3010 Comp Security

-Applied Crypto 3 Hash Functions

Dr Tay Kian Boon

NTU, CCDS

2025 Semester 1

Overview

- Introduction to Hash Functions (crypto, not the hash in algorithms)
- Applications of Hash Functions
- Desired Properties of Hash Functions
- Examples of Historical and Important Hash Functions
  - MD5 Hash Function
  - SHA1 Hash Function
- KECCAK - SHA3 winner Hash Function

Hash Functions - Introduction

- Hash functions—such as MD5, SHA-1, SHA-256, SHA-3, and BLAKE2—comprise the cryptographer’s Swiss Army Knife
- Applications:
  - digital signatures,
  - public-key encryption,
  - integrity verification,
  - message authentication,
  - password protection,
  - key agreement protocols, and many other cryptographic protocols.

Hash Functions - Introduction

- There's a hash function somewhere under the hood, whether you're
  - encrypting an email,
  - sending a message on your mobile phone,
  - connecting to an HTTPS website, or
  - connecting to a remote machine through IPSec or SSH,
- It is a fundamental pillar in cryptography

Hash Functions – Main Applications

- Hash functions are most versatile and ubiquitous of all crypto algorithms.
- Many other examples of their use in the real world:
  - cloud storage systems use them to identify identical files & to detect modified files;
  - the Git revision control system uses them to identify files in a repository;
  - host-based intrusion detection systems (HIDS) use them to detect modified files;
  - network-based intrusion detection systems (NIDS) use hashes to detect known-malicious data going through a network;
  - forensic analysts use hash values to prove that digital artifacts have not been modified;
  - Blockchain uses hash function to ensure integrity of previous transactions!

Hash Functions

- A hash function takes in files of ANY length and hashes them into a fixed size file, typically 256 or 512-bits in modern context
- Files can be text, video, photos, audio etc
- In math terminology, hash is a function that maps

  $H: \{0,1\}^m \rightarrow \{0,1\}^n$, where m (arbitrary) > n (fixed)
- Because m>n, this map is many to one function, not one-one!
- So there are cases where different files hash to SAME value!

