

# Hashes and certificates

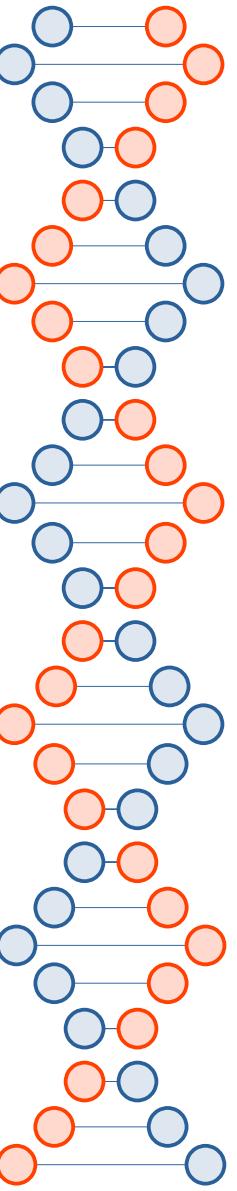
CSE 548 Spring 2026  
[jedimaestro@asu.edu](mailto:jedimaestro@asu.edu)

How do you know who you're talking to when you do encryption?

Authentication and non-repudiation

What if your private key gets stolen?

Forward secrecy and future secrecy



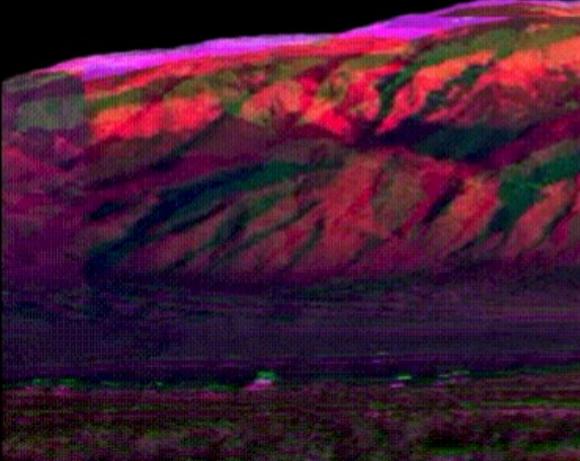
# Check out...

[https://media.ccc.de/v/25c3-3023-en-making\\_the\\_theoretical\\_possible](https://media.ccc.de/v/25c3-3023-en-making_the_theoretical_possible)

Also check out:

<https://www.win.tue.nl/hashclash/rogue-ca/>

# BREAKPOINT



## Certificate Viewer: breakpointingbad.com

### General Details

#### Issued To

Common Name (CN) breakpointingbad.com  
Organization (O) <Not Part Of Certificate>  
Organizational Unit (OU) <Not Part Of Certificate>

#### Issued By

Common Name (CN) R3  
Organization (O) Let's Encrypt  
Organizational Unit (OU) <Not Part Of Certificate>

#### Validity Period

Issued On Thursday, December 22, 2022 at 10:57:13 PM  
Expires On Wednesday, March 22, 2023 at 10:57:12 PM

#### Fingerprints

SHA-256 Fingerprint 81 05 41 B0 19 8B 06 9C 90 20 7F B3 EE 60 2E AB  
BD 64 25 F9 D8 DE 87 7D FD 70 34 AC F9 F5 DE 92  
SHA-1 Fingerprint C1 95 59 2C 66 12 BC 36 71 7E 99 C9 60 98 12 0A  
B8 02 1D 47

We are a non-profit founded in 20  
combined experience focusing on

## Certificate Viewer: breakpointingbad.com

X

General

Details

### Issued To

Common Name (CN) breakpointingbad.com  
Organization (O) <Not Part Of Certificate>  
Organizational Unit (OU) <Not Part Of Certificate>

### Issued By

Common Name (CN) R3  
Organization (O) Let's Encrypt  
Organizational Unit (OU) <Not Part Of Certificate>

### Validity Period

Issued On Thursday, December 22, 2022 at 10:57:13 PM  
Expires On Wednesday, March 22, 2023 at 10:57:12 PM

### Fingerprints

SHA-256 Fingerprint 81 05 41 B0 19 8B 06 9C 90 20 7F B3 EE 60 2E AB  
BD 64 25 F9 D8 DE 87 7D FD 70 34 AC F9 F5 DE 92  
SHA-1 Fingerprint C1 95 59 2C 66 12 BC 36 71 7E 99 C9 60 98 12 0A  
B8 02 1D 47

## Certificate Fields

▼ breakpointingbad.com

    ▼ Certificate

        Version

        Serial Number

        Certificate Signature Algorithm

        Issuer

    ▼ Validity

        Not Before

## Field Value

04:B5:2A:1D:FD:B3:AC:F1:34:37:27:94:9F:F5:A8:5A:12:E6

Export...

breakpointingbad.com

## Certificate Fields

▼ breakpointingbad.com

    ▼ Certificate

        Version

        Serial Number

        Certificate Signature Algorithm

        Issuer

    ▼ Validity

        Not Before

## Field Value

PKCS #1 SHA-256 With RSA Encryption

## Certificate Fields

▼ breakpointingbad.com

▼ Certificate

Version

Serial Number

Certificate Signature Algorithm

Issuer

▼ Validity

Not Before

## Field Value

CN = R3

O = Let's Encrypt

C = US

## Certificate Fields

Certificate Signature Algorithm

Issuer

▼ Validity

Not Before

Not After

Subject

▼ Subject Public Key Info

Subject Public Key Algorithm

## Field Value

3/22/23, 10:57:12 PM MST

Export...

## Certificate Fields

Certificate Signature Algorithm

Issuer

▼ Validity

Not Before

Not After

Subject

▼ Subject Public Key Info

Subject Public Key Algorithm

## Field Value

CN = breakpointingbad.com

## Certificate Fields

Not After

Subject

▼ Subject Public Key Info

Subject Public Key Algorithm

Subject's Public Key

▼ Extensions

Certificate Key Usage

Extended Key Usage

## Field Value

Modulus (2048 bits):

DC DA F0 96 47 5C 62 91 27 27 AD B2 95 EE 3D 51  
CF 26 EB EC 27 EE ED 2E 9F DA 1D BF 83 2F 12 F0  
EA CC 96 5B 8C C1 3E A1 C6 46 90 4D E5 93 20 E1  
5C 9B 62 BB 82 3A 7F 77 7C 85 CB 8C F3 0F B9 0D  
38 24 9C 0D 39 8C FF F4 B5 AD 0A 94 75 AA F9 41

Export

## Certificate Fields

▼ breakpointingbad.com

▼ Certificate

Version

Serial Number

Certificate Signature Algorithm

Issuer

▼ Validity

Not Before

## Field Value

98 2F 52 F3 68 5E 6D BC 18 2C 93 42 8B C5 41 D1  
40 B4 0F 53 D9 BD BA 22 F9 52 90 76 37 F0 C4 56  
31 F8 8D C7 B8 21 3E FB 0F 83 B8 A7 CF F3 B4 A1

Public Exponent (17 bits):  
01 00 01

Export...

