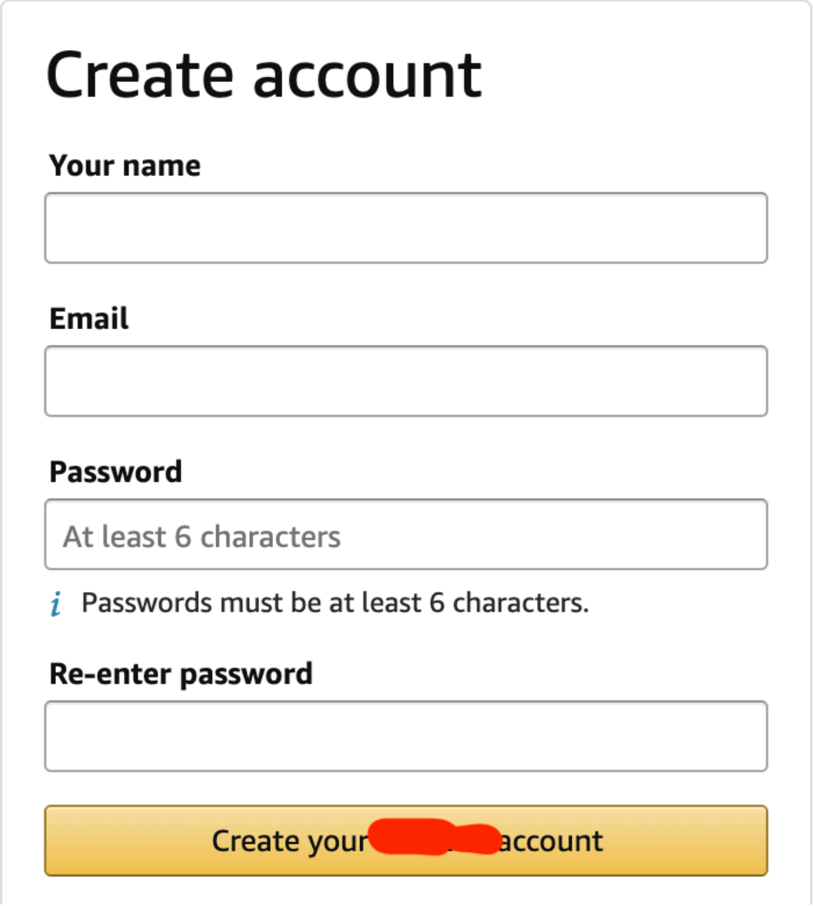
Collision-Resistant Hash

Shusen Wang

Application: Storing Passwords in Database

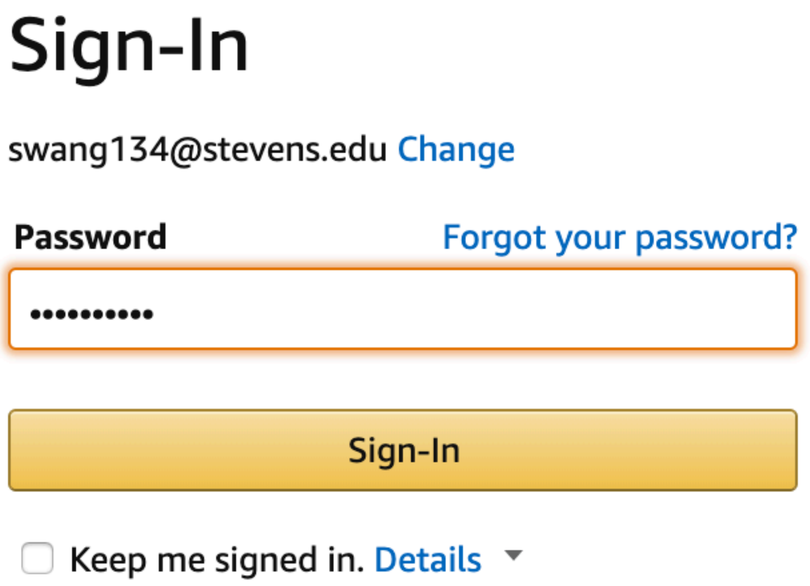
Naively store plain text passwords in database



Database

Email	Password
john@gmail.com	123john455
david@gmail.edu	smith780817
mary@yahoo.com	maryjohnson86
lee@gmail.com	123456
swang134@stevens.edu	wangshusen123
jeff@gmail.com	admin
linda@hotmail.com	Laidfi3j!=32

Naively store plain text passwords in database

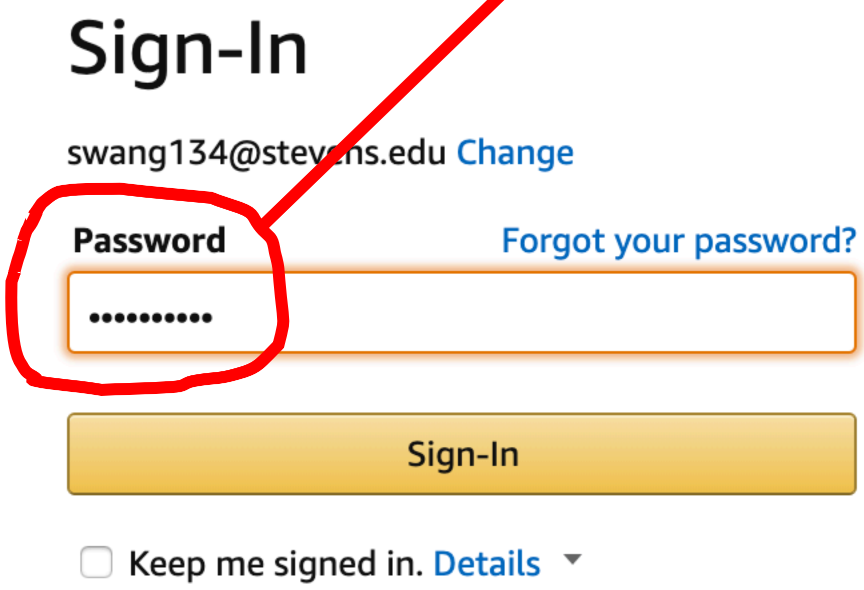


Database

Email	Password
john@gmail.com	123john455
david@gmail.edu	smith780817
mary@yahoo.com	maryjohnson86
lee@gmail.com	123456
swang134@stevens.edu	wangshusen123
jeff@gmail.com	admin
linda@hotmail.com	Laidfi3j!=32

Naively store plain text passwords in database

Do they match?