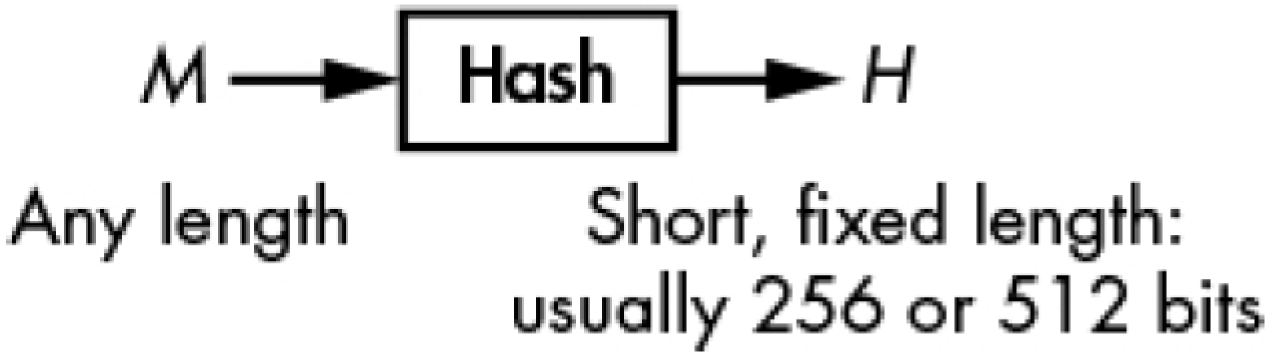


Figure 6-1: A hash function's input and output

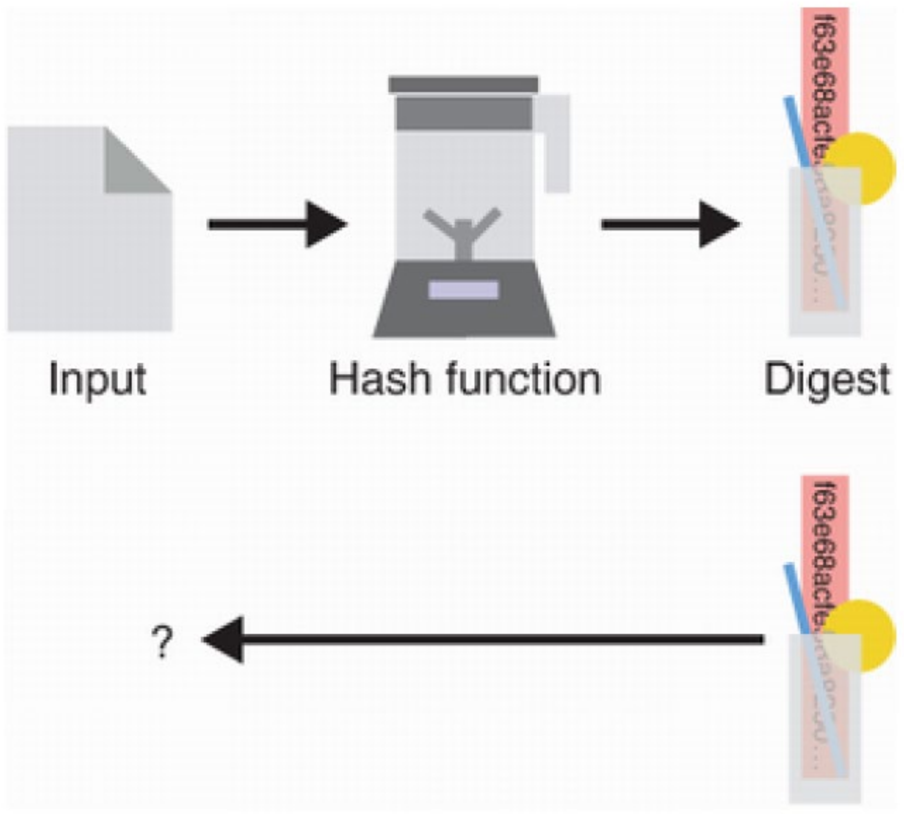
Question

- What are the desired properties of a secure hash function?
- 3 Properties!

1. Pre-Image Resistance:

- This property ensures that **no one** is able to reverse the hash function in order to recover the input given an output.
- In figure 2.3, we illustrate this "one-wayness" by imagining that our hash function is like a blender, making it impossible to recover the ingredients from the produced smoothie.

1. Pre-Image Resistant



2. Second Pre-image resistance

- The property says the following: if I give you an input and the digest it hashes to, you should not be able to find a different input that hashes to the same digest. Next Figure illustrates this principle.
- Note that attacker have no control over first input (analogy: a password hash was captured by attacker. He/she need to find any word that hashes to the same password hash!)

2. Second Pre-image resistance

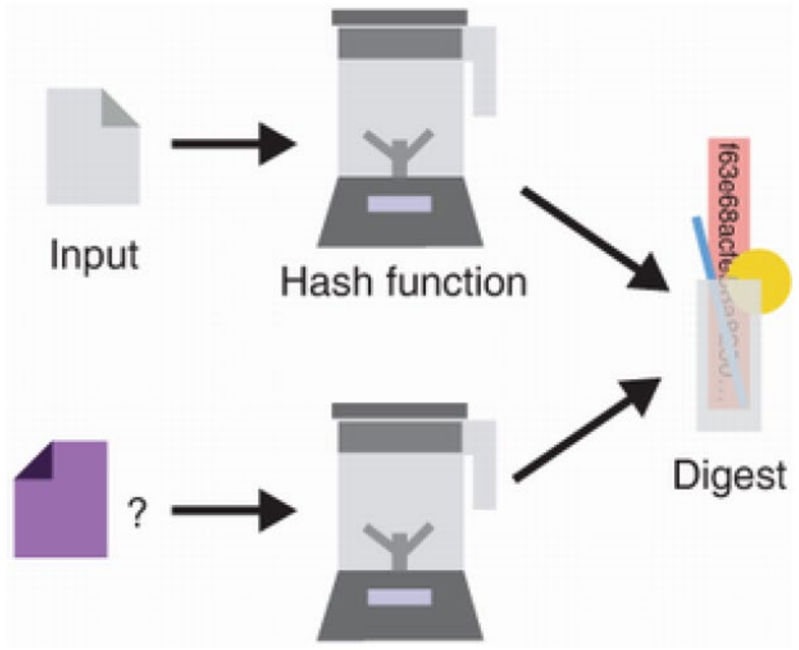


Figure 2.4 Considering an input and its associated digest, one should never be able to find a different input that hashes to the same output.

