1 A quick summary of the course so far, and plan for today's lecture

- We started the semester talking about encryption schemes that are secure against 'read-only' adversaries, and saw how to build these encryption schemes using PRFs/PRPs. Here is the sequence of topics that we looked at:
  - perfect security - it's perfect, but requires very large keys
  - no-query semantic security - computational security, guarantees security against comp. bounded adversaries.
  - semantic security for bounded-length messages - can be constructed from PRFs
  - semantic security for general messages - can also be built from PRFs
- Next, we discussed that in order to handle stronger adversaries, we need tools that can guarantee integrity of the ciphertext — cryptographic primitives that ensure that the adversary has not tampered with the ciphertext. In order to do this, we started discussing MACs, a cryptographic object that is interesting in itself, and has applications beyond encryption.
  - MACs for bounded-length messages - can be built from PRFs
  - MACs for unbounded-length messages - can also be built from PRFs. In fact, we can even ensure that the signature is 'short'. We discussed two approaches for this: a counter-based approach, and a cipher-block chaining approach.

Today, we will discuss a third approach for signing unbounded-length messages. This approach uses a new cryptographic tool that has immense applications beyond signing/encryption. This new tool, *collision-resistant hash functions*, will be our focus for the next two lectures.

2 A third approach for signing long messages

Suppose $\mathsf{MAC}_{\mathrm{bdd}}$ is a MAC scheme that can sign n-bit messages. A natural idea is to compress our long message into a short, n-bit digest that can be signed using $\mathsf{MAC}_{\mathrm{bdd}}$. Clearly, the security of this approach depends on what we use for compressing the long message into a short digest. We saw a few approaches that do not work.

- Break the message into n-bit chunks, and compute the XOR of all chunks. This does not work because given a signature on $(m_1, m_2, \ldots, m_\ell)$, the same signature will also be a valid signature on $(m_2, m_1, m_3, \ldots, m_\ell)$.

More concretely, we want a compressing function $H : (\{0,1\}^n)^\ell \to \{0,1\}^n$ such that it is *hard to find two inputs that map to the same output.* If we have such a compressing function, then we can sign long messages as follows: first apply the compressing function, then sign the compressed string. Such compressing functions are called 'hash functions' in cryptography, and this approach of signing is called the *hash and sign* approach.

3 Hash functions

Hash functions are one of the most widely used objects in cryptography, and even outside cryptography. A word of caution: you may have encountered hash functions in a data structures/algorithms course. The goals are quite similar (in a hashmap, you want to 'hash' elements from a universe to a smaller space). However, the parameters and terminologies are different, and might cause some confusion. Therefore, I would recommend that you approach this object with a fresh mindset.

Intuitively, a hash function is an efficient deterministic function that must compress inputs, and it must be hard to find two inputs that map to the same output. Two inputs that map to the same output are called a 'collision'.

However, these two requirements are contradictory in some sense - given any deterministic function $H : \{0,1\}^\ell \to \{0,1\}^n$, if $\ell > n$, then there exist two inputs x_1, x_2 that map to the same output. And given the description of H, there always *exists* an adversary $\mathcal{A}_H$ that has x_1, x_2 hardwired, and simply outputs (x_1, x_2) as a collision. To address this issue, we will need a keyed family of hash functions. [1]

Formally, a hash function family is a set of deterministic functions $\mathcal{H} = \{H_k\}_{k \in \mathcal{K}}$, where each function H_k is indexed by a key k belonging to some key space. The hash function has an input space $\{0,1\}^\ell$, an output space $\{0,1\}^n$ where $\ell > n$. This key is often referred to as **salt**, especially in the computer security literature.

3.1 Security: Does the adversary know the key?

Informally, the hash function is secure if no p.p.t. adversary can find a collision — two inputs that map to the same output. In order to formally define security, we must address the following immediate question: does the adversary know the hash key? Depending on whether the adversary gets the key or not, we end up with very different cryptographic objects.

