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

Assignment #3

Due: Monday, Mar. 11, 2019, by Gradescope (each answer on a separate page).

Problem 1. Let's explore why in the RSA trapdoor permutation every party has to be assigned a different modulus $n = pq$. Suppose we try to use the same modulus $n = pq$ for everyone. Every party is assigned a public exponent $e_i \in \mathbb{Z}$ and a private exponent $d_i \in \mathbb{Z}$ such that $e_i \cdot d_i = 1 \bmod \varphi(n)$. At first this appears to work fine: to sign a message $m \in \mathcal{M}$, Alice would publish the signature $\sigma_\mathrm{a} \leftarrow H(m)^{d_\mathrm{a}} \in \mathbb{Z}_n$ where $H : \mathcal{M} \to \mathbb{Z}_n^*$ is a hash function. Similarly, Bob would publish the signature $\sigma_\mathrm{b} \leftarrow H(m)^{d_\mathrm{b}} \in \mathbb{Z}_n$. Since Alice is the only one who knows $d_\mathrm{a} \in \mathbb{Z}$ and Bob is the only one who knows $d_\mathrm{b} \in \mathbb{Z}$, this seems fine.

Let's show that this is completely insecure: Bob can use his secret key d_b to sign messages on behalf of Alice.

a. Show that Bob can use his public-private key pair $(e_\mathrm{b}, d_\mathrm{b})$ to obtain a multiple of $\varphi(n)$. Let us denote that integer by V.

b. Now, suppose Bob knows Alice's public key e_a. Show that for any message $m \in \mathcal{M}$, Bob can compute $\sigma \leftarrow H(m)^{1/e_\mathrm{a}} \in \mathbb{Z}_n$. In other words, Bob can invert Alice's trapdoor permutation and obtain her signature on m.
Hint: First, suppose e_a is relatively prime to V. Then Bob can find an integer d such that $d \cdot e_\mathrm{a} = 1 \bmod V$. Show that d can be used to efficiently compute σ. Next, show how to make your algorithm work even if e_a is not relatively prime to V.

Note: In fact, one can show that Bob can completely factor the global modulus n.

Problem 2. Consider again the RSA-FDH signature scheme. The public key is a pair (N, e) where N is an RSA modulus, and a signature on a message $m \in \mathcal{M}$ is defined as $\sigma := H(m)^{1/e} \in \mathbb{Z}_N$, where $H : \mathcal{M} \to \mathbb{Z}_N$ is a hash function. Suppose the adversary could find three messages $m_1, m_2, m_3 \in \mathcal{M}$ such that $H(m_1) \cdot H(m_2) = H(m_3)$ in $\mathbb{Z}_N$. Show that the resulting RSA-FDH signature scheme is no longer existentially unforgeable under a chosen message attack.

More generally, your attack shows that for security of the signature scheme, it should be difficult to find a set of inputs to H where the corresponding outputs have a known algebraic relation in $\mathbb{Z}_N$. One can show that this is indeed the case for a random function $H : \mathcal{M} \to \mathbb{Z}_N$, which is what we assumed when proving security of RSA-FDH.

Problem 3. A bad choice of primes for RSA. Let's see why when choosing an RSA modulus $n = pq$ it is important to choose the two primes p and q *independently* at random. Suppose n is generated by choosing the prime p at random, and then choosing the prime q dependent on p. In particular, suppose that p and q are close, namely $|p - q| < n^{1/4}$. Let's show that the resulting n can be easily factored.

a. Let $A = (p + q)/2$ be the arithmetic mean of p and q. Recall that $\sqrt{n}$ is the geometric mean of p and q. Show that when $|p - q| < n^{1/4}$ we have that

$$A - \sqrt{n} < 1.$$

Hint: one way to prove this is by multiplying both sides by $A + \sqrt{n}$ and then using the fact that $A \geq \sqrt{n}$ by the AGM inequality.

b. Because p and q are odd primes, we know that A is an integer. Then by part (a) we can deduce that $A = \lceil \sqrt{n} \rceil$, and therefore it is easy to calculate A from n. Show that using A and n it is easy to factor n.

As an <u>optional</u>, more challenging question, show how to efficiently factor $n = pq$ when you are told that $|p - 2q| < n^{1/4}/4$.

Problem 4. A commitment scheme enables Alice to commit a value x to Bob. The scheme is *hiding* if the commitment does not reveal to Bob any information about the committed value x. At a later time Alice may *open* the commitment and convince Bob that the committed value is x. The commitment is *binding* if Alice cannot convince Bob that the committed value is some $x' \neq x$. Here is an example commitment scheme:

Public values: A group $\mathbb{G}$ of prime order q and two generators $g, h \in \mathbb{G}$.

