HASH FUNCTIONS

DR. O. I. ADELAIYE

HASH FUNCTION ???

- A one-way function
  - A one-way transformation
    - If *h* is a hash function such that *y=h(x)*, then it is **computationally infeasible** for a user who has *h* and *y* to find x
  - Collision free
    - If *h* is a hash function such that *y=h(x)*, then it is **computationally infeasible** for a user who has *h* and *y* to find an x' such that *h(x')=y)*
  - Gives a fixed length output, whatever the input size is
    - MD5's is 128, SHA-1's is 160
  - The output is aka hash, digest or checksum.
    - MD5: Message Digest 5
    - SHA-1: Secure Hash Algorithm

FEASIBLE OR NOT??

- In theory, is it possible to find two strings with the same hash?
  - Yes ! Hash all possible 161 bit strings
- In practice?
  - Computationally infeasible !
- Is it possible to find m such that h(m)=H
  - In theory, yes !
  - In practice, computationally infeasible
- Is finding two strings with the same hash of the same difficulty as finding one string matching a particular hash value?
  - Is finding m, m’ such that h(m)=h(m’) as difficult as finding m such that h(m)=H?

THE BIRTHDAY PARADOX

- What is the probability of picking someone whose birthday date is the 22nd of October?
  - 1/365
- If you have a group of n persons, what is the probability of picking 2 persons who have the same birthday?
  - You pick a first person
  - The probability that the second person does not have the same date of birth is 364/365
  - The probability that the third person does not have the date of birth of any of the first two is 363/365
  - …
  - The probability that the nth person does not have the date of birth of any of the first (n-1) persons is (365-(n-1))/365
  - The probability that none of n persons have the same date of birth is

    $365xxxxx\ldots x(365-(n-1)/365$
  - In particular, for n=23 persons, the probability that they all have different birthdays is: $365xxxxx\ldots x(365-(22)/365$ = 0.4927
  - That is, the probability that two persons have the same date of birth in a group of 23 persons is: 1-0.4927=0.5073

BIRTHDAY PARADOX

- If you have 23 persons, two of them have the same date of birth with a probability higher than 0.5
  - More generally if you have n possible values, it is enough to have $\sqrt{n}$ samples of these values in order to get two equal values with a probability higher than 0.5
- Finding two messages with the same hash is similar to a Birthday problem !
  - Two messages Two individuals
  - Same hash Same birthday
- For SHA-1 (2^{160} possible hashes), it is enough to hash 2^{80} different strings in order to get a collision with a probability higher than 0.5
  - In the case of MD5 (2^{128} possible hashes), only 2^{64} hashes are required

HASH FUNCTION FOR MIC/MAC

INTEGRITY CHECK

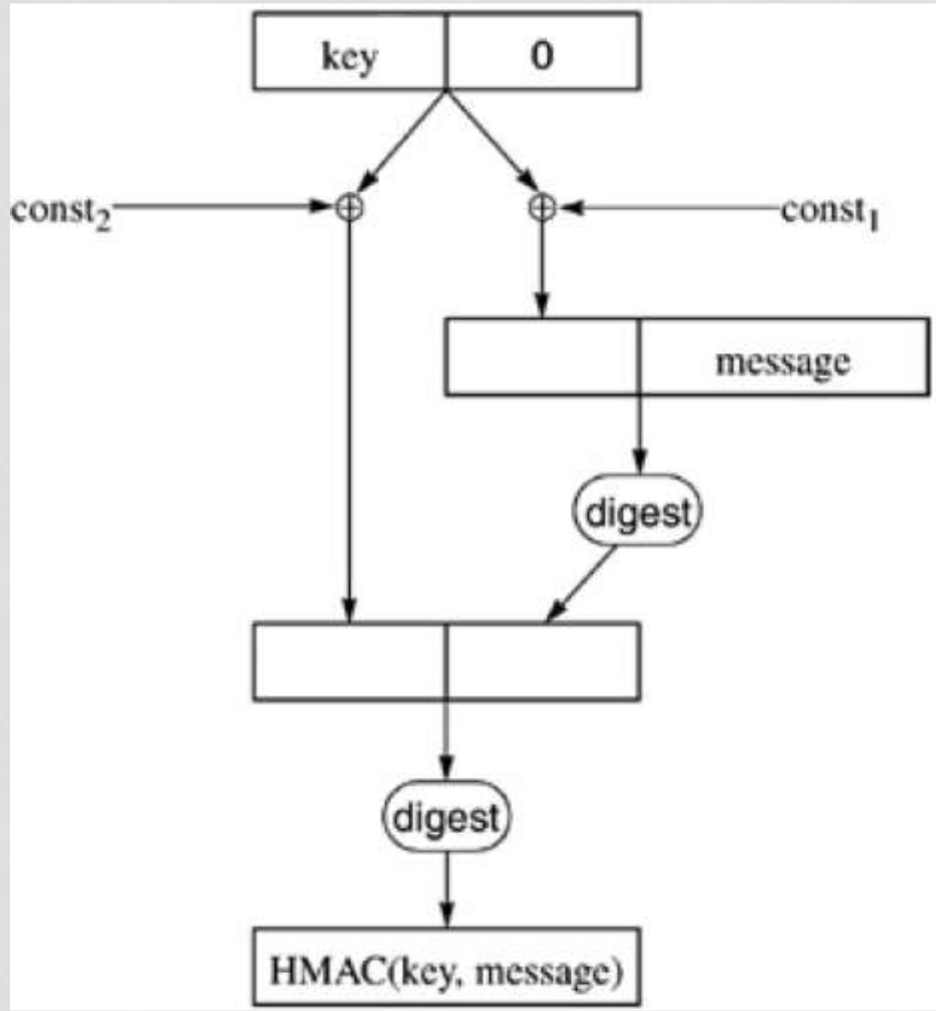
- Assume Alice and Bob want to exchange some messages with data integrity
  - Is hashing the message and sending the hash along with the message enough?
    - Must use a **keyed hash**
  - Is hashing the K | message enough?
    - K is a pre-shared secret
  - Problem: algorithms that compute the hash in an iterative way
    - Hash of the message up to chunk *n* can be calculated using the hash up to chunk *n-1*
    - As is the case with MD4, MD5, and SHA-1
    - H(K | message | forgery) can be calculated from h(K | message)

HASHING FOR INTEGRITY CHECK

- Possible solutions:
  - Use only a subset of the bits as a hash
    - So the attacker does not get hold of the full hash
  - Use h(message|K) instead of h(K|message)
    - But $h(m1)=h(m2) \Rightarrow h(m1|K)=h(m2|K)$ for the algorithms that iteratively calculate the hash
- Use h(K|message|K)

HMAC

- HMAC has two phases
  - Phase 1: compute h(K1|message) = H
  - Phase 2: compute h(K2|H)
  - K1 and K2 are derived from K
- HMAC pads the key to get 512bit key
  - If the key is longer than 512 bits, then the first 512 bits of the key are hashed, then the hash padded to 512 bits
- Two constants are XORed to the padded key for the different operations
- Does it solve the problem?



EN

D