Cryptographic Hash Functions & Message Authentication Codes

Cryptography - CS 411 / CS 507

Erkay Savaş

Department of Computer Science and Engineering
Sabancı University

October 25, 2019

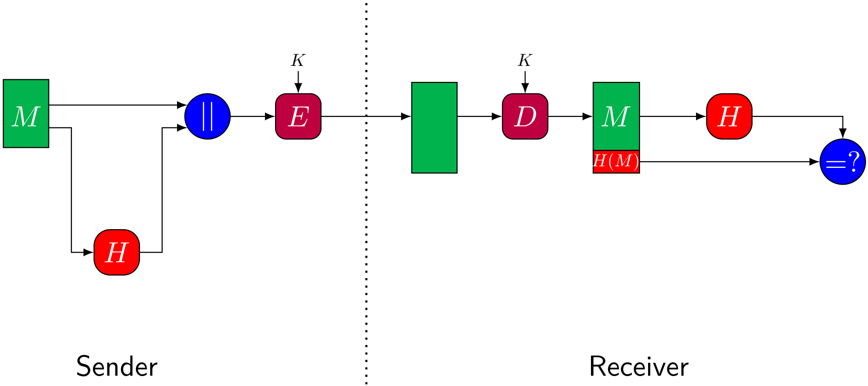
Outline

- Cryptographic Hash functions
- Message Authentication Codes (MAC)

Hash Functions

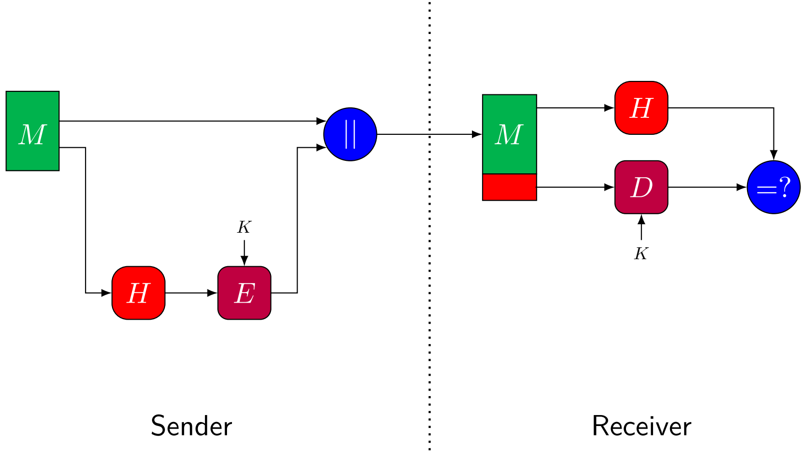
- One-way hash functions do not use secret key.
- A hash function accepts an arbitrary length message M and
  - produces a fixed-length output, referred as hash code, or shortly hash, $h = H(M)$;
  - message digest and hash value are also used.
- A message representative
  - A change to any bit (or bits) in the message results in a (big) change to the hash value.
- Hash functions are widely used in message authentication and digital signatures

Basic Use of Hash functions



- Both encryption and message authentication

- Message authentication only



Requirements for a Hash Function

- One-way property
  a) $H(x)$ is relatively easy to compute for any given x,
     - making both hardware and software implementations practical
  b) For any given value h, it is computationally infeasible to find x such that $H(x) = h$
- Weak collision resistance
  - For any given message x, it is computationally infeasible to find $y \neq x$ such that $H(x) = H(y)$
- Strong collision resistance
  - It is computationally infeasible to find <u>any</u> pair (x, y) with $x \neq y$ such that $H(x) = H(y)$

How Hard to Find Collision?

- Depends on the hash length, n-bit
- Ideally, after $2^{n/2}$ trials it is likely to find collisions due to "birthday attacks"

Collision Attacks on Hash Functions

- In 2004, many collisions were found for MD4, MD5, HAVAL-128, and RIPEMD
- MD5 collisions have been used to create two different and "meaningful" documents with the same hash.
- Lenstra et al. showed how to produce examples of X.509 certificates with the same hash

Secure Hash Standard (SHA-1)

- Proposed by NIST as standard hash function for certain US federal government applications (1995).
- The hash value is 160-bits.
- Five 32-bit chaining variables are used.
- Similar to DES, the chaining variables are processed in 20 rounds.
- For more information see `http://www.nist.gov` and Handbook of Applied Cryptography.