3.1.1 Universal hash functions: when the adversary DOES NOT know the key

There are many scenarios where the adversary does not know the key used for hashing, and must still find a collision. Consider the 'hash-then-sign' approach for signing long messages. Here, the salt/key used for hashing can be part of the MAC key. Let us consider the following security game, which is perhaps the most basic security game in this setting. For simplicity of notation, I am skipping the dependence on the security parameter. Strictly speaking, the key space is an infinite sequence of sets, where we have a different key space for every security parameter n.

UHF

1. **Setup:** Challenger chooses a UHF key k (**does not** send it to $\mathcal{A}$).
2. **Collision:** The adversary sends two messages $m_0, m_1 \in \mathcal{M}$, and wins if $H_k(m_0) = H_k(m_1)$.

Figure 1: Security Game for UHFs

Definition 21.01. A hash function family $\mathcal{H} = \{H_k\}_{k \in \mathcal{K}}$ with key space $\mathcal{K}$ and message space $\mathcal{M}$ is said to be a secure universal hash function if for any p.p.t. adversary, there exists a negligible function $\mu(\cdot)$ such that for all n

$$\Pr\Big[\mathcal{A} \text{ wins the UHF game (Figure 1)}\Big] \leq \mu(n).$$

Note: the output of the UHF does not necessarily hide the key!

3.1.2 Collision resistant hash functions: when the adversary knows the key

There are many natural scenarios where it makes sense to allow the adversary to have the hash key in the security game. For instance, later in the course, we will talk about digital signature schemes, which are very similar to MACs, except that the verification algorithm does not use the secret key that's used for signing. In such a scenario, we cannot use a UHF if we want to 'hash-and-sign'. A collision resistant hash function family guarantees that it is hard to find collisions, even if the adversary knows the key.

There are other natural 'real-world' applications, one of which we will discuss briefly next class.

[1] Note that this is a 'definitional issue'. In real-world, we do have keyless hash functions. And while there exist adversaries that can efficiently output a collision, it is 'hard' to find such adversaries. In the next lecture, we will discuss these real-world keyless hash functions too, and how we can make sense of the security guarantees that they offer.

CRHF

1. **Setup:** Challenger chooses a UHF key k and sends it to $\mathcal{A}$.
2. **Collision:** The adversary sends two messages $m_0, m_1 \in \mathcal{M}$, and wins if $H_k(m_0) = H_k(m_1)$.

Figure 2: Security Game for CRHFs

Definition 21.02. A hash function family $\mathcal{H} = \{H_k\}_{k\in\mathcal{K}}$ with key space $\mathcal{K}$ and message space $\mathcal{M}$ is said to be a secure collision-resistant hash function if for any p.p.t. adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all n,

$$\Pr\Big[\mathcal{A} \text{ wins the CRHF game (Figure 2)}\Big] \le \mu(n).$$

4 Constructing UHFs

Any MAC scheme with message space $\mathcal{M}$ and signature space $\mathcal{T}$ is also a UHF with input space $\mathcal{M}$, and output space $\mathcal{T}$ (assuming the messages are longer than the signatures). Similarly, given a PRF $F : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$, if the inputs are longer than the outputs, then F is also a UHF (with same input/ouptut/key space).

Somewhat surprisingly, **UHFs can be constructed unconditionally** (that is, without any computational assumptions)! A couple of constructions are given below. We did not discuss the first one in class, but since it is a fairly simple construction/proof, I might skip the detailed in-class discussion, and will take related queries (if any).