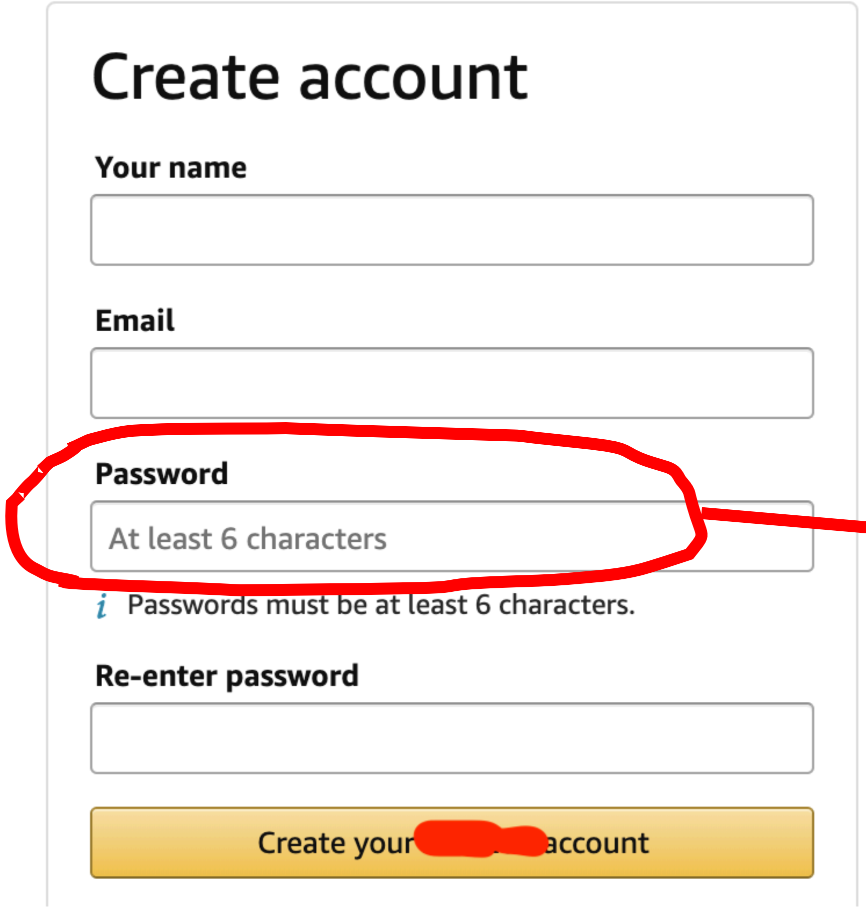
Database

Email	Password
john@gmail.com	123john455
david@gmail.edu	smith780817
mary@yahoo.com	maryjohnson86
lee@gmail.com	123456
swang134@stevens.edu	wangshusen123
jeff@gmail.com	admin
linda@hotmail.com	Laidfi3j!=32

Naively store plain text passwords in database

- Storing plain text passwords in the database is a sin!
- What if a user's (username, password) pair is compromised? (E.g., by hacker or database administrator.)
- 55% of net users use the same password for most, if not all, websites.
    - The attacker will use the (username, password) pair for most websites, e.g., banks, Gmail, FB, Amazon, etc.
- There are over 30% websites which store your passwords in plain text.

Using hashed passwords

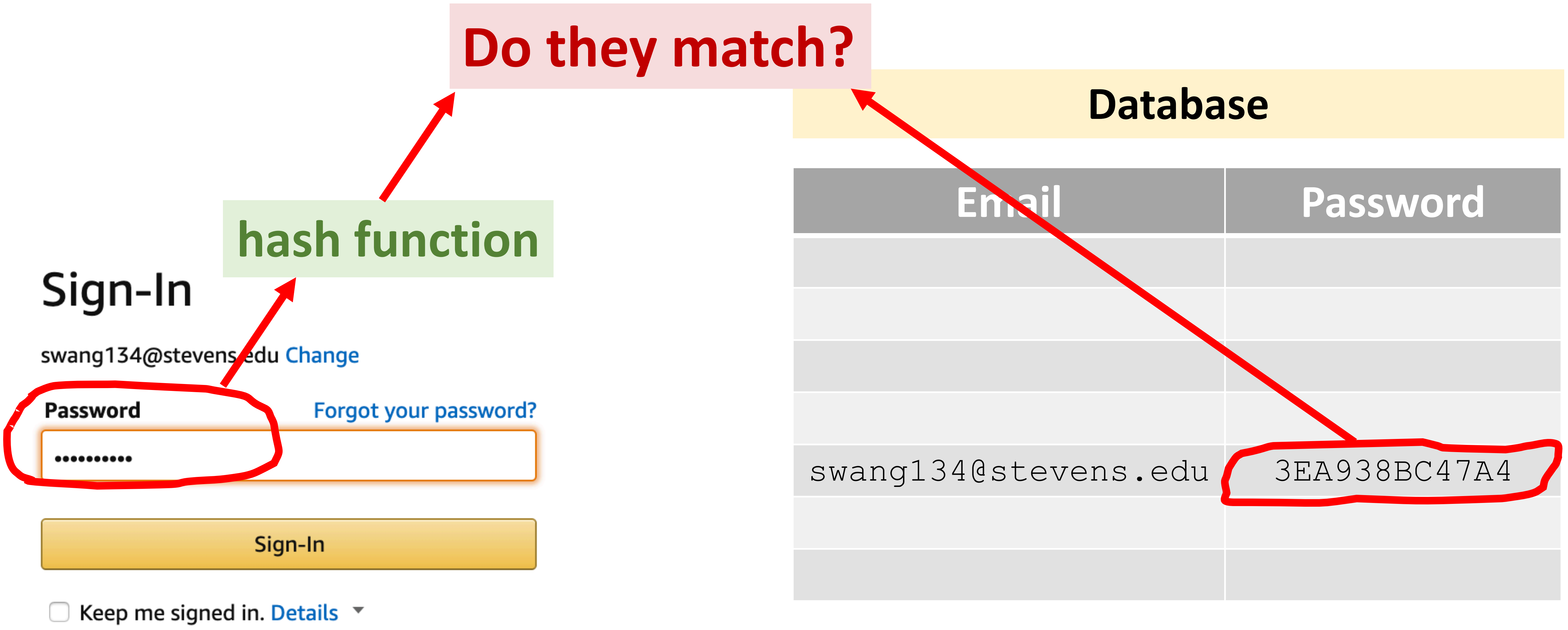


hash function

Database

Email	Password
swang134@stevens.edu	3EA938BC47A4

Using hashed passwords



What if the hashed password is stolen?

