

# CS 5830

# Cryptography

Instructor: Tom Ristenpart

TAs: Yan Ji, Sanketh Menda



**CORNELL  
TECH**

HOME OF THE  
**JACOBS  
INSTITUTE**

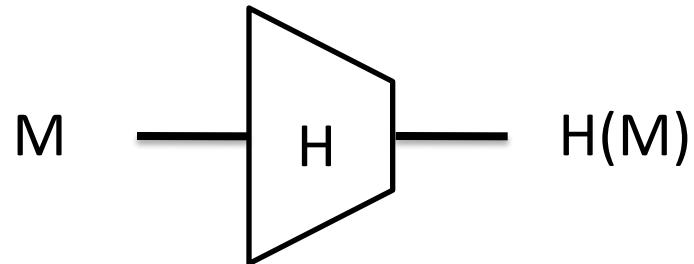


# Where we're at

- **Message authentication schemes (MACs)**
  - Secret key + message => authentication tag
  - Examples: CBC-MAC
- **Authenticated encryption (with associated data)**
  - Confidentiality + ciphertext integrity
- **Today: cryptographic hash functions**

# Cryptographic hash functions

A function  $H$  that maps (essentially) arbitrary bit string to fixed length string of size  $n$



MD5:  $n = 128$  bits

SHA-1:  $n = 160$  bits

SHA-256:  $n = 256$  bits

Many security goals asked of hash functions

Ideally, they behave as if they were a (public) random function

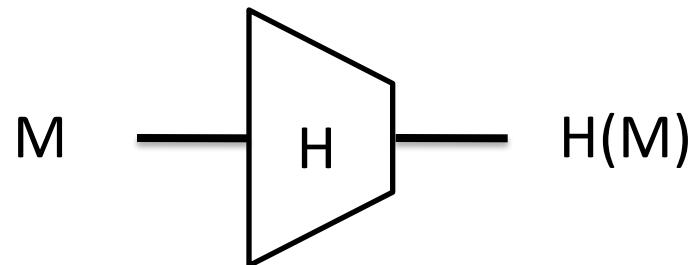
# Applications of hashing

- File comparison
- Digital signatures (coming up later)
- Password hashing and password-based encryption
- Message authentication codes
- Key derivation
- Randomness extraction (in random number generators)

~~H(file) ≠ H(file')~~  
~~iff~~  
~~file ≠ file'~~  
~~✓~~

# Cryptographic hash functions

A function  $H$  that maps (essentially) arbitrary bit string to fixed length string of size  $n$



