. Also, (G_{sig}, S, V) is a signature scheme and $(G_{\mathrm{pub}}, E_{\mathrm{pub}}, D_{\mathrm{pub}})$ is a public-key encryption scheme. Bob decrypts by doing:

$$D'\big(\mathrm{pk}_{\mathrm{sig}}, id_{\mathrm{A}}, \mathrm{sk}_{\mathrm{enc}}, id_{\mathrm{B}},\ (c_0, c_1, \sigma)\big) := \left\{ \begin{array}{l} \text{if } V(\mathrm{pk}_{\mathrm{sig}},\ (c_0, c_1, id_{\mathrm{B}}),\ \sigma) = \text{reject, output reject} \\ k \leftarrow D_{\mathrm{pub}}(\mathrm{sk}_{\mathrm{enc}},\ c_0), \quad (id, m) \leftarrow D_{\mathrm{sym}}(k, c_1) \\ \text{if } id \neq id_{\mathrm{A}} \text{ output reject} \\ \text{otherwise, output } m \end{array} \right\}$$

Notice that the encryption algorithm signs the symmetric ciphertext c_1 using Alice's secret signing key. One might think that the signature on c_1 adequately provides ciphertext integrity, and therefore it suffices for $(E_{\mathrm{sym}}, D_{\mathrm{sym}})$ to be CPA-secure, rather than an authenticated encryption cipher. In particular, suppose the developers use randomized counter mode as the cipher $(E_{\mathrm{sym}}, D_{\mathrm{sym}})$.

Let's show that the encryption scheme (E', D') is insecure. Suppose that as in part (a), a parity bit is appended to the plaintext during encryption and this bit is checked during decryption. Use your attack from part (a) to show that given a ciphertext (c_0, c_1, σ) from Alice, an adversary can completely decrypt this ciphertext. As in part (a), the adversary issues a stream of ciphertexts to Bob, and Bob responds with accept or reject to each. The sequence of responses should enable the adversary to decrypt (c_0, c_1, σ). You may assume that the adversary is a user in the system and has its own signing key pair $(\mathrm{pk}_{\mathrm{adv}}, \mathrm{sk}_{\mathrm{adv}})$.

Note: Once again, the lesson is to only use authenticated encryption symmetric ciphers.

Problem 5. Exponentiation algorithms. Let $\mathbb{G}$ be a finite cyclic group of order p with generator g. In class we discussed the repeated squaring algorithm for computing $g^x \in \mathbb{G}$ for $0 \leq x < p$. The algorithm needed at most $2\log_2 p$ multiplications in $\mathbb{G}$.

In this question we develop a faster exponentiation algorithm. For some small constant w, called the window size, the algorithm begins by building a table T of size 2^w defined as follows:

$$\text{set } \ T[k] := g^k \ \text{ for } k = 0, \ldots, 2^w - 1\ . \tag{1}$$

a. Show that once the table T is computed, we can compute g^x using only $(1 + 1/w)(\log_2 p)$ multiplications in $\mathbb{G}$. Your algorithm shows that when the base of the exponentiation g is fixed forever, as in the Diffie-Hellman protocol, the table T can be pre-computed once and for all. Then exponentiation is faster than with repeated squaring.
Hint: Start by writing the exponent x base 2^w so that:

$$x = x_0 + x_1 2^w + x_2 (2^w)^2 + \ldots + x_{d-1}(2^w)^{d-1} \quad \text{where } 0 \le x_i < 2^w \text{ for all } i = 0, \ldots, d-1.$$

Here there are d digits in the representation of x base 2^w. Start the exponentiation algorithm with x_{d-1} and work your way down, squaring the accumulator w times at every iteration.

b. Suppose every exponentiation is done relative to a different base, so that a new table T must be re-computed for every exponentiation. What is the worse case number of multiplications as a function of w and $\log_2 p$?

c. Continuing with Part (b), compute the optimal window size w when $\log_2 p = 256$, namely the w that minimizes the overall worst-case running time. What is the worst-case running time with this w? (counting only multiplications in $\mathbb{G}$)

d. Set $d := \lceil (\log_2 p)/w \rceil$. Going back to the case of a fixed generator g, show how one can pre-compute a table of size $2^w \cdot d$ once and for all, so that using this table exponentiation can be done with at most d multiplications in $\mathbb{G}$. This is a factor $2w$ improvement over repeated squaring, but requires storing a large pre-computed table.
Hint: Try creating d tables $T_0, \ldots, T_{d-1}$, where each table has a structure similar to T from (1).