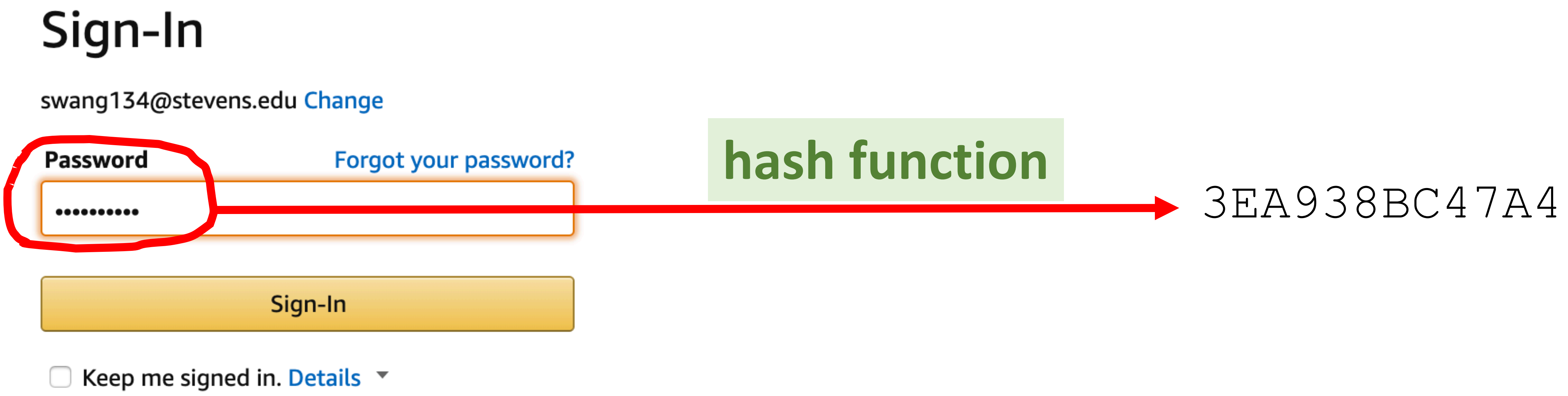
- What if the attacker knows my username and hashed password (3EA938BC47A4)?
- He wants to find a string x such that

$$h(\mathrm{x}) = \texttt{3EA938BC47A4}.$$

What if the hashed password is stolen?

- What if the attacker knows my username and hashed password (3EA938BC47A4)?
- He wants to find a string x such that

$$h(\mathrm{x}) = \mathtt{3EA938BC47A4}.$$



What if the hashed password is stolen?

- What if the attacker knows my username and hashed password (`3EA938BC47A4`)?
- He wants to find a string x such that

$$h(\mathrm{x}) = \mathtt{3EA938BC47A4}.$$

- This is a hash collision:

$$h(\mathrm{MyRealPassword}) = h(\mathrm{x}).$$

- Even if x is not my password, the attacker will be granted access to my account.

How to defend the attack?

Collision-Resistant Hash

Brute-Force Attack

Question: Given i, how to find x such that $h(\mathrm{x}) = i$.

Brute-force attack: Enumerate all x until $h(\mathrm{x}) = i$.

- Brute-force attack is inefficient.

- Assume i has 128 bits.
- On average, there is one collision per 2^{128} $(= 3.4\times10^{38})$ trials.

- Assume i has 256 bits.
- On average, there is one collision per 2^{256} $(= 1.2\times10^{77})$ trials.

Brute-Force Attack

Question: Given i, how to find x such that $h(\mathrm{x}) = i$.

Brute-force attack: Enumerate all x until $h(\mathrm{x}) = i$.

- Knowing my hashed password is i=`3EA938...BC47A4` (256 bits), the attacker needs to perform 1.2×10^{77} trials to find a hash collision.
- Is the benefit of stealing my account worth his computational cost?

Collision-Resistant Hash

- **Property 1:** Easy to evaluate.
- Given x, it cost little time to calculate $i = h(\mathrm{x})$.

- **Property 2:** Hard to invert.
- Given i, it is hard to find x such that $h(\mathrm{x}) = i$.
- There is not an easier way than brute-force enumeration.