3. Collision Resistance

- It guarantees that no one is able to produce two different inputs that hash to the same output (as seen in figure 2.5).
- Here an attacker can choose the two inputs, unlike the previous property that fixes one of the inputs.

3. Collision Resistance

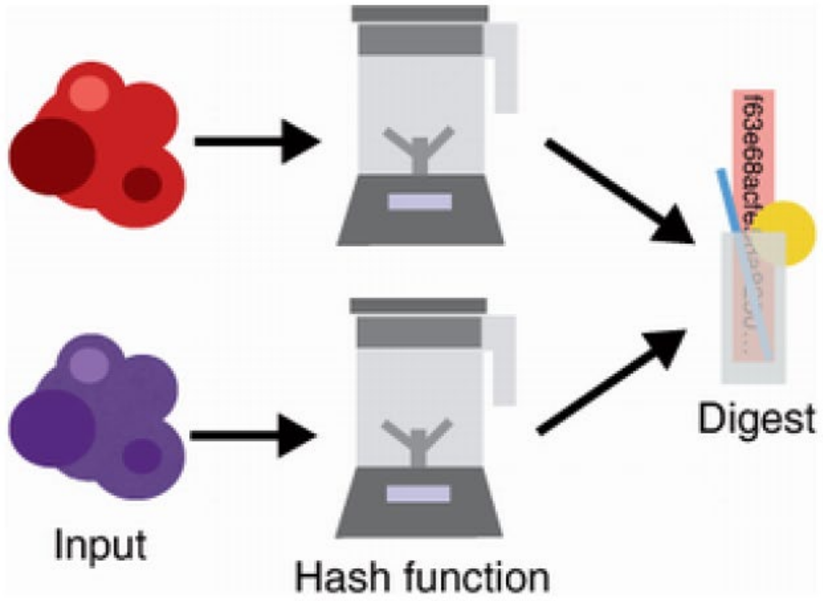
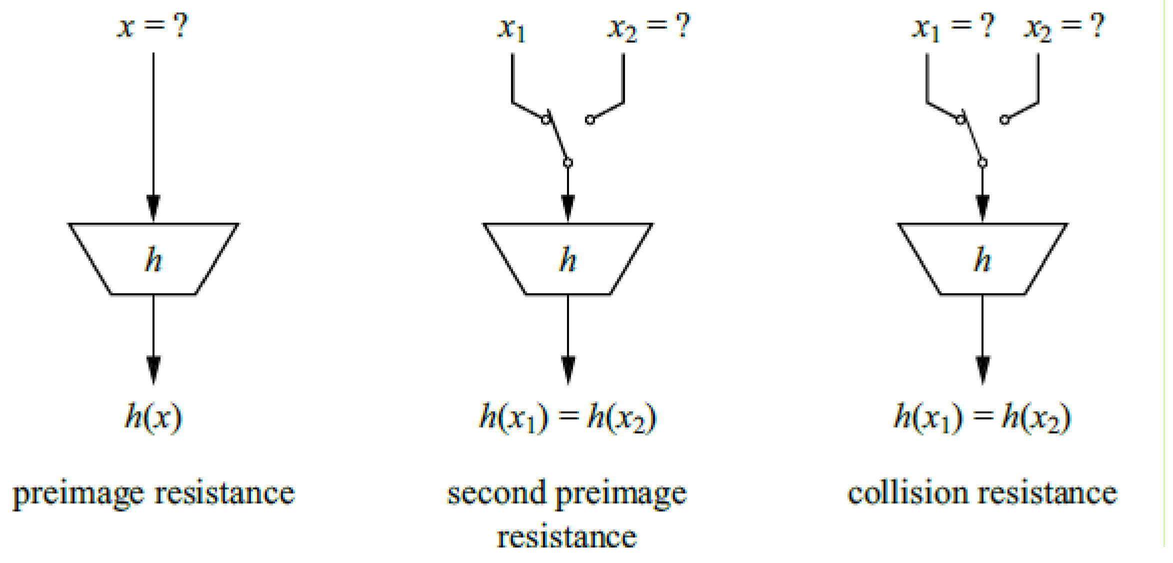


Figure 2.5 One should never be able to find two inputs (represented on the left as two random blobs of data) that hash to the same output value (on the right). This security property is called *collision resistance*.