Cryptanalysis of SHA-1

- In Feb. 2005, Wang, Yin, and Yu announced that their attacks can find collisions requiring fewer than 2^{69} operations.
  - A brute-force search would require about 2^{80} operations due to "birthday attacks"
- In August 2006, Wang, Yao and Yao announced that finding collisions requires 2^{63} operations.
- In 2017, Google announced the SHAttered attack, in which they generated two different PDF files with the same SHA-1 hash in roughly $2^{63.1}$ SHA-1 evaluations.
- NIST proposes SHA-2 variants
  - SHA-256, SHA-384, and SHA-512.
  - Attacks on SHA-1 have not beed extended to SHA-2 variants yet.

NIST hash function competition

- The NIST hash function competition
  - an open competition held by the US NIST for a new **SHA-3** function to replace the older SHA-1 and SHA-2 hash functions,
  - formally announced on November 2, 2007.
  - NIST selected 51 entries for the Round 1, and 14 of them advanced to Round 2.
  - Sarmal did not pass to round 2
  - Spectral Hash with substantial weakness.
  - SHAMATA retracted
  - Hamsi accepted for round 2.
  - Winner was announced in 2012. (Keccak)
    - http://keccak.noekeon.org/

Birthday Paradox

- Probability results are sometimes counterintuitive.
- In birthday paradox, we are looking for the smallest value of k such that $P(365, k) \geq 0.5$.
  - probability that at least two people in a group of k people have the same birthday is greater than 0.5. (Ignore the leap year)
- The problem statement:
  - $P(n, k) = \Pr(a_i = a_j)$ where $a_1, a_2, \ldots, a_k$ and $1 \leq a_i, a_j \leq n$ and $i \neq j$
  - Assume $n \geq k$.
  - each item is able to take one of n values equally likely
  - at least one duplicate in k items
  - What is the minimum value of k such that the $Pr(a_i = a_j) \geq 0.5$

Birthday Paradox

- Consider the number of different ways, N, that we can have k values with no duplicates.
  - $N = 365 \times 364 \times 363 \times \ldots \times (365 - k + 1) = \dfrac{365!}{(365-k)!}$
- The total number of possibilities is 365^k.
- Then the probability

$$P(365, k) = 1 - \frac{365!}{(365-k)!365^k}$$

- When the probabilities are calculated $P(365, 23) = 0.5073$

Birthday Paradox

- If there are 23 people in a room, the probability of two people having the same birthday is greater than 0.5.
  - The probability is 89% that there is a match among 40 people.
- A useful inequality:

$$P(n,k) > 1 - e^{-k^2/2n}$$

if n is large enough

$$P(n,k) = 1 - \frac{n!}{(n-k)!n^k}$$

Birthday Paradox in Cryptography

- Two rooms with k people each.
  - Probability of a pair of people with the same birthday, elements of pairs from a different room:
  - $\approx 1 - e^{-k^2/n}$.
  - Example:
    - $k = 19 \rightarrow \approx 63\%, 19 \cong 365^{1/2}$
    - $k = 30 \rightarrow 91.5\%$
- Application to hash functions
  - Two sets of messages with $k = 2^{m/2}$ messages each.
  - Messages choose one of the hash values of m-bit $(n = 2^m)$ <u>at random</u>.

Birthday Attacks 1/3

- Assume hash code is a 64-bit value, $m = 64$
- Alice signs the hash $H(M)$ of a message M.
- An opponent would need to find M' such that
  $H(M) = H(M')$
  to substitute another message.
- After trying about 2^{64} different messages, we have high probability to find a message M' that gives the same hash as M.
- However, a different attack based on birthday paradox is much more feasible.

Birthday Attacks 2/3

- Opponent forms two sets of messages:
  1. 2^{32} variations on the original message, all of which convey essentially the same meaning
  2. Prepares an equal number of messages, all of which are variations on the fake message to be substituted for the original message
- These two sets of messages are compared to find a pair of messages that produces the same hash value.
  - The probability of success, by the birthday paradox, is greater than 0.5.
- If no match is found, additional messages are generated for the two sets.

Birthday Attacks 3/3

- The opponent offers the valid variation to Alice to sign.
- Alice generates a hash value for this message and signs it.
- The opponent replaces the original message with the fraudulent message that generates the same hash value.
- Now, the fradulent message has a valid signature.
- Since the hash value is 64-bit, the level of effort required is only on the order of 2^{32}.

Why Birthday Attacks Work?