MD5:  $n = 128$  bits  
SHA-1:  $n = 160$  bits  
SHA-256:  $n = 256$  bits

## Collision resistance:

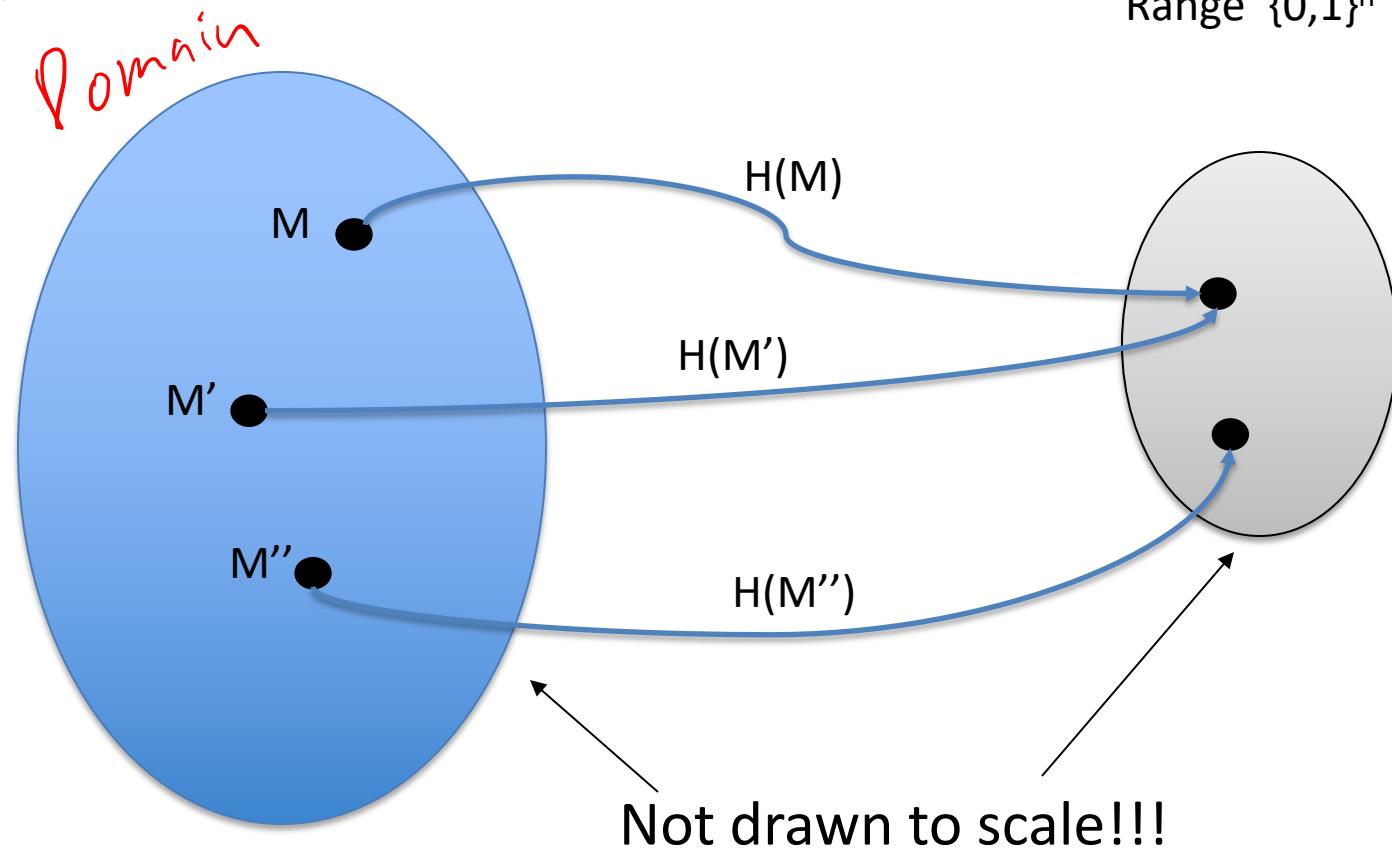
No computationally efficient adversary can find  $M \neq M'$  such that  $H(M) = H(M')$

CR( $H, \mathcal{A}$ ):  
 $(M, M') \leftarrow \$ \mathcal{A}$   
If  $M \neq M'$  and  $H(M) = H(M')$   
    Return 1  
Return 0

# Collisions always exist

Domain (usually all strings up to some length)

SHA-1: up to length  $2^{64}-1$



**Pigeonhole principle:**

size of domain larger than size of range implies there must be collisions

# Hash functions: Keyed or unkeyed?

- Hash functions traditionally are *unkeyed*
  - Function  $H: \mathcal{M} \rightarrow \{0,1\}^n$  mapping a message to fixed-length string
  - $\mathcal{M}$  is message space (e.g., all bit strings up to length  $2^{64}$ )
  - Examples: MD5, SHA1, SHA256
- Some hash functions also defined *with keys*
  - Function  $H: \{0,1\}^k \times \mathcal{M} \rightarrow \{0,1\}^n$  mapping a key, message pair to fixed-length bit string
  - Most often used in cases where we need message authentication, not just CR
  - Examples: HMAC (we will see later), Envelope MAC

