Lecture 4: Message Integrity

Ryan Cunningham
University of Illinois
ECE 422/CS 461 – Fall 2017

Security News

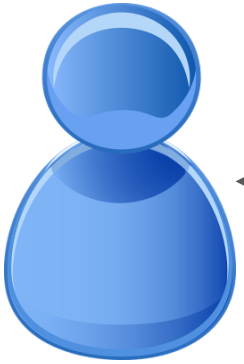
- Krebs investigated MalwareTech/Marcus Hutchins. Looks like he did some bad stuff.
- TechCrunch journalist finds T-Mobile SIM card spoofing used for bitcoin phishing scams
- Kenyan election thrown out by supreme court, opposition alleges voting machines hacked

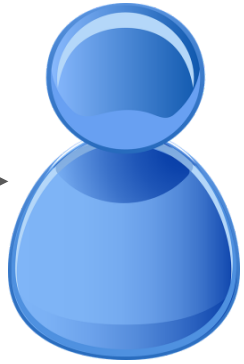
INTRO TO CRYPTOGRAPHY

Cryptography is the study/practice of techniques for s**ecure communication**, even in the presence of powerful **adversaries** who have **control** over the **underlying channel**

Eve (or Mallory)

Wiretaps the channel
Drops messages
Tampers with messages





Send messages to each other over a channel (e.g., a shoe string, a copper wire, a TCP socket)

Alice

Bob

Goals of cryptography module

1. **Understand the interfaces of basic crypto primitives:**
   Hashes, MACs, symmetric encryption, public key encryption, digital signatures, key exchange, proof of work
2. **Apply the adversarial mindset to crypto protocols**
3. **Appreciate the wisdom of the saying:**
   "Don't roll your own crypto!"
4. **Familiarity with concepts, vocabulary**
   Lectures are for breadth

Cryptography is **NOT** just about encryption!

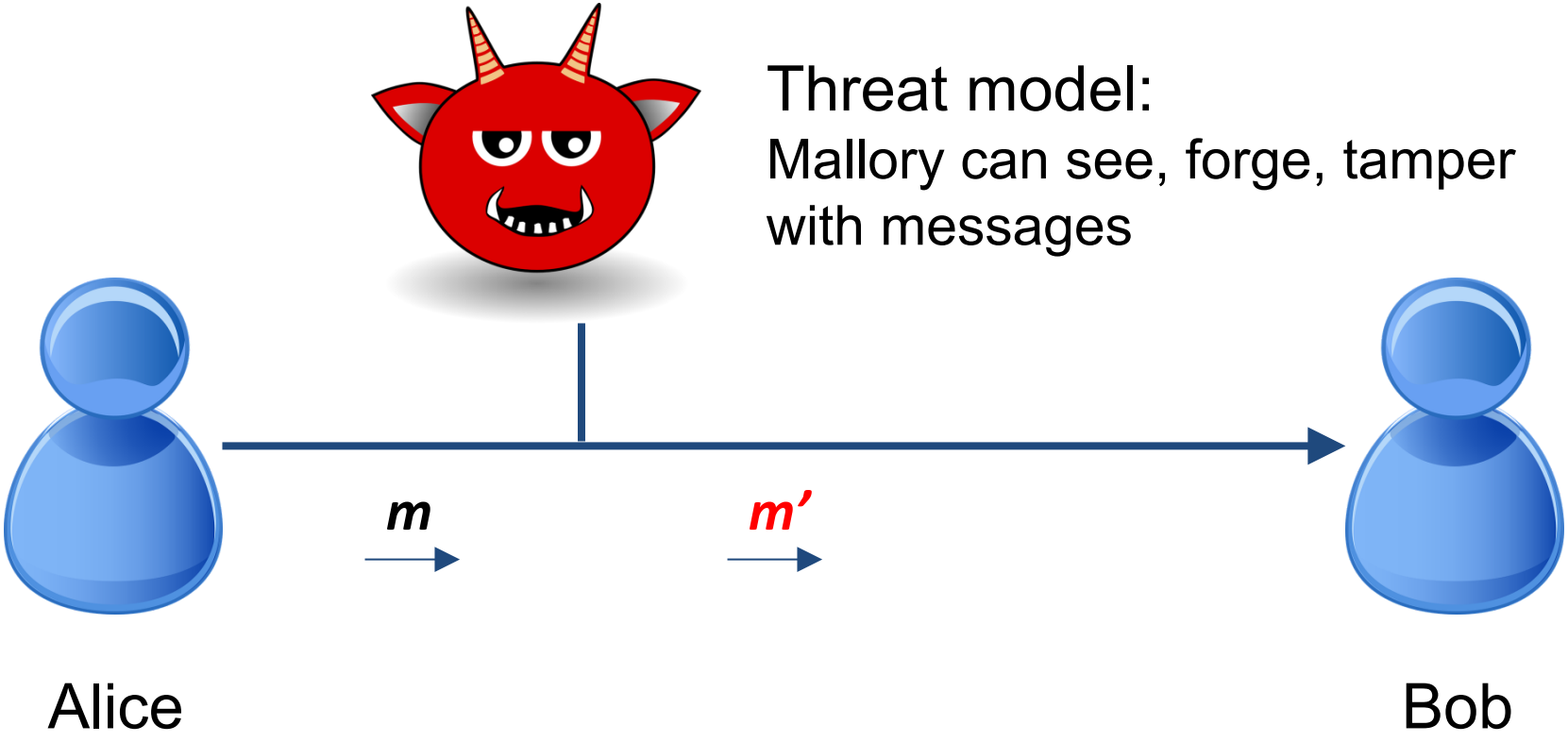
- Confidentiality (secrecy & privacy)
- Integrity (tamper resilience)
- Availability
- Non-repudiability (non-deniability)