3 Security Properties of Hash (in 1 pix)



Summary: Important Properties of Hash

- In order to be a useful function in crypto that can play the role of message integrity, hash needs to satisfy 3 important properties

1. One-way Function (infeasible to invert), i.e. given only the hash of a **file h_0**, it is computationally hard to find M s.t. $H(\mathbf{M}) = \mathbf{h_0}$ (First pre-image resistanct)
2. Given a file M & hash H(M), it is computationally infeasible for anyone to produce another file N such that $H(M) = H(N)$! (Second pre-image resistant)
3. It is computationally infeasible to find 2 files $M \neq N$ with identical hashes! i.e. $H(M) = H(N)$! (collision resistance)

Important Properties of Hash Functions

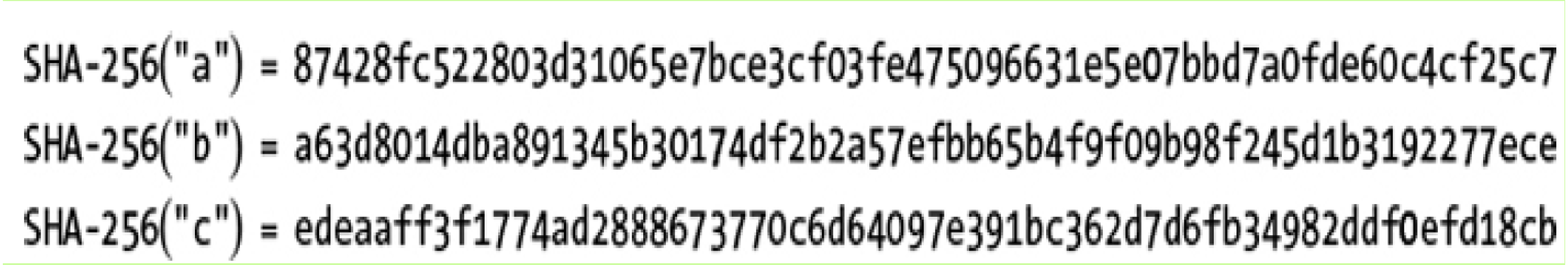
- People often confuse collision resistance and second pre-image resistance.
- Take a moment to understand the differences.
- Amongst these 3 properties, 1st and 2nd ones the hardest to crack!
- For a n-bit hash function, its security (in terms of finding hash collisions) is only at best n/2 bit strength (birthday attack)
  - 50% chance 2 files with same hash after 2^(n/2) hash random files
- That is why a strong 256-bit hash is paired with 128-bit AES in applications.

Other Good Properties of Hash

Three other obvious properties we want hash to have:

1. Fast (for integrity), slow (for password hashing)-why? (tutorial)
2. Long length, at least 256 bit long
3. Unpredictable: minute change in M will affect many bits in its hash