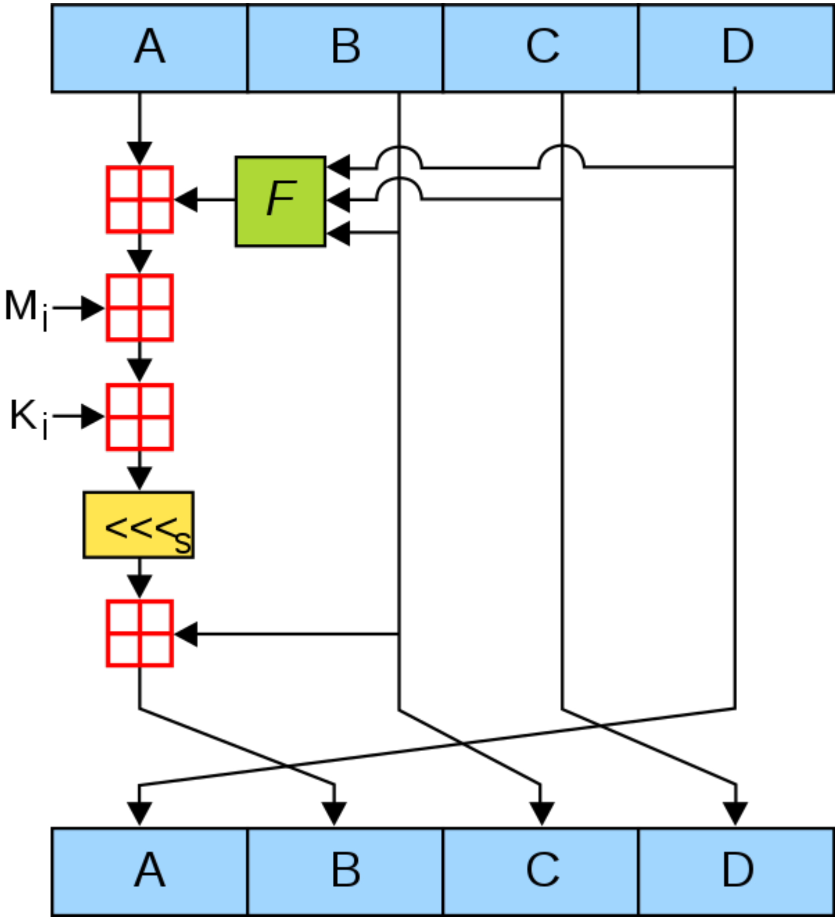
Collision-Resistant Hash

- Many cryptographic applications requires collision-resistant hash functions.
- If there is an easier method than brute-force enumeration, the hash function is considered flawed.

Example

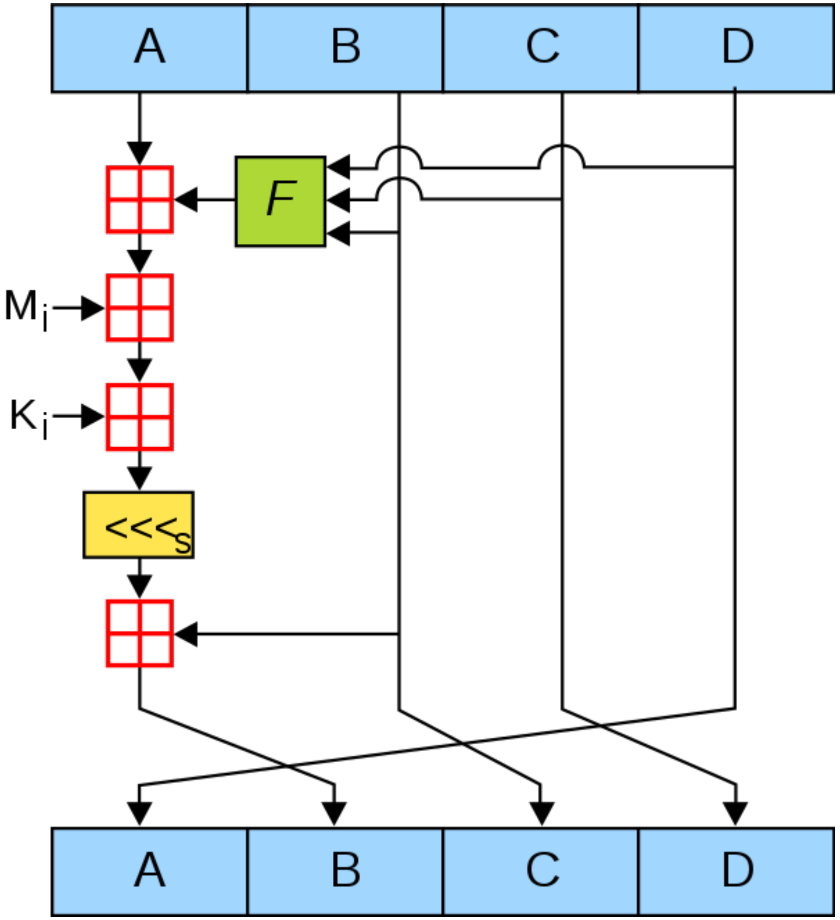
- Modulo is not collision-resistant.
- For example, $h(x) = x \bmod 100003$.
- Given i, we can immediately find x such that $h(x) = i$.
- E.g., $x = i + 100003$.
- We do not need to perform brute-force enumeration.

MD5 Hash Function



- MD5 is a very sophisticated hash function.
- MD5 is a widely used hash function producing a 128-bit hash value.
  - MD5 is a function that maps everything (disregarding its size) to 128 bits.
  - E.g., it maps a 10GB file to 128 bits.
- See Wikipedia: https://en.wikipedia.org/wiki/MD5

MD5 Hash Function

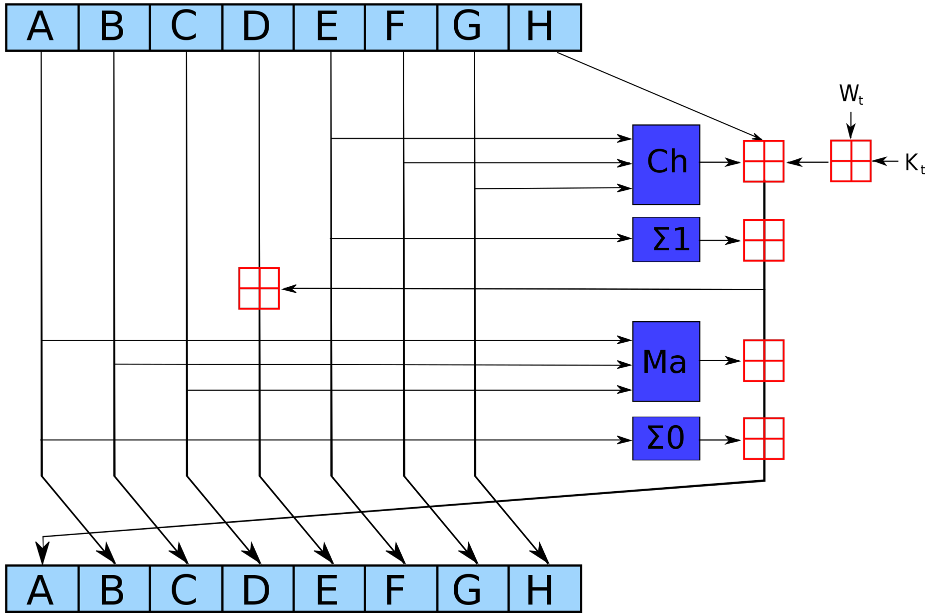


- MD5 is flawed; published papers [1,2] showed techniques more efficient than brute force.

