

Quiz 1 Review

Problem 1-1. Warm-Up [Fall 2024 Quiz 1]

- (a) Setting a task of recognizing the existence of objects (e.g., bridges) in a set of images before creating a user account is used to try to prevent bots from creating thousands of bogus accounts. An attacker using crowd sourcing to solve object recognition is an example of:

- An insufficient security goal.
- An insufficient attack model.
- A buggy implementation.

Solution: An insufficient attack model.

- (b) T/F Every one-way function is collision resistant.

Solution: False

- (c) T/F A secure signature scheme (existentially unforgeable against adaptive chosen message attacks) must be randomized.

Solution: False

- (d) T/F A secure digital signature scheme provides privacy and authenticity of the message being signed.

Solution: False

- (e) Consider the MAC scheme defined by $\text{MAC}(K, M) = \text{AES}(K, M)$.

Which of the following assumptions on AES is sufficient to make the MAC secure (existentially unforgeable against chosen message attacks)

- Collision resistant hash function
- One-way function
- Pseudorandom function

Solution: Pseudorandom function

Problem 1-2. Message Authentication Recall that in class we mentioned the hash-then-MAC paradigm, which uses a MAC for messages of fixed length n (such as AES, in which case $n = 128$), and a hash function with range $\{0, 1\}^n$, to MAC messages of arbitrary length, by first hashing the message to an element in $\{0, 1\}^n$ and then applying the underlying MAC to the hash value.

In class we mentioned that for the resulting MAC to be secure the hash function must be collision resistant, and thus this paradigm cannot be used with AES since its domain consists of messages of length 128, and there does not exist a collision resistant hash function with 128-bit security that maps to tags of bitlength 128, due to the Birthday Paradox.

In this problem we consider using the hash-then-MAC paradigm with AES but with a hash function with a *secret* seed. In particular, consider the following hash function H : It takes as input a (secret) seed $(a_1, \dots, a_t)$ and a message $(M_1, \dots, M_t)$, where each a_i and M_i is a block of 128 bits. The hash function outputs

$$H((a_1, \dots, a_t), (M_1, \dots, M_t)) = \sum_{i=1}^t a_i M_i.$$

The arithmetic is done over a finite field of size 2^{128} (known as the Galois Field and denoted by $\text{GF}[2^{128}]$), though how the arithmetic is done is not important, the only important thing is that the output is in $\{0, 1\}^{128}$, and that in every (finite) field (and in particular in $\text{GF}[2^{128}]$) every non-zero element has an multiplicative inverse.

- (a) Define the MAC scheme obtained by applying the hash-then-MAC paradigm to the hash function $H : (\{0, 1\}^{128})^t \times (\{0, 1\}^{128})^t \rightarrow \{0, 1\}^{128}$ and AES as the underlying MAC.

Solution:

$$\text{MAC}'((K, (a_1, \dots, a_t)), (M_1, \dots, M_t)) = \text{MAC}(K, H((a_1, \dots, a_t), (M_1, \dots, M_t)))$$

- (b) 1. Recall the attack for the hash-then-MAC using AES and a hash function without a secret seed (assuming the message space is $(\{0, 1\}^{128})^t$ for $t \geq 2$). *Hint:* This attack takes time roughly 2^{64} .

Solution: The attack is to generate random messages $M_1, \dots, M_T \in (\{0, 1\}^{128})^t$ until there exists $i \in [T - 1]$ such that $H(M_T) = H(M_i)$. Note that H can be computed by the adversary since it does not have a secret seed! By the birthday paradox $T \approx 2^{64}$, and since the messages were chosen at random from a very large message space ($t \geq 2$) then with overwhelming probability $M_T \neq M_i$, in which case the adversary can query the signing oracle for a signature of M_i , and output this signature as a valid signature for M_T .

2. Explain why this attack fails if we use the hash function above (with a secret seed), assuming the attacker sees many fewer than 2^{64} tags. **Solution:**

When the hash function has a secret seed the adversary does not know if $H(M_T) = H(M_i)$, and hence cannot execute the above attack.