…. other properties, too!

Message Integrity (Hashes & MACs)

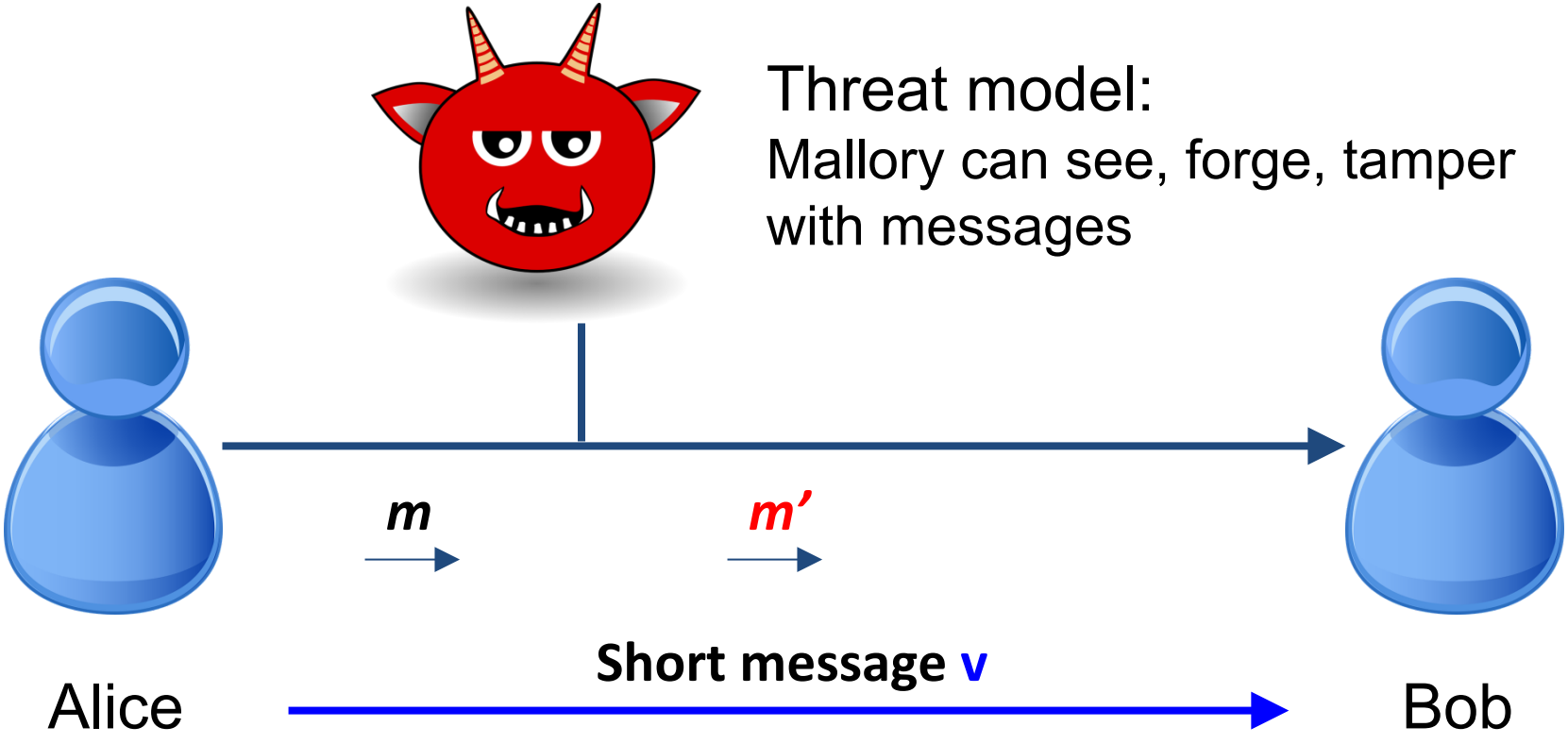
Goal: Secure File Transfer

Alice wants to send file ***m*** to Bob (let's say, a 4 Gigabyte movie)
Mallory wants to trick Bob into accepting a file Alice didn't send



Goal: Secure File Transfer

Alice wants to send file ***m*** to Bob (let's say, a 4 Gigabyte movie)
Mallory wants to trick Bob into accepting a file Alice didn't send

