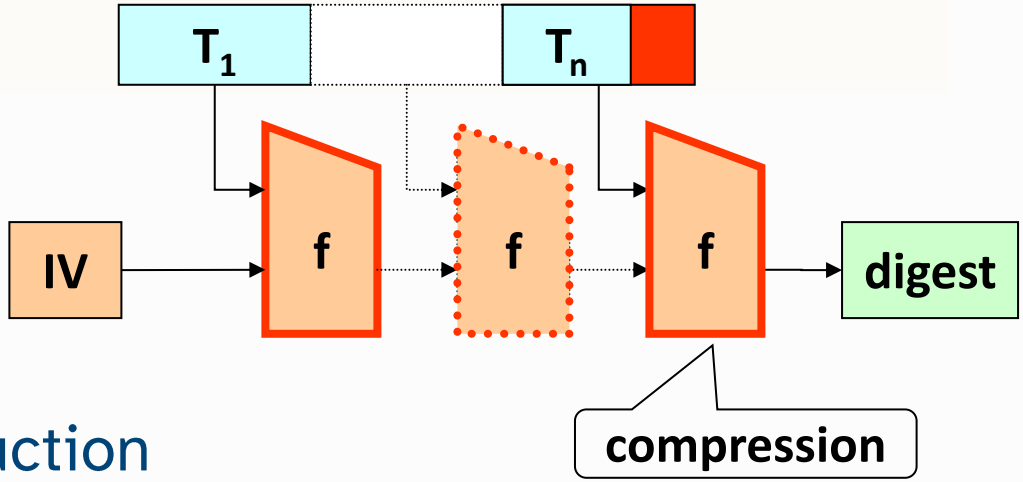
Cryptographic Hashing (digest functions)

Digest functions

- Give a fixed-length value from a variable-length text
  - Sort of text “fingerprint”
- Produce very different values for similar texts
  - Cryptographic one-way hash functions
- Relevant properties:
  - Preimage resistance
    - Given a digest, it is infeasible to find an original text producing it
  - 2nd-preimage resistance
    - Given a text, it is infeasible to find another one with the same digest
  - Collision resistance
    - It is infeasible to find any two texts with the same digest
    - Birthday paradox

Digest functions: aproaches



- Merkle-Damgård construction
  - Iterative compression
  - Collision-resistent, one-way compression functions
  - Length padding (1, followed by zeros, followed by length)

- Sponge functions
  - **Absorption**: update a finite internal state (entropy pool) from a variable-length, padded input stream
  - **Squeezing**: produce an arbitrary-length output from the internal state

Digest functions: common algorithms

- MD4 (128 bits)
  - Still used, but very easy to break
- MD5 (128 bits)
  - Very easy to find collisions!
  - Disclaimer: it can be used when collisions are not an issue
- SHA-1 (Secure Hash Algorithm, 160 bits)
  - Also no longer secure … (collisions found in 2017)
- RIPEMD (128 and 160)
- HAVAL (128, 160, 192, 224, 256)
- SHA-2 family (SHA-256, SHA-348, SHA-512)
- SHA-3 family (SHA3-224, SHA3-256, SHA3-384, SHA3-512)
- Blake2s (128, 160, 192, 224, 256)
- Blake2b (160, 256, 384, 512)

Digest functions: what are they good for?

- To produce robust or secure checksums
  - Message Integrity Codes (MIC)
  - Message Authentication Codes (MAC)
- To produce fingerprints of documents
  - Acts as a document “identity”
  - Useful for signing purposes
- To implement key derivation processes
  - Password → key
  - Key → key
- To implement cryptopuzzles
  - E.g. Bitcoin mining

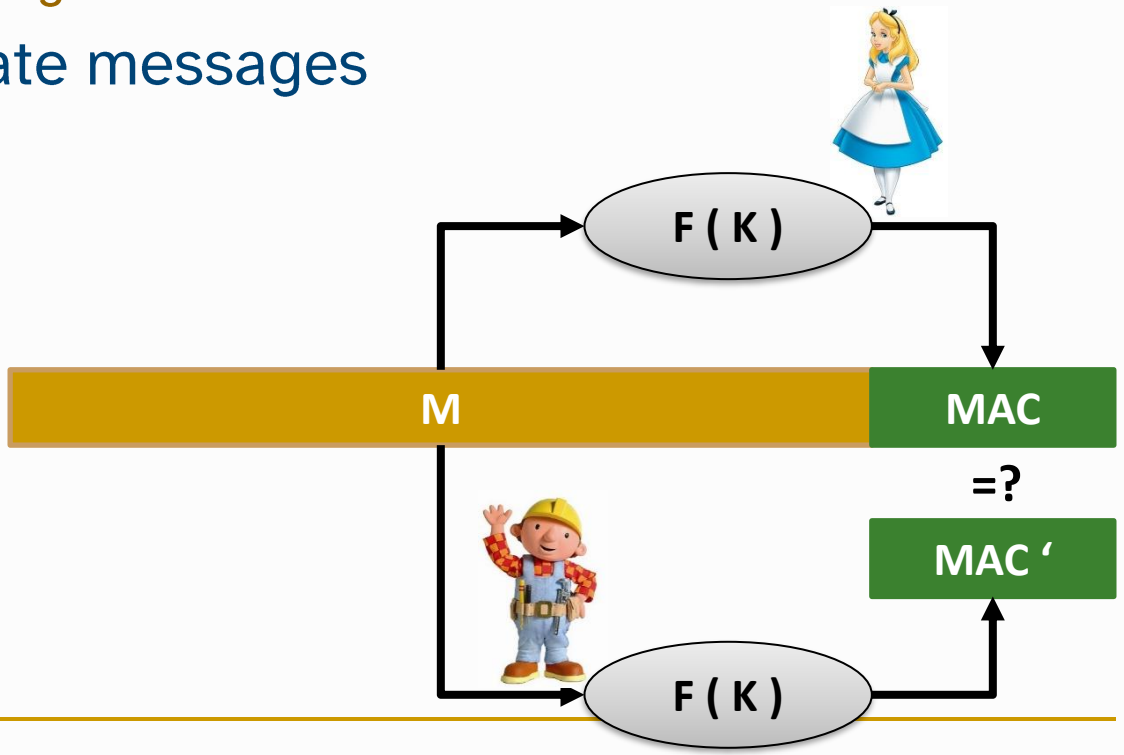
Message Integrity Code (MIC)

- Provide the capability to detect changes by devices
  - Communication/storage errors
  - From a random process or without control

- Send: Calculate MIC and send T + MIC
  - T = Text
  - MIC = digest(T)
- Receive: Receive data (T') and check if H(T) = MIC
  - Calculate MIC'=digest(T')
  - Validate if MIC' = MIC
- Does not protect from planned changes to the text
  - Attacker can manipulate T into T'' and calculate a new MIC''

Message Authentication Codes (MAC)

- Hash, or digest, computed with a key
  - Only key holders can generate and validate the MAC
- Used to authenticate messages
  - M' = M | MAC(M)



MAC functions: Hash-based approaches

- Adding a key to the hashed data
  - Keyed-MD5 (128 bits)
    - MD5(K, keyfill, text, K, MD5fill)
  - HMAC (Hashed-based MAC)
    - Generic construction, uses a hash function H
    - Output length depends on H
    - HMAC-MD5, HMAC-SHA, etc.

H (K, opad, H (K, ipad, text))

ipad = 0x36 B times

opad = 0x5C B times