## Certificate Hierarchy

▼ Builtin Object Token:ISRG Root X1

▼ R3

breakpointingbad.com

## Certificate Fields

Subject's Public Key

▼ Extensions

Certificate Key Usage

Extended Key Usage

Certificate Basic Constraints

Certificate Subject Key ID

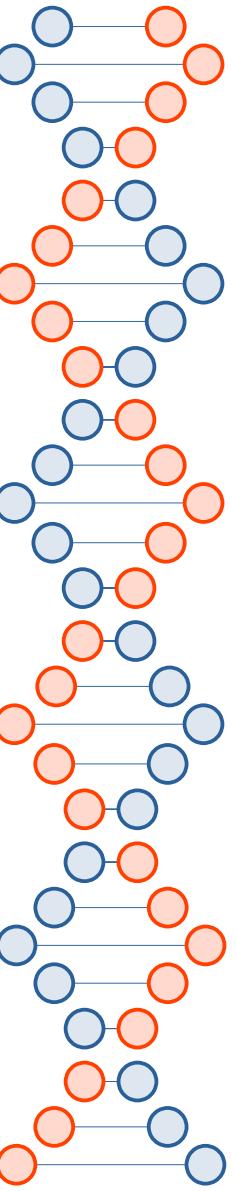
Certification Authority Key ID

Authority Information Access

## Field Value

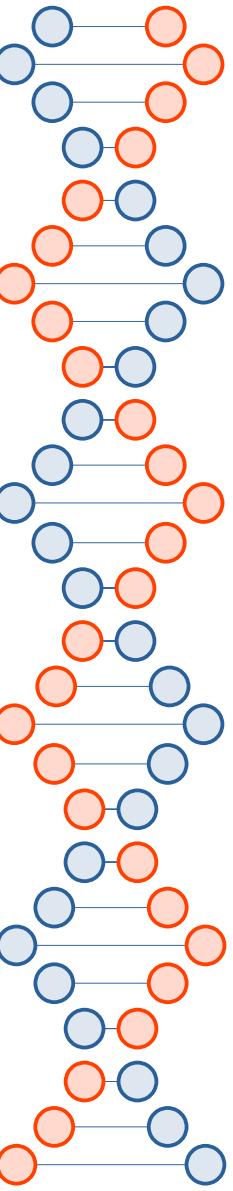
Critical

Is not a Certification Authority



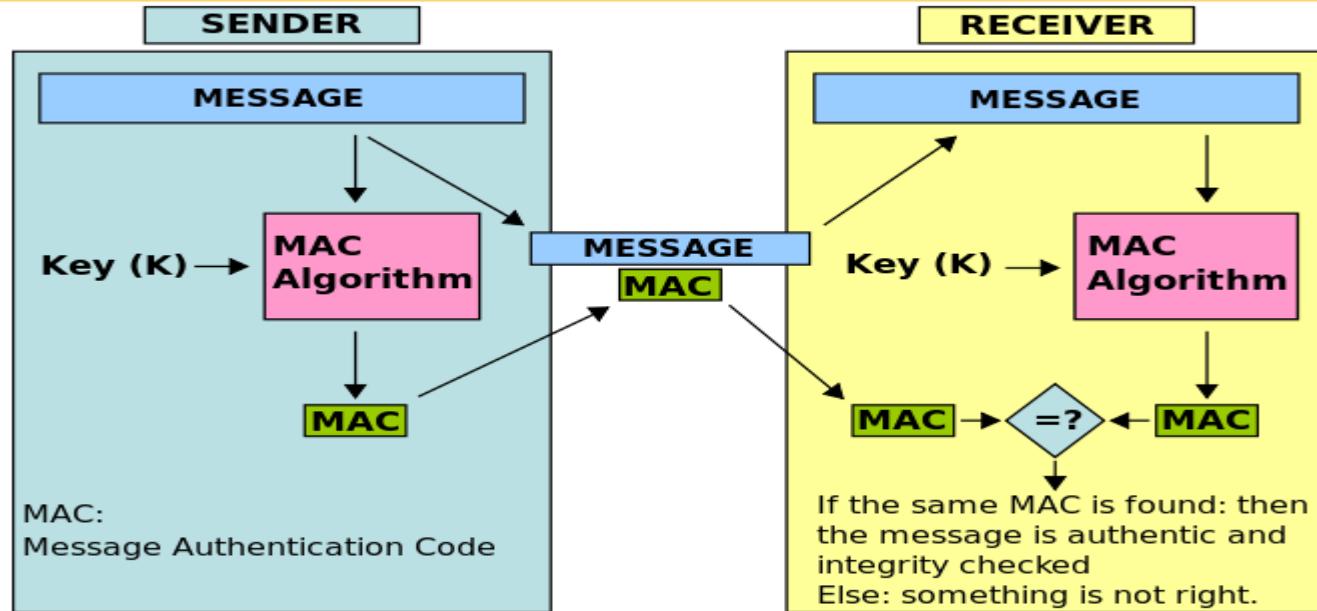
# Why hash functions?

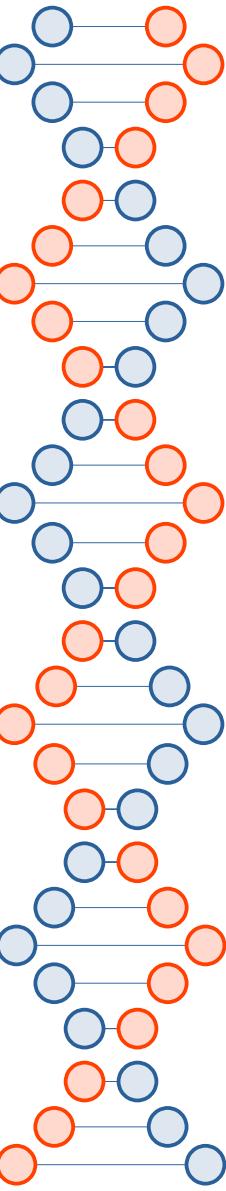
- Speed
  - Symmetric crypto is generally faster than asymmetric
  - Hashes are generally faster than either
- Error detection (e.g., checksum)
- Security and privacy