# Hash functions: Keyed or unkeyed?

Transforming keyed to unkeyed

- Pick a key  $K$  and make it fixed, public parameter, hardcode it
- Pick random key  $K$  per application, or per use of hash
  - Rarely done

Transforming unkeyed to keyed

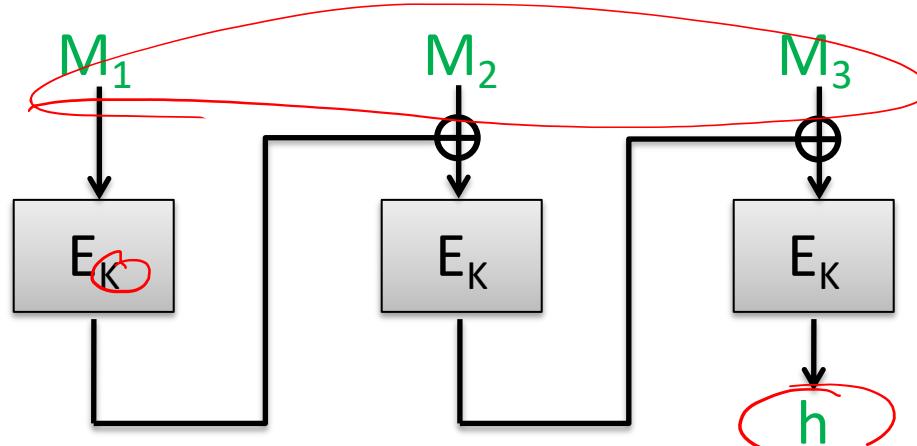
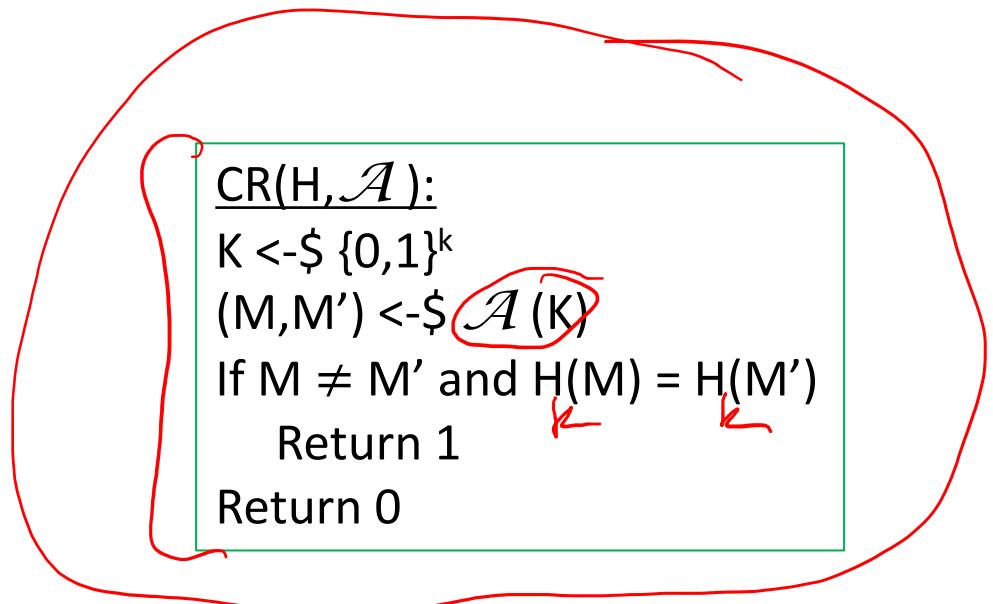
- Need a keying strategy. We will see some soon (e.g., HMAC)

# CBC-MAC as CR hash?

Consider CBC-MAC as keyed hash

Key was secret in message authentication setting

If key is public is CBC-MAC collision resistant?