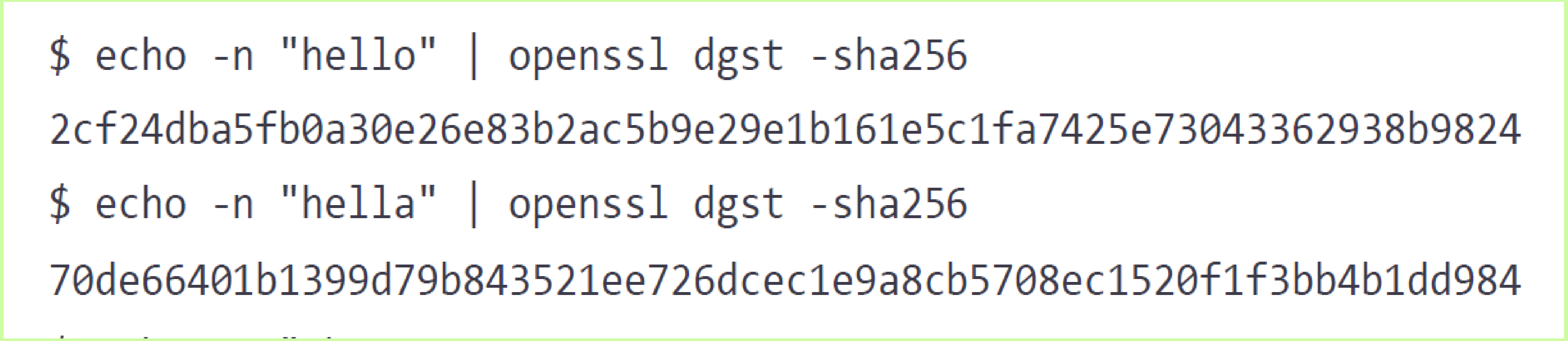
Want 1-bit change affects whole Hash- Note {a,b,c} differs in only 1 bit in ASCII!



```
SHA-256("a") = 87428fc522803d31065e7bce3cf03fe475096631e5e07bbd7a0fde60c4cf25c7
SHA-256("b") = a63d8014dba891345b30174df2b2a57efbb65b4f9f09b98f245d1b3192277ece
SHA-256("c") = edeaaff3f1774ad2888673770c6d64097e391bc362d7d6fb34982ddf0efd18cb
```

- See Next page how to compute SHA256 of a message

Important Properties of Hash Functions



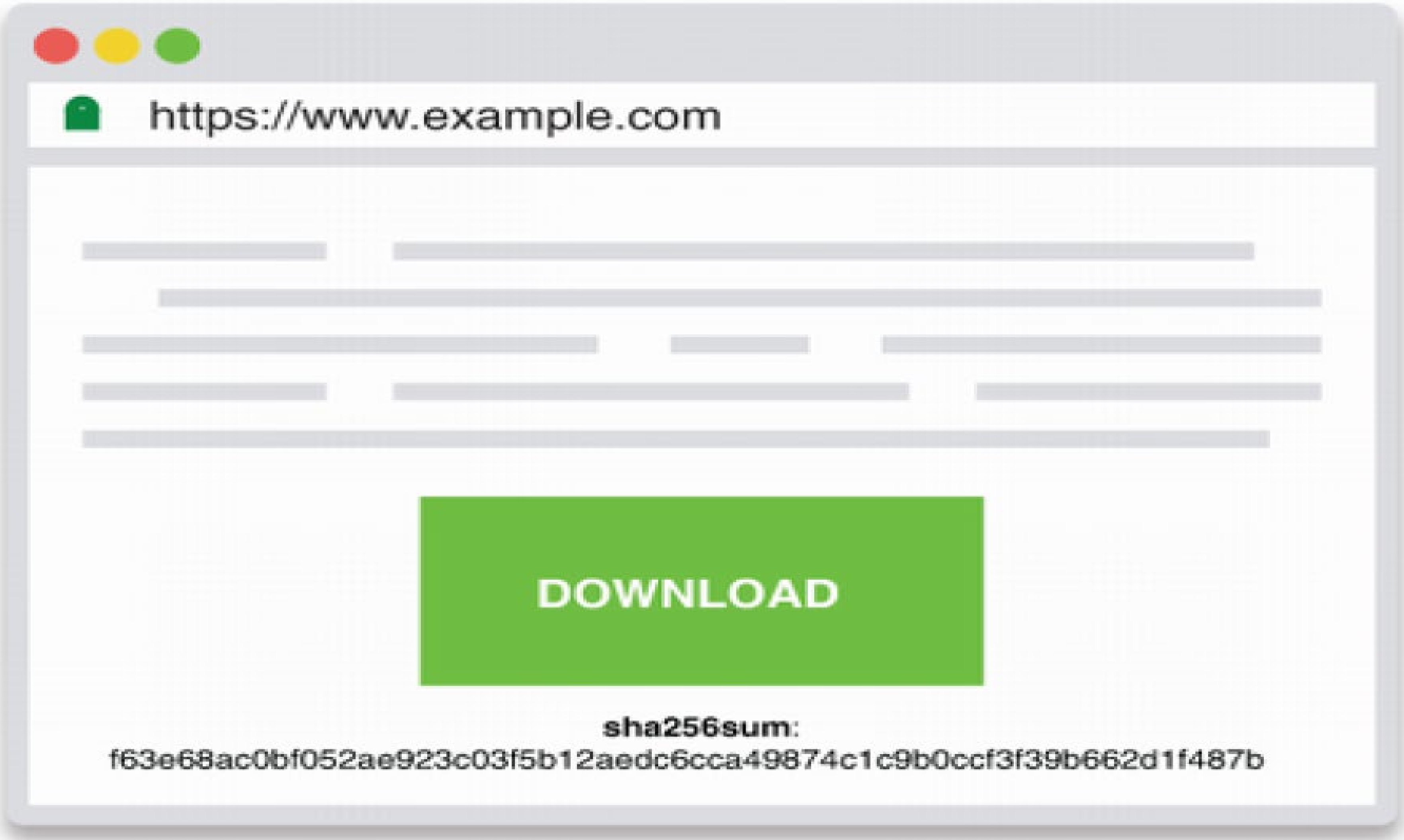
```
$ echo -n "hello" | openssl dgst -sha256
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
$ echo -n "hella" | openssl dgst -sha256
70de66401b1399d79b843521ee726dcec1e9a8cb5708ec1520f1f3bb4b1dd984
```