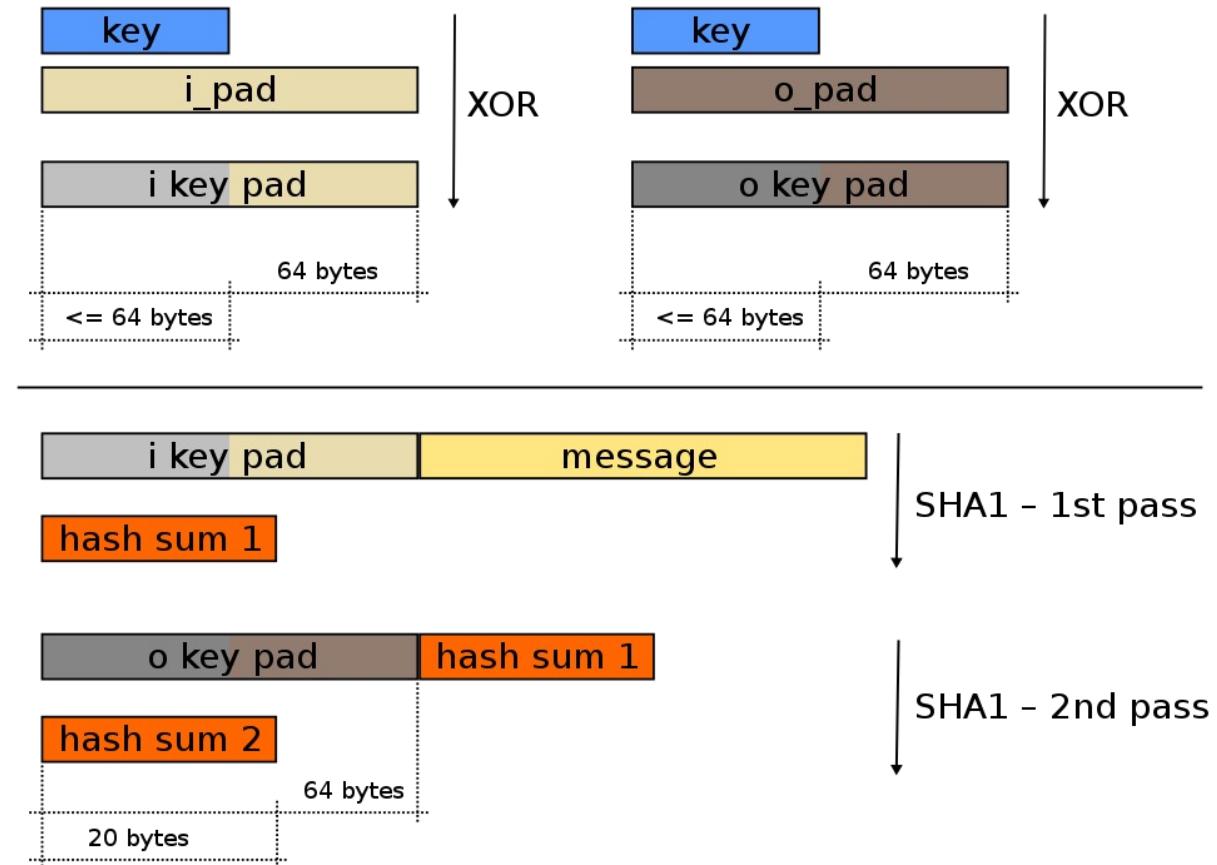
# Why cryptographic hash functions?

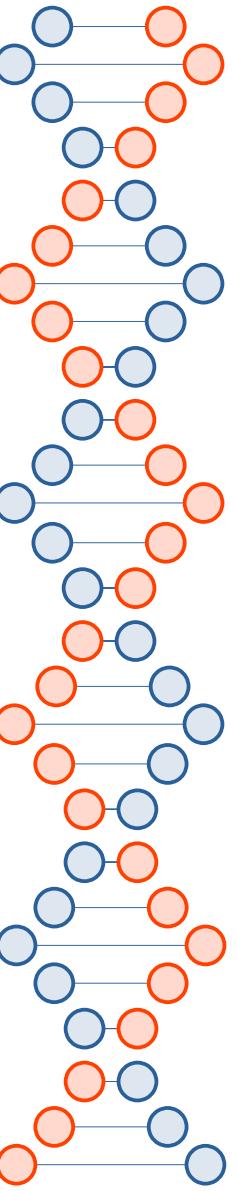
- Unique identifier for an object
- Integrity of an object
  - *E.g.*, message authentication codes
- Digital signatures
  - Sign the digest
    - *E.g.*, 1024-bit RSA, 100MB message, 256-bit digest
- Passwords
- Proof of work





