CSE 5014: Cryptography and Network Security
2024 Spring Semester Written Assignment # 4
Due: May 28th, 2024, please submit at the beginning of class
Sample Solutions

Q.1 Let (Gen, H) be a collision-resistant hash function with inputs of arbitrary size. We define a MAC for arbitrary-length message by

$$Mac_{s,k}(m) = H^s(k||m).$$

Show that this is not a secure MAC if H is constructed by the Merkle-Damgård transform from an arbitrary collision-resistant hash function h. (Assume that s is known to the attacker)

Solution:

Let h be the collision-resistant hash function from which H is constructed by applying the Merkle-Damgård transform. We show that the MAC is not secure by constructing an adversary: we first query an arbitrary message m of length n and obtain

$$t = Mac_k(m) = H(k||m) = h(h(0^n||k)||m).$$

The adversary outputs $m' = m||t$ and $t' = h(t||t)$. Now it holds that

$$\begin{aligned} Mac_k m' &= H(k||m') \\ &= H(k||m||t) \\ &= h(h(h(0^n||k)||m)||t) \\ &= h(t||t) \\ &= t'. \end{aligned}$$

It follows that the adversary wins with probability 1, which is certainly not negligible.

□

Q.2

1. We say that a number $y \in \mathbb{Z}_n^*$ is a quadratic residue (QR) if $y = x^2$ for some $x \in \mathbb{Z}_n^*$. Prove that the set of QRs is a subgroup of $\mathbb{Z}_n^*$.

2. Let $p > 1$ be a prime. It can be shown that $(\mathbb{Z}_p^*, \times_p)$ is a cyclic group, that is, there exists a generator $g \in \mathbb{Z}_p^*$ such that $\mathbb{Z}_p^* = \{g^1, g^2, \ldots, g^{p-1}\}$. For $y \in \mathbb{Z}_p^*$, let $\log_g(y)$ denote the smallest nonnegative integer i for which $g^i = y$. For example $\log_g(1) = 0$, and $\log_g(g) = 1$. Show that y is a QR in $\mathbb{Z}_p^*$ if and only if $\log_g(y)$ is an even number.

Solution:

1. Denote the set of QRs by S. For two elements $a, b \in S$, i.e., $a = x^2$ and $b = y^2$ for some $x, y \in \mathbb{Z}_n^*$, we then have $a \cdot b = x^2y^2 = (xy)^2 \in S$. The closure property is proved.

   Clearly, for $a, b, c \in S$, we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$. And, $1 = 1^2 \in S$ is the identity elements.

   It then suffices to prove the existence of the inverse element for each element in S. For $a = x^2 \in S$, we have $x^2 \cdot x^{-2} = (x \cdot x^{-1})^2 = 1 \in S$, which means $a^{-1} = x^{-2} \in S$.

2. (Note: There are various ways to prove that $\mathbb{Z}_p^*$ is a cyclic group, which is equivalent to prove that there is an element in $\mathbb{Z}_p^*$ whose order is exactly $p - 1$) Suppose that $y \in S$, the set of QRs. This means $y = x^2$ for some element $x \in \mathbb{Z}_p^*$. For a fixed generator $g \in \mathbb{Z}_p^*$, suppose that $x = g^i$ for a certain i with $0 \leq i \leq p - 2$. Then we have $y = x^2 = g^{2i}$, and equivalently, $\log_g(y) = 2i$, which is an even number.

   It remains to prove the "if" part. If $\log_g(y) = 2k$, i.e., $y = g^{2k}$. Then $y = (g^k)^2 = x^2 \in S$, where $x = g^k$.

$\square$

Q.3

Formally define the CDH assumption. Prove that hardness of the CDH problem relative to $\mathcal{G}$ implies hardness of the discrete-logarithm problem relative to $\mathcal{G}$, and that hardness of the DDH problem relative to $\mathcal{G}$ implies hardness of the CDH problem relative to $\mathcal{G}$.

Solution: We first formally define what it means for the CDH problem to be hard.

The CDH **experiment** $\mathrm{CDH}_{\mathcal{A},\mathcal{G}}(n)$:

(a) Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q (with $\|q\| = n$), and g is a generator of $\mathbb{G}$.

