Hashes (keyless)

Hashes

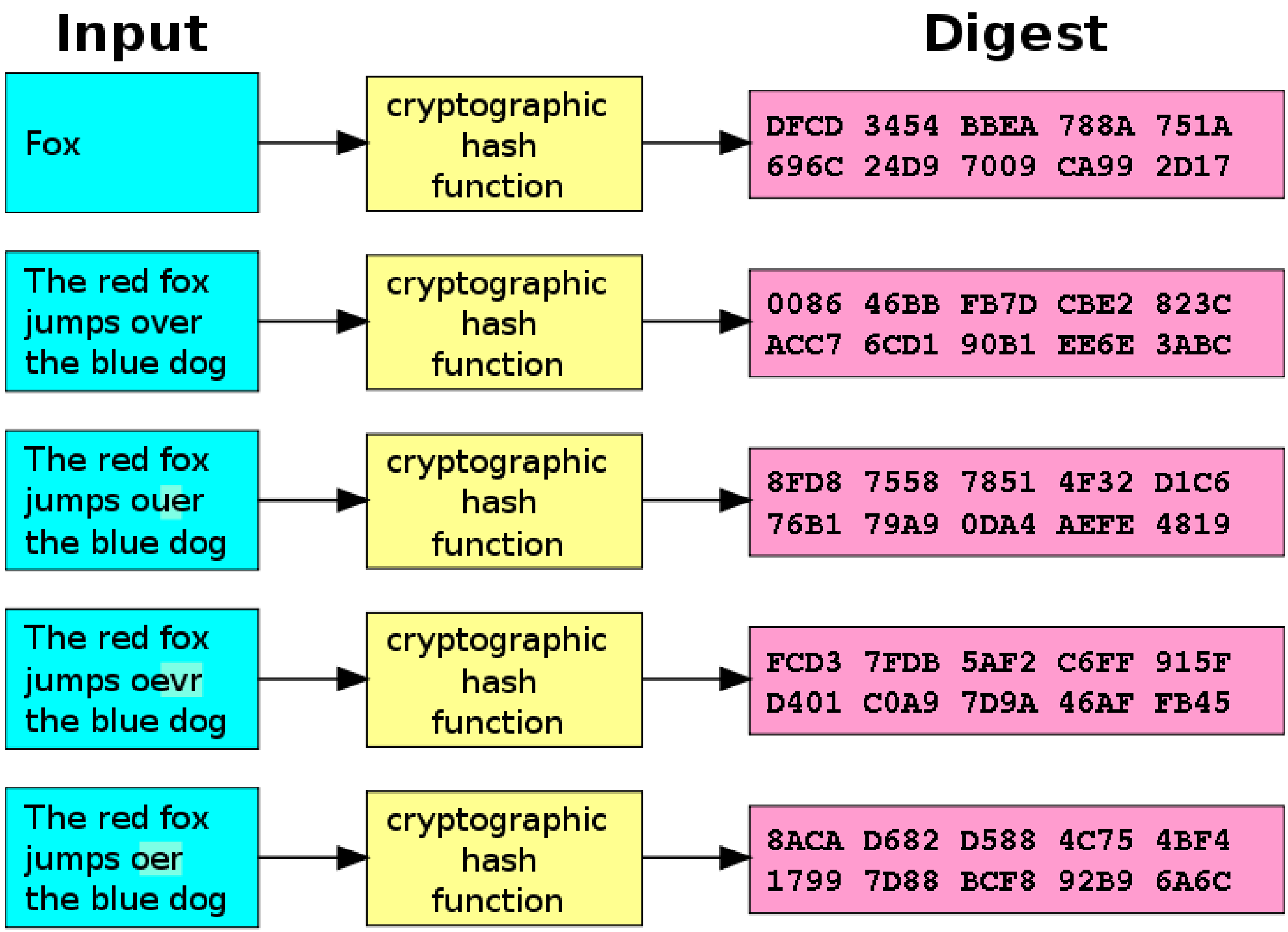
Hash -- Definition

A hash is an efficient function that takes an arbitrary length input and produces a fixed length output (digest, hash)
Let $H : \mathcal{M} \longrightarrow \mathcal{T}$ be a function

- $\mathcal{M}$ = message space
- $\mathcal{T}$ = digest space

Uses

- Suppose you want to check if Alice and Bob have the same version of some file (**File integrity**)
  - They compute $H(a), H(b)$
  - They check if $H(a) = H(b)$



Proprieties

- Pre-image Image Resistance
- Second Pre-image resistance
- Resistant to collisions

1. Pre-Image Resistance

The hash function must be a one way function. Given $t \in \mathcal{T}$ it is hard to find $m \in \mathcal{M}$ s.t $H(m) = t$

Intuition

- It should be unfeasable to reverse a hash function ($\mathcal{O}(2^l)$ time where l is the number of output bits)
- This propriety prevents an attacker to find the original message from a hash

2. Second Pre-Image Resistance

Given m it should be hard to find $m' \neq m$ with $H(m') = H(m)$

Attack game

- An adversary $\mathcal{A}$ is given a message m and outputs a message $m' \neq m$
- $\mathcal{A}$ wins the game if he finds $H(m) = H(m')$
- His advantage is $Pr[\mathcal{A} \text{ finds a second preimage}]$

