CS255: Intro. to Cryptography **Winter 2025**

Assignment #4

Due: Wed., Mar. 12, 2025, by Gradescope (each answer on a separate page).

Problem 1. (*RSA modulus*) 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.

Answer: Given $e_i \in \mathbb{Z}, d_i \in \mathbb{Z}, e_i \cdot d_i = 1 \bmod \varphi(n)$, we can get $e_i \cdot d_i = V + 1, V = C\varphi(n)$, so $e_i \cdot d_i - 1 = V$
Basically Bob just multiply e_b and d_b then minus 1, the result would be a multiple of $\varphi(n)$

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.

Answer: suppose e_a is relatively prime to V, by Euclid algorithm, we can find (d, b) s.t. $de_a + bV = 1$ by $log(e_a + V)$ time. Taking mod(V) on both sides, here we get $d \cdot e_a = 1 \bmod (V)$. Given V is a multiple of $\varphi(n)$, so $d \cdot e_a = 1 \bmod \varphi(n)$. Now we find the private key $d_a = $ d. Bob can construct signature by raise H(m) to d to construct $\sigma \leftarrow H(m)^d \in \mathbb{Z}_n$.
When e_a is not co-prime to V, $g = GCD(e_a, V)$. Given V is a multiple of $\phi(n)$, we have $de_a + bV = g$. Since $e_a \cdot d_a = 1 \mod (\phi(n))$, we know $G(e_a, \phi(n)) = 1$. Combining $g = GCD(e_a, V)$, we know $G(g, \phi(n)) = 1$, meaning $\frac{V}{g}$ is still a multiple of $\phi(n)$. Also from $G(g, \phi(n)) = 1$, we know there exist an inverse of g, g^{-1}
From $\frac{de_a}{g} + \frac{bV}{g} = 1$, taking $\mod (\phi(n))$ on both sides, we know $\frac{d}{g}e_a = 1 \mod (\phi(n))$, Bob can construct signature by raise H(m) to $\frac{d}{g}$ to construct $\sigma \leftarrow H(m)^{\frac{d}{g}} \in \mathbb{Z}_n$.

Note: In fact, one can show that Bob can completely factor the global modulus n. Then Bob can compute Alice's secret key d_a.

Problem 2. (*RSA signatures*) 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 can 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.

Answer: we have $\sigma_1 := H(m_1)^{1/e} \in \mathbb{Z}_N$ and $\sigma_2 := H(m_2)^{1/e} \in \mathbb{Z}_N$
If $H(m_1) \cdot H(m_2) = H(m_3)$ in $\mathbb{Z}_N$, then $\sigma_1 \cdot \sigma_2 = H(m_1)^{1/e} \cdot H(m_2)^{1/e} = H(m_3)^{1/e} = \sigma_3 \in \mathbb{Z}_N$ Here we forge the signature of m_3, by querying the signature of m_1, m_2

Problem 3. (*Commitment schemes*) 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 element $x \in \mathbb{Z}_q$ 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 $b = g^{x'} h^{r'}$.

Answer: It is obvious that $b, g, h \in G$, suppose c_1 is an integer s.t. $g^{c_1} = h$ and c_2 is an integer that $g^{c_2} = b$, then for any $x' \in Z_q$ we can solve r' by x', c_1 and c_2. (here we don't need to know the exact value of c_1 and c_2, but we know they must exist)
$g^{c_2} = b = g^{x'} \cdot h^{r'} = g^{x'} \cdot g^{c_1 \cdot r'}$ then $c_2 = x' + c_1 \cdot r' \bmod (q)$, $c_2 - x' = c_1 \cdot r' \bmod (q)$
Since h is a generator, $c_1 \neq 0$ so $GCD(c_1, q) = 1$, c_1 has an inverse. Then $r' = (c_2 - x')(c_1)^{-1} \bmod (q)$, this proves that for any $x' \in Z_q$ there exist an $r' \in Z_q$.
Then we can prove r' is unique. Suppose r' is not unique, there exist $x'' \neq x', x'' \in Z_q, x' \in Z_q$ s.t. $(c_2 - x')(c_1)^{-1} = (c_2 - x'')(c_1)^{-1}$, which is equivalent to $(x'' - x')(c_1)^{-1} = 0$. We already know $c_1 \neq 0$, so $(x'' - x')$ must be 0, which leads to a contradiction.

b. To prove the binding property show that if Alice can find a commitment b and two openings (x, r) and (x', r'), where $x \neq x'$, then Alice can compute the discrete log of h base g. Conclude that if the discrete log problem is hard in $\mathbb{G}$, and h is chosen uniformly in $\mathbb{G}$, then the commitment scheme must be binding.
Hint: use the fact that $b = g^x h^r = g^{x'} h^{r'}$, where Alice knows x, r, x', r', to find the discrete log of h base g.