Commitment: To commit to an integer $x \in \mathbb{Z}_q$ Alice does the following: (1) she chooses a random $r \in \mathbb{Z}_q$, (2) she computes $b = g^x \cdot h^r \in \mathbb{G}$, and (3) she sends b to Bob as her commitment to x.

Open: To open the commitment Alice sends (x, r) to Bob. Bob verifies that $b = g^x \cdot h^r$.

Show that this scheme is hiding and binding.

a. To prove the hiding property show that b reveals no information about x. In other words, show that given b, the committed value can be any element x' in $\mathbb{Z}_q$.
Hint: show that for any $x' \in \mathbb{Z}_q$ there exists a unique $r' \in \mathbb{Z}_q$ so that $b = g^{x'} h^{r'}$.

b. To prove the binding property show that if Alice can open the commitment as (x', r'), where $x \neq x'$, then Alice can compute the discrete log of h base g. In other words, show that if Alice can find an (x', r') such that $b = g^{x'} h^{r'}$ and $x \neq x'$ then she can find the discrete log of h base g. Recall that Alice also knows the (x, r) used to create b.

c. Show that the commitment is *additively homomorphic*: given a commitment to $x \in \mathbb{Z}_q$ and a commitment to $y \in \mathbb{Z}_q$, Bob can construct a commitment to $z = ax + by$, for any $a, b \in \mathbb{Z}_q$ of his choice.

Problem 5. Fast one-time signatures from discrete-log. Let's see another application for the commitment scheme from the previous problem. Let $\mathbb{G}$ be a group of prime order q with generator g. Consider the following signature system for signing messages in $\mathbb{Z}_q$:

> KeyGen: choose $x, y \xleftarrow{\text{R}} \mathbb{Z}_q$, set $h := g^x$ and $u := g^y$.
> output $\text{sk} := (x, y)$ and $pk := (g, h, u) \in \mathbb{G}^3$.
>
> $\mathsf{Sign}(\text{sk},\ m \in \mathbb{Z}_q)$: output $s \in \mathbb{Z}_q$ such that $u = g^m h^s$.
>
> $\mathsf{Verify}(pk, m, s)$: output 'yes' if $u = g^m h^s$ and 'no' otherwise.

a. Explain how the signing algorithm works. That is, show how to find s using sk. Note that signing is super fast.

b. Show that the signature scheme is weakly one-time secure assuming the discrete-log problem in $\mathbb{G}$ is hard. The weak one-time security game is defined as follows:

> the adversary $\mathcal{A}$ first outputs a message $m \in \mathbb{Z}_q$ and in response is given the public key pk and a valid signature s on m relative to pk. The adversary's goal is to output a signature forgery (m^*, s^*) where $m \neq m^*$.

Show how to use $\mathcal{A}$ to compute discrete-log in $\mathbb{G}$. This will prove that the signature is secure in this weak sense as long as the adversary sees at most one signature.

[Recall that in the standard game defined in class the adversary is first given the public-key and only then outputs a message m. In the weak game above the adversary is forced to choose the message m *before* seeing the public-key. The standard game from class gives the adversary more power and more accurately models the real world.]

Hint: Your goal is to construct an algorithm $\mathcal{B}$ that given a random $h \in \mathbb{G}$ outputs an $x \in \mathbb{Z}_q$ such that $h = g^x$. Your algorithm $\mathcal{B}$ runs adversary $\mathcal{A}$ and receives a message m from $\mathcal{A}$. Show how $\mathcal{B}$ can generate a public key $pk = (g, h, u)$ so that it has a signature s for m. Your algorithm $\mathcal{B}$ then sends pk and s to $\mathcal{A}$ and receives from $\mathcal{A}$ a signature forgery (m^*, s^*). Show how to use the signatures on m^* and m to compute the discrete-log of h base g.

c. Show that this signature scheme is not 2-time secure. Given the signature on two distinct messages $m_0, m_1 \in \mathbb{Z}_q$ show how to forge a signature for any other message $m \in \mathbb{Z}_q$.

Problem 6. Oblivious transfer from ElGamal encryption. Alice runs an online bookstore with books $m_1, \ldots, m_n \in \mathcal{M}$. Bob wants to read book number $1 \leq i \leq n$. He pays Alice for the book and wants to download the book (assume the price is the same for all books). However, he does not want to reveal to Alice what book he is interested in. Similarly, Alice wants to make sure Bob gets exactly one book. This problem is called *oblivious transfer*. They need a protocol that reveals m_i (and nothing else) to Bob, and reveals nothing at all to Alice.