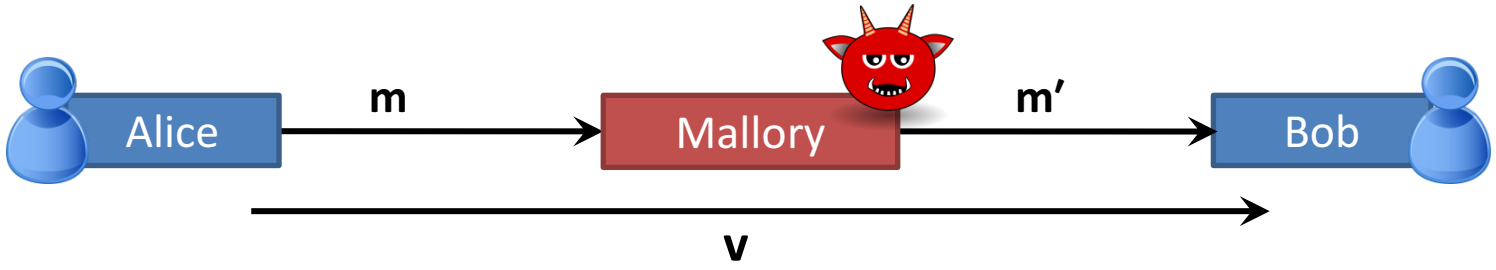


Setup assumption: *Securely transfer a short message!*

Solution: Collision Resistant Hash Function (CRHF)

Hash Function ***h***: $\{0,1\}^* \rightarrow \{0,1\}^{256}$ (or other fixed number)

1. Alice computes **v** := ***h***(**m**)
2. Alice transfers **v** over secure channel, **m** over insecure channel



3. Bob verifies that **v** = ***h***(**m'**), accepts file iff this is true

Function ***h*** ? We're sunk if Mallory can compute **m'** ≠ **m** where ***h***(**m**) = ***h***(**m'**)! A *collision*!

Contrast with: "checksums" e.g. CRC32.... defend against random errors, not a deliberate attacker!

(Bad) Example

Hash function that XORs all the bytes in the data

Input: arbitrary string

Output: one byte

`h(“Hello”)=`

`0x48^0x65^0x6c^0x6c^0x6f=0x42`

Hash function properties

Good hash functions should make it difficult to find:

First pre-image:

given h(m), find m

Second pre-image:

given m_1, find m_2 s.t. $h(m_1) = h(m_2)$

Collision:

find *any* m_1 != m_2 s.t. $h(m_1) = h(m_2)$

Preimage resistant (one-way)

Given h, hard to find m such that h(m)=x

We cannot undo the hash function

Can’t work back from the hash to find the input

Assuming m is a message, an attacker can’t figure out the contents using the hash

XORing bytes is **not** preimage resistant

‘H’ = 1001000b

‘h’ = 1101000b

0x42 =1000010b

Second preimage resistant

Given m, hard to find m≠m’ such that H(m)=H(m’)

For a given input, we can’t find a different input that produces the same output

Assuming m is a message, an attacker can’t find a different message with the same hash

XORing bits is **not** second preimage resistant

`h(“Frenchify”)=0x42`

`h(“Ninja”)=0x42`

`h(“Tyrannosaurus”)=0x42`

Collision resistant

Hard to find m and m’ such that H(m)=H(m’)

Can’t find ***pair*** of inputs that produce the same output

Attacker can’t produce two messages with the same hash

XORing bits does **not** have this property

```
h(“anatomopathologic”)=0x7b

h(“undoingness”)=0x7b
```

Avalanche Effect

Property of good cryptographic hashes (and block ciphers)

Flipping one bit of the input causes all output bits to change with 50% probability.

