1 Hash Functions

Definition 1 (Merkle-Damgard Cascaded Hash Function Construction)

Merkle-Damgard construction is a particular method of constructing a hash function, where if the underlying compression function \( h \) is collision resistant, then the hash function \( H \), is also collision-resistant. The construction is depicted in Figure 1.

The idea is as follows. A message \( m \) is split into \( n \) blocks, \( m_1, ..., m_n \) (each block being of a fixed length), with appropriate padding whenever needed. The first block, and the IV, is passed as an input to a compression function \( h \) to yield the first chaining vector \( CV_1 = h(IV, m_1) \). \( CV_1 \) and \( m_2 \), are passed as input to \( h \) again to yield the second chaining vector \( CV_2 = h(CV_1, m_2) \). This chaining goes on until \( m_n \), and \( CV_{n-1} \), are passed to \( h \), which outputs \( CV_n = h(CV_{n-1}, m_n) \). This entire construction is denoted by \( H \), therefore the output of hash function becomes \( H(m) = CV_n \).

A key property of this construction is that it has an avalanche effect. In other words, even if a single bit in the original message \( m \) is modified, then \( H(m) \) becomes completely different and uncorrelated. Thus, any two messages which differ in only in a single bit, their hash values will be completely different. This will reduce the likelihood of finding collisions.

The length of a message to be hashed might not be a multiple of the block length. Therefore, the message needs to be padded with an appropriate number of bits. added to
data to maintain that property. In order to maintain the security (or collision resistance) of hash functions, one must make sure that appropriate padding is incorporated into the design. If appropriate padding is not used, then an attacker might be able to find collisions with less effort. In particular, the padded message, denoted by $pad(m)$, should satisfy the following properties. Note that $H(m)$ is basically equal to $H(pad(m))$.

**Properties of padding**

1. If $|m_1| = |m_2|$, then $|pad(m_1)| = |pad(m_2)|$.

2. If $|m_1| \neq |m_2|$, then at least the last block of $pad(m_1)$, and $pad(m_2)$ should be different.

To understand the second property: Let’s say $m_1 = [00]$, and $m_2 = [000]$, and the block length being 6. If we pad with all zeroes, then $pad(m_1) = [00][0000]$, and to $pad(m_2) = [000][000]$. The two messages will result in the same hash value, and a collision, even though the messages themselves are different. If we changed the last blocks of padding to be different, then this issue would not occur.

**Theorem 1 (If h is CR2, then H is CR2.)**

**Proof.** We prove the contrapositive: $\neg H$-CR2 $\Rightarrow \neg h$-CR2. That is, we prove if $H$ is not CR2, then $h$ is not CR2. In other words, we show if $\exists A$, which can find collisions in $H$, then we can construct $B$, which can find collisions in $h$.

Let’s say $A$ outputs two messages $m_1$, and $m_2$ ($m_1 \neq m_2$), such that $H(m_1) = H(m_2)$. Also, let’s assume that $pad(m_1) = [m_{1,1}, m_{1,2}, ..., m_{1,p}]$, where $m_{1,1}$ is first block of message $pad(m_1)$, $m_{1,2}$ is second block of message block $pad(m_1)$, etc., and $p$ is the number of blocks. Then, $pad(m_2) = [m_{2,1}, m_{2,2}, ..., m_{2,1}]$, where $q$ is the number of blocks in $pad(m_2)$.

**Case: 1 |$m_1$| $\neq |m_2|$**

Since $|m_1| \neq |m_2|$, then from the second property of padding, we have, $m_{1,p} \neq m_{2,q}$. Since the two messages collide, we have $CV_{1,p} = CV_{2,q}$. Therefore, $h(CV_{1,p-1}, m_{1p}) = h(CV_{2,q-1}, m_{2q})$. Since $m_{1,p} \neq m_{2,q}$, $(CV_{1,p-1}, m_{1p}) \neq (CV_{2,q-1}, m_{2q})$, and therefore $(CV_{1,p-1}, m_{1p})$ and $(CV_{2,q-1}, m_{2q})$ are two messages that yield collison on $h$ function.

**Case 2: |$m_1$| = |$m_2$|**

From the first property of padding, $|pad(m_1)| = |pad(m_2)|$. So, the last block might not necessarily be the same and might not yield a collision on $h$. However, since the two messages are indeed different, they will differ in at least one block, and this is what we will use to find the collison on $h$.

If $(CV_{1,p-1}, m_{1p}) \neq (CV_{2,q-1}, m_{2q})$, then $h(CV_{1,p-1}, m_{1p}) = h(CV_{2,q-1}, m_{2q})$, thus causing a collision. If $(CV_{1,p-1}, m_{1p}) = (CV_{2,q-1}, m_{2q})$, then maybe $(CV_{1,p-2}, m_{1p-1}) = \ldots$
(CV₂,q−2,m₂q−1) are same. Working backwards through the chain, eventually some pair must collide.

For i = 0, 1, ..., min(p, q) − 1
If (CVᵢ,p−1−i,mᵢ,p−i) ≠ (CV₂,q−1−i,m₂,q−i)
Return ̂M₁ = (CVᵢ,p−1−i,mᵢ,p−i), and ̂M₂ = (CV₂,q−1−i,m₂,q−i)

2 Message Authentication

Authentication is used to verify that a party is indeed who he/she claims to be. Some possible ways of providing authentication:

1. A sends message m, along with hash value H(m) to B. This is not adequate because anybody can hash any message.
2. A sends message m, along with Encₖ(m) to B. This will work, but for large messages this will be inefficient.
3. A sends message m, along with Encₖ(H(m)) to B. This is better, because authentication does not depend on size of m.

Note: Not every encryption yields message authentication. One-time pad is one such encryption scheme. If m, and m⊕k, is sent, then one can retrieve the key by m⊕(m⊕k).

Definition 2 (Message Authentication Code (MAC))

Data appended to the message, which allows us to verify the message.

Key Generation: k ←$ K.
MAC(): takes input message m, and key k. Therefore, σ = MAC(m, k). (m, σ) is sent as the output.
Verify: The receiver calculates the MAC for m, and checks it against the received σ.

2.1 Security Notions for Message Authentication

(I) Key Recovery (KR):
Given a set of valid messages and MAC pairs, (m₁, σ₁), ..., (mₙ, σₙ), it should be hard for an adversary to compute the key k. Note that the knowledge of key will allow
the adversary to produce MACs on any messages of its choice. Thus, every message authentication scheme should be secure against key recovery.

(II) No-Message Attack (NMA):

Given no message and corresponding MAC values, the adversary comes with a valid \((m, \sigma)\).

(III) Existential forgery under adaptive chosen message attacks (CMA):

The adversary has access to a “Signing Oracle”, and a “Verification Oracle” (see Figure 2 below). It first sends \(q\) adaptive queries to the signing oracle and obtains the corresponding MAC values. Based on the values queried and the responses received from the signing oracle, the adversary produces a new message \(m_{\text{new}}\), which differs from all other messages queried to Signing Oracle, and a valid MAC \(\sigma_{\text{new}}\) of \(m_{\text{new}}\). If the verification oracle accepts \((m_{\text{new}}, \sigma_{\text{new}})\) as a valid message and MAC pair, then the adversary is said to have created a valid forgery and broken the message authentication scheme. A message authentication scheme is said to be secure against the CMA attack if for all adversaries \(A\) playing such a game, \(\text{Adv}_{\text{MAC}}^{\text{CMA}}(A) \leq \epsilon\).

![Figure 2: Chosen Message Attack](image)

One can easily show that: \(CMA \Rightarrow NMA \Rightarrow KR\).