3. Is this hash-then-MAC scheme (with H as defined above and AES) secure if the attacker gets to see more than 2^{64} tags for messages of his choice?
Hint: Use the fact that AES is injective (i.e., if $\text{AES}(K, M) = \text{AES}(K, M')$ then $M = M'$), and the fact that H is a linear function, together with the fact that one can efficiently solve t linear equations with t variables (using Gaussian elimination).

Solution: No! An adversary can query the signing oracle to sign many messages until he finds t collisions; namely, until he finds t message pairs $\{M_i, M'_i\}_{i \in [t]}$ such that $M_i \neq M'_i$ and

$$\text{AES}(K, H((a_1, \dots, a_t), M_i)) = \text{AES}(K, H((a_1, \dots, a_t), M'_i)).$$

This implies that $H((a_1, \dots, a_t), M_i) = H((a_1, \dots, a_t), M'_i)$, which reveals one linear equation about the secret $a_1, \dots, a_t$. Repeating this t times we get t linear equations about $a_1, \dots, a_t$, and these linear equations are likely to be independent. As a result the adversary can find the secret seed $a_1, \dots, a_t$ using Gaussian elimination. Once the adversary finds the seed he can break the scheme as above.

Here's a cleverer break that only requires one collision due to Adam Zheng from Fall 2021's 6.S060.

Suppose that the CMA adversary obtained 2^{64} tags, and it found a collision. As noted in the above solution, suppose we have found two messages $M_1 \neq M_2$ which cause a collision, i.e.,

$$\text{AES}(k, H(a, M_1)) = \text{AES}(k, H(a, M_2)) \implies H(a, M_1) = H(a, M_2)$$

This implies $a * M_1 = a * M_2$ (where $*$ is the dot product), and thus $a * (M_1 - M_2) = 0, \text{mod } 2^{128}$.

Now have the CMA adversary pick any message M_3 (not already submitted to the oracle) which has tag $\text{AES}(k, H(a, M_3))$. Then, $M_3 + (M_1 - M_2)$ also has that tag, since $H(a, M_3 + (M_1 - M_2)) = a * (M_3 + (M_1 - M_2)) = a * M_3 + a * (M_1 - M_2) = a * M_3 = H(a, M_3)$ by linearity of H . Since $M_1 \neq M_2$, $M_3 + (M_1 - M_2) \neq M_3$.

Problem 1-3. Message Signing

Let $(\text{Gen}, \text{Sig}, \text{Ver})$ be a signature scheme with message space $\{0, 1\}^k$ (where k is the security parameter), and let H be a seeded hash function with domain $\{0, 1\}^*$ and range

$\{0, 1\}^k$. Consider the new signature scheme $(\text{Gen}', \text{Sig}', \text{Ver}')$, with message space $\{0, 1\}^*$, defined via the following “hash-then-sign” paradigm:

- Gen' runs Gen to generate a pair (sk, vk) and samples a seed s for H . It outputs $sk' = (sk, s)$ and $vk' = (vk, s)$.
 - Sig' takes as input a secret key $sk' = (sk, s)$ and a message M , and outputs a $\text{Sig}(sk, H_s(M))$, i.e., it signs the hashed message $H_s(M)$.
 - Ver' , given the verification key $vk' = (vk, s)$, a message M , and a signature σ , outputs 1 if and only if $\text{Ver}(vk, H_s(M), \sigma) = 1$.
- (a) Suppose that $(\text{Gen}, \text{Sig}, \text{Ver})$ is secure (existentially against adaptive chosen message attack) then which of the following properties of H do we need to ensure that $(\text{Gen}', \text{Sig}', \text{Ver}')$ is also secure?
1. One-wayness.
 2. Target collision resistance
 3. Collision resistance.

Solution: Collision resistance. If an adversary can find two different messages M_1, M_2 such that $H_s(M_1) = H_s(M_2)$ then it can ask the oracle to sign M_1 and this signature is a valid signature also for M_2 . Note that one-wayness and target collision resistance do not suffice, since it may be easy to find collisions in such functions. If H is collision resistant then the resulting signature scheme is secure since an adversary that asks (adaptively) to sign polynomially many messages $M_1, \dots, M_t$, and obtains signatures $\sigma_i = \text{Sig}(sk, H_s(M_i))$ for every $i \in [t]$, cannot forge a signature to a new message for the following reason: Suppose he forges a signature to a new message M^* . If $H_s(M^*) \neq H_s(M_i)$ for every $i \in [t]$ then this adversary can be used to break the security of the original signature scheme, and if $H_s(M^*) = H_s(M_i)$ for some $i \in [t]$ then this adversary found a collision thus breaking the collision resistance assumption.

Problem 1-4. RSA [Fall 2023 Midterm]

Let N be an RSA modulus with public exponent $e = 3$ and private exponent d (i.e., $ed \equiv 1 \pmod{\phi(N)}$). The full-domain-hash signature scheme uses a hash function $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$. The scheme computes the signature on a message $m \in \{0, 1\}^*$ as:

$$\sigma \leftarrow H(m)^d \pmod{N}.$$

- (a) Consider the full-domain-hash signature scheme instantiated with some hash function H . Say that an attacker can find two messages $m_0, m_1 \in \{0, 1\}^*$, such that $H(m_0) = H(m_1)^2 \in \mathbb{Z}_N^*$. Explain how the attacker can use (m_0, m_1) to win the signature security game.

Solution:

- The attacker asks for a signature on m_0 . This is the value $\sigma = H(m_0)^d$.
- The attacker outputs $\sigma^* \leftarrow \sigma^2 \in \mathbb{Z}_N^*$ as its forged signature.

The forged signature σ^* is valid since

$$(\sigma^*)^e = ((H(m_0)^d)^2)^e = (H(m_0)^2)^{ed} = H(m_0)^2 \in \mathbb{Z}_N^*.$$

- (b) Model the hash function $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ as a truly random function. How many times would an attacker have to evaluate the hash function H , on average, to find messages $m_0, m_1 \in \mathbb{Z}_N^*$ such that $H(m_0) = H(m_1)^2 \in \mathbb{Z}_N^*$.

Solution: The analysis here is similar to the Birthday Problem. The average number is around $\sqrt{N}$.

Additional Problems

Problem 1-5. Hash Function Properties

Let $h: \{0, 1\}^{\leq 2n} \rightarrow \{0, 1\}^n$ be a hash function that is collision resistant. Let $h': \{0, 1\}^{\leq n+1} \rightarrow \{0, 1\}^{n+1}$ be the hash function given by the rule

$$h'(x) = \begin{cases} 0||x & \text{if } x \in \{0, 1\}^n \\ 1||h(x) & \text{otherwise} \end{cases}$$

- (a) Prove that h' is not one-way.

Definition 1 A function $f: X_n \rightarrow Y_n$ is said to be one-way if for every efficient adversary $\mathcal{A}$, the probability that $\mathcal{A}$, on input n and $y = f(x)$ for a random $x \in X_n$, outputs any x' such that $f(x') = y$, is negligible.

Solution: The modified hash function h' is not one-way, since for any hash value y of the form $0||x$, a preimage is x . Therefore, we can find a preimage for at least one half of all possible hash values.

- (b) Prove that h' is collision resistant.

Definition 2 A function $f: X_n \rightarrow Y_n$ is said to be collision resistant (CR) if for every efficient adversary $\mathcal{A}$, the probability that $\mathcal{A}$ on input n , outputs any distinct $x, x' \in X_n$ such that $f(x) = f(x')$, is negligible.

Solution:

Next, we prove that h' inherits collision resistance from h . We show that if we can find a collision for h' , then we can easily do so for h . Suppose

$$\exists x_0 \neq x_1 : h'(x_0) = h'(x_1)$$

We have two cases:

- (a) First bit of $h'(x_0)$ is 0. Impossible as implies $x_0 = x_1$.
- (b) First bit of $h'(x_0)$ is 1. Then, $h(x_0) = h(x_1)$ a contradiction, as h is collision resistant.

- (c) Prove that h' is target collision resistant if h is target collision resistant. For this problem part, assume that h has the form $h : \{0, 1\}^{\leq n+1} \rightarrow \{0, 1\}^n$.