Probability each bit flips is independent of others

h_good(“Hello”)=10110111

h_good(“hello”)=10001011

1 bit in input flipped, 4/8 output bits flipped

Confusion and diffusion

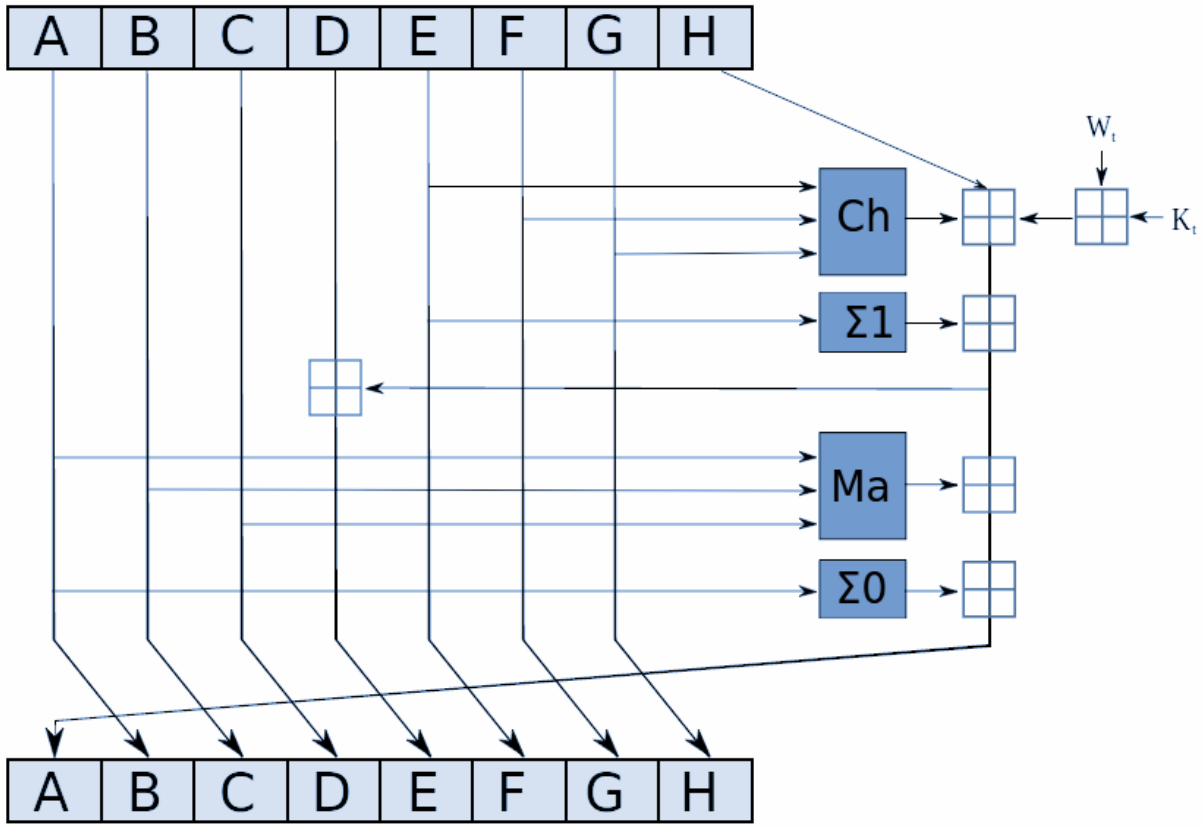
- Two desirable properties of a crypto hash (and block ciphers)
- **Diffusion** - changing a character in the input should change many characters in the output (and vice versa)
- **Confusion** - each character of output should depend on several parts of the input
- Proposed in 1945 by Claude Shannon

