In this lecture we mainly discuss data integrity. We begin by briefly recalling CBC mode. Next, we started talking about Message Authentication Codes and Unpredictability vs Psuedorandomness. We then discussed PRFs as fixed length MACs and difference with variable length MACs. We briefly started HMACs for the sake of the hw due the following week.

1 Data Integrity

1.1 CBC mode

CBC (Cipher Block Chaining) mode just happens to provides some integrity protections, but we need something that specifically protects integrity.

1.2 What is Data Integrity?

Message integrity protects a message from being changed by an attacker. In data integrity, both Bob and Alice have a key, but their new goal is to make modification to their message detectable. A non-solution to this problem is to use parity checks. Parity checks do not attain our new goal because they are used to check when an error was made randomly, not maliciously. We need a way to tell if an attacker, Mallory, changed our message between Bob and Alice.

2 Message Authentication Schemes

2.1 Message Authentication Codes

![Figure 1: Alice and Bob with MACs](image-url)
Definition 1. A message authentication code (or MAC) is a tuple of probabilistic polynomial-time algorithms $(\text{Gen}, \text{Mac}, \text{Vrfy})$ such that:

1. Gen: A key-generation algorithm that takes as input the security parameter $1^n$ and outputs a key $k$ with $|k| \geq n$.

2. Mac: The tag-generation algorithm that takes as input a key $k$ and a message $m \in \{0,1\}^*$, and outputs a tag $t$. Since this algorithm may be randomized, we write this as $t \leftarrow \text{Mac}_k(m)$.

3. Vrfy: The verification algorithm that takes as input a key $k$, a message $m$, and a tag $t$. It outputs a bit $b$, with $b = 1$ meaning valid and $b = 0$ meaning invalid. We assume without loss of generality that Vrfy is deterministic, and so write this as $b := \text{Vrfy}_k(m, t)$.

2.2 Existentially Unforgeability against Chosen-Message-Attack

Definition 2. A message authentication code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ is existentially unforgeable under an adaptive chosen-message attack, or just secure, if for all probabilistic polynomial-time adversaries $A$, there exists a negligible function $\text{negl}$ such that:

$$\Pr[\text{Mac} - \text{forge}_A, \Pi(n) = 1] \leq \text{negl}(n)$$

Is a PRF a good MAC? YES!

3 Unpredictable vs. Pseudorandom

Unpredictable consists of something difficult to compute where being pseudorandom, is when two things are difficult to tell apart. Unpredictability is easier to achieve since it is harder for the attacker to compute something correctly, rather than just having to guess two things apart. It requires more effort from the attacker and less from the owner.

3.1 One Way Functions and Permutations

Definition 3. A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is one-way if the following two conditions hold:

1. (Easy to Compute) There exists a polynomial-time algorithm $M_f$ computing $f$; that is, $M_f(x) = f(x)$ for all $x$.

2. (Hard to invert) For every probabilistic polynomial-time algorithm $A$, there exists a negligible function $\text{negl}$ such that

$$\Pr[\text{Invert}_A, f(n) = 1] \leq \text{negl}(n)$$

Definition 4. Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be length-preserving, and let $f_n$ be the restriction of $f$ to the domain $\{0,1\}^n$ (i.e., $f_n$ is only defined for $x \in \{0,1\}^n$, in which case $f_n(x) = f(x)$). A one-way function $f$ is called a one-way permutation if for every $n$, the function $\{n\}$ is a bijection.

Given $x, y = \text{OWF}(x)$ is easy to compute. But given $y$, getting $x$ s.t. $y = \text{OWF}(x)$ is difficult to compute (unpredictable). However, given $\text{OWF}(x)$, $x$ is NOT pseudorandom. But you can build pseudorandomness from unpredictability.
3.2 Variable-length MAC

PRFs can be used for MACs, but they do not satisfy EU-CMA if variable lengths are allowed. Therefore we need solutions that can work for more practical uses with variable length messages.

1. Apply the pseudorandom function (block cipher) to the length \( l \) of the input message in order to obtain a key \( k_l \) (i.e., set \( k_l := F_k(l) \)). Then, compute the basic CBC-MAC using the key \( k_l \). This ensures that different (and computationally independent) keys are used to authenticate messages of different lengths.

2. Prepend the message with its length \( |m| \) (encoded as an \( n \)-bit string), and then compute the basic CBC-MAC on the resulting message.

3. Change the scheme so that key generation chooses two different keys \( k_1 \leftarrow \{0,1\}^n \) and \( k_2 \leftarrow \{0,1\}^n \). Then, to authenticate a message \( m \) first compute the basic CBC-MAC of \( m \) using \( k_1 \) and let \( t \) be the result; output the tag \( \hat{t} := F_{k_2}(t) \).

4 Keyed Hash Functions

With hash functions, it is difficult to find collisions (according to Pigeonhole principle, there will be collisions, no matter how rare, if the domain is larger than the range).

Definition 5. A Hash function \( \Pi = (Gen, H) \) is collision resistant if for all probabilistic polynomial-time adversaries \( A \) there exists a negligible function \( \text{negl} \) such that

\[
\Pr[Hash \text{-- col}_A, \Pi(n) = 1] \leq \text{negl}(n)
\]

4.1 HMAC

Definition 6. Let \((\widetilde{Gen}, H)\) be a variable-length collision-resistant hash function. Let \( IV, opad, \) and \( ipad \) be fixed constants of length \( n \). HMAC defines a MAC as follows:

1. Gen: on input \( 1^n \), run \( \widetilde{Gen}(1^n) \) to obtain a key \( s \). Also choose \( k \leftarrow \{0,1\}^n \) at random. Output the key \((s,k)\).

2. Mac: on input a key \((s,k)\) and a message \( m \in \{0,1\}^* \) of length \( L \), output the tag

\[
t := H^s_{IV}((k \oplus opad)||H^s_{IV}((k \oplus ipad)||m))
\]

3. Vrfy: on input a key \((s,k)\), a message \( m \in \{0,1\}^* \), and a tag \( t \), output 1 if and only if \( t = \hat{t} := Mac_{s,k}(m) \).
4.2 Types of Hash Functions

1. *Universal*: This way is the hardest for the attacker because $k \in 0, 1^\lambda$ is given after the Attacker has guessed $x_0$ or $x_1$.

2. *One way Universal*: This way is in between because $k \in 0, 1^\lambda$ is given after $x_0$ is guessed, but before $x_1$ is guessed.

3. *Collision Resistant*: This way is easiest for the attacker because $k \in 0, 1^\lambda$ is given before the Attacker guesses $x_0$ or $x_1$.

References