$m_1 = 0^n$

$m_i = \text{Adversary } \mathcal{A}(K)$

$H_K(m_i) = E_K(0^n)$

$H_K(R \parallel E_K(R)) = E_K(E_K(0^n) \oplus E_K(R)) = E_K(H_K(m_i))$

$H_K(R \parallel E_K(R)) = E_K(E_K(0^n) \oplus E_K(R)) = E_K(H_K(m_i))$

# Universal hash functions as CR hashes?

A keyed function  $H: \{0, 1\}^k \times \mathcal{M} \rightarrow \{0, 1\}^n$  is an  $\epsilon$ -almost universal (AU) hash function if for all  $M \neq M'$

$$\Pr [ H_K(M) = H_K(M') ] \leq \epsilon = \frac{1}{2^n}$$

where the probability is over choice of  $K$ .

What is difference between this and CR notion?

CR: Adversary **can** choose message as function of  $K$

Universal: Adversary **cannot** choose message as function of  $K$

# Universal hash functions as CR hashes?

Example universal hash from Carter-Wegman:

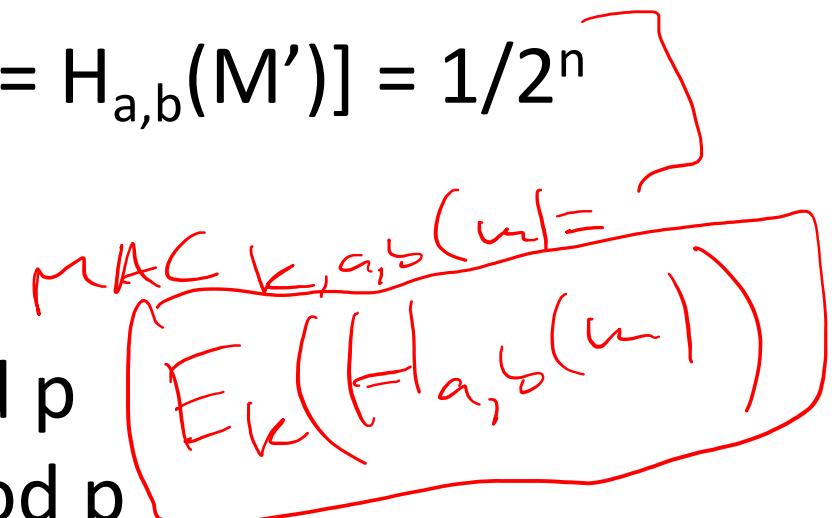
$$H_{a,b}(M) = ((aM+b) \bmod p) \bmod 2^n \quad p > 2^n \text{ is a prime}$$

Universality: for any  $M \neq M'$ ,  $\Pr[H_{a,b}(M) = H_{a,b}(M')] = 1/2^n$

What is a CR adversary?

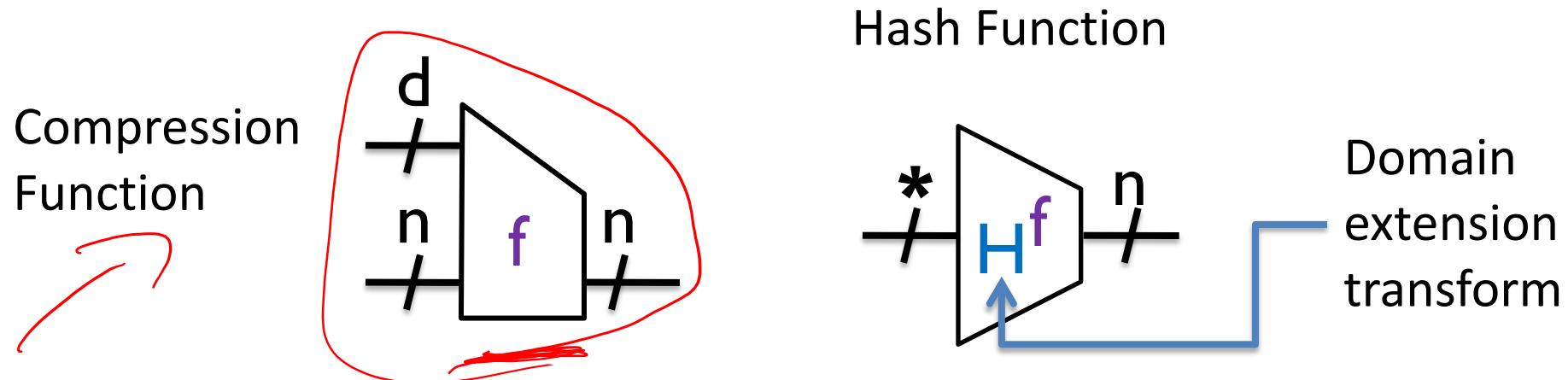
Pick any  $M$ , compute  $y = ((aM+b) \bmod p)$

Solve linear equation:  $y = aM' + b \bmod p$

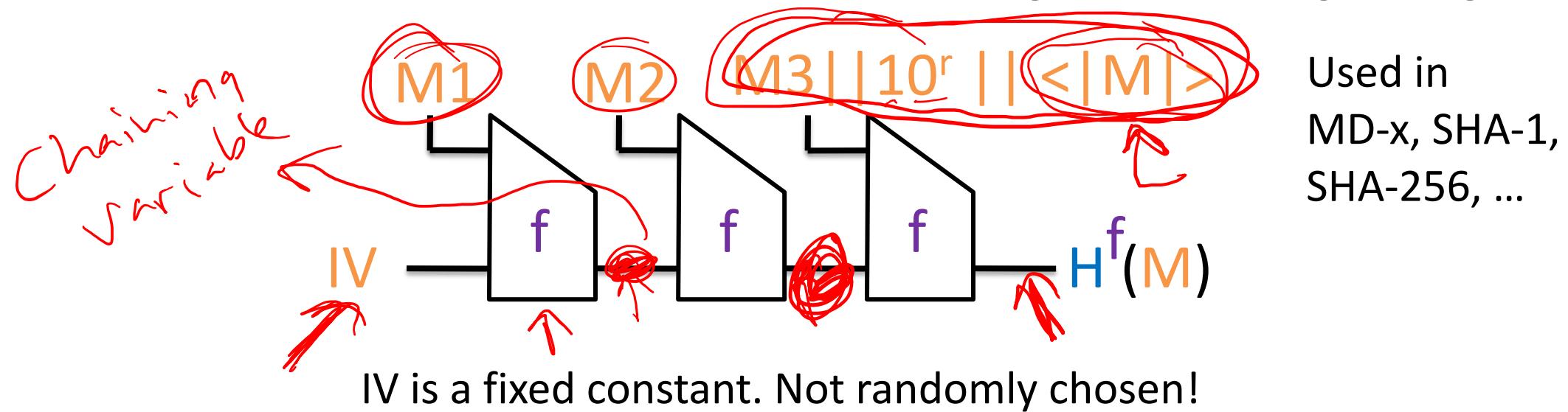


Universal hash functions don't provide collision-resistance

# Two-step design approach for hash functions



Domain extension called “Merkle-Damgård with strengthening”



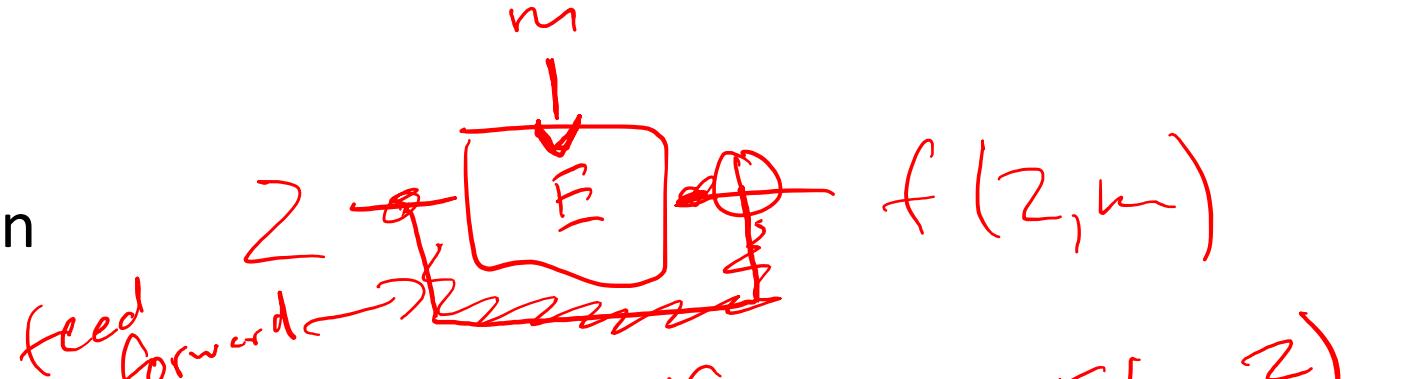
# Building compression functions

Λ V

- Can build compression functions from suitable block ciphers

$$f(z, m) = E(m, z) \oplus z$$

Called Davies-Meyer construction



- Can use AES, but security too low. Why?

- SHA-1 uses custom E with  $k = d = 512$  and  $n = 160$

- Message block length of SHA-1 is 512 bits

$$\text{SHA-256 is } 1024 = d$$

$$\begin{aligned} f(z, m) &= E(m, z) \\ &\quad \text{if } m \neq m \text{ then:} \\ &D(m, 0^n) = z \\ &D(m, 0^n) = z \\ &D(m, 0^n) = z \\ &f(z, m) = \tilde{f}(z, m) \\ &\quad = 0^n \end{aligned}$$

# SHA-1 compression function

Expand 512-bit message  $m$  into

$W_1, \dots, W_{80}$  strings of length 32 bit values  
(Think of this as “key schedule”)

Chaining variable is 160 bits, 5 32-bit values

$A, B, C, D, E$

$F(B, C, D)$  function that changes over rounds:

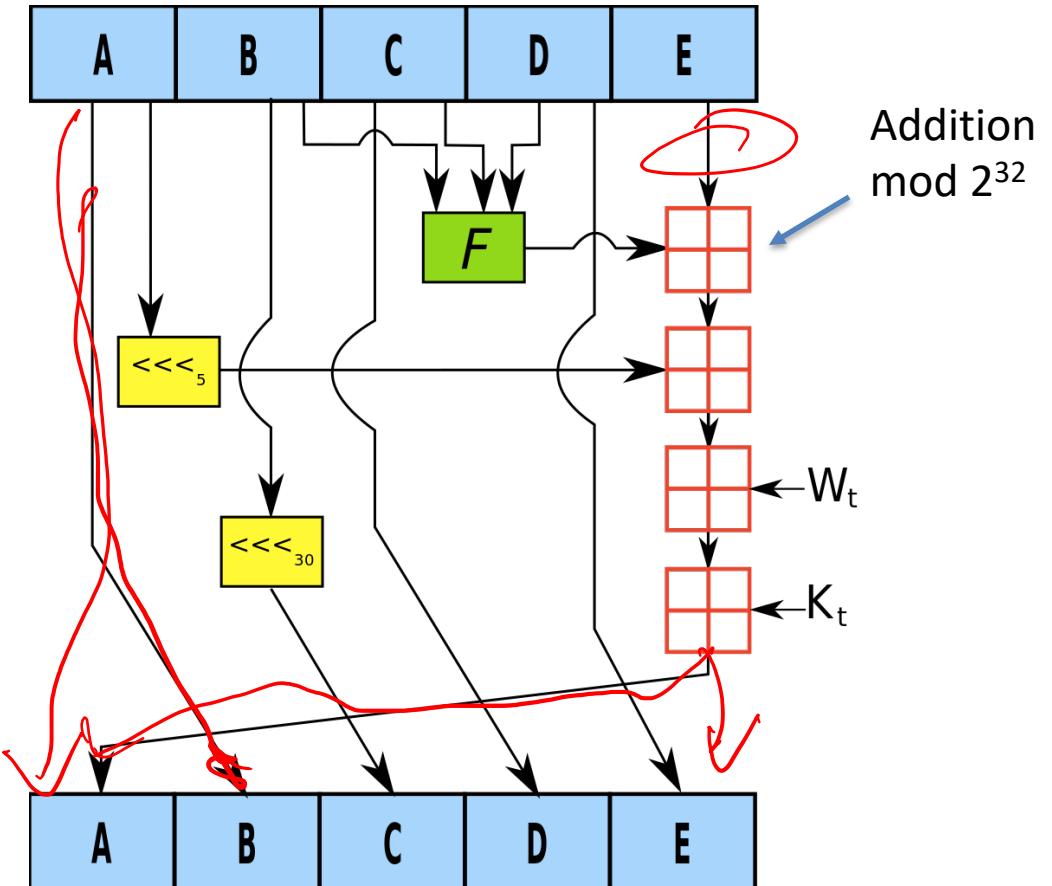
0-19: **(B and C) or ((not B) and D)**

20-39:  **$B \text{ xor } C \text{ xor } D$**

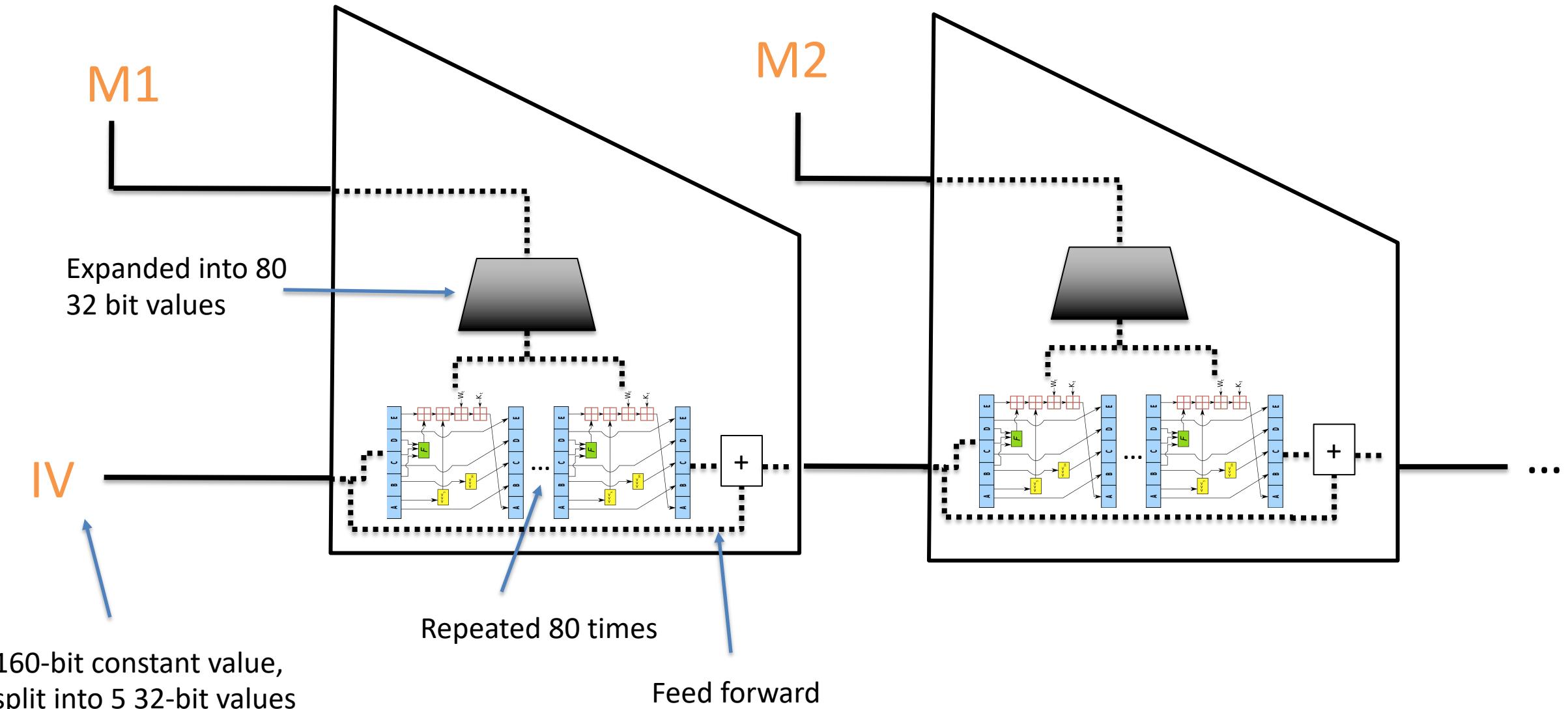
40-59: **(B and C) or (B and C) or (C and D)**

60-79:  **$B \text{ xor } C \text{ xor } D$**

Constants  $K_1, \dots, K_{80}$  differ across rounds



# SHA1 hash function



# Birthday attacks

- What is best possible security achievable by hash function with output length  $n$  bits?
- Answer: security is only achievable up to  $2^{n/2}$  hash computations

# The birthday problem

Choose  $q$  values  $Y_1, \dots, Y_q$  from  $\{0,1\}^n$  at random. What is probability that two are the same?

Let  $\text{Coll}_i$  be event that  $Y_i = Y_j$  for some  $j < i$

$$\begin{aligned}\Pr[\text{Coll}] &= \Pr[\text{Coll}_1 \vee \dots \vee \text{Coll}_q] \\ &\leq \Pr[\text{Coll}_1] + \dots + \Pr[\text{Coll}_q] \\ &= \frac{0}{2^n} + \frac{1}{2^n} + \frac{2}{2^n} + \dots + \frac{q}{2^n} \\ &= \frac{q(q-1)}{2^n}\end{aligned}$$

Another proof shows that if  $q \leq 2^{(n+1)/2}$

$$\Pr[\text{Coll}] \geq \frac{0.3 \cdot q(q-1)}{2^n}$$

# The birthday attack

Let  $m$  be some length in domain of hash function  $H$

Adversary A:

For  $i = 1$  to  $q$  do:

$$X_i \leftarrow \{0,1\}^m$$

$$h_i \leftarrow H(X_i)$$

If exists  $i, j$  s.t.  $X_i \neq X_j$  and  $h_i = h_j$  then

Return  $(X_i, X_j)$

Return fail

Same # of domain points  
map to each range point

If  $H$  is *regular* then probability of success is  
exactly birthday probability

$$\Pr[A \text{ finds collision}] \geq \frac{0.3 \cdot q(q-1)}{2^n}$$

# Birthday attack run times

MD5:	$n = 128$ bits	$2^{64}$	MD5 computations
SHA-1:	$n = 160$ bits	$2^{80}$	SHA-1 computations
SHA-256:	$n = 256$ bits	$2^{128}$	SHA-256 computations

$2^{64}$  too small by today's standards!

Bitcoin network computes about  $\sim 2^{67}$  SHA-256 hashes *per second*

<https://blockchain.info/charts/hash-rate>

# Faster attacks than birthday?

- 2004: Full break of MD5 announced by Xiaoyun Wang and co-authors
  - MD5 is easy to find collisions against
  - You can download programs to do it on your laptop
- 2005: Announced faster than  $2^{80}$  attack against SHA-1 by Wang et al.
  - Not practical to run ( $2^{69}$  estimated cost)
- 2017: CWI and Google announce first demonstrated collision

# SHAttered attack

Attacker-chosen prefix P. Find two pairs of message blocks  $M_1, M_2$  and  $M_1', M_2'$  such that for any suffix S:

