CS255: Cryptography and Computer Security **Winter 2019**

Final Exam

Instructions:

– **Answer all five questions.**
– Write your answers in the space allocated in this printed exam. The exam will be digitally scanned so please do not write answers on the back side of any page.
– The exam is open book and open notes. Laptops are allowed in airplane mode only. 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 and a half hours**.

I acknowledge and accept the Stanford Honor Code.

(Signature)

(SUNet ID)

(Print your name, legibly!*)*

Problem 1. Questions from all over.

a. Error correcting codes are used to correct random transmission errors. When combining encryption and an error correcting code would you encrypt and then apply the error correcting code or vice versa? Encryption here refers to authenticated encryption, such as encrypt then MAC. Briefly justify your answer. Note that error correction is not a security mechanism so that the composed system need not provide ciphertext integrity.

b. Suppose an attacker steals the long-term secret TLS 1.3 key of a banking web site.

i. Can the attacker act as a passive eavesdropper on future HTTPS connections to that banking web site?

ii. Can the attacker impersonate the banking web sites to bank customers?

c. In class we showed how to instantiate the ElGamal public key system with one of two groups: (1) the group $\mathbb{Z}_{p_1}^*$ for some prime p_1, and (2) the group of points of an elliptic curve $E(\mathbb{F}_{p_2})$ for some prime p_2. Why is it that p_1 must be at least 2048 bits, but p_2 need only be 256 bits, to obtain comparable levels of security?

Problem 2. Committing encryption. A common mistake is to assume that encryption commits the encryptor to the encrypted message. Let (E, D) be a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Suppose Alice chooses some $k \in \mathcal{K}$ and $m \in \mathcal{M}$ and publishes $c := E(k, m)$. This ciphertext c is then stored in a system that prevents any modification to c. Later, Alice is asked to decrypt this c by revealing her key k. We say that the encryption scheme is committing if Alice cannot produce a $k' \in \mathcal{K}$ such that $D(k', c) = m'$ where $m' \neq m$ and $m' \neq \mathsf{reject}$.

a. Give a complete game based definition for committing encryption. Your game need only capture the commitment aspect of the scheme, not confidentiality. Hint: in your game, the challenger does nothing, and the attacker should output two keys k and k', along with some other data.

b. Let CTR denote counter mode encryption with a random IV, with key space $\mathcal{K}_{\mathrm{e}}$. Let (S, V) be a secure MAC with key space $\mathcal{K}_{\mathrm{m}}$. Let (E', D') be the derived CTR-then-MAC cipher whose key space is $\mathcal{K}_{\mathrm{e}} \times \mathcal{K}_{\mathrm{m}}$. We know that (E', D') provides authenticated encryption. Show that (E', D') is not a committing encryption scheme.

c. Let's show that any cipher can be made to be committing. Recall that in homework #3 we defined the concept of a commitment scheme. Such a scheme has a commitment algorithm $\mathsf{commit} : \mathcal{K} \times \mathcal{R} \to \mathcal{C}_{\text{com}}$ used to commit to a key $k \in \mathcal{K}$ using randomness $r \xleftarrow{\text{R}} \mathcal{R}$. The scheme must be *hiding* and *binding*. Show that any authenticated encryption cipher (E, D) defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ can be converted into a committing encryption scheme (E', D') defined over some $(\mathcal{K}', \mathcal{M}, \mathcal{C}')$ by using a commitment scheme. Your answer should describe algorithms E' and D' as well as the key space $\mathcal{K}'$ and ciphertext space $\mathcal{C}'$. Briefly explain why your (E', D') is committing and provides authenticated encryption.

d. We can extend our discussion of committing encryption to public-key encryption. The only modification to your game based definition from part (a) is that the adversary must also output a single public key pk along with the ciphertext c, and the released secret keys must be compatible with this pk. Show that ElGamal encryption is a committing encryption scheme.