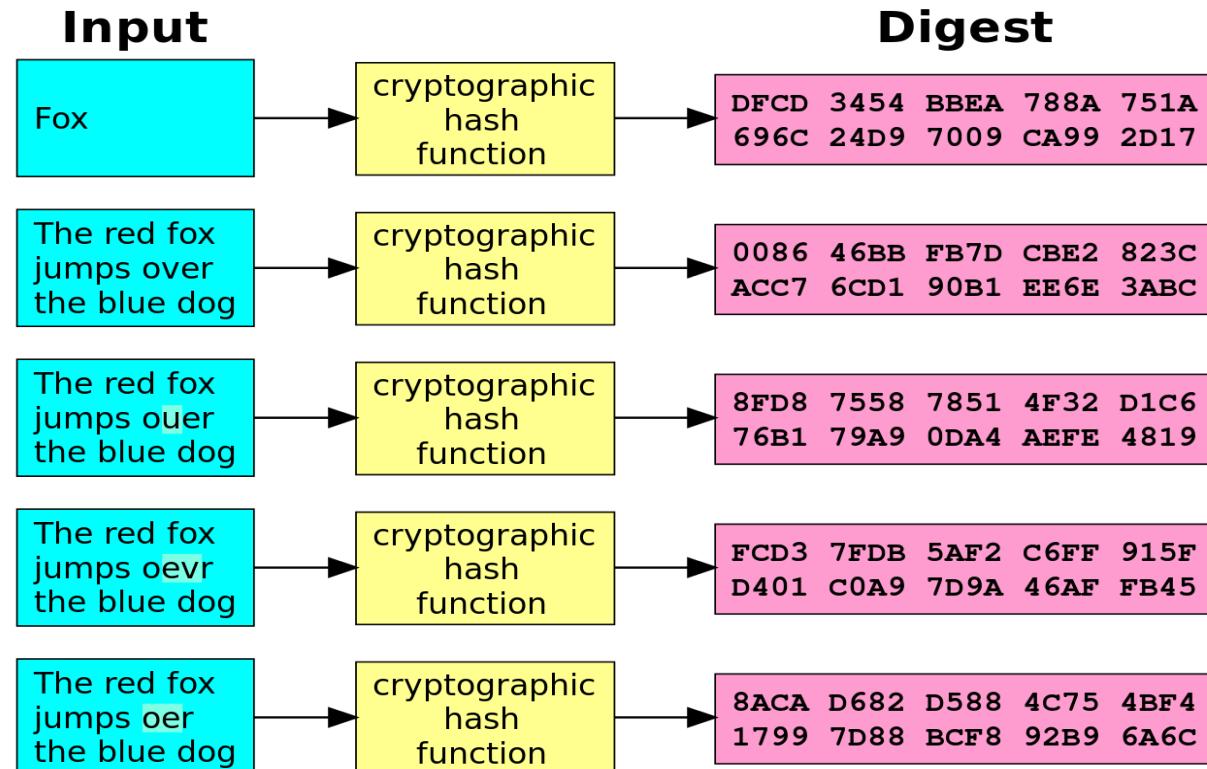
# <https://en.wikipedia.org/wiki/HMAC>

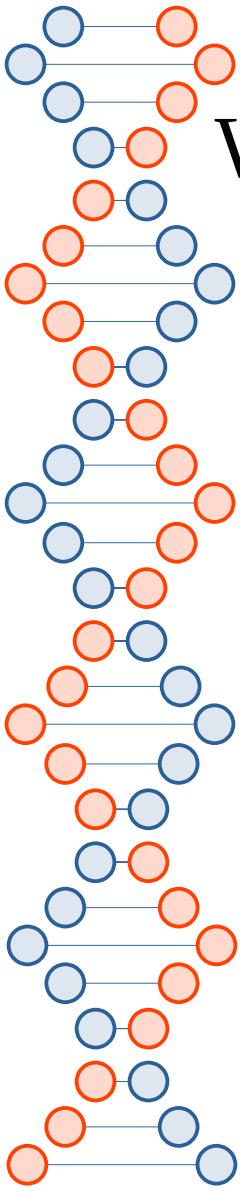




<review>

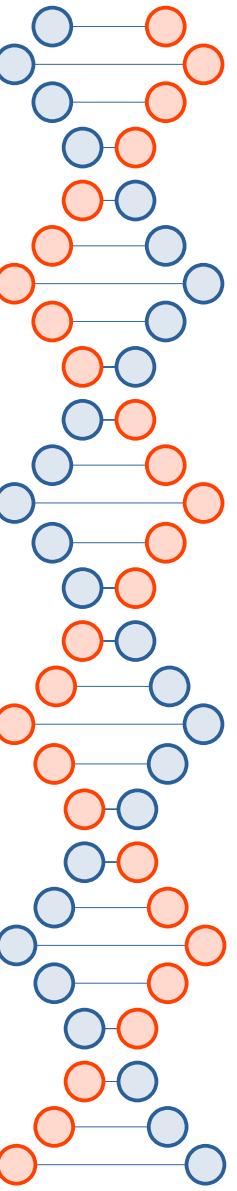
# Hash function example





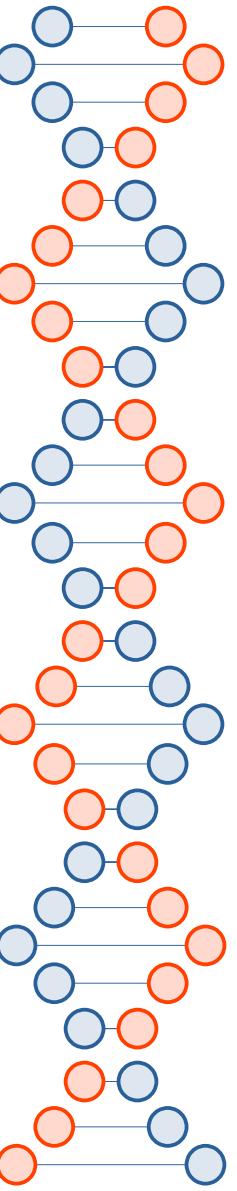
# What makes a hash function cryptographic?

- One-way function
- Deterministic (same input, same output)
- Infeasible to find message that digests to specific hash value
- Infeasible to find two messages that digest to the same hash
- Avalanche effect (small change in message leads to big changes in digest---digests seemingly uncorrelated)
- *Still want it to be quick*



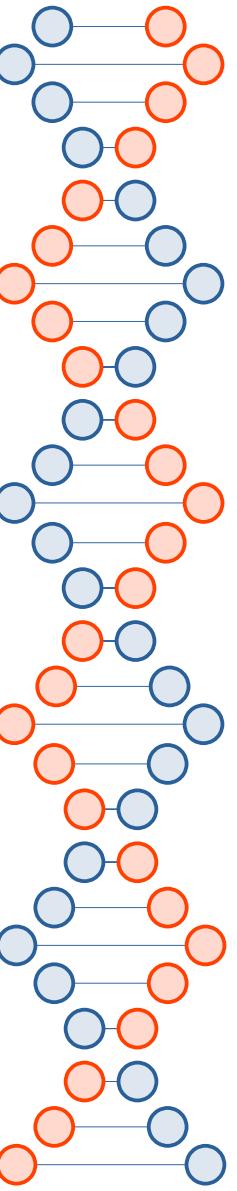
# Example algorithms

- MD5: 128-bit digest
  - seriously broken
- SHA-1: 160-bit digest
  - not secure against well-funded adversaries
- SHA-2: 224 to 512 bit digest
  - Merkle–Damgård construction
- SHA-3: 224 to 512 bit digest
  - Sponge construction
  - adopted in August of 2015
- CRC32: not cryptographic, very poor choice



# Example algorithms

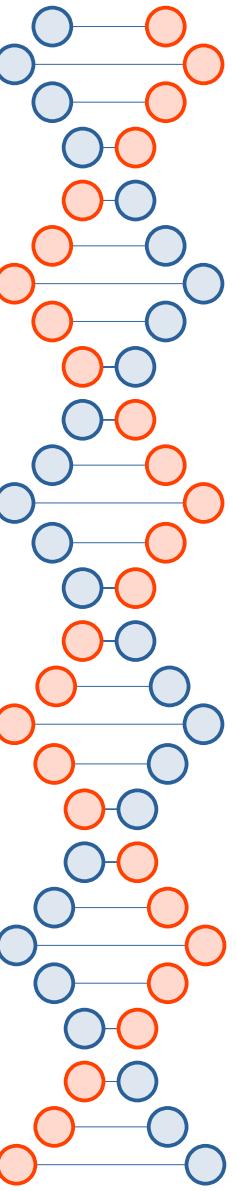
- MD5: 128-bit digest, seriously broken
- SHA-1: 160-bit digest, not secure against well-funded adversaries
- SHA-3: 224 to 512 bit digest, adopted in August of 2015
- CRC32: not cryptographic, very poor choice



## Property #1

- Pre-image resistance
- Given  $h$ , it should be infeasible to find  $m$  such that  $h = \text{hash}(m)$

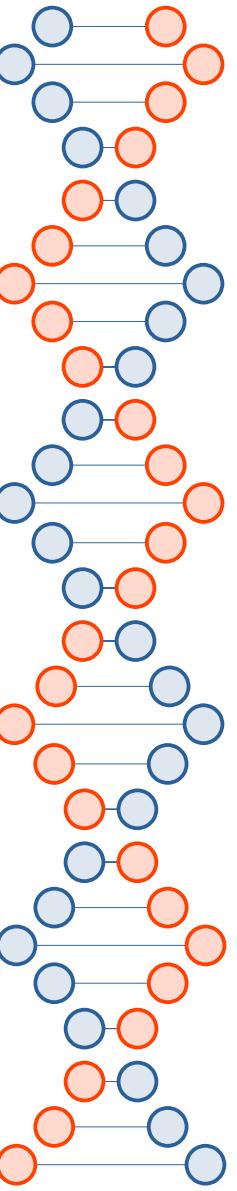
Neither MD5 nor SHA-3 are broken in this way, but MD5 digests are small.



