

# Cryptographic Hash Functions

Beyond CRCs

## Hash functions

- A hash function is a mathematical, efficiently computable function that has fixed size output:
  - $F : \{0, 1\}^N \rightarrow \{0, 1\}^n$ , where  $N > n$
  - $F : \{0, 1\}^* \rightarrow \{0, 1\}^n$
- In cryptography, the first type of hash function is often called a compression function, with the name hash function reserved for the unbounded domain type.

## Checksums and CRCs

- Used to provide integrity checks against **random faults**.
- **Not** sufficient for **protection against malicious or intentional modification**.
  - Easy to make changes and re-compute the CRC to match.
- In the past, it was believed that the use of CRCs within encryption was sufficient to provide integrity. However, that is no longer considered adequate:
  - Example: The use of CRCs in the WEP protocol resulted in a serious vulnerability, allowing for powerful active attacks.

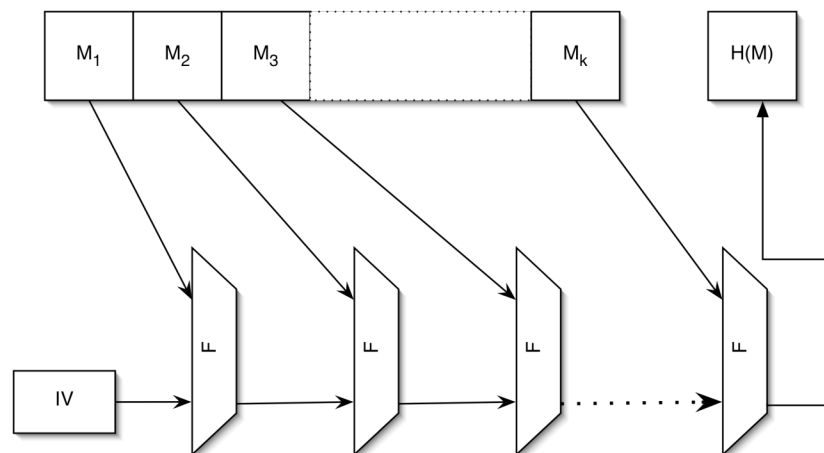
## Cryptographic hash functions

- There security of hash functions is defined empirically, if the following problems are found to be computationally infeasible:
  - One way:
    - Given  $y$ , find  $x$  such that  $h(x) = y$
  - Second pre-image resistant:
    - Given  $x$ , find  $y \neq x$  such that  $h(x) = h(y)$
  - Collision-resistant:
    - Find  $y, x$ , with  $y \neq x$  such that  $h(x) = h(y)$

## Constructing hash functions

- Since constructing secure hash functions is a difficult problem, the following approach has been taken in practice:
  - Construct a good compression function. Since the domain of compression functions are “small” they are easier to test for the desired properties.
- Use the MD construction (next) to turn a one-way, collision-resistant compression function into a hash function with similar properties.

## Merkle-Damgard (MD)



## Applications of Hash Functions

- System integrity protection:
- For password verification, eliminating the need to keep passwords
- As building blocks for **message authentication codes** (MACs) and **digital signature** algorithms.

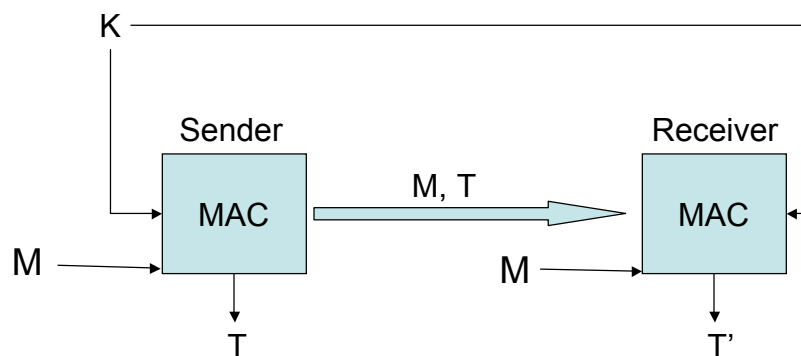
## Message-extension attacks

- Since most hash functions are built using the Merkle-Damgard construction, they are vulnerable to length-extension attacks:
  - Given  $h(M)$  and length of  $M = \text{len}(M)$  and adversary can find  $h(M \parallel M')$  for chosen  $M'$ .
  - To prevent this, MAC should be used instead of hashes to provide integrity of message transmission.

## Message Authentication Codes

- Message authentication codes, like block ciphers, are *symmetrically-keyed* cryptographic primitives.
- Like cryptographic hash functions, MACs take arbitrary-length input and produce a fixed length output.
- Not invertible.
- Require a key.

## Using MACs



## Building MACs from hash functions

- HMAC is a MAC from hash functions.
- Let  $h$  be the hash function. Assume
- $L$  = block length of compression function input
- Let  $K$  be the key,  $K'$  be the key padded with 0's to length  $L$ .
- $\text{HMAC}(K, M) = h(K' \oplus \text{opad} || h(K' \oplus \text{ipad} || M))$
- $\text{ipad} = 0x363636\dots3636$
- $\text{opad} = 0x5c5c5c\dots5c5c$  (both of length  $L$ )