Problem 3. ElGamal tricks. Bob's corporate mail server publishes a public-key $\mathrm{pk}_{\mathrm{bob}}$ so that all incoming emails to Bob are encrypted under $\mathrm{pk}_{\mathrm{bob}}$. When Bob goes on vacation he instructs the company's mail server to forward all his incoming email to his colleague Alice. Alice's public key is $\mathrm{pk}_{\mathrm{alice}}$. The mail server needs a way to translate an email encrypted under public-key $\mathrm{pk}_{\mathrm{bob}}$ into an email encrypted under public-key $\mathrm{pk}_{\mathrm{alice}}$. This would be easy if the mail server had $\mathrm{sk}_{\mathrm{bob}}$, but then the mail server could read all of Bob's incoming email, which is undesirable.

Let $\mathbb{G}$ be a group of prime order q with generator $g \in \mathbb{G}$. Consider a minor variation of the ElGamal encryption scheme from lecture where encryption using a public key $\mathrm{pk} := u = g^{\alpha} \in \mathbb{G}$ works as follows:

$$E(\mathrm{pk}, m) = \left\{ \beta \xleftarrow{\mathrm{R}} \mathbb{Z}_q,\ v \leftarrow g^{\beta},\ k \leftarrow H(u^{\beta}),\ c \leftarrow E_{\mathrm{sym}}(k, m),\ \text{output } (v, c) \right\}.$$

Here E_{sym} is the encryption algorithm of a symmetric cipher with key space $\mathcal{K}_{\mathrm{sym}}$, and H is a hash function $H : \mathbb{G} \to \mathcal{K}_{\mathrm{sym}}$.

Suppose that $\mathrm{pk}_{\mathrm{bob}}$ and $\mathrm{pk}_{\mathrm{alice}}$ are public keys for this scheme with corresponding secret keys $\mathrm{sk}_{\mathrm{bob}} = \alpha \in \mathbb{Z}_q$ and $\mathrm{sk}_{\mathrm{alice}} = \alpha' \in \mathbb{Z}_q$. To enable private translation of ciphertexts from $\mathrm{pk}_{\mathrm{bob}}$ to $\mathrm{pk}_{\mathrm{alice}}$, Alice and Bob get together to compute $\tau := \alpha/\alpha' \in \mathbb{Z}_q$. They give τ to the mail server.

a. Explain how the mail server uses τ to translate a ciphertext $c \xleftarrow{\mathrm{R}} E(\mathrm{pk}_{\mathrm{bob}}, m)$ to a ciphertext c' for $\mathrm{pk}_{\mathrm{alice}}$ for the same message m. Justify your answer.

b. Show that τ can also be used to translate in the other direction. That is, if $c \xleftarrow{R} E(\text{pk}_{\text{alice}}, m)$ then the mail server can construct a ciphertext c' for pk_{bob} for the same message m.
Note that this is an unintended consequence that Alice did not want. It is not difficult to modify the scheme to prevent this unintended feature, but we will not do that here.

c. Show that if the mail server could use τ to decrypt messages for Bob, then this would lead to a total break of this ElGamal encryption scheme. That is, suppose there is an efficient adversary $\mathcal{A}$ that given $\big(\text{pk}_{\text{bob}},\ \text{pk}_{\text{alice}},\ \tau,\ c \xleftarrow{R} E(\text{pk}_{\text{bob}}, m)\big)$ as input, outputs m. Show that there is an efficient adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break semantic security of this ElGamal scheme (without knowledge of τ).

d. When Bob comes back from vacation, what should he do to make sure that Alice can no longer read his emails?