## Property #2

- Second pre-image resistance
- Given a message  $m_1$ , it should be infeasible to find another message  $m_2$  such that...  
$$\text{hash}(m_1) = \text{hash}(m_2)$$

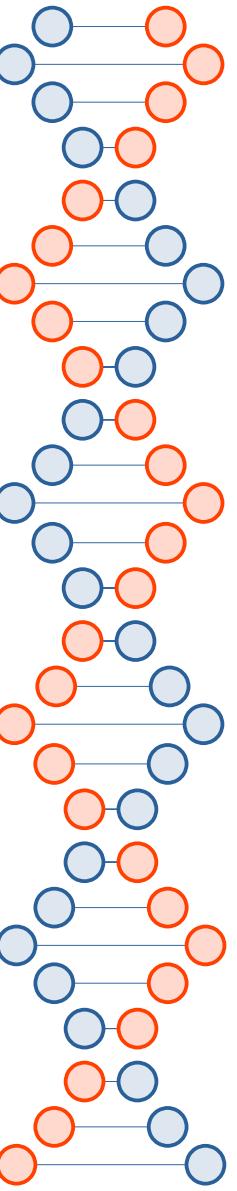
Neither MD5 nor SHA-3 are broken in this way, but MD5 digests are small.



## Property #3

- Collision resistance
- It should be infeasible to find two messages,  $m_1$  and  $m_2$  such that...  
$$\text{hash}(m_1) = \text{hash}(m_2)$$

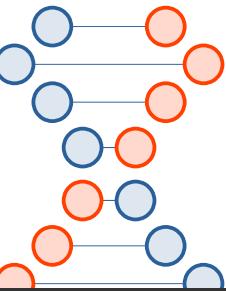
SHA-3 is not broken in this way, MD5 broken in seconds on your laptop, SHA-1 with \$100K or so.



# Wang Xiaoyun

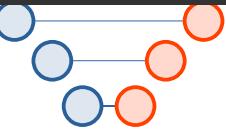
- Tsinghua University
- Contributed a lot of ideas to cracking MD5, SHA-0, and SHA-1



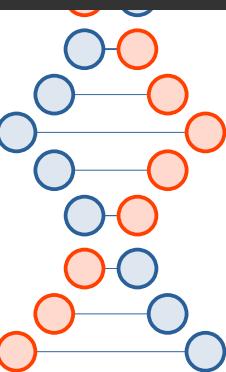


# Length extension attack

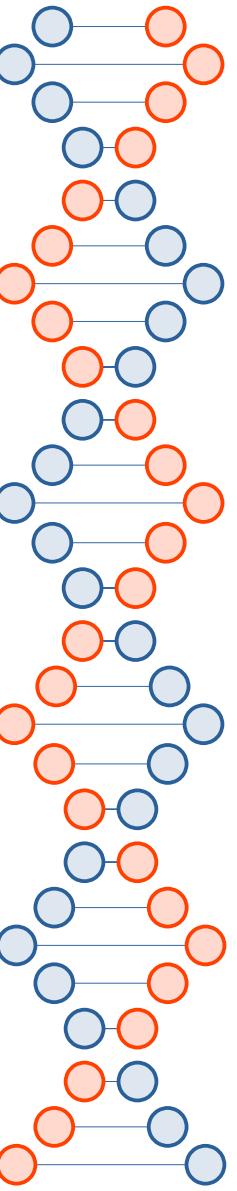
```
jedi@mariposa:~$ echo "password='lDEnr45#d3'&donut=choc&quantity=1" | md5sum  
91a9fc74a98997dba291a26a91c9648e -  
jedi@mariposa:~$ echo "password='lDEnr45#d3'&donut=choc&quantity=100" | md5sum  
8fdd2d4515bcba887b1b80a653f21e0c -
```



```
jedi@mariposa:~$ echo "password=[REDACTED]&donut=choc&quantity=1" | md5sum  
91a9fc74a98997dba291a26a91c9648e -  
jedi@mariposa:~$ echo "password=[REDACTED]&donut=choc&quantity=100" | md5sum  
8fdd2d4515bcba887b1b80a653f21e0c -
```

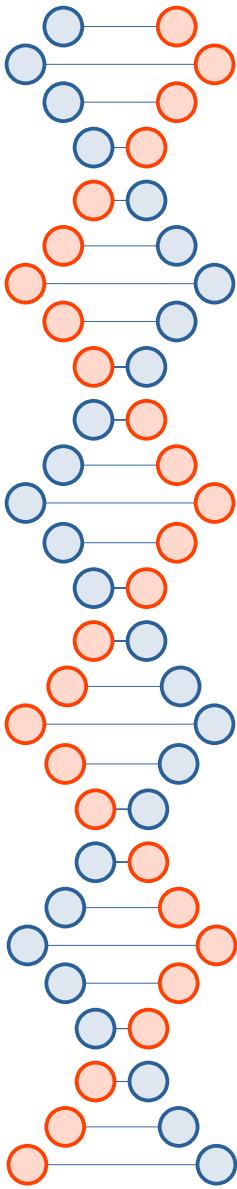


MD5 and SHA-1 vulnerable, SHA-2 basically is, SHA-3 is not



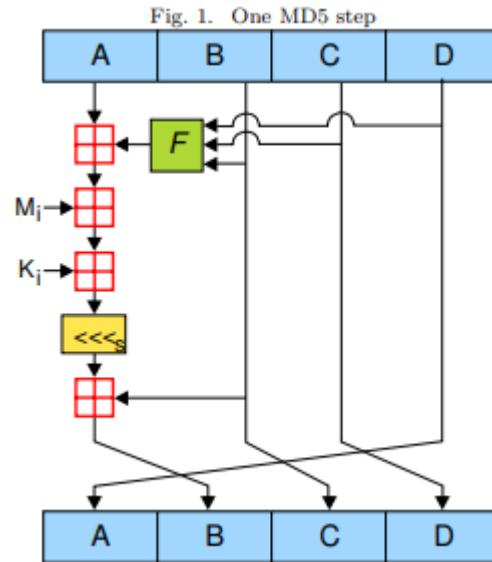
# Length extension attack

- One issue is if the attacker doesn't know the password
- Another issue is if the password is different but the attacker finds a collision later on



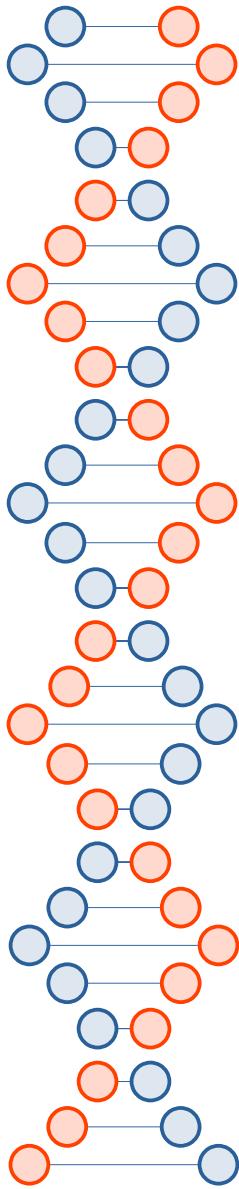
# MD5

- Pad to multiple of 512 bits
- 4 rounds
- 4 32-bit words at a time
- Concatenate them at the end for a 128-bit digest
- $F$  is non-linear, varies by round