Reference

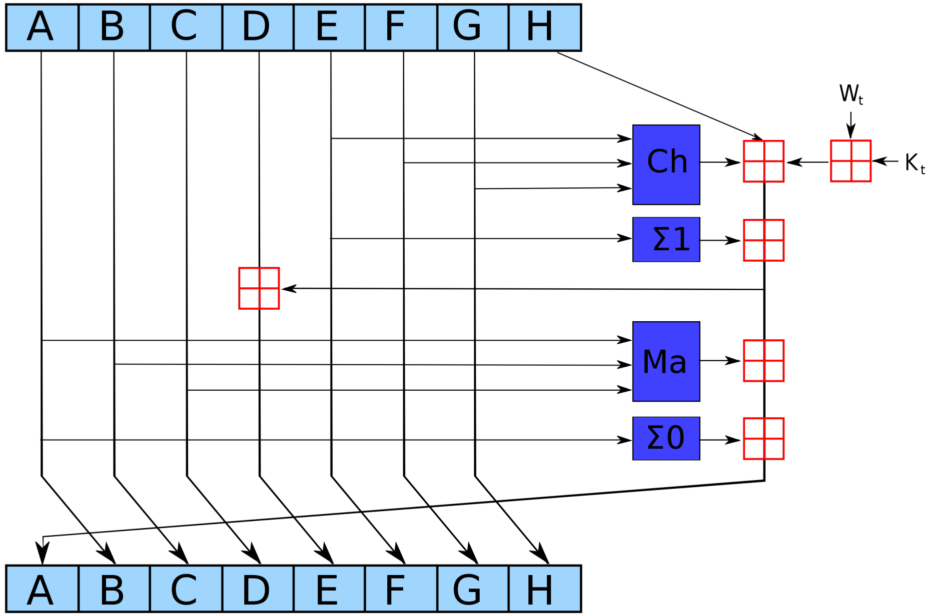
[1] Dobbertin: The status of MD5 after a recent attack. *Crypto-Bytes*, 1996.
[2] Wang & Yu: How to break MD5 and other hash functions. In *Annual international conference on the theory and applications of cryptographic techniques*, 2005.

SHA-2 (Secure Hash Algorithm 2)



- SHA-2 is a set of hash functions designed by the US National Security Agency (NSA) and first published in 2001.
- SHA-2 includes SHA-224, SHA-256, SHA-384, SHA-512, etc.
- See Wikipedia: https://en.wikipedia.org/wiki/SHA-2

SHA-256



- SHA-256 maps string to 256-bit integer.
  - It maps an empty string to a 256-bit integer.
  - It maps a 10 GB file to a 256-bit integer.
- SHA-256 may be collision-resistant.
  - There is no mathematical proof of collision resistance.
  - People do not know easier collision attack than brute-force.

Property of Collison-Resistant Hashing

A tiny change of the input will result in totally different output.

- `SHA256("Shusen Wang") =`

  `258862049feb1afdc4fd85f9bdeaf14dd770ec546f7087ffe4571ee515b64225`

- `SHA256("Shusen Wong") =`

  `3db0e717905af8c776c9fa223bd253b5662a157ac5a4f4d87ae87cad9075c453`

- `SHA256("ShusenWang") =`

  `ff0e88ec73231263032 0ccc8fdee24736fe73efb4f86ee91ef4aee54e6dbf68b`

Property of Collison-Resistant Hashing

A tiny change of the input will result in totally different output.

- What if a hash function $h(\cdot)$ does not have such a property?

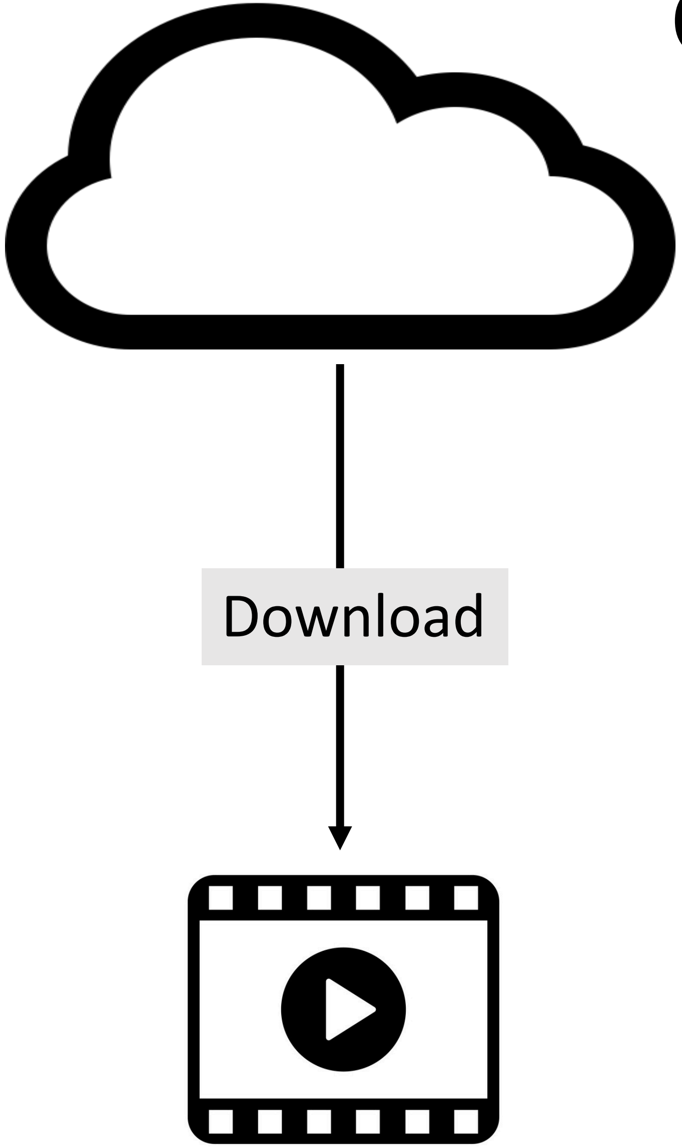
Property of Collison-Resistant Hashing

A tiny change of the input will result in totally different output.

- What if a hash function $h(\cdot)$ does not have such a property?
- Then, small $\left|\left|\mathbf{x} - \mathbf{y}\right|\right|$ <==> $\left|\left|\mathbf{h(x)} - \mathbf{h(y)}\right|\right|$.
- Collision attack will be easier than brute-force enumeration.
  - Goal of attack: Given `i`, find `x` such that `h(x)=i`.
  - Suppose the attacker finds such a `y` that `h(y)` is close to `i`.
  - Then the attacker can search `x` in the neighborhood of `y`.