Important Note

- Hash is not an encryption! It is not meant to be!
- It is mainly used as a message integrity check
- Remember the CIA rule in computer security
  - Confidentiality – (strong encryption)
  - Integrity – (strong hash)
  - Accessibility

A Real Scenario

- In front of you, a download button is taking a good chunk of the page.
- You can read the letters *DOWNLOAD*, and clicking this seems to redirect you to a different website containing a file.
- Below it, lies a long string of unintelligible letters: `f63e68ac0bf052ae923c03f5b12aedc6cca49874c1c9b0ccf3f39b662d1f487b`
- It is followed by what looks like an acronym of some sort: sha256sum
- Sound familiar? You've probably downloaded something in your past life that was also accompanied with such an odd string (figure 2.1).



sha256sum:
f63e68ac0bf052ae923c03f5b12aedc6cca49874c1c9b0ccf3f39b662d1f487b

Figure 2.1 A web page linking to an external website containing a file.

A Real Scenario

- If you’ve ever wondered what was to be done with that long string:

1. Click the button to download the file
2. Use the SHA-256 hash algorithm to *hash* the downloaded file
3. Compare the output (the digest) with the long string displayed on the web page

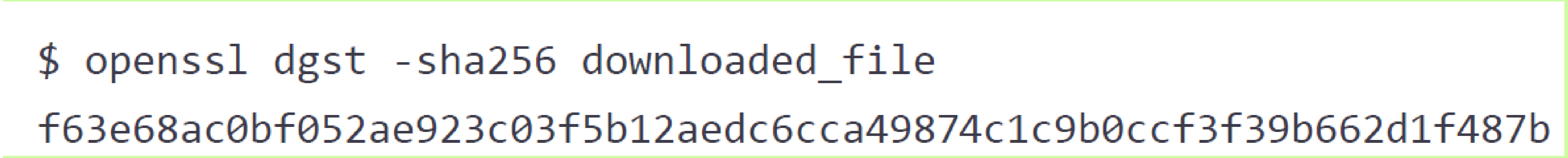
- This allows you to verify that you downloaded the right file.

A Real Scenario

- There is only 1 caveat to what I just said in previous slide.
- The remaining question: if the hash reflected on the web page of download coincides with your computed hash, does it always mean the files have not been tampered with, assuming the hash used is a real secured hash?
- See Tutorial!