SHA-256

What is SHA256?

```
$ sha256sum file.dat
```

The **SHA256** compression function, **h**



Cryptographic hash

Input: arbitrary length data
(No key)
Output: 256 bits

Built with compression function, ***h***

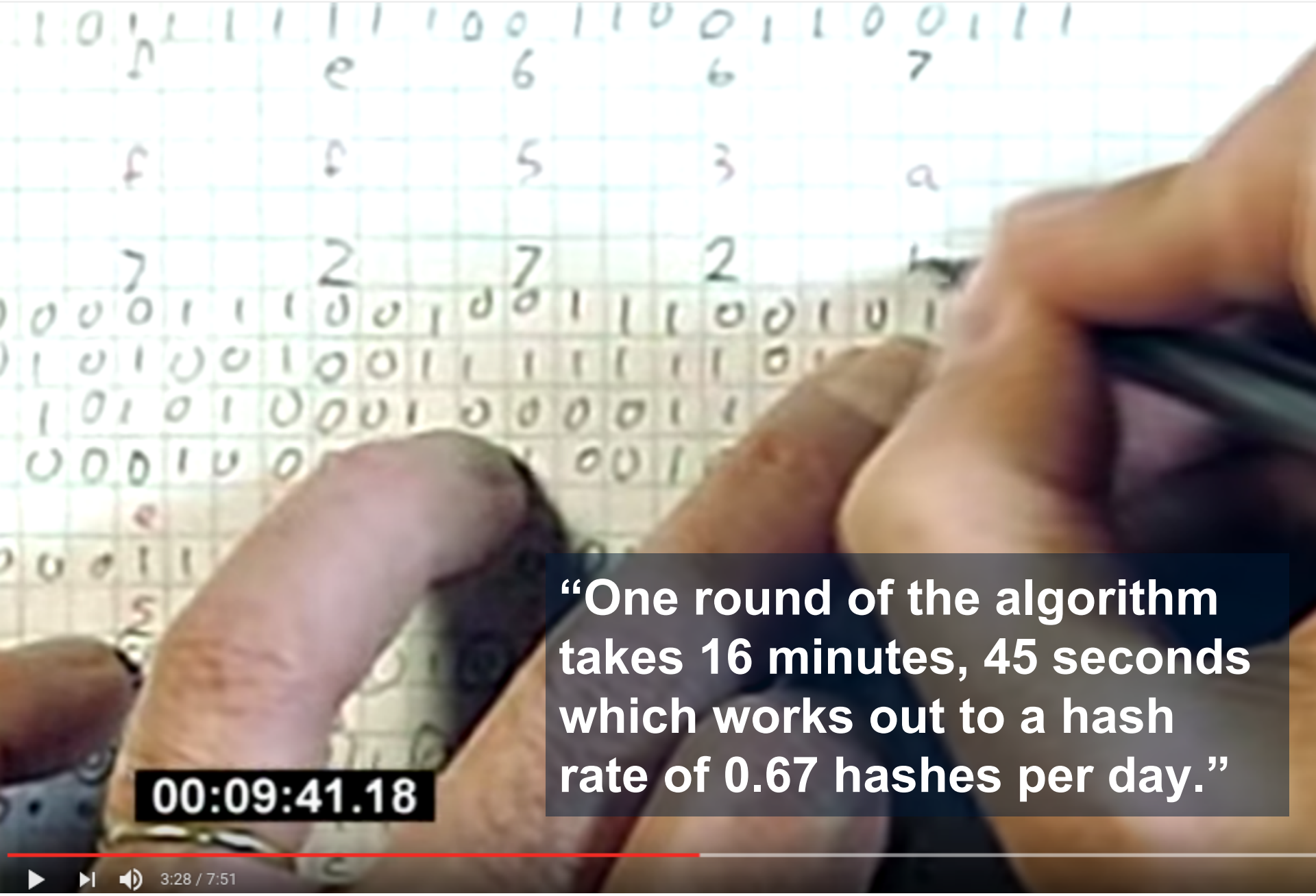
(256 bits, 512 bits) in → 256 bits out

Designed to be really hairy (64 rounds of this)!

Confusion and Diffusion

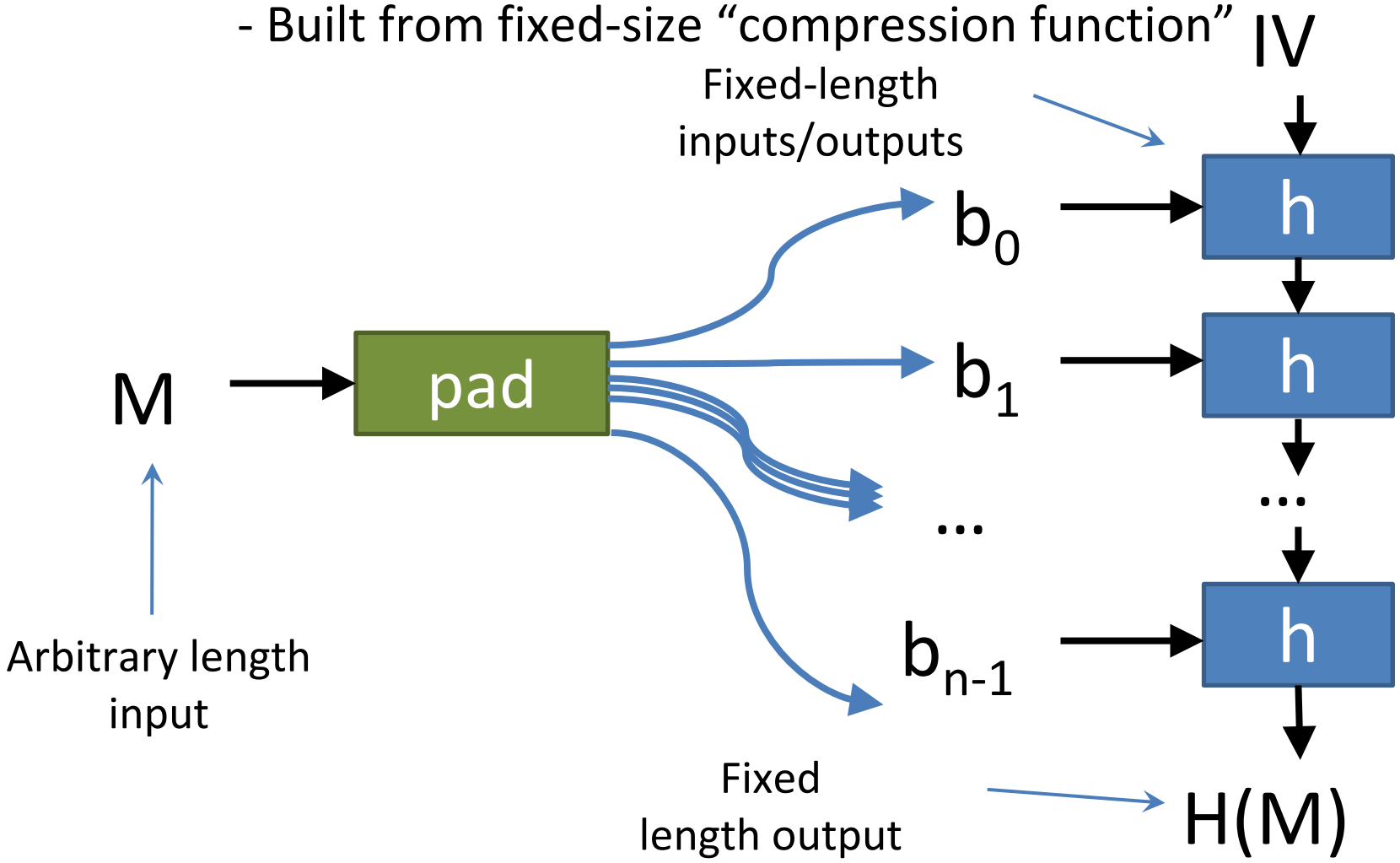
$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