Answer: From the problem statement we can get $b = g^x h^r = g^{x'} h^{r'}$, and $b, g^x, h^r, g^{x'}, h^{r'} \in \mathbb{G}$, so we can take their inverse and get $g^{x-x'} = h^{r'-r}$, take discrete log based on g both left and right, we have $(r' - r)logh = x - x' \bmod (q)$. In the above equation, x, r are given, and x', r' are constructed value. So in order to successfully construct another opening, Alice needs to solve a discrete log problem $log(h)$ based generator g, which is believe to be a computationally hard problem. In other words, it is hard for Alice to construct another opening other than the existing witness (x, r).

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.

Answer: Suppose the first randomness for commitment is r_1, $b_x = g^x h^{r_1}$. Suppose the second randomness is r_2, $b_y = g^y h^{r2}$, then $b_x^a \cdot b_y^b = g^{ax} h^{ar_1} \cdot g^{by} h^{br_2} = g^{ax+by} h^{ar_1+br_2}$. Here we construct the commitment to z,
$commit(z) = b_x^a \cdot b_y^b$ and the randomness is $ar_1 + br_2$

Problem 4. (*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: choose 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 $O(t)$.

Answer:
Core idea: we slice the chain of permutation mapping into shorter ones, and store the "checkpoints" of the chain for lookup. To compute the pre-image, we keep running f and check if it matches any "checkpoints" in the table. If there is a match, it is very likely the pre-image is in the slice ending with the matched checkpoints. Details is described as follows:
Let $t := \lceil |X|/B \rceil$, and construct a table T as follows
for $i = 1, \ldots, B$:
    $z_i \xleftarrow{R} X$
    $z_{ii} \leftarrow f^{(t)}(z_i) \in X$
end for
output $L := \{(z_0, z_{00}), \ldots, (z_B, z_{BB}) \subseteq \mathcal{P}^2$
Here we construct table T that contains B pairs of element. This can be done offline. When we have $x \in X$ satisfying $f(x) = y$ and we want to compute pre-image of y, we do the following
Let z = f(y) for $i = 1, \ldots, t$:
    if there is a $(y, z) \in T$
        $c \leftarrow f^{(t-i)}(y) \in X$
        if h(c) = y
            output c
    y = z, z = f(z)
output fail
The algorithm needs t times evaluation of f and at most t times lookup in table.
However this algo won't always work due to 1) we may not cover all connected component in permutation f 2) if two chains collide, they will merge which can produce false alarm on z, a wrong ending element, and consequently, wrong starting element.
A mitigation is to build many small independent tables, and each table contains a small number of chains of ensuring that no collisions occur within a single table. This can greatly reduce the chain collision problem.

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.

Problem 5. (*Identification protocols*) In the lecture on identification protocols we saw a protocol called S/key that uses an iterated one-way function. In this question we explore the security of iterated one-way functions.

a. Let's show that the iteration of a one-way function need not be one-way. To do so, let $f : \mathcal{X} \to \mathcal{X}$ be a one-way function, where $0 \in \mathcal{X}$. Let $\hat{f} : \mathcal{X}^2 \to \mathcal{X}^2$ be defined as:

$$\hat{f}(x, y) = \begin{cases} (0, 0) & \text{if } y = 0 \\ (f(x), 0) & \text{otherwise} \end{cases}$$

Show that $\hat{f}$ is one-way, but $\hat{f}^{(2)}(x, y) := \hat{f}(\hat{f}(x, y))$ is not.

Answer: To recap, the definition of one-way function is that, the probability of computing the pre-image of $f(x)$ is negligible if x is randomly sampled from the input space.
First, we prove $\hat{f}$ is one way by contrapositive. Assume $\hat{f}(x, y)$ is not one way, then when $y \neq 0$, we can efficiently find (x', y') s.t. $\hat{f}(x', y') = \hat{f}(x', y') = (f(x), 0)$. This equation also indicates that we efficiently find a pre-image of x, s.t. $f(x) = f(x')$. This contradicts the assumption that f is one-way.
Second, we prove $\hat{f}^{(2)}(x, y)$ is not one-way. For $\forall(x, y) \in \mathcal{X}^2$, $\hat{f}^{(2)}(x, y) = 0$, meaning any (x, y) is trivially a pre-image for any (x, y). So the probability of computing an pre-image when in input is sampled uniformly from $\mathcal{X}^2$ is 1, which is not negligible

b. Let's show that the iteration of a one-way permutation is also one-way (recall that a permutation is a one-to-one function). Suppose $f : \mathcal{X} \to \mathcal{X}$ is a one-way permutation. Show that $f^{(2)}(x) := f(f(x))$ is also one-way. As usual, prove the contrapositive.