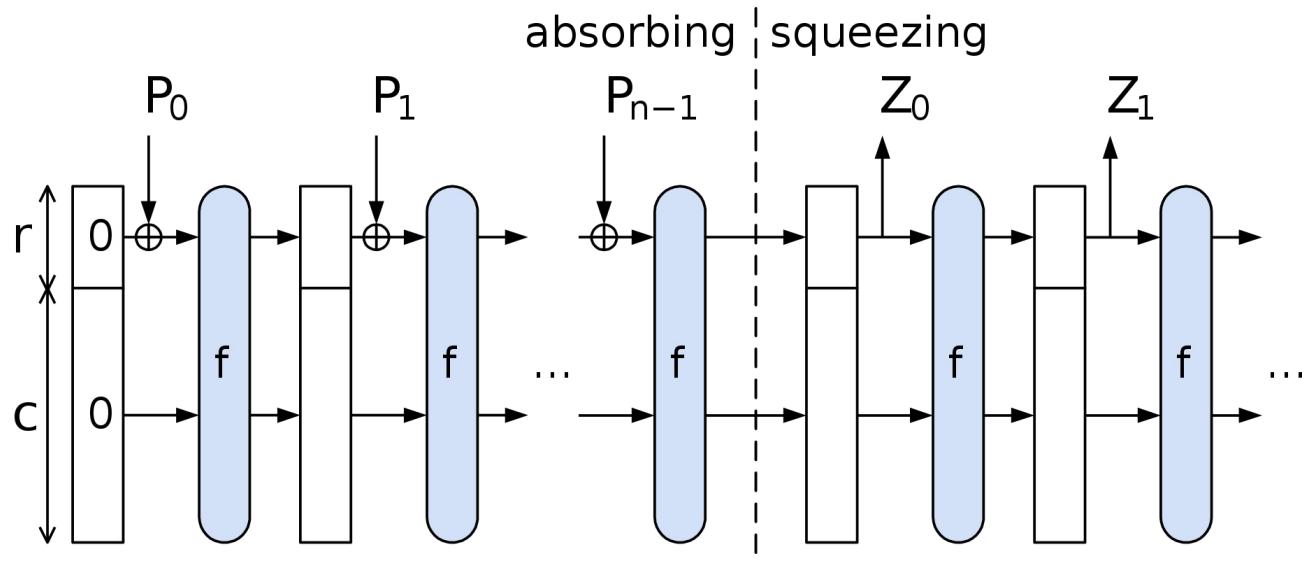
Round ( $i$ )	$F(X, Y, Z)$	$g$
0	$(X \wedge Y) \vee (\neg X \wedge Z)$	$i$
1	$(X \wedge Z) \vee (Y \wedge \neg Z)$	$(5 \times i + 1) \bmod 16$
2	$(X \oplus Y \oplus Z)$	$i(3 \times i + 5) \bmod 16$
3	$(Y \oplus (X \vee \neg Z))$	$(7 \times i) \bmod 16$