They use the following protocol that uses a group $\mathbb{G}$ of prime order q with generator $g \in \mathbb{G}$:

- Alice sends a random $v \xleftarrow{R} \mathbb{G}$ to Bob,
- Bob chooses $\alpha \xleftarrow{R} \mathbb{Z}_q$ and sends $u \leftarrow g^\alpha v^{-i} \in \mathbb{G}$ to Alice,
- For $j = 1, \ldots, n$ Alice encrypts book m_j using the ElGamal public-key $u_j \leftarrow uv^j$ to obtain an ElGamal ciphertext c_j. She sends all n ElGamal ciphertexts $c_1, \ldots, c_n$ to Bob.

a. Explain how Bob can recover m_i from the data it receives from Alice.
Hint: there is something special about the public-key u_i.

b. Explain why Alice learns nothing about i.

c. Let's argue that Bob learns nothing other than m_i. First, suppose there is an efficient algorithm $\mathcal{A}$ that takes as input $g^r \in \mathbb{G}$ and outputs $g^{(r^2)} \in \mathbb{G}$. Show that $\mathcal{A}$ can be used to solve the Computational Diffie-Hellman (CDH) problem in $\mathbb{G}$. That is, construct an efficient algorithm $\mathcal{B}$ that takes as input $(g^a, g^b) \in \mathbb{G}^2$ and outputs g^{ab}. Your algorithm $\mathcal{B}$ shows that if CDH is hard in $\mathbb{G}$, then so is the squaring problem.
Hint: use the relation $(a+b)^2 = a^2 + 2ab + b^2$. Your algorithm $\mathcal{B}$ will use algorithm $\mathcal{A}$ as a black box subroutine.

d. Now, suppose Bob sends a malicious $u \in \mathbb{G}$ that later lets him learn something about two books m_{i_1} and m_{i_2}, for some $i_1 \neq i_2$. One can show that this means that Bob has an efficient algorithm $\mathcal{A}'$ for the following problem: $\mathcal{A}'$ takes as input random $(v,\ c_1 = g^{r_1},\ c_2 = g^{r_2}) \in \mathbb{G}^3$ and outputs (u, i_1, i_2, w_1, w_2) where $w_1 = (uv^{i_1})^{r_1}$ and $w_2 = (uv^{i_2})^{r_2}$. Show that $\mathcal{A}'$ can be used to solve the squaring problem in $\mathbb{G}$. That is, construct an efficient algorithm $\mathcal{B}'$ that takes as input $g^a \in \mathbb{G}$ and outputs $g^{(a^2)}$ by using $\mathcal{A}'$ as a black box.

You just showed that if squaring is hard in $\mathbb{G}$ (which follows from CDH by part (c)), then $\mathcal{A}'$ cannot exist and therefore Bob cannot gain information about two books.

Hint: Given random g^a as input, your algorithm $\mathcal{B}'$ sets $v \leftarrow g^a$ and sets $c_1 \leftarrow v^{s_1} = g^{as_1}$ and $c_2 \leftarrow v^{s_2} = g^{as_2}$ for random $s_1, s_2 \xleftarrow{R} \mathbb{Z}_q$. It runs $\mathcal{A}'$ on (v, c_1, c_2) and obtains (u, i_1, i_2, w_1, w_2) where $w_1 = (uv^{i_1})^{as_1}$ and $w_2 = (uv^{i_2})^{as_2}$. Use the output from $\mathcal{A}'$ to compute $g^{(a^2)}$.

Problem 7. Time-space tradeoff. Let $f : X \to X$ be a one-way permutation (i.e., a one-to-one function on X). Show that one can build a table T of size $2B$ elements of X ($B \ll |X|$)

that enables an attacker to invert f in time $O(|X|/B)$. More precisely, construct an $O(|X|/B)$-time deterministic algorithm $\mathcal{A}$ that takes as input the table T and a $y \in X$, and outputs an $x \in X$ satisfying $f(x) = y$. This result suggests that the more memory the attacker has, the easier it becomes to invert functions.
Hint: Pick a random point $z \in X$ and compute the sequence

$$z_0 := z, \;\; z_1 := f(z), \;\; z_2 := f(f(z)), \;\; z_3 := f(f(f(z))), \;\; \ldots$$

Since f is a permutation, this sequence must come back to z at some point (i.e. there exists some $j > 0$ such that $z_j = z$). We call the resulting sequence $(z_0, z_1, \ldots, z_j)$ an f-cycle. Let $t := \lceil |X|/B \rceil$. Try storing $(z_0, z_t, z_{2t}, z_{3t}, \ldots)$ in memory. Use this table (or perhaps, several such tables) to invert an input $y \in X$ in time $O(t)$.

Discussion: Time-space tradeoffs of this nature can be used to attack unsalted hashed passwords, as discussed in class. Time-space tradeoffs also exist for general one-way functions (not just permutations), but their performance is not as good as your time-space tradeoff above. These algorithms are called *Hellman tables* and discussed in Section 18.7 in the book.