https://www.youtube.com/watch?v=y3dqhixzGVo



Merkle–Damgård Construction

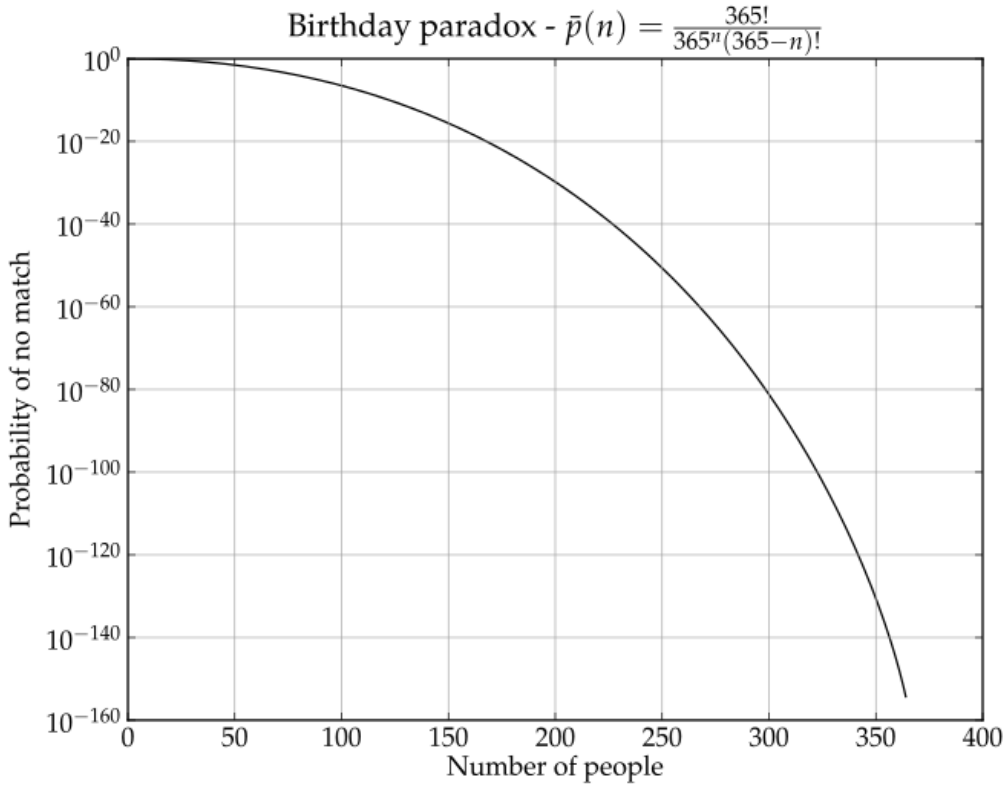
- Arbitrary-length input
- Fixed-length output
- Built from fixed-size “compression function”



Attacking hashes

Birthday problem

In a group of n people, what is the probability two share the same birthday?



Birthday problem

As size of a set increases *linearly*

Number of pairs increases as the *square*

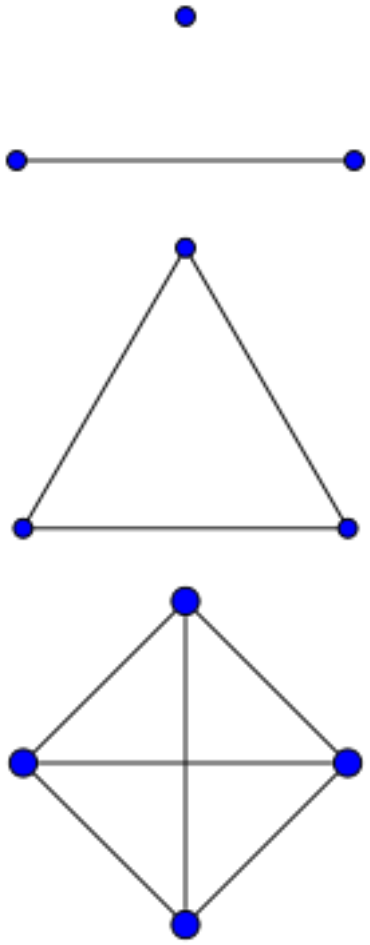
2 people = 1 pair

3 people = 2 pairs

4 people = 6 pairs

… $\binom{n}{2}$

n people = = n(n-1)/2 = $O(n^2)$



How do you find a collision?