Answer: Prove the contrapositive.
Suppose $f^{(2)}(x)$ is not one-way, given x is randomly sampled from $\mathcal{X}$, there exists an efficient algorithm A that computes the pre-image of $z = f^{(2)}(x)$. Then we can use it to compute pre-image of $f(x)$. Algo B does follows: when asked to compute pre-image $y = f(x)$, B just use A to compute the pre-image of $z = f(y) = f^{(2)}(x)$. The output of A will be the preimage of y. Moreover, if A is efficient, then B is also efficient, which prove $f(x)$ is not one-way.
So we proved that if $f(x)$ is one-way, then $f^{(2)}(x)$ is one-way

c. Explain why your proof from part (b) does not apply to a one-way function. Where does the proof fail?

Answer:
In (b), for a permutation f, we have that f is bijective (one-to-one and onto), so f^{-1} exists as a function.
In (a), for a general one-way function that is not a permutation, f may map multiple inputs to the same output, therefore, f^{-1} is not well-defined as a function for all values in the range of f. Without the permutation property, finding a preimage of $f^{(2)}(x)$ doesn't necessarily help find a preimage of $f(x)$

Problem 6. (*authenticated key exchange*) In class we saw a one-sided AKE (authenticated key exchange protocol) with forward-secrecy and a two-sided AKE without forward-secrecy. Let's try to construct the best of both worlds: a two-sided AKE with forward-secrecy.

Consider the following two-sided AKE with forward-secrecy between Alice and Bank: They each have a certificate for a signing key and we denote by $S_{\text{alice}}(\text{data})$ and $S_{\text{bank}}(\text{data})$ their respective signatures on 'data'. They fix a group $\mathbb{G}$ of order q and generator $g \in \mathbb{G}$. Alice chooses a random a and Bank chooses a random b, both in $\mathbb{Z}_q$. They exchange the following messages:

$$\begin{array}{lcl} \textit{Alice} & \xrightarrow{\quad g^a\,,\quad \text{cert}_{\text{alice}}\,,\quad S_{\text{alice}}\,(g^a) \quad} & \textit{Bank} \\ & \xleftarrow{\quad g^b\,,\quad \text{cert}_{\text{bank}}\,,\quad S_{\text{bank}}\,\left(g^a,\ g^b\right) \quad} & \\ k \leftarrow H(g^{ab}, \text{“alice”}) & & k \leftarrow H(g^{ab}, \text{id from cert}_{\text{alice}}) \end{array}$$

Both sides compute the same key k using a hash function $H : \mathbb{G} \times \mathcal{ID} \to \mathcal{K}$, and each side deletes its secret a or b. If all the certificates and signatures verify correctly then Alice thinks she is speaking with Bank and Bank thinks it is speaking with Alice. The protocol provides forward-secrecy because a compromise of the server or the client does not compromise past sessions.

Since the Diffie-Hellman messages in this protocol are signed by the participants, one might expect that the protocol is secure against a person-in-the-middle attack. Unfortunately that is incorrect: the protocol is vulnerable to an identity misbinding attack. An attacker can cause the protocol to terminate successfully where Bank thinks it is talking to Alice, but Alice thinks she is talking to the attacker.

Describe the attack. What does the attacker do? Recall that the attacker can block messages and replace them with its own messages.

Answer:
Alice sends the message $(g^a, cert_{alice}, S_{alice}(g^a))$ to the bank, bank receives that message. Bank replied to that message $(g^b, cert_{bank}, S_{bank}(g^a, g^b))$. The attacker intercepts the message and change the message to be $(g^{b'}, cert_{evil}, S_{evil}(g^a, g^{b'}))$ and sends it to Alice. Alice and bank both derive key k and k' respectively. At this point the protocol terminate successfully.
From the bank's perspective, it receives Alice's public key and certificates, and derives $kH(g^{ab}, alice)$, it is exactly the same when it is talking to Alice.
From Alice perspective, she receives attackers public key and certificates, and then derives $k' = H(g^{ab'}, alice)$, so Alice thinks she is talking to the attacker.

a. WARNING: I don't fully understand this problem. (Did we cover this?) I assume this refers to Section 18.6.1 (page 740 in B&S).

   To start, the verifier V receives a public key p from the prover P. The prover, of course, has a secret key s. Our protocol is as follows:

   - V chooses a random r of length ℓ and sends a (maybe long) challenge $c = E(p, r)$ to P
   - P computes $t = D(s, c)$ and sends t back to V
   - V checks that $t = r$

   If, at the end of Attack Game 18.3, an adversary is given cipher text c of a message $m \in \{0,1\}^\ell$, in a secure encryption scheme, the best they can do is guess m randomly. Thus, their likelihood of success is $\frac{1}{2^\ell}$.

b. WARNING: I don't fully understand this problem.

   Anyway, from above, at the end of Attack Game 18.3, the verifier sends $c = E(p, r)$ to where $r \in \{0,1\}^\ell$. However, also has p. Therefore, they can try to encrypt every bit string in $\{0,1\}^\ell$. One of these, of course, will be r. Thus, will come upon $E(p, r) = c$. They can then send r to P.

   We brute force 2^ℓ options, as required.

   See page 438 of B&S. looks good.="?