Definition 3 A function $f : X_n \rightarrow Y_n$ is said to be target collision resistant (TCR) if for every efficient adversary $\mathcal{A}$, the probability that $\mathcal{A}$ on input n and a random $x \in X_n$, outputs $x' \in X_n$ such that $x' \neq x$ and $f(x) = f(x')$, is negligible.

Solution: Given an algorithm $\mathcal{A}$ that breaks TCR for h' , we construct an algorithm $\mathcal{B}$ that breaks TCR for h , which is a contradiction.

The algorithm $\mathcal{B}$ operates as follows, given a random TCR challenge $x \in \{0, 1\}^{\leq n+1}$:

- If $|x| = n$, output FAIL.
- Otherwise, run $x' \leftarrow \mathcal{A}(n, x)$.
- Output x' .

We first argue that, whenever $\mathcal{B}$ does not fail, it breaks the TCR property of h . Algorithm $\mathcal{A}$ produces a string x' such that $h'(x) = h'(x')$ and $x \neq x'$. Since h' is length-preserving, we know that $|x| = |x'| \neq n$. Then, by the definition of h' it holds that $1 \| h(x) = 1 \| h(x')$, which implies that $h(x) = h(x')$. Since $x \neq x'$, x' is a valid solution to the TCR challenge.

To complete the argument, we need only to show that $\mathcal{B}$ does not fail often. The algorithm $\mathcal{B}$ only outputs FAIL when the TCR challenge x has length n . (Algorithm $\mathcal{B}$ also fails if $\mathcal{A}$ fails, but the probability that $\mathcal{A}$ fails is non-negligible by assumption.) The probability of this bad event is at most

$$\frac{2^n}{(2^0 + 2^1 + 2^2 + \dots + 2^{n-1} + 2^{n+1})} \leq \frac{2^n}{2^{n+2}} \leq \frac{1}{4}.$$

Therefore algorithm $\mathcal{B}$ breaks the TCR of h with probability better than negligible and we are done.

Problem 1-6. Pseudo-Random Functions

Let F be a pseudorandom function (PRF) that takes messages in $\{0, 1\}^n$ to messages in $\{0, 1\}^n$.

- (a) We wish to use F to construct a PRF that takes messages in $\{0, 1\}^{2n}$ to messages in $\{0, 1\}^{2n}$. Suppose the key to F is also in $\{0, 1\}^n$.

Below are four proposals of such a PRF, where $x_0, x_1, K, K_0, K_1 \in \{0, 1\}^n$ and where we use $||$ to denote concatenation. Notice that the constructions 1, 3, and 4 use a key in $\{0, 1\}^n$ whereas the second construction uses a key in $\{0, 1\}^{2n}$.

1. $G_1(K, x_0||x_1) = F(K, x_0)||F(K, x_1)$.
2. $G_2(K_0||K_1, x_0||x_1) = F(K_0, x_0)||F(K_1, x_1)$.
3. $G_3(K, x_0||x_1) = F(K', 0^n)||F(K', 1^n)$, where $K' = F(F(K, x_0), x_1)$
4. $G_4(K, x_0||x_1) = F(F(K, x_0), x_1)||F(F(K, x_0), x_1 \oplus 1^n)$.

Only one of the above four proposals is a secure PRF. Which one is the secure one? For each of the three others, show an attack that distinguishes it from a truly random function (recall the definition of a PRF given in the lecture).

Solution: G_3 is the only PRF.

G_1 is not a PRF since by querying it on a single query of the form $x||x$ for any $x \in \{0, 1\}^n$, the outcome will have the form $y||y$ for $y \in \{0, 1\}^n$, as opposed to being truly random in $\{0, 1\}^{2n}$.

G_2 is not a PRF since by querying it on (x_0, x_1) and (x'_0, x_1) the two outcomes will not look random (jointly) since the last n bits will be the same in both!

G_4 is not a PRF since by querying it on $x_0||x_1$ and $x_0||x_1 \oplus 1^n$ one can distinguish the output from uniform.