- Variations are obtained by adding a space at the end of a line, modifying the punctuation, changing the wording slightly, etc.
- In two sets there are $k = 2^{m/2}$ messages each.
- The probability that a message from the first set of k produces the same hash value as a message from the second set of k is given by a similar formula with approximation

$$1 - e^{-k^2/n}$$

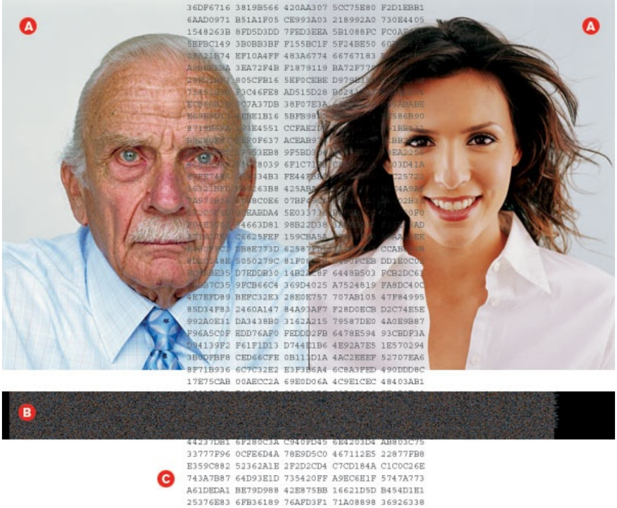
Why Birthday Attacks Work?

- $n = 2^m \rightarrow$ Probability that there is a match between the hash values of two messages from the two sets is approximately

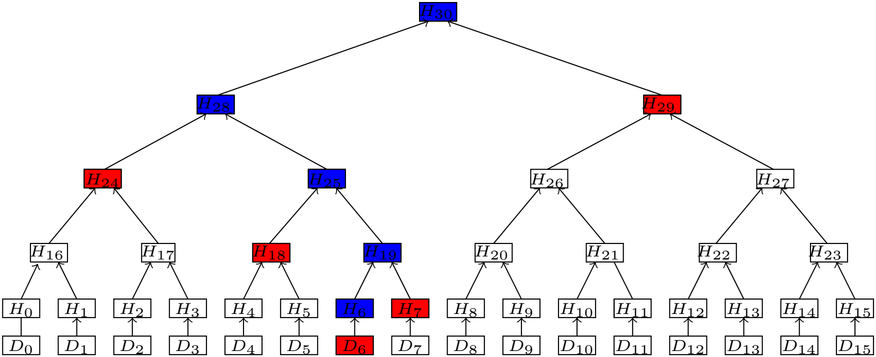
  $1 - e^{-k^2/n} = 1 - e^{-1} = 0.63 > 0.5$

  http://www.win.tue.nl/hashclash/

Collision in MD5



Merkle Hash Tree



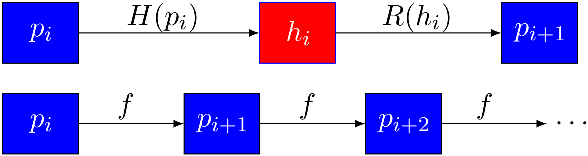
Password Protection

- The password is hashed and the hash value is stored.
- The user enters his/her password for authentication
- Its hash is computed and the hashes are compared
- If comparison is passed, the user is authenticated
- To increase security, **salt** is used ($h = \mathrm{H}(\textsc{pwd}||\textsc{salt})$, where $\textsc{salt}$ is a k-bit random integer.)

Dictionary Attack

- Dictionary Attack
- Generate a dictionary of all possible passwords (or as many passwords as possible): $\textsc{pwd}_i$ for $i = 1, 2, \ldots$
- Compute the hashes of all passwords in your dictionary: $\mathrm{H}(\textsc{pwd}_i)$ for $i = 1, 2, \ldots$
- You have now pairs in your table: $[\mathrm{H}(\textsc{pwd}_i), \textsc{pwd}_i]$ for $i = 1, 2, \ldots$
- If you use salt, then the table size increases significantly
  - Assume 16-bit salts
  - Then, for every password candidate $\textsc{pwd}_i$, we need to have 2^{16} possible hash values such that $\mathrm{H}(\textsc{pwd}_i||0)$, $\mathrm{H}(\textsc{pwd}_i||1), \ldots \mathrm{H}(\textsc{pwd}_i||65535)$