(Unconditional) Universal Hash Functions: Construction 1

Let $p = \Theta(2^n)$ be a prime. The input space for our hash function will be $(\mathbb{Z}_p)^*$, key space $\mathbb{Z}_p$ and output space $\mathbb{Z}_p$. Given a key $k \in \mathbb{Z}_p$, evaluation on $x = (x_0, x_1, \ldots, x_{d-1})$ is defined as

$$H_k(x) = x_0 + x_1 \cdot k + x_2 \cdot k^2 + \ldots + x_{d-1} \cdot k^{d-1} + k^d \mod p.$$

Theorem 21.01. Let $m_1 \in \mathbb{Z}_p^{\ell_1}$, $m_2 \in \mathbb{Z}_p^{\ell_2}$. Then

$$\Pr_{k \leftarrow \mathbb{Z}_p}[H_k(m_1) = H_k(m_2)] \le \max\{\ell_1, \ell_2\}/p.$$

Proof. The proof uses one simple idea: take any **non-zero** polynomial $w(\cdot)$ of degree d. Then this polynomial has at most d roots in $\mathbb{Z}_p$. That is, there are at most d numbers $u \in \mathbb{Z}_p$ such that $w(u) = 0$.

Fix any $m_1 \in \mathbb{Z}_p^{\ell_1}$, $m_2 \in \mathbb{Z}_p^{\ell_2}$, and for simplicity of notation, let us assume $\ell_1 = \ell_2 = \ell$. Consider the polynomial

$$w_{m_1,m_2}(z) = \sum_{i=0}^{\ell-1} (m_{1,i} - m_{2,i}) \cdot z^i.$$

This polynomial has degree at most $\ell - 1$, and therefore has at most $\ell - 1$ roots. Note that $H_k(m_1) = H_k(m_2)$ if and only if k is a root of w_{m_1,m_2}. Since there are at most $\ell - 1$ roots,

$$\Pr_{k \leftarrow \mathbb{Z}_p}[H_k(m_1) = H_k(m_2)] \le \frac{\ell}{p} = \mathsf{negl}(n)$$

□

Another neat and simple construction was proposed in class (and proven after class). I'm listing the construction below, you are encouraged to think about the proof.

(Unconditional) Universal Hash Functions: Construction 2

The input space for our hash function will be $(\{0,1\})^*$, key space $\{0,1\}^n$ and output space $\{0,1\}^n$. Given a key $k \in \{0,1\}^n$, interpret the key as a n-bit number. Interpret the input x also as a number, and output $x \mod k$ (this will be a number less than 2^n, and therefore can be represented using n bits).

5 Constructing CRHFs

Constructing CRHFs is far more challenging, and we don't have any CRHF constructions based on OWFs (and hence no CRHF constructions based on PRGs/PRFs). We tried the following few candidate constructions in class, and discussed why they are not secure.

- The CBC-MAC construction: Recall, the CBC-MAC is a keyed function which maps $(\{0,1\}^n)^\ell$ to $\{0,1\}^n$, using an n bit key. This is not a secure CRHF. Given the key (which is the PRF key in this case), one can compute $t_0 = F(0^n, k)$, and output $(m_0 = 0^n, m_1 = (0^n, t_0))$ as the collision. Note that both strings map to t_0.
- Consider a PRF whose input space is larger than the output space. This keyed function is compressing, but may not be a secure CRHF. That is, there exist keyed functions that are secure PRFs, but not secure CRHFs.

  Let $F : \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^n$ be a secure PRF, where $\ell > n$. Consider a keyed function F' which is defined as follows:

  $$F'(x,k) = \begin{cases} F(x,k) \text{ if } [x]_n \neq k & ([x]_n \text{ denotes the first } n \text{ bits of } x) \\ F(0^n,k) \text{ if } [x]_n = k \end{cases}$$

  The function F' is a secure PRF, assuming F is. However, it is not a secure CRHF. Given the key k, one can find two inputs that map to the same output.

