ECE 443/518 – Computer Cyber Security
Lecture 06 Cryptographic Hash Functions

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

September 12, 2022

Outline

Reading Assignment

- This lecture: UC 11.2, 11.3, 11.5
- Next lecture: UC 12, 5.1.6

Outline

Motivation

- How should we address active adversaries?
  - Who can modify messages or even introduce messages.
- Three steps
  - Integrity without a secret key: Cryptographic Hash Functions
  - Integrity with a secret key: Message Authentication Codes
  - Confidentiality and integrity: Authenticated Encryption

Integrity without Secret Key

- Alice has developed a marvelous game and wants everyone to play it.
- The installation package is huge – Alice decides to seek help from third parties for distribution.
  - Because required bandwidth is either too expensive or technically infeasible.
  - E.g. via BitTorrent.
- It is not possible for Bob, who wants to download the game, to setup a secret key with Alice.
- Oscar, who participates in package distribution, plans to add his/her own adware to the package to make some profit.
- Integrity: how to design a mechanism to ensure Bob to receive the authentic package from Alice?

Hash Functions

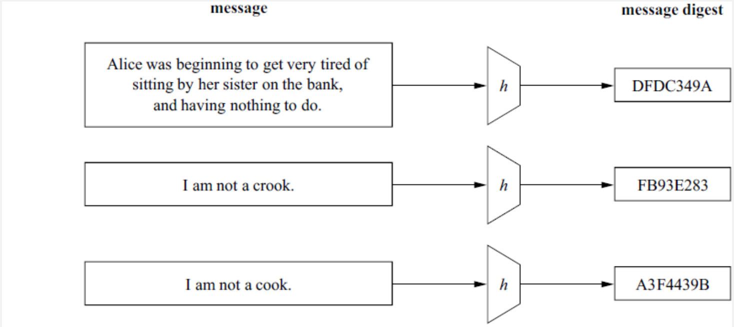
message
message digest
Alice was beginning to get very tired of sitting by her sister on the bank, and having nothing to do.
h
DFDC349A
I am not a crook.
h
FB93E283
I am not a cook.
h
A3F4439B

Fig. 11.3 Principal input–output behavior of hash functions

(Paar and Pelzl)

- Input x: messages of arbitrary lengths
- Output $z = h(x)$: message digest, a.k.a fingerprint, with fixed size, say m bits.

Preimage Resistance (One-Wayness)

Given a hash function h and a message digest z, find a message x such that:

$$z == h(x).$$

- Prevent someone to recover x from z.
  - Not related to our game distribution example but a must have for "good" hash functions.
- A "good" hash function should be one-way.
  - To allow infinite many messages to map to any single z since there are only finite z's.

Alice's Mechanism

- From the package x, Alice publishes the message digest $z = h(x)$ on her website.
  - The message digest is so short, e.g. $m = 256$, that Alice doesn't need to worry about bandwidth.
- Bob obtains the package x', computes $z' = h(x')$, and verifies that $z == z'$.
  - Can Bob be sure $x == x'$ now? Don't try to answer it now – state your assumptions and think of attacks!
- Assumption: Oscar can't modify z on Alice's website.
  - I.e. an authentic channel that guarentees only integrity – anyone can see but no one could modify z.
  - In comparison with the secure channel that guarentees both confidentiality and integrity to setup secret keys.
- Attack: Oscar create a package with the same message digest so that Bob won't find out what he received is not authentic.

Second Preimage Resistance (Weak Collision Resistance)

Given a hash function h, a message x_1 and its message digest $z_1 = h(x_1)$, find a message $x_2 \neq x_1$ such that for its message digest $z_2 = h(x_2)$,

$$z_2 == z_1.$$

- Weak collision is unavoidable: x_2 always exists.
  - Collision: different messages map to the same message digest.
  - The practical question is how easily Oscar can find one.
- Oscar's attack: choose x_2 randomly and compute $z_2 = h(x_2)$.
  - $z_2 == z_1$ with a probability of at least $\frac{1}{2^m}$ for some z_1.
- If Oscar repeats the attack N times, the probability of finding x_2 is $1 - (1 - \frac{1}{2^m})^N$.
  - About 63% for $N = 2^m$.
  - Not a concern if m is large enough when Oscar is computationally bounded.
- What about cryptanalysis that uses properties of h and x_1?

Oscar's Trick

- Knowing there may exist little hope to modify Alice's package without being caught, Oscar decides to create his/her own game package to distribute the adware.
- Oscar's trick: create two packages x and x' such that
  - $h(x) == h(x')$
  - Good package x: just the game.
  - Bad package x': the game and the adware.
- Oscar then delivers x' to Bob through third parties.
- If Bob finds the adware in x', Oscar shows Bob x and claims someone else creates x'.
- Will second preimage resistance help?

(Strong) Collision Resistance

Given a hash function h, find two messages $x_1 \neq x_2$ such that:

$$h(x_2) == h(x_1).$$

- Birthday Attack: what is the probability that two in our class have the same birthday?
  - How many students are needed to have a 50% chance of two colliding birthdays? 23.
- Roughly speaking, if Oscar creates $2^{\frac{m}{2}}$ random packages, then there is 50% chance of collision.
- Bob may still resist such attack by requesting m to be large enough.
  - But what about cryptanalysis?

Cryptographic Hash Functions

- Cryptographic Hash Functions: a hash function that is
  - Preimage resistant
  - Second preimage resistant
  - (Strong) collision resistant
- With a proper choice of m.
  - As of now, consider $m = 256$ or more.
- Be so even under cryptanalysis.
  - A "bad" choice of h may lead to attack of second preimage resistance using far less than 2^m messages, or attack of strong collision resistance using far less than $2^{\frac{m}{2}}$ messages.
  - E.g. cyclic redundancy check (CRC) is a good hash function against data corruption but not a good cryptographic hash function.

Outline

The MD4 Family

- MD5: RFC 1321 (1992), 128-bit
  - Was widely used, "no longer acceptable where collision resistance is required" per RFC 6151.
- SHA-1: FIPS PUB 180-1 (1995), 160-bit
  - Successful recent efforts to generate collision.
  - Should be phased out.
- SHA-2: FIPS PUB 180-2 (2001), FIPS PUB 180-4 (2015)
  - SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256.
  - Were adopted slowly but widely in use now – Bitcoin contributes to $10^{20} \approx 2^{67}$ SHA-256 hashes per second as of recently.
  - A lot of ongoing attacking efforts.

SHA-3

- FIPS PUB 202 (2015)
- Via an open selection process like AES starting 2006.
  - Not meant to replace SHA-2, but as an alternative.
- Finalists
  - BLAKE: based on a stream cipher
  - Groestl: use a lot of constructs from AES
  - JH
  - Keccak: based on sponge construction
  - Skein: based on a block cipher and a variant of Matyas-Meyer-Oseas.
- Winner: Keccak

Summary

- Cryptographic hash functions need to be preimage resistant, second preimage resistant, and (strong) collision resistant.
- As of now, we should use hash functions with at least 256 bits hashes.
  - Use SHA-2 and SHA-3.
  - Avoid MD5 and SHA-1.