Application: Checksum

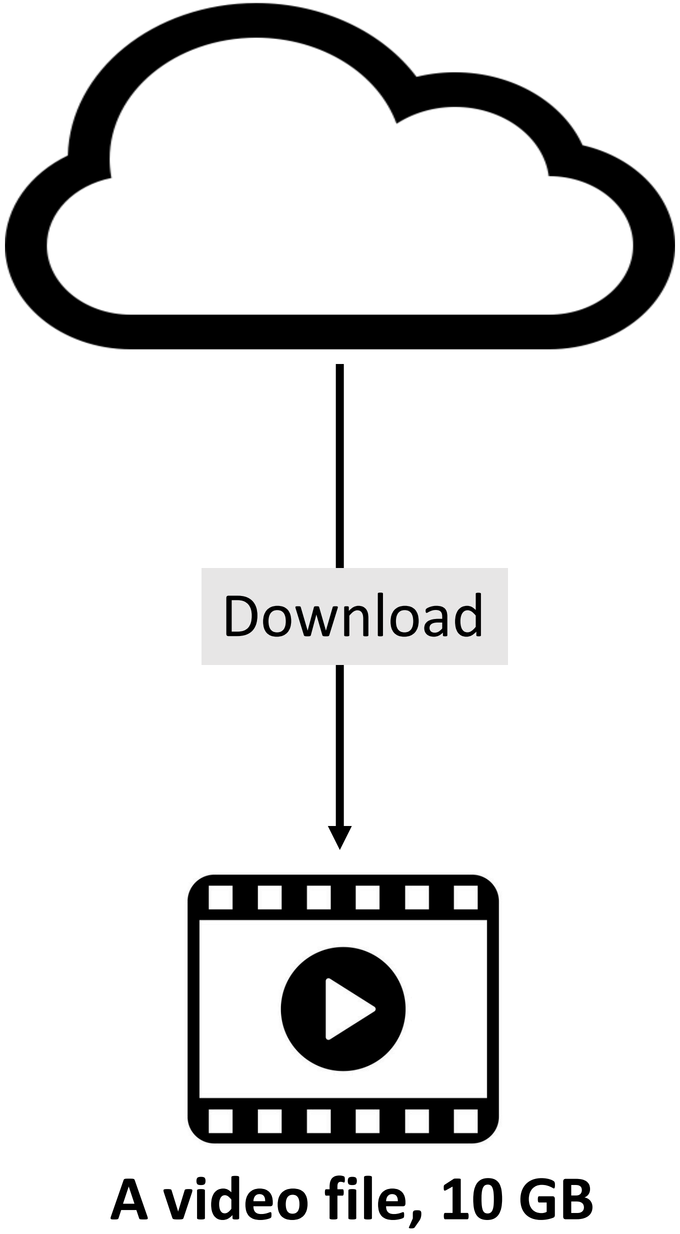
Question: Has big files arrived intact?



A video file, 10 GB

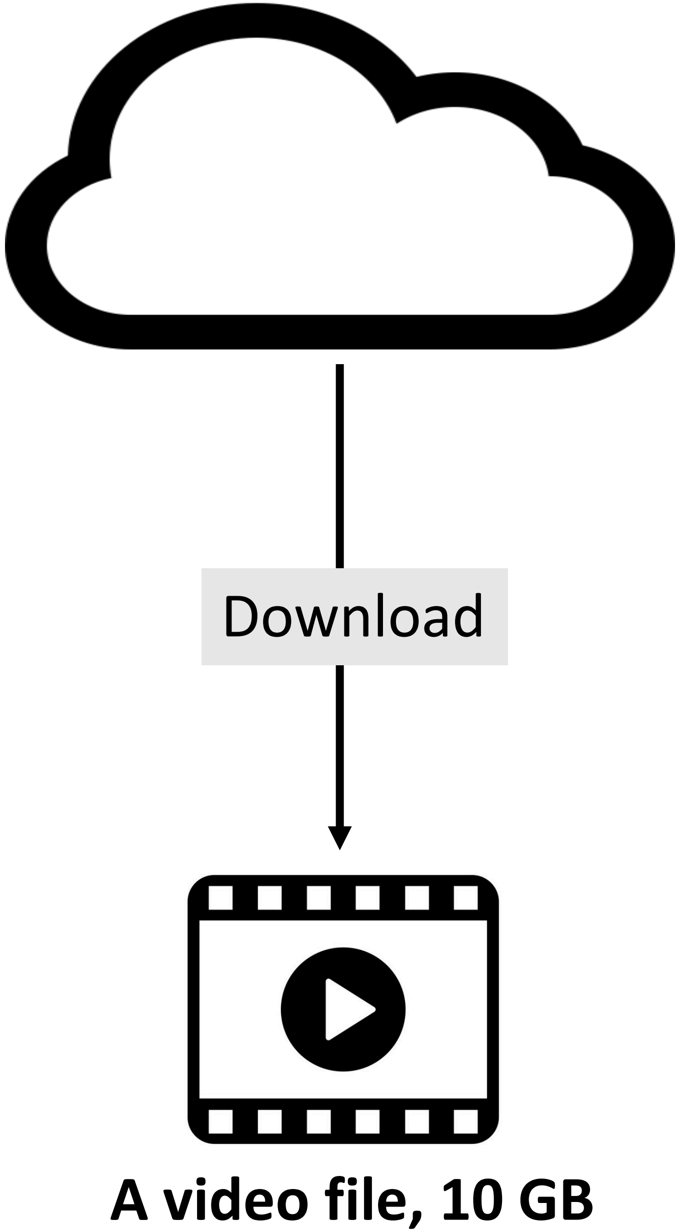
- The big file was partitioned to many small packages in order to be downloaded.
- There can be errors during the transmission.
- How do we know the big file is intact?