Checking Hash

- To try hashing something, you can use the popular OpenSSL library.
- It offers a multipurpose command-line interface (CLI) that comes by default in a number of systems including macOS.
- For example, this can be done by opening the terminal and writing the following line:



```
$ openssl dgst -sha256 downloaded_file
f63e68ac0bf052ae923c03f5b12aedc6cca49874c1c9b0ccf3f39b662d1f487b
```

Some Common & Important Hash Functions

- 1992: MD5 (Rivest, 128-bit hash) – broken
- 1993: SHA1 (NSA, 160-bit hash) - broken
- 2001: SHA2 (SHA256, SHA512) - NSA
- 2012: SHA3 KECCAK (Joan Daemen, designer of AES)
  - 256-bit, 512-bit etc
  - KECCAK (pronounced "catch-ack"),winner of international new generation hash function competition

Classical Hash Functions

- Prior to 2003, no one believe hash functions such as MD5, SHA1 can be broken

Hash Queen Wang XiaoYun: 王小云

- Wang Xiaoyun is a Chinese cryptographer. She is a professor in the Department of Math of Shandong University and an academician of the Chinese Academy of Sciences
- At CRYPTO 2004, she demonstrated collision attacks against MD5
- They received a standing ovation for their work

30 Nov 2007: Hash King - Marc Stevens

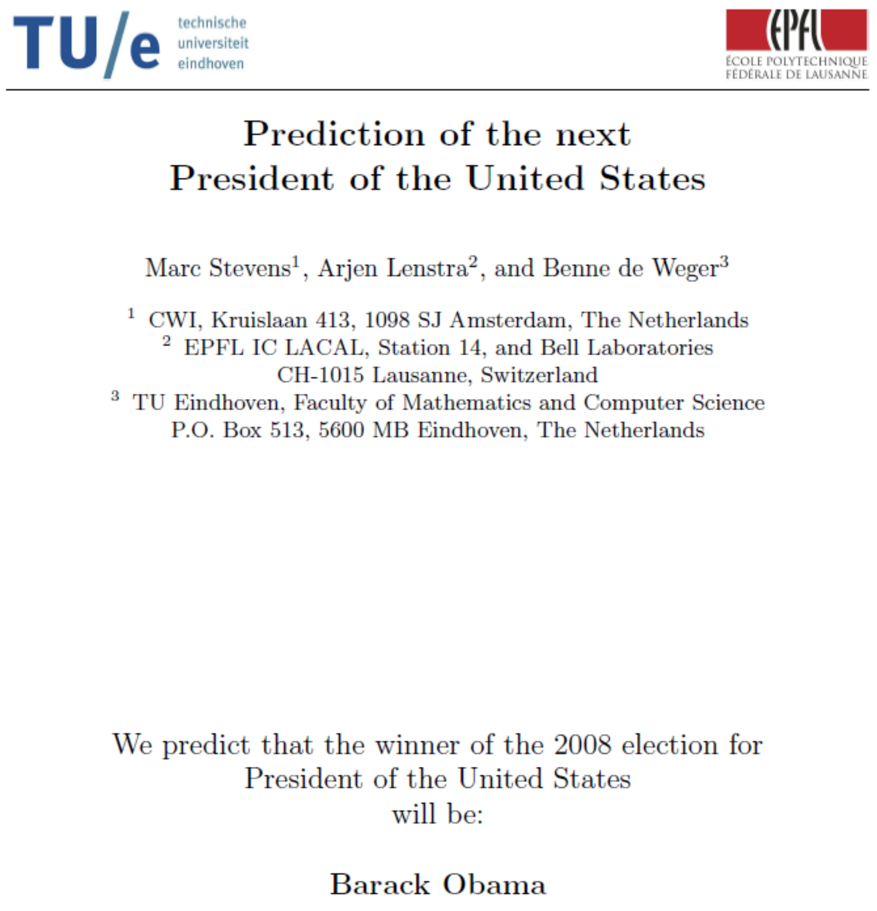
- **We have used a Sony Playstation 3 to correctly predict the outcome of the 2008 US presidential elections. In order not to influence the voters we keep our prediction secret, but commit to it by publishing its cryptographic hash on this website. The document with the correct prediction and matching hash will be revealed after the elections.**