<http://koclab.cs.ucsb.edu/teaching/cren/project/2008/savage.pdf>

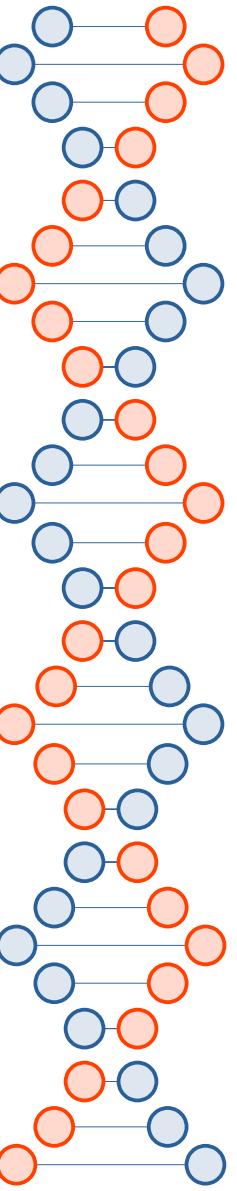


# SHA-3

- Sponge construction, 1600 bits of internal state



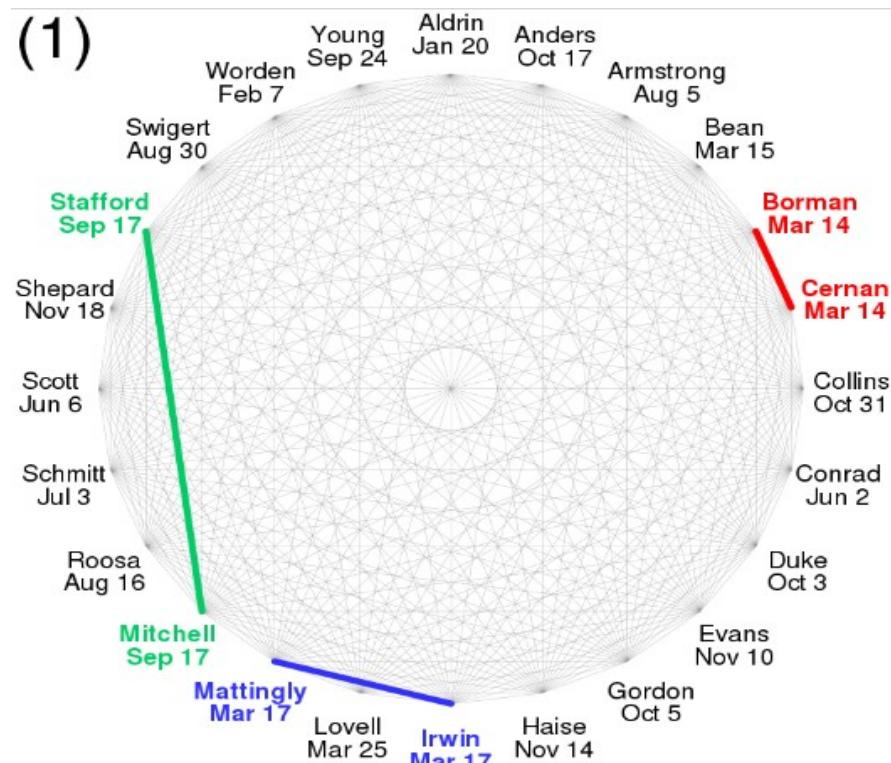
<https://en.wikipedia.org/wiki/SHA-3>

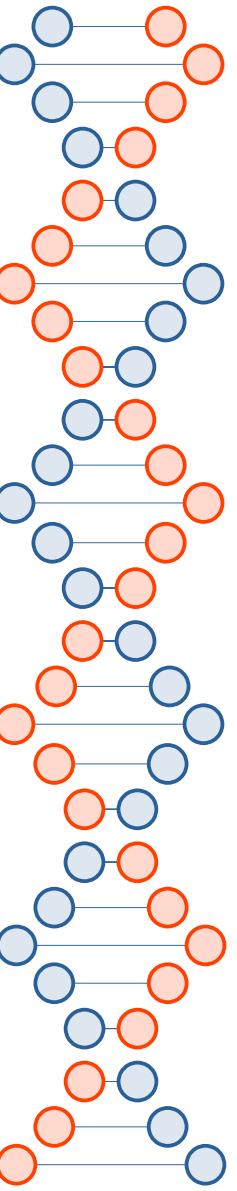


# Review: Birthday attack

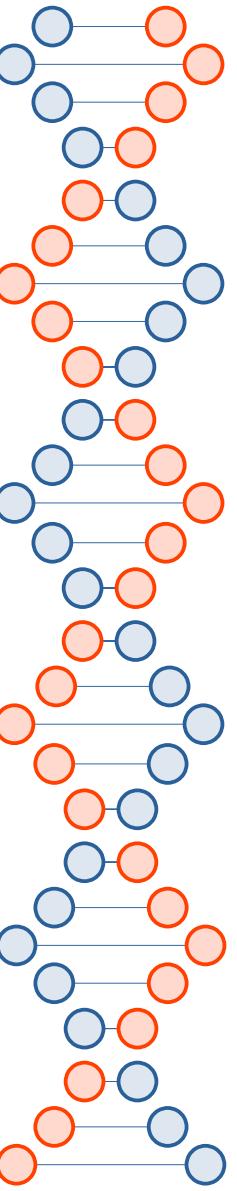
- Probability of collision is  $1$  in  $2^n$ , but the expected number of hashes until two of them collide is  $\sqrt{2^n} = 2^{n/2}$ 
  - Why? Third try has two opportunities to collide, fourth has three opportunities, fifth has six, and so on...

# 24 people, same birthday?



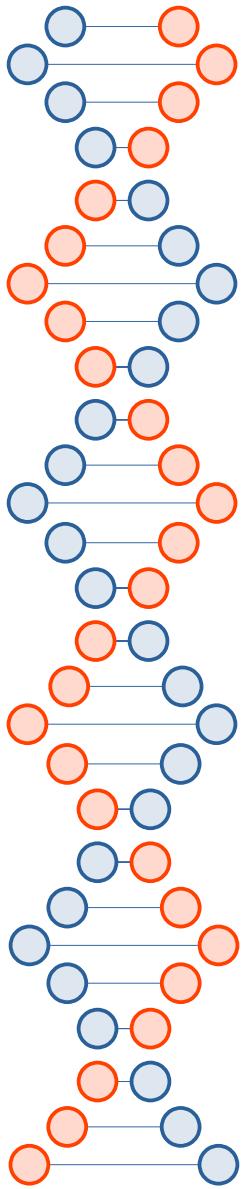


<review>



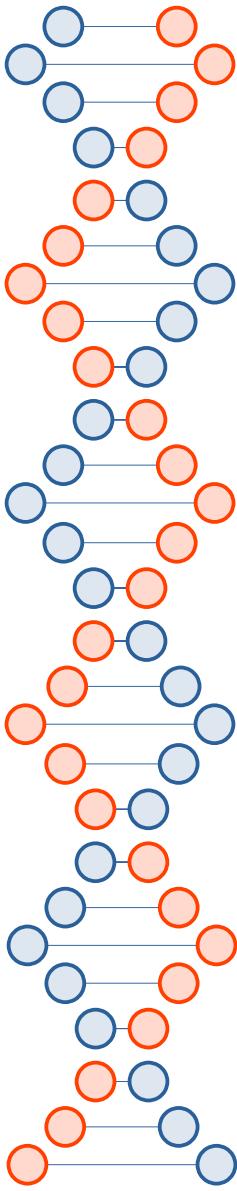
# Chosen-prefix collision attack

- Given two prefixes  $p_1$  and  $p_2$ , find  $m_1$  and  $m_2$  such that  $\text{hash}(p_1 \parallel m_1) = \text{hash}(p_2 \parallel m_2)$
- p1 and p2 could be domain names in a certificate, images, PDFs, etc. ... any digital image.



# Ingredients for a practical chosen prefix attack on MD5

- Collision attack on MD5
  - That works for any initialization vector (so you can put bits in front)
- Length extension attack
  - So you can put identical bits on the end
- Birthday attack
  - So you can bridge the prefix to a block that meets the requirements of the collision attack

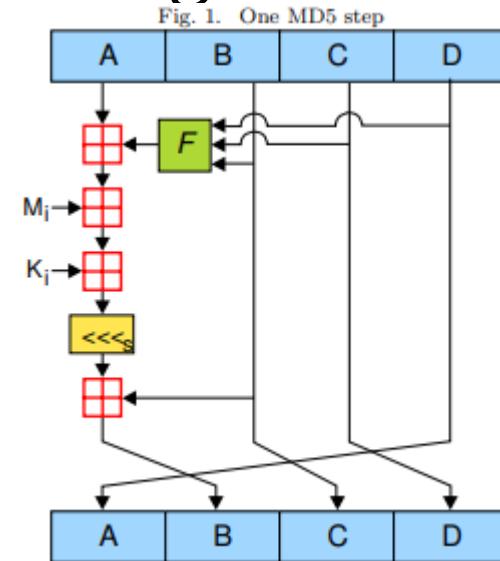


# MD5 collision attack by Wang and Yu

$$C_0 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, 2^{15}, 0, 0, 2^{31}, 0)$$

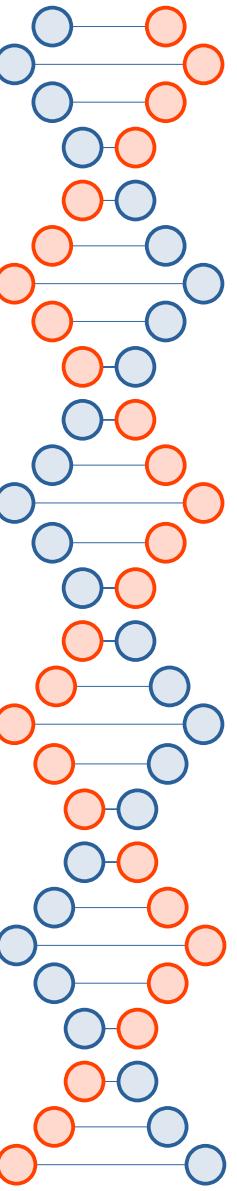
and

$$C_1 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, -2^{15}, 0, 0, 2^{31}, 0)$$



Round ( $i$ )	$F(X, Y, Z)$	$g$
0	$(X \wedge Y) \vee (\neg X \wedge Z)$	$i$
1	$(X \wedge Z) \vee (Y \wedge \neg Z)$	$(5 \times i + 1) \bmod 16$
2	$(X \oplus Y \oplus Z)$	$i(3 \times i + 5) \bmod 16$
3	$(Y \oplus (X \vee \neg Z))$	$(7 \times i) \bmod 16$

<http://koclab.cs.ucsb.edu/teaching/cren/project/2008/savage.pdf>



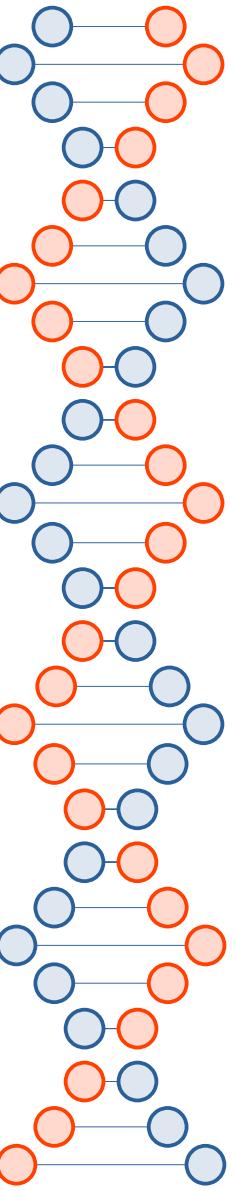
# An example

Both have digest 79054025255fb1a26e4bc422aef54eb4

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbd f280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