- Time-memory trade-off
- $\mathcal{P}$: set of all passwords; $p \in \mathcal{P}$
- $\mathcal{H}$: image of hash function H; $h \in \mathcal{H}$
- Let $R : \mathcal{H} \to \mathcal{P}$
- Generate a chain of passwords



where

$$f : \mathcal{P} \to \mathcal{P}(p_{i+1} = R(H(p_i)))$$

Password Crack 2/2

- Create the following table and store only $p_{j,0}$ and $p_{j,t-1}$

$$
\begin{array}{ccccccccc}
p_{1,0} & \xrightarrow{f} & p_{1,1} & \xrightarrow{f} & p_{1,2} & \xrightarrow{f} & \cdots & \xrightarrow{f} & p_{1,t-1} \\
p_{2,0} & \xrightarrow{f} & p_{2,1} & \xrightarrow{f} & p_{2,2} & \xrightarrow{f} & \cdots & \xrightarrow{f} & p_{2,t-1} \\
\vdots & & \vdots & & \vdots & & & & \vdots \\
p_{m,0} & \xrightarrow{f} & p_{m,1} & \xrightarrow{f} & p_{m,2} & \xrightarrow{f} & \cdots & \xrightarrow{f} & p_{m,t-1}
\end{array}
$$

- Given a hash value h corresponding to a password, compute
  - $p_0 = R(h)$
  - $p_i = f(p_{i-1})$ for $i = 1, 2, \ldots, t-2$
- Compare p_i for $i = 0, \ldots, t-2$ with $p_{j,t-1}$ for $j = 1, \ldots, m$
- For more information, search for rainbowcrack

Why It Works

- Case 1:
  - $p_0 = R(h)$
  - compare p_0 and $p_{j,t-1}$ for $j = 1, \ldots, m$
  - If $p_0 = p_{j,t-1}$; then $R(h) = R(H(p_{j,t-2}))$
  - $h = H(p_{j,t-2})$
  - Then the password is $p_{j,t-2}$
  - Compute $p_{j,0} \rightarrow p_{j,1} \rightarrow \ldots \rightarrow p_{j,t-2}$.
  - $t-2$ hash computations in total
  - m comparisons

Why It Works

- Case 2:
  - $p_0 = R(h)$ and $p_0 \neq p_{j,t-1}$ for $j = 1, \ldots, m$
  - then compare $p_1 = f(p_0)$ and $p_{j,t-1}$ for $j = 1, \ldots, m$
  - If $p_1 = p_{j,t-1}$ then $R(H(p_0)) = R(H(p_{j,t-2}))$
  - $H(p_0) = H(p_{j,t-2})$
  - $p_0 = p_{j,t-2}$ then $R(h) = R(H(p_{j,t-3})) \rightarrow p_0 = p_{j,t-3}$
  - Compute $p_{j,0} \rightarrow p_{j,1} \rightarrow \ldots \rightarrow p_{j,t-3}$.
  - $t - 2$ hash computations in total
    - One hash computation from p_0 to p_1.
    - $t - 3$ hash computations from $p_{j,0}$ to $p_{j,t-3}$

Extendable-Output Functions (XOF)

- XOF is a generalization of a cryptographic hash function.
- Instead of creating a fixed-length digest (e.g. 32 bytes like SHA-2/256), it can produce outputs of any desired length.
- Used in cryptographic schemes
- Example: SHAKE-128 and SHAKE-256 based on SHA3

Message Authentication

- Message authentication ensures the integrity of a message
- i.e. its content has not been changed by unauthorized parties