Pigeonhole principle: collisions must exist

Input space {0,1}* larger than output $\{0,1\}^{256}$

Birthday attack: build a table with 2^{128} entries

With ~50% probability, have a collision

Cycle finding: "Tortoise and hare" algorithm

$h(x),\ h(h(x)),\ h(h(h(x)),\ ..,\ h^i(x)$

These are **generic** - actual attacks rely on **structure** of the particular hash function

Most cryptographic primitives come with a **security parameter**

k or λ

- Often Corresponds to a key size
- Cryptography protocols run in **polynomial** time

i.e., as a function of λ

O(poly(λ))

- Ideally, we can show that the chance of failure is **negligible**, or **vanishingly** small as a function of λ

O(negl(λ))

Concrete Parameterization

How large of a digest size should we choose?

1. Estimate an attacker's budget

e.g., the entire NSA

2. Factor in hardware improvements

3. Consider the best known attacks

Reduction from protocol to well-studied problem

4. Add a safety margin

If all goes well, adding 1 bit increases search space by 2x

SHA-512 (assuming no weakness…)

Second preimage resistance:

Try $2^{512}=1.3\text{x}10^{154}$ inputs

Far, far more than the number of atoms in the universe

Collision resistance:

Try $2^{512/2}=2^{256}=1.6\text{x}10^{77}$ pairs of inputs

Checking 10^{13} pairs/micro second would take $1.8\text{x}10^{56}$ years

Other hash functions:

MD5

- Once ubiquitous
- Broken in 2004
- Easy to find collisions today

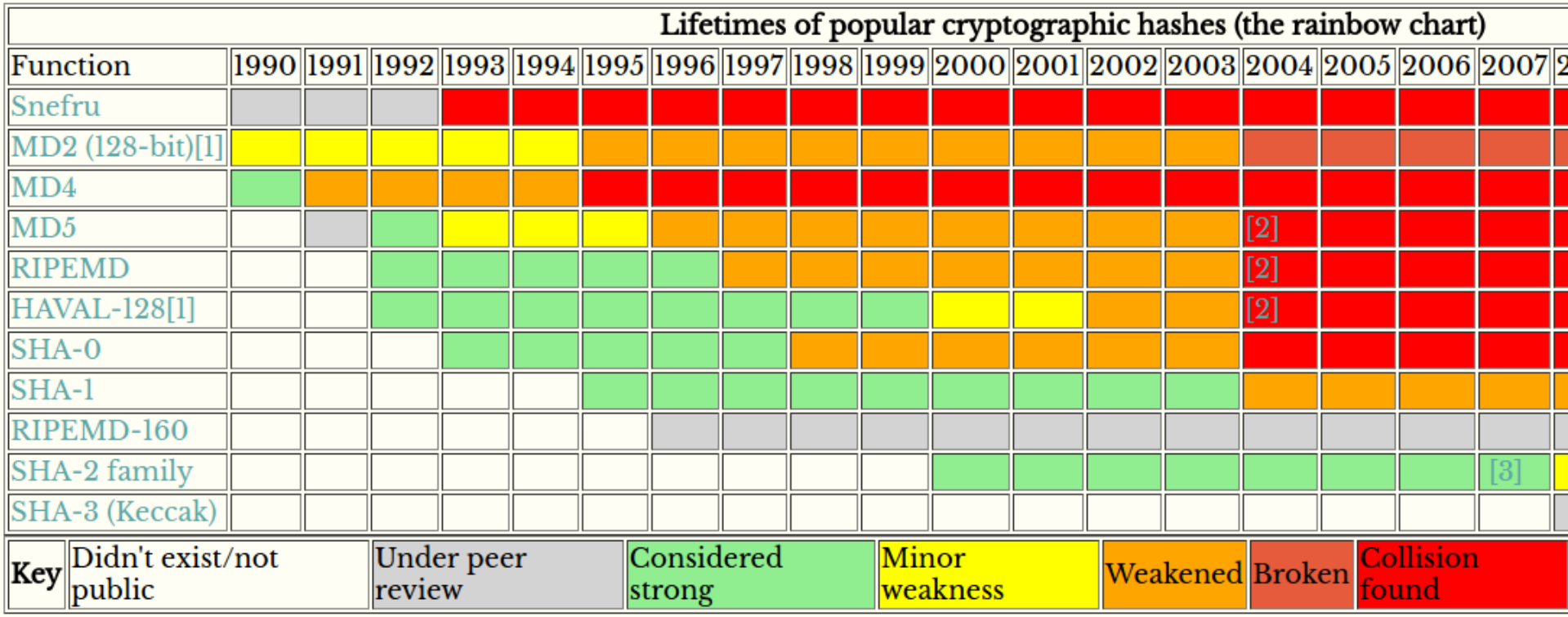
SHA1

- Currently widely used
- Collisions recently found!
- Don’t use in new applications

SHA3

- Different construction: “Sponge”
- Not susceptible to ***length-extension***

http://valerieaurora.org/hash.html



[1] Note that 128-bit hashes are at best 2^64 complexity to break; using a 128-bit hash is irresponsible based on sheer digest length.