Solution: Hash as a checksum



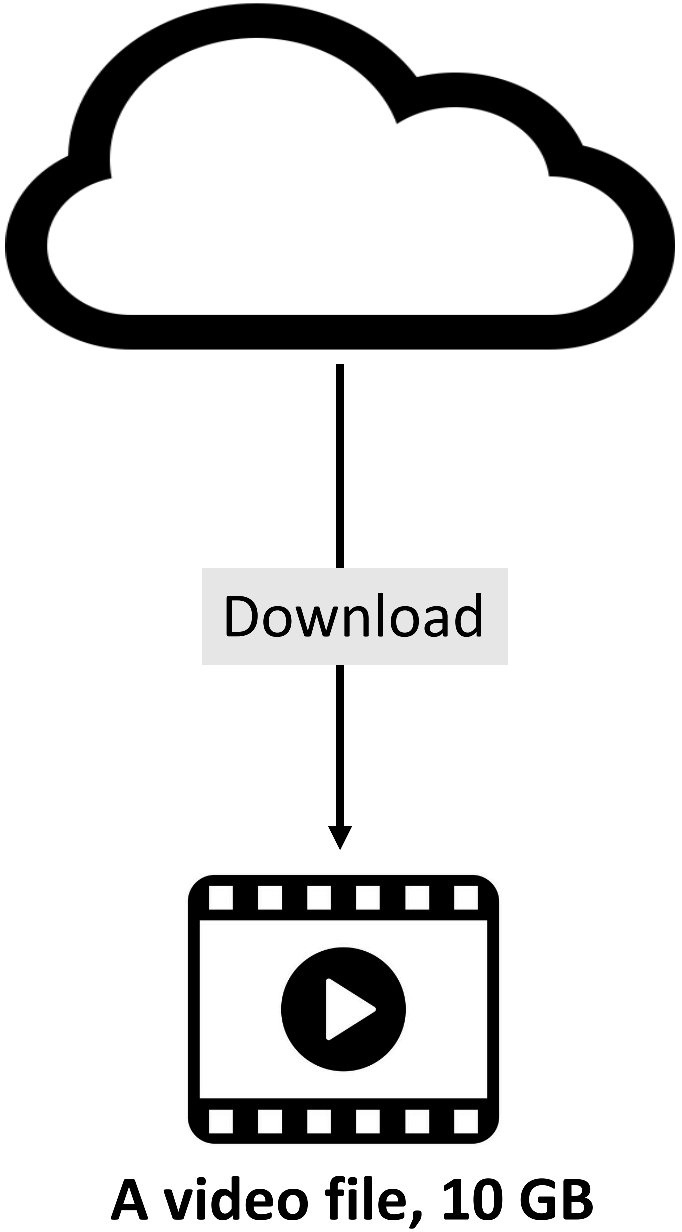
- The cloud compute the MD5 of the big file.
- User downloads both the file and the MD5.
- The MD5 has merely 128 bits, so it is very unlikely wrong during the transmission.
- The user computes the MD5 of the file, and compare it to the downloaded MD5.

Solution: Hash as a checksum



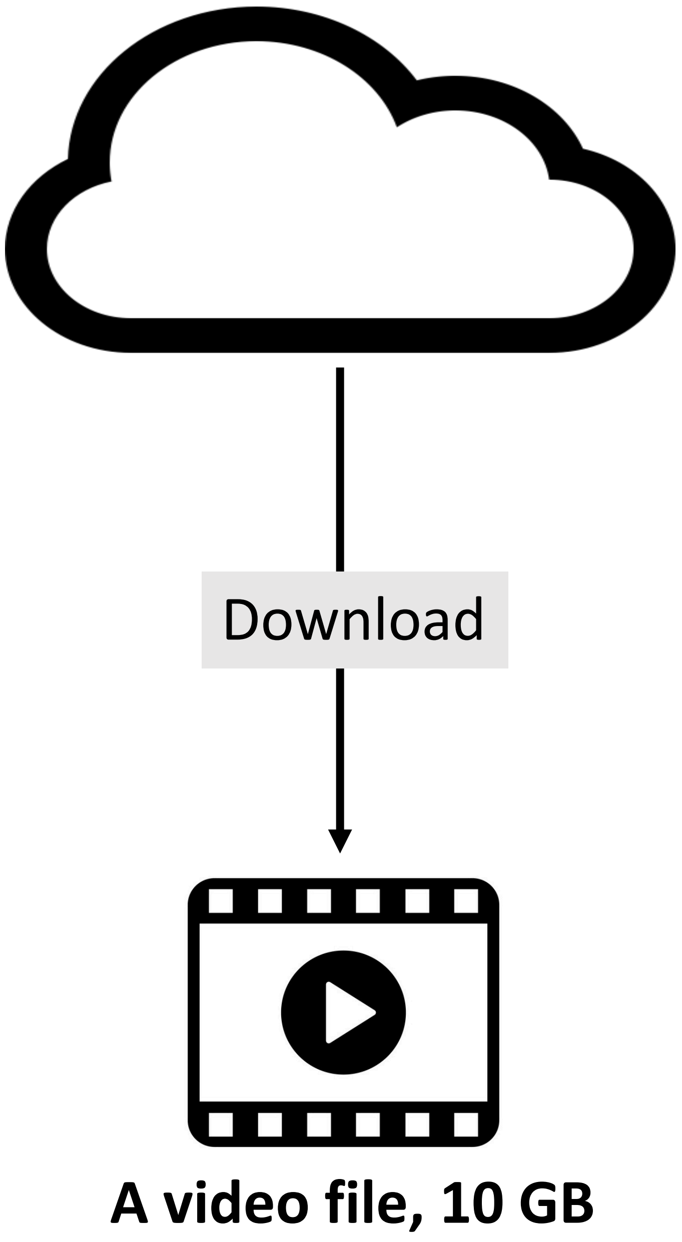
- If the two MD5s are different, we are certain that the file has errors.
- If the two MD5s match, then the file is very likely intact. (Why?)

Why does MD5 checksum work?



- F: the original file.
- $\tilde{F}$: the downloaded file.
- $F \neq \tilde{F}$: the file is broken.

Why does MD5 checksum work?



- F: the original file.
- $\tilde{F}$: the downloaded file.
- $F \neq \tilde{F}$: the file is broken.
- Can $\mathrm{MD5}(F)$ and $\mathrm{MD5}(\tilde{F})$ be the same?
- It is equivalent to a hash collision.
- If the error is random, then collision happens with probability $1/2^{128}$.