(b) Choose $h_1, h_2 \leftarrow \mathbb{G}$.

(c) $\mathcal{A}$ is given $\mathbb{G}, q, g, h_1, h_2$, and outputs $h_3 \in \mathbb{G}$.

(d) The output of the experiment is defined to be 1 if $h_3 = \mathrm{DH}_g(h_1, h_2)$, and 0 otherwise.

As described, it is unclear how to test whether $h_3 = \mathrm{DH}_g(h_1, h_2)$ efficiently. However, by choosing uniform h_1 with $\log_g h_1$ known (which can be done by choosing uniform $x_1 \in \mathbb{Z}_q$ and setting $h_1 := g^{x_1}$) it becomes easy to perform.

DEFINITION We say that the CDH problem is hard relative to $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathrm{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1\right] \leq \mathrm{negl}(n).$$

Assume the CDH problem is hard relative to $\mathcal{G}$. Let $\mathcal{A}$ be a probabilistic show how $\mathcal{A}$ can be used by a probabilistic polynomial-time algorithm $\mathcal{A}'$ to solve the CDH problem with success probability $\varepsilon(n)$: **Algorithm $\mathcal{A}'$:**
The algorithm is given $\mathbb{G}, q, g, h_1, h_2$ as input.

(a) Run $\mathcal{A}(\mathbb{G}, q, g, h_1)$ and obtain output x_1.

(b) Return $h_2^{x_1}$.

Observe that the input given to $\mathcal{A}$ when run as a subroutine by $\mathcal{A}'$ is distributed exactly as in experiment $\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n)$. So with probability $\varepsilon(n)$ it holds that $x_1 = \log_g h_1$. Whenever this occurs, $\mathcal{A}'$ outputs the correct answer $\mathrm{DH}_g(h_1, h_2)$. Since the CDH problem was assumed to be hard, this implies that $\varepsilon(n)$ is negligible and so the discrete logarithm problem must be hard as well. We now proceed to prove that the hardness of the DDH problem relative to $\mathcal{G}$ implies the hardness of the CDH problem relative to $\mathcal{G}$. Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm and set $\varepsilon(n) \overset{\text{def}}{=} \Pr\left[\mathrm{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1\right]$. We show how $\mathcal{A}$ can be used by a probabilistic polynomial-time algorithm $\mathcal{A}'$ to solve the DDH problem with roughly the same probability:

Algorithm $\mathcal{A}'$:
The algorithm is given $\mathbb{G}, q, g, h_1, h_2, h_3$ as input.

(a) Run $\mathcal{A}(\mathbb{G}, q, g, h_1, h_2)$ and obtain output h'_3.

(b) If $h'_3 = h_3$ output 1; else output 0.

We need to analyze the probability that $\mathcal{A}'$ outputs 1 when the final component of its input h_3 is equal to $\mathrm{DH}_g(h_1, h_2)$, as compared to the probability that it outputs 1 when h_3 is chosen uniformly at random from G. The input given to $\mathcal{A}$ when run as a subroutine by $\mathcal{A}'$ is distributed exactly as in experiment $\mathrm{CDH}_{\mathcal{A},\mathcal{G}}(n)$. So with probability $\varepsilon(n)$, it holds that $h'_3 = \mathrm{DH}_g(h_1, h_2)$. It follows that when $h_3 = \mathrm{DH}_g(h_1, h_2)$, algorithm $\mathcal{A}'$ outputs 1 with probability at least $\varepsilon(n)$.

On the other hand, when h_3 is chosen uniformly at random from $\mathbb{G}$, then regardless of the behavior of $\mathcal{A}'$ the probability that $h'_3 = h_3$ is exactly $1/|G| = 1/q$. Thus,

$$\begin{aligned}|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \\ \geq \varepsilon(n) - 1/q\end{aligned}$$

Since the above must be negligible by the assumption that the DDH problem is hard, and $1/q$ is negligible, it follows that $\varepsilon(n)$ must be negligible. We conclude that the CDH problem is hard as well.

$\square$

Q.4

The discrete logarithm problem is easy in $(\mathbb{Z}_N, +_N)$ for any integer N and for any generator. Explain this.