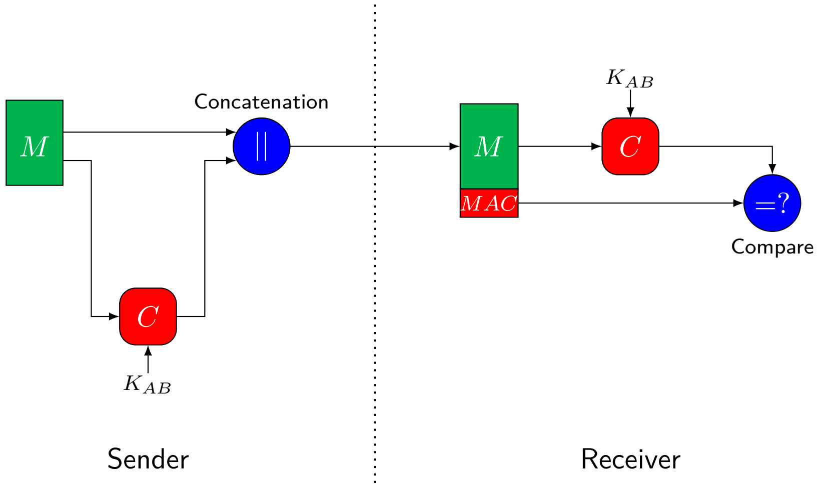
Authentication with Encryption

- The ciphertext of the message serves as authenticator.
  - If the ciphertext decrypts into a meaningful plaintext, then the message is authentic.
- Several scenarios in which this scheme is not suitable:
  - May be hard to distinguish a meaningful message.
  - Authentication cannot be done on selective basis.
  - one destination is interested in the authentication while the others are interested only in confidentiality
  - Separation of authentication and confidentiality may offer architectural flexibility

Message Authentication Codes

- MAC or cryptographic checksum is a short, fixed-length bit string derived from a message of arbitrary length using a secret key.
- This technique assumes that two communicating parties, say A and B, share a secret key K_{AB}.
- When A has a message M to send to B, it calculates the MAC as a function of the message and the key:
  $\texttt{MAC} = C_{K_{AB}}(M)$
  where C is the MAC function.
- The message and MAC are transmitted to B.

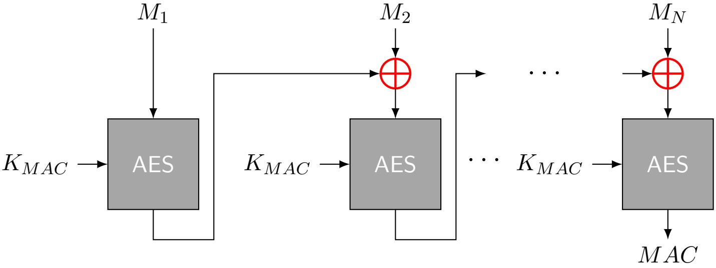
Basic Use of MAC



- Message authentication without confidentiality

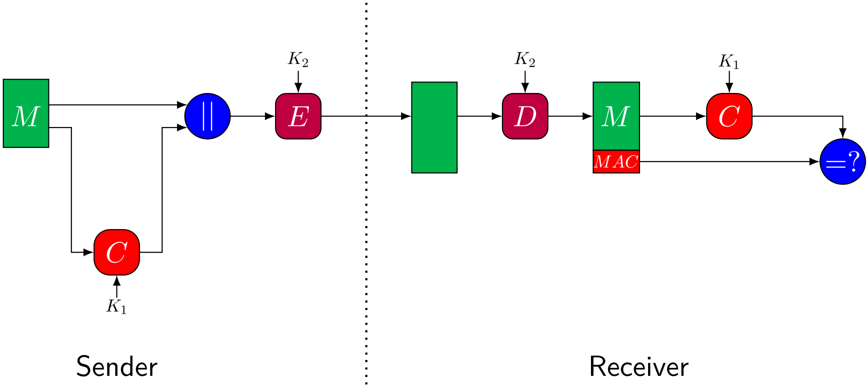
Properties of MAC Function

- A MAC function is similar to encryption in many ways.
- But it does not have to be reversible.
- A MAC function is a many-to-one function



Basic Uses of MAC

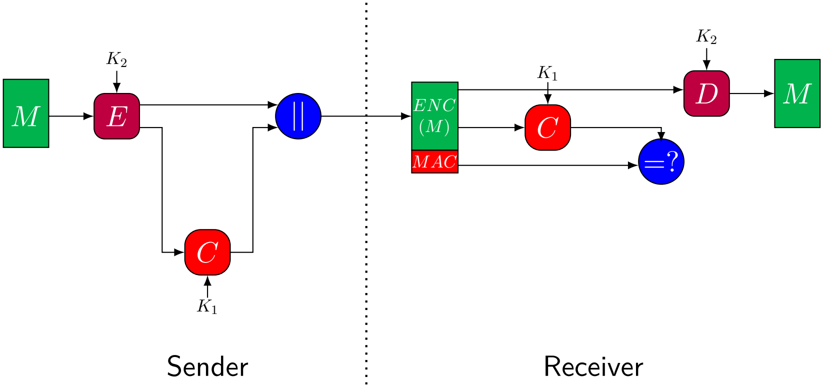
- Message authentication and confidentiality



- Authentication code is tied to plaintext

Basic Uses of MAC

- Message authentication and confidentiality



- Authentication is tied to ciphertext