Application: Proof of Work

Proof of Work

- SHA-256 hash function f maps everything to 256 bits.

Proof of Work

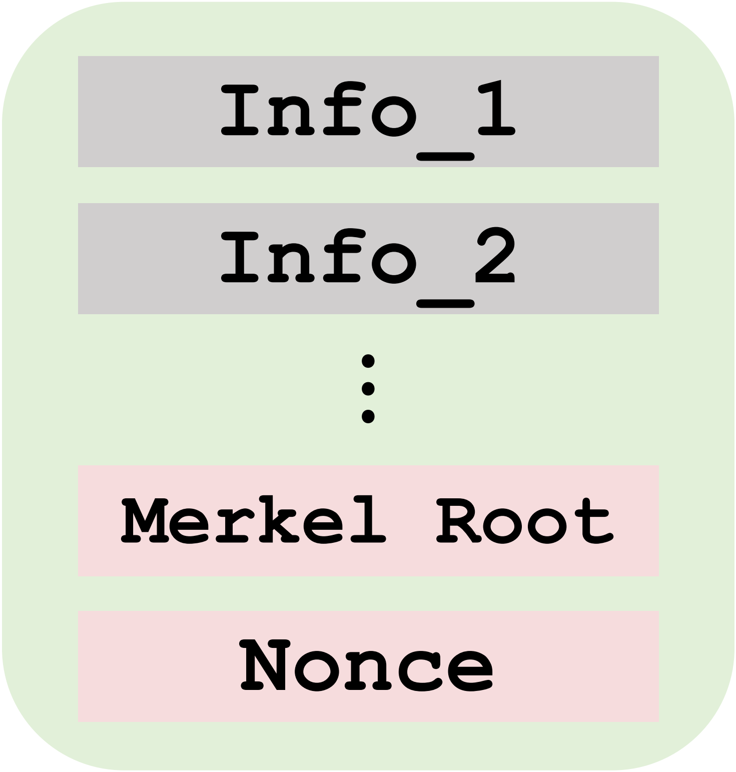
- SHA-256 hash function f maps everything to 256 bits.
- You are asked to find x such that the first 60 bits of $h(x)$ are all zeros.
- How to find such an x?
    - Brute-force enumeration.
    - On average, you need to enumerate 2^{60} numbers.

Proof of Work

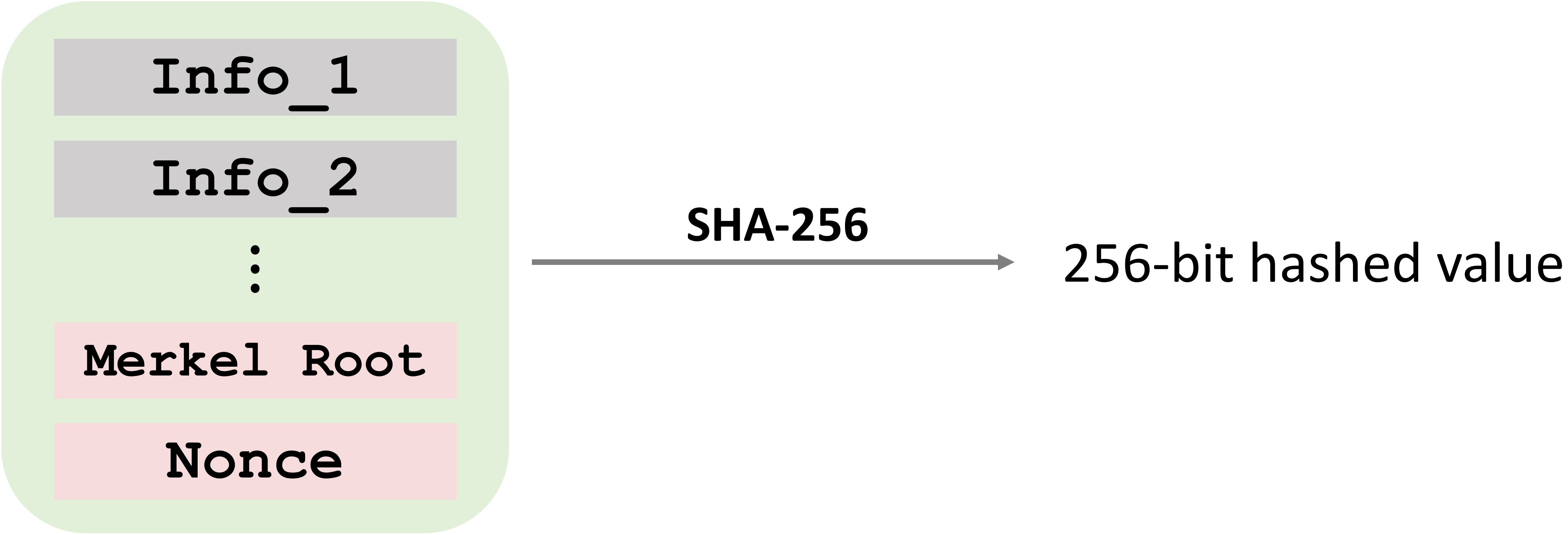
- **Property 1:** Hard to compute.
- No easier way than brute-force enumeration.
- E.g., repeatedly evaluate the hash function $h(\cdot)$ for 2^{60} times.

- **Property 2:** Easy to verify.
- Given x, evaluate $h(x)$ only once to verify whether the first 60 digits are all zeros.

Application in Bitcoin



Application in Bitcoin



- Adjust the Merkel root and nonce so that the first n bits of the hashed value are all zeros.

Application in Bitcoin

- As of June 2020, the first $n = 78$ digits of the hashed value must be all zeros.
- If you are the first to find a nonce which leads to such a hashed value, you can build the next block.
- The builder of a block is rewarded some Bitcoins.
  - As of June 2020, the reward is 6.25 BTCs.
  - As of June 1, 2020, one BTC is worth $10,000.

Summary

Collision-Resistant Hash

- Collision: Given i, find x such that $h(x) = i$.
- Collision-resistant hash functions are easy to evaluate but hard to invert.
- MD5 looks like collision-resistant but is flawed.
- SHA-256 is very widely used.

Applications

- Store hashed passwords in database.
- Checksum.
- Proof of work.
- Digital signature.

Thank You!