[2] What happened in 2004? Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu happened.

[3] In 2007, the NIST launched the SHA-3 competition because "Although there is no specific reason to believe that a practical attack on any of the SHA-2 family of hash functions is imminent, a successful collision attack on an algorithm in the SHA-2 family could have catastrophic effects for digital signatures." One year later the first strength reduction was published.

The Hash Function Lounge has an excellent list of references for most of the dates. Wikipedia now has references to the rest.

Schneier on Security

SHA-1 Collision Found

The first collision in the SHA-1 hash function has been found.

This is not a surprise. We've all expected this for over a decade, watching computing power increase. This is why NIST standardized SHA-3 in 2012.

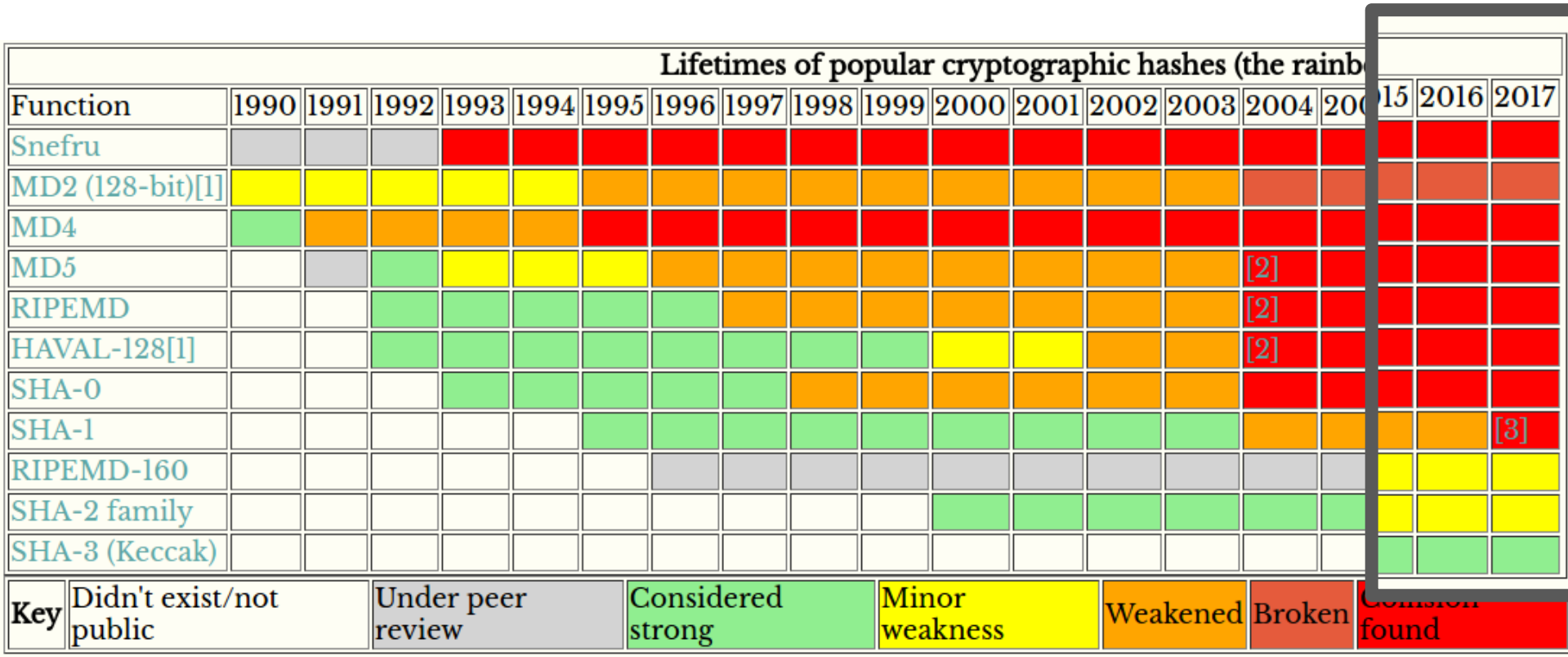
EDITED TO ADD (2/24): Website for the collision. (Yes, this brute-force example has its own website.)

EDITED TO ADD (3/7): This 2012 cost estimate was pretty accurate.

Tags: cryptanalysis, cryptography, hashes, SHA-1

Posted on February 23, 2017 at 3:29 PM • 31 Comments

http://valerieaurora.org/hash.html



[1] Note that 128-bit hashes are at best 2^64 complexity to break; using a 128-bit hash is irresponsible based on sheer digest length.

[2] What happened in 2004? Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu happened.

[3] In 2007, the NIST launched the SHA-3 competition because "Although there is no specific reason to believe that a practical attack on any of the SHA-2 family of hash functions is imminent, a successful collision attack on an algorithm in the SHA-2 family could have catastrophic effects for digital signatures." One year later the first strength reduction was published.

The Hash Function Lounge has an excellent list of references for most of the dates. Wikipedia now has references to the rest.