Solution:

The discrete logarithm problem in $\mathbb{Z}_N$ is that: given a generator g and $y = xg \bmod N$, find x. Since g is a generator, we have $\gcd(g, N) = 1$. Thus, g has an inverse modulo N, and we can use the extended Euclidean algorithm to get the inverse g^{-1} of g. The linear congruential equation can be thereby solved to get x.

$\square$

Q.5

Consider the cyclic group $\mathbb{Z}^*_{17} = \{1, 2, \ldots, 16\}$ and the mapping f defined by $f(x) = x^2 \bmod 17$ for all x in the group.

1. What is the size of the image set of f, i.e., the set $S = \{f(x) : x \in \mathbb{Z}_{17}^*\}$?
2. How many generators are there in $\mathbb{Z}_{17}^*$?
3. Pick a generator g. What is the probability that, for a randomly chosen $a, b \in \{0, 1, \ldots, 15\}$, the value of g^{ab} is in S?

Solution:

1. $|S| = 8$, i.e., squaring is a 2-to-1 mapping over $\mathbb{Z}_{17}^*$.
2. This is equivalent to count the number of elements in $\mathbb{Z}_{17}^*$ whose order is 15. The number is $\phi(\phi(17)) = \phi(16) = 8$.
3. The probability is 3/4.

$\square$

Q.6

When p and q are distinct odd primes and $N = pq$, the elements in $\mathbb{Z}_N^*$ have either 0 or 4 square roots. A quarter (1/4) of the elements have 4 square roots; the rest have no square root. The four square roots of $x \in \mathbb{Z}_N^*$ look like $\pm a, \pm b$ (of course, $-a$ means $N - a$ since we always work modulo N). Suppose that you are given an efficient deterministic algorithm A that, on input x that has square roots, finds some square root. (If x does not have a square root, it returns $\perp$.)

Use A to make an efficient *probabilistic* algorithm A' that factors N. (Hint: If you can find two square roots of a number, call them a and b, which are not of the form $a = \pm b \bmod N$, then you can factor N. Show how.] **Note**: you only get to call A as a black-box, so you don't know a *priori* which of the square roots it will find.

Solution:

Consider the following algorithm A'. On input N, it picks $x \leftarrow_R \mathbb{Z}_N^*$, and computes $y \leftarrow x^2 \bmod N$. (Note that sampling from $\mathbb{Z}_N^*$ is effectively done by sampling from $\mathbb{Z}_N$, because if you manged to find an x that wasn't in $\mathbb{Z}_N^*$, then you could factor N immediately) It then runs $z \leftarrow A(y)$. If $z = \pm x$ then it samples a new x and repeats the process; it does this until $z \neq \pm x$. Once this loop is broken, we know that $z^2 = x^2 \bmod N$, or $z^2 - x^2 = 0 \bmod N$.

By simple factoring this gives $(z-x)(z+x) = 0 \bmod N$ and since $z \neq \pm x$ we know that neither factor is zero. But then it must be the case that $\gcd((z-x) \bmod N, N)$ is one of the two factors of N, and the other is found immediately. Thus, A' can factor N in this way.

As for efficiency, we know that $A(y)$ is some fixed values, but we don't know which a priori. Since two of the four square roots of y "work" for us, the probability that $A(y)$ returns one of these is 1/2. Thus, A' requires only two samples on average.

□

Q.7 Describe a man-in-the-middle attack on the Diffie-Hellman protocol where the adversary shares a key k_A with Alice and a (different) key k_B with Bob, and Alice and Bob cannot detect that anything is wrong.

Solution:

The man-in-the-middle attack consists of simply running independent executions of the protocol with Alice and with Bob. (I.e., the adversary plays the role of Bob when interacting with Alice, and plays the role of Alice when interacting with Bob.) This results in a key k_A output by Alice and a key k_B output by Bob, both of which are known to the adversary. Since the adversary ran the protocol honestly with each party, neither party can detect that anything is amiss.

□

Q.8 Consider the following key-exchange protocol:
(a) Alice chooses uniform $k, r \in \{0,1\}^n$, and sends $s := k \oplus r$ to Bob.
(b) Bob chooses uniform $t \in \{0,1\}^n$, and sends $u := s \oplus t$ to Alice.
(c) Alice computes $w := u \oplus r$ and sends w to Bob.
(d) Alice outputs k and Bob outputs $w \oplus t$.