Marc released MD5 hash of his predictions for 2008 president usa elections on Nov 2007

3D515DEAD7AA16560ABA3E9DF05CBC80

https://www.win.tue.nl/hashclash/Nostradamus/

After 2008 Elections, they released Obama.pdf with correct MD5 hash!



technische universiteit eindhoven

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Prediction of the next
President of the United States

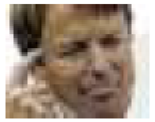
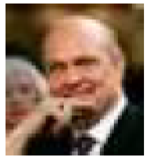
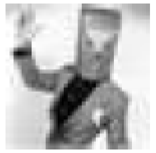
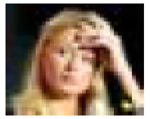
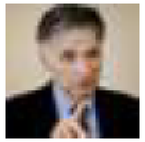
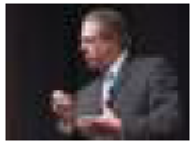
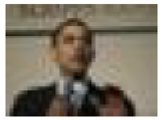
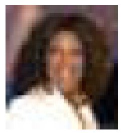
Marc Stevens[1], Arjen Lenstra[2], and Benne de Weger[3]

[1] CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
[2] EPFL IC LACAL, Station 14, and Bell Laboratories
CH-1015 Lausanne, Switzerland
[3] TU Eindhoven, Faculty of Mathematics and Computer Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

We predict that the winner of the 2008 election for
President of the United States
will be:

Barack Obama

We have prepared twelve different predictions, ten of which are shown in the table below.

A:		John Edwards.pdf	G:		Fred Thompson.pdf
B:		John McCain.pdf	H:		(hidden)
C:		Mitt Romney.pdf	I:		Paris Hilton.pdf
D:		Ralph Nader.pdf	J:		Al Gore.pdf
E:		(hidden)	K:		Jeb Bush.pdf
F:		Barack Obama.pdf	L:		Oprah Winfrey.pdf

All twelve documents we prepared, the ten given above and two hidden ones, have the MD5 hash value

```
3D515DEAD7AA16560ABA3E9DF05CBC80.
```

Colliding X.509 Certificates for Different Identities
By Marc Stevens, Dec 2008

- Here are the certificates (downloadable):

  colliding certificate number 1 (on the name of "Arjen K. Lenstra")
  colliding certificate number 2 (on the name of "Marc Stevens")

  Here is the CA certificate with which the colliding certificates can be validated:

  CA certificate (on the name of "Hash Collision CA")

We announce a pair of X.509 certificates with the following properties:

colliding

The to-be-signed parts of the certificates form a collision for the MD5 hash function.
This means that their signatures are identical.
These signatures have been generated by a Certification Authority that uses MD5.

different identities

One certificate has "Arjen K. Lenstra" as Common Name, the other one has "Marc Stevens" instead.
They also have different O-fields (for Organization) and serial numbers.
These values were chosen before the collision construction started.
This aspect is the main difference with our Colliding X.509 Certificates construction of last year,
where we had colliding certificates but with identical owner names.

MD5 TOTALLY BROKEN!

2 NTU Experts in crypto: (SPMS)

- Prof Wu Hong Jun (hash, stream & block ciphers)
  - Top 5 eStream
  - Top 5 SHA3 HASH COMPETITION

- Prof Thomas Peyrin (hash & block ciphers)

NIST SHA3 Competition

- In 2006, NIST started to organize the NIST hash function competition to create a new hash standard, SHA-3..
- Keccak (Joan Daemen) was accepted as one of the 51 candidates.
- In July 2009, 14 algorithms were selected for the second round. Keccak advanced to the last round in December 2010.
- On October 2, 2012, Keccak was selected as the winner of the competition.
- https://www.youtube.com/watch?v=ucw5ZW291V0

FINAL WORDS ON HASH

- If need to use (fast secure) hash, go for KECCAK
- Dont attempt to create or believe someone’s proprietary hash
- For ascertaining large size file integrity, we do not “sign” the large file
- We sign the hash of the large file, of course by strong hash!
- SHA256,SHA512 in use, no attacks yet (they are known as SHA2)
- If possible, migrate to KECCAK