Consider the following 'number-theoretic' attempt to build a secure hash function: let p be an n bit prime. The key space is $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$, input space is $\mathbb{Z}_p \times \mathbb{Z}_p$ (recall, $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ and $\mathbb{Z}_p = \mathbb{Z}_p^* \cup \{0\}$). The output space is $\mathbb{Z}_p^*$. Given a key $k = (x, y)$, the hash function is defined as follows:

$$H_k(a,b) \equiv x^a \cdot y^b \mod p$$

Clearly, the function maps two numbers in $\mathbb{Z}_p$ to a single number in $\mathbb{Z}_p^*$, and therefore is compressing. However, it is not a secure CRHF. Using Fermat's Little Theorem, we know that $x^{p-1} = 1 \mod p$ for all $x \in \mathbb{Z}_p^*$, and therefore $H_k(0, p-1) = H_k(p-1, 0) = 1$.

Towards the end of the lecture, one of the students asked if the above construction is secure if we remove 0 from the input space (that is, restrict ourselves to $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$). There are other potential vulnerabilities in this attempt, and even if we restrict the input space to be $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$, we need to be careful with our choice of key (x, y). We also need to be careful with our choice of prime p. For instance, suppose $p = 2q + 1$, where both p and q are prime. Then, for any choice of x and y, there exists a collision that can be found efficiently, even if the input space is $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$.

6 Lecture summary, plan for next lecture, additional resources

Summary: In this lecture, we introduced hash functions. We saw that the hash function must be 'keyed'. There are two well-studied notions of hash functions: universal hash functions (where the adversary must find a collision without seeing the key), and collision-resistant hash functions (where the adversary must find a collision after given a random key from the keyspace). Universal hash functions can be built information-theoretically, while CRHFs are harder objects to build. We saw a few failed attempts for building a CRHF.

Next lecture: Next lecture, we will first prove that a UHF can be combined with a PRF to obtain a secure MAC using the 'hash-and-sign' approach. Next, we will see how to extend the input domain of a CRHF. Finally, we will discuss some real-world and theoretical constructions of CRHFs.

Number theory results used in this lecture: Starting with this lecture, we will gradually introduce a few results from number theory. You are encouraged to understand the proofs of these theorems. However, for the purposes of this course, it suffices to know these results.

- (Fermat's Little Theorem) For any prime p and number $a \in \mathbb{Z}_p^*$, $a^{p-1} = 1 \mod p$. Moreover, $b = a^{p-1} \mod p$ is the unique number in $\mathbb{Z}_p^*$ such that $a \cdot b = 1 \mod p$.

- Let p be a prime. For any non-zero, degree-d polynomial $h(x) = h_0 + h_1 x + \ldots + h_{d-1}x^{d-1} + h_d x^d$ where each $h_i \in \mathbb{Z}_p$,
$$|\{w \in \mathbb{Z}_p : h(w) = 0 \mod p\}| \leq d.$$

Relevant sections from textbook [Boneh-Shoup]: Sections 7.1 and 7.2 contain the definitions and constructions of UHFs. Section 8.1 discusses the definition of CRHFs (the textbook refers to CRHFs as 'keyless hash functions', since the adversary knows the key; see discussion in Section 8.1). Appendix A of the textbook provides a summary of number theoretic results that will be useful for this course.

7 Questions

Question 21.01. Let p, q be n bit primes such that $p = 2q + 1$. Consider the following hash function $\mathcal{H}$, with key space and input space equal to $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$, and output space $\mathbb{Z}_p^*$. For a key $k = (x, y)$, the function $H_k(a, b) = x^a \cdot y^b \mod p$. Show that this is not a secure CRHF.

Hint: Consider any number $x \in \mathbb{Z}_p^*$. Let $\mathsf{ord}(x)$ denote the smallest integer greater than 0 such that $x^{\mathsf{ord}(x)} = 1 \mod p$. We know that $\mathsf{ord}(x) \leq p - 1$ (using Fermat's Little Theorem). One can also show that $\mathsf{ord}(x)$ must divide $p - 1$ (should be easy to prove, given Euclid's Extended Algorithm).
As a result, for any key $(x, y) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$, $\mathsf{ord}(x)$ and $\mathsf{ord}(y)$ must either be 1, 2, q or $2q$. For each of these cases, show that you can find a collision.