Show that Alice and Bob output the same key. Analyze the security of the scheme (i.e., either prove its security or show a concrete attack).

Solution:

Alice outputs k, while Bob outputs

$$w \oplus t = (u \oplus r) \oplus t = ((s \oplus t) \oplus r) \oplus t = (((k \oplus r) \oplus t) \oplus r) \oplus t = k.$$

The scheme, however, is not secure. Given a transcript (s, u, w) of an execution of the protocol, an adversary can compute $s \oplus u \oplus w$ and this is equal to the key since

$$s \oplus u \oplus w = (k \oplus r) \oplus u \oplus (u \oplus r) = k.$$

$\square$

Q.9 Show that the regular RSA signature scheme is *arbitrarily forgeable* (forging the signature of any challenge message m) if the attacker is allowed to ask the signing oracle. Note that the challenge message m cannot be queried to the signing oracle. (Recall that the RSA signature is $m^d \bmod N$, where d is the private key and $N = pq$)

Solution:
We forge the RSA signature σ of any challenge message m by querying the signing oracle the message $m' = m \cdot r^e \bmod N$, where $r \in_R \mathbb{Z}_N^*$ is chosen randomly. The signing oracle will return the signature $\sigma' = m'^d = m^d \cdot r \bmod N$. Then, we can compute the signature $\sigma = \sigma'/r = m^d \bmod N$.

$\square$

Q.10 Recall the El Gamal encryption scheme: the public key is (p, g, h), where g is a generator of $\mathbb{Z}_p^*$ and $h = g^x$, and the private key is x; the encryption scheme is $Enc(m) = (g^y, h^y \cdot m)$, where $y \leftarrow_R \mathbb{Z}_p^*$; the decryption scheme is $Dec(c_1, c_2) = c_2/c_1^x$. The El Gamal signature scheme is: To sign on a message m, $k \leftarrow_R \mathbb{Z}_p^*$ with $\gcd(k, p-1) = 1$,

$$\sigma = Sign_{sk}(m) = (r, s) = (g^k, k^{-1}(m - rx) \bmod (p-1)).$$

To verify a signature $\sigma = (r, s)$, it is accepted if $h^r r^s = g^m$.

(1) Show that El Gamal encryption scheme is *not* secure against the chosen ciphertext attack.

(2) Is El Gamal signature scheme secure against the chosen message attack (allowing to ask the signing oracle) if the *hash-and-sign paradigm* is used.

(3) Assume that the hash-and-sign paradigm is *not* used. Can we forge a signature for any given message m by asking the signing oracle. Note that you cannot ask the oracle about the signature of m.

Solution:

(1) If such a oracle exists, then Eve who wants to decrypt the ciphertext $c = (c_1, c_2)$, with $c_1 = g^y$ and $c_2 = h^y \cdot m$, chooses random elements k' and m' and gets oracle to decrypt $c' = (c_1 \cdot g^{y'}, m \cdot m' \cdot h^{y+y'})$. Oracle send mm', the plaintext of $c' = (g^{y+y'}, mm'h^{y+y'})$ to Eve. Eve simply divides by m' and obtains the plaintext m of c.

(2) When El Gamal signature used without a hash function ,it is existential forgeable as discussed in the slides. El Gamal signature scheme is secure against the chosen message attack if a hash function h is applied to the original message, and it is the hash value that is signed. Thus, to forge the signature of a real message is not easy. Adversary Eve has to find some meaningful message m' which $h(m') = m$. If h is collision-resistant hash function, her probability of success is negligible.

(3) We can query the oracle for any message except m. Therefore, we design a forger algorithm as follows.

(i) Query the oracle for message m', where $m/m' = u \bmod (p-1)$. (Oracle returns $(r = g^k \bmod p, s = k^{-1}(m' - rx) \bmod (p-1)$.

(ii) Compute $s' = su \bmod (p-1)$ and r' such that $r' \equiv ru \bmod (p-1)$ and $r' \equiv r \bmod p$.

(iii) Now we check the verification step:

$$h^{r'} r^{s'} = h^{ru} r^{su} = (h^r r^s)^u = (g^{m'})^u = g^m.$$

(iv) Return (m, r', s').

□