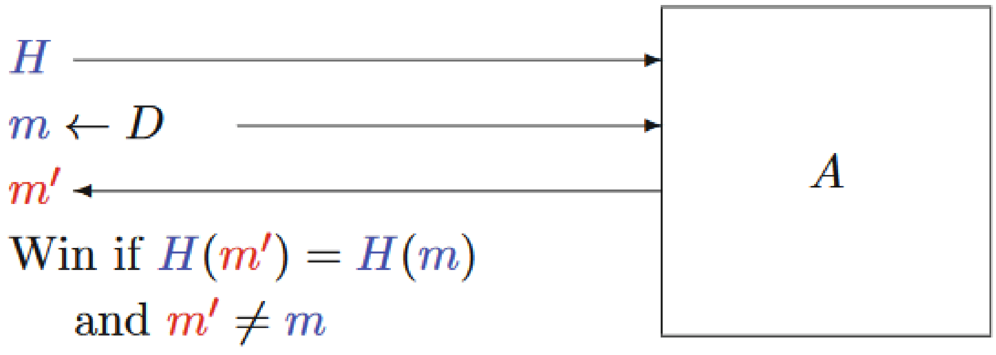


FIGURE 14.1. Security game for second preimage resistance

Remarks

- In practice a hash function with l bits output should need 2^l queries before one can find a second preimage
- This propriety prevents an attacker to substitute a message with another and get the same hash

3. Hash Collisions

Intuition - A hash collision happens when we have two different messages that have the same hash

Why do we care about hash collisions?

- Since hashes are used to fastly verify a message integrity if two messages have the same hash then we can replace one with another => We can play with data
- Now, we want to hash big files and big messages so $|\mathcal{M}| >> |\mathcal{T}|$ => It would appear that hash collisions are possible
- Natural collisions are normal to happen and we consider them improbable if $\mathcal{T}$ is big enough (SHA256 => T = $\{0, 1\}^{256}$)
- Yet, we don't want hash collisions to be computable
    - We don't want an attacker to be able to craft collisions or find collisions given a message

Let's throw some definitions

Attack game

- An adversary $\mathcal{A}$ outputs two messages $m_0 \neq m_1$
- $\mathcal{A}$ wins the game if he finds $H(m_0) = H(m_1)$
- His advantage is $Pr[\text{Adversary finds a collision}]$

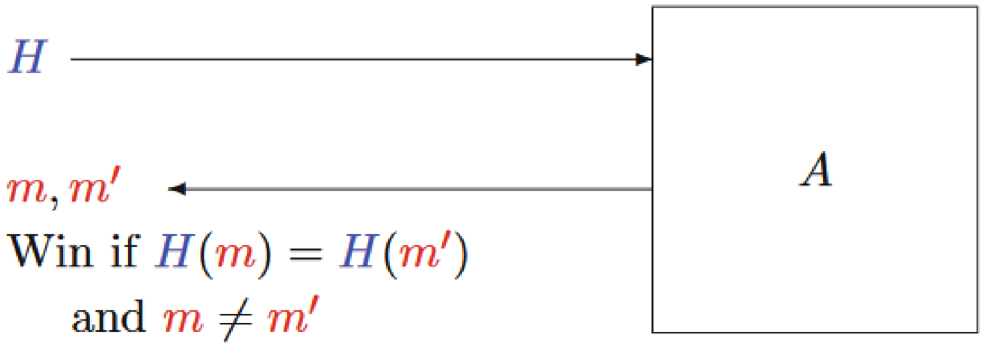


FIGURE 14.2. Security game for collision resistance of a function

Security

A hash function H is collision resistat if for all efficient and **explicit** adversaries the advantage is negligible

Intuition

- We know hash collisions exist (therefore an efficient adversary must exist) and that is easy to prove therefore we request an explicit algorithm that finds these collisions
- This propriety makes it difficult for an attacker to find 2 input values with the same hash

Difference from 2nd preimage

- There is a fundamental difference in how hard it is to break collision resistance and second-preimage resistance.
    - Breaking collision-resistance is like inviting more people into the room until the room **contains 2 people with the same birthday**.
    - Breaking second-preimage resistance is like inviting more people into the room until the room **contains another person with your birthday**.
- One of these fundamentally takes longer than theother

Implications

Lemma 1

> Assuming a function H is preimage resistant for every element of the range of H is a **weaker** assumption than assuming it is either collision resistant or second preimage resistant.

Note

- Provisional implication
- https://crypto.stackexchange.com/questions/10602/why-does-second-pre-image-resistance-imply-pre-image-resistance?rq=1
- https://crypto.stackexchange.com/questions/9684/pre-image-resistant-but-not-2nd-pre-image-resistant

Lemma 2

> Assuming a function is second preimage resistant is a **weaker** assumption than assuming it is collision resistant.

Resources

- https://www.youtube.com/watch?v=b4b8ktEV4Bg - computerphile
- https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm
- https://www.cs.ucdavis.edu/~rogaway/papers/relates.pdf - Good read for more details