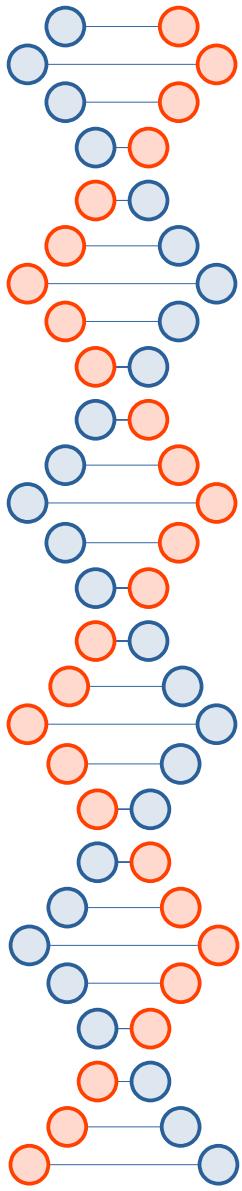
```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
```

<https://www.mscs.dal.ca/~selinger/md5collision/>



# Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate

Marc Stevens<sup>1</sup>, Alexander Sotirov<sup>2</sup>,  
Jacob Appelbaum<sup>3</sup>, Arjen Lenstra<sup>4,5</sup>, David Molnar<sup>6</sup>,  
Dag Arne Osvik<sup>4</sup>, and Benne de Weger<sup>7</sup>



legitimate website certificate	rogue CA certificate
serial number	serial number
commercial CA name	commercial CA name
validity period	validity period
domain name	rogue CA name
2048 bit RSA public key	1024 bit RSA public key
v3 extensions "CA = FALSE"	v3 extensions "CA = TRUE"

chosen prefixes

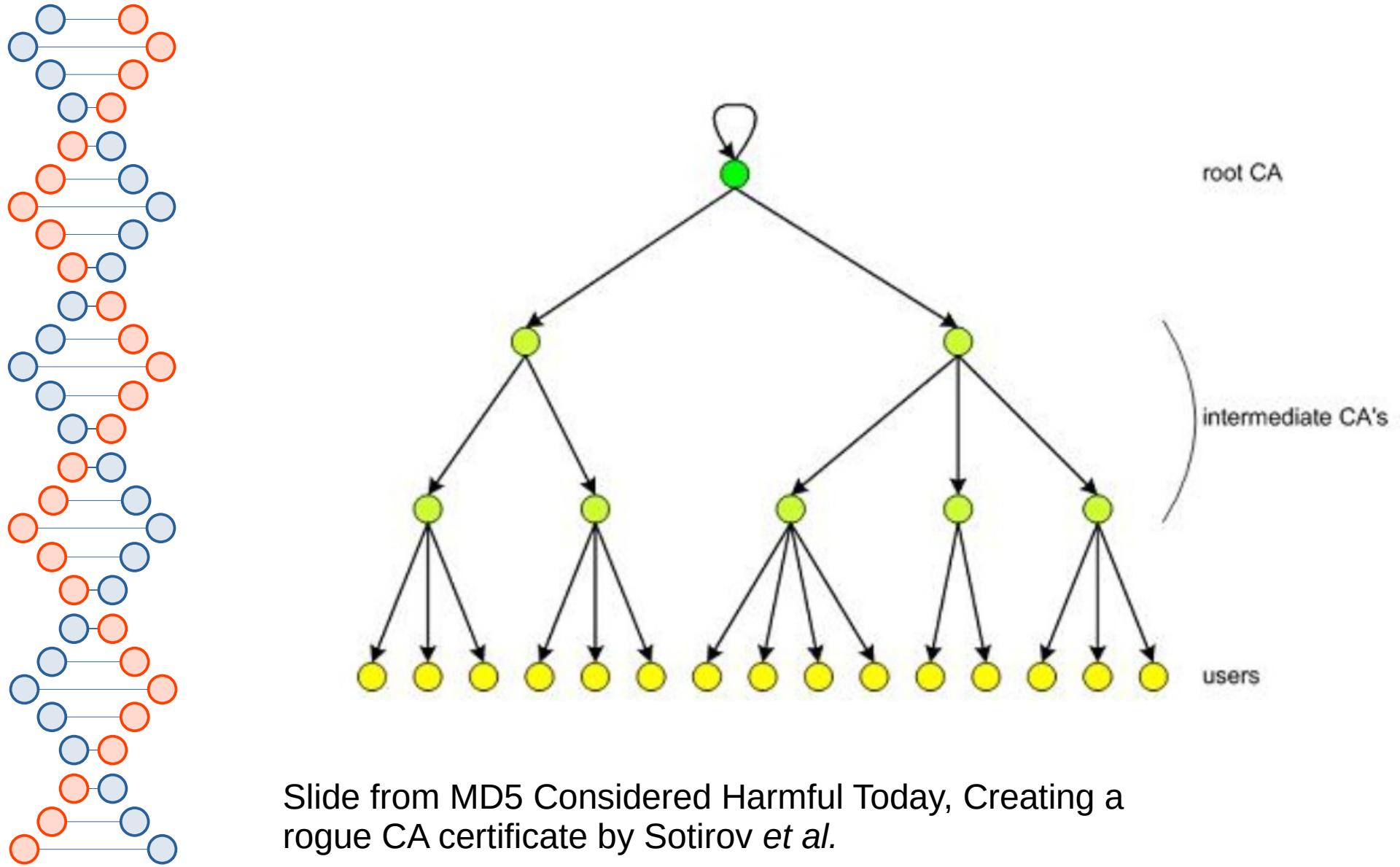
collision bits

identical suffixes

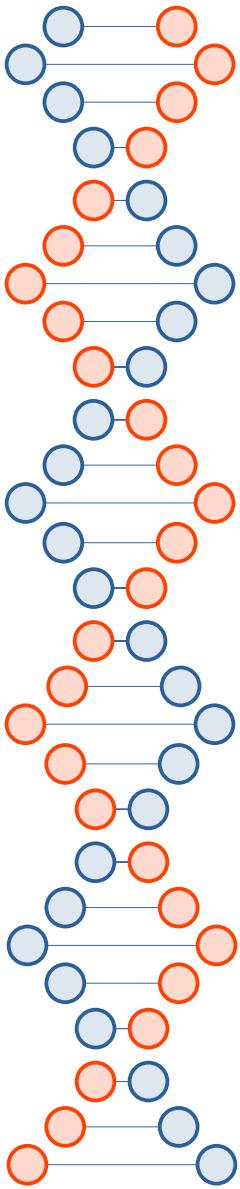
tumor

**Fig. 1.** The to-be-signed parts of the colliding certificates





Slide from MD5 Considered Harmful Today, Creating a rogue CA certificate by Sotirov *et al.*



# *Cryptography Engineering* by Ferguson *et al.*