Problem 4. Attacks on CBC.

a. In class we discussed the ECBC (encrypted CBC) MAC for messages in $\mathcal{X}^{\leq L}$ where $\mathcal{X} = \{0,1\}^n$. Recall that RawCBC is the same as ECBC, but without the very last encryption step. We showed that RawCBC is an insecure MAC for variable length messages. Here we show a more devastating attack on RawCBC. Let m_1 and m_2 be two multi-block messages. Show that by asking the signer for the MAC tag on m_1 and for the MAC tag on one additional multi-block message m_2' of the same length as m_2, the attacker can obtain the MAC tag on $m = m_1 \parallel m_2$, the concatenation of m_1 and m_2.

b. Let's see a real-world attack on CBC encryption with a predictable IV. Suppose Bob uses AES-CBC encryption with key k to encrypt blocks on disk, where each block is 4096 bytes long (4KB), the default block size for Linux. Disk block number i is CBC encrypted with key k and using an IV equal to the binary representation of i. This ensures that if two blocks on disk hold the same data, they result in different ciphertexts. Note that there is no need to store the IV with the ciphertext because the IV is derived from the block number. Moreover, if a single file spans multiple disk blocks, then each disk block is AES-CBC encrypted on its own. All disk blocks are encrypted using the same secret key k.

Suppose an authoritarian regime publishes a subversive video m and Bob stores m encrypted on his disk. Later, Bob's machine is seized and the authoritarian regime wants to test if Bob's disk contains an encrypted copy of m (if so, Bob may have some explaining to do). If this were possible, it would be a serious violation of semantic security.

Show that the regime can create a multi-block video m for which it is easy to test if the encrypted m is stored on Bob's disk. You do not know the block number where m will be stored on Bob's disk. However, you may assume that m is block-aligned, that is, the first byte of m is stored as the first byte of some block on disk, byte number 4097 of m is the first byte of the next consecutive block, etc. If m is stored on Bob's disk then your test should always say "yes". For simplicity, you may assume that all other content stored on Bob's encrypted disk is random bytes.

Hint: Try to design a message m that when encrypted as described above results in a sequence of encrypted disk blocks, each 4096 bytes long, where some two consecutive encrypted blocks begin with the same sequence of 16 bytes. It may help to first answer the question assuming the first encrypted block of the message m is stored in block number i, where i is an (unknown) even number. Then generalize your answer to an arbitrary (unknown) i.

Note: in practice, disk encryption systems that use AES-CBC, set the IV for block i to be $\text{AES}(k', i)$, where k' is an independent secret key. This prevents your attack.

Problem 5. Group key exchange. n parties $A_1, \ldots, A_n$ want to setup a group secret key that they can use for encrypted group messaging. For simplicity, let's assume n is a power of two. They have at their disposal a public bulletin board (a cloud server) that they can all post messages to and that will be visible to all of them, as well as to the rest of the world. Our goal here is to design a group key exchange that is secure against a passive eavesdropper.

a. Let $\mathbb{G}$ be a group of prime order q with generator $g \in \mathbb{G}$. Let $H : \mathbb{G} \to \mathbb{Z}_q$ be a random hash function. Your goal is to design a protocol based on two-party (anonymous) Diffie-Hellman that runs in at most $\log_2 n$ rounds. In each round every party posts at most one group element in $\mathbb{G}$ to the bulletin board and reads at most one group element from the bulletin board. The parties can only send messages to the bulletin board; no peer-to-peer communication is allowed. At the end of the protocol, after $\log_2 n$ rounds, all parties obtain the same secret key and there are at most $2n$ group elements on the bulletin board. You may assume that the parties know each other's identities and can order themselves lexicographically by identity. Hint: think of an n-leaf binary tree where each leaf corresponds to one party.

b. Suppose one of the n parties has a malfunctioning random number generator – whenever that party needs to sample a random number in $\mathbb{Z}_q$, its random number generator always returns $5 \in \mathbb{Z}_q$. Show that this will sink the entire protocol from part (a), even if all the other participants have well functioning random number generators. Specifically, show that an eavesdropper will learn the group secret key.

c. Is it possible to design a group key exchange protocol among n parties that is secure against passive eavesdropping even if at most one of the participants suffers from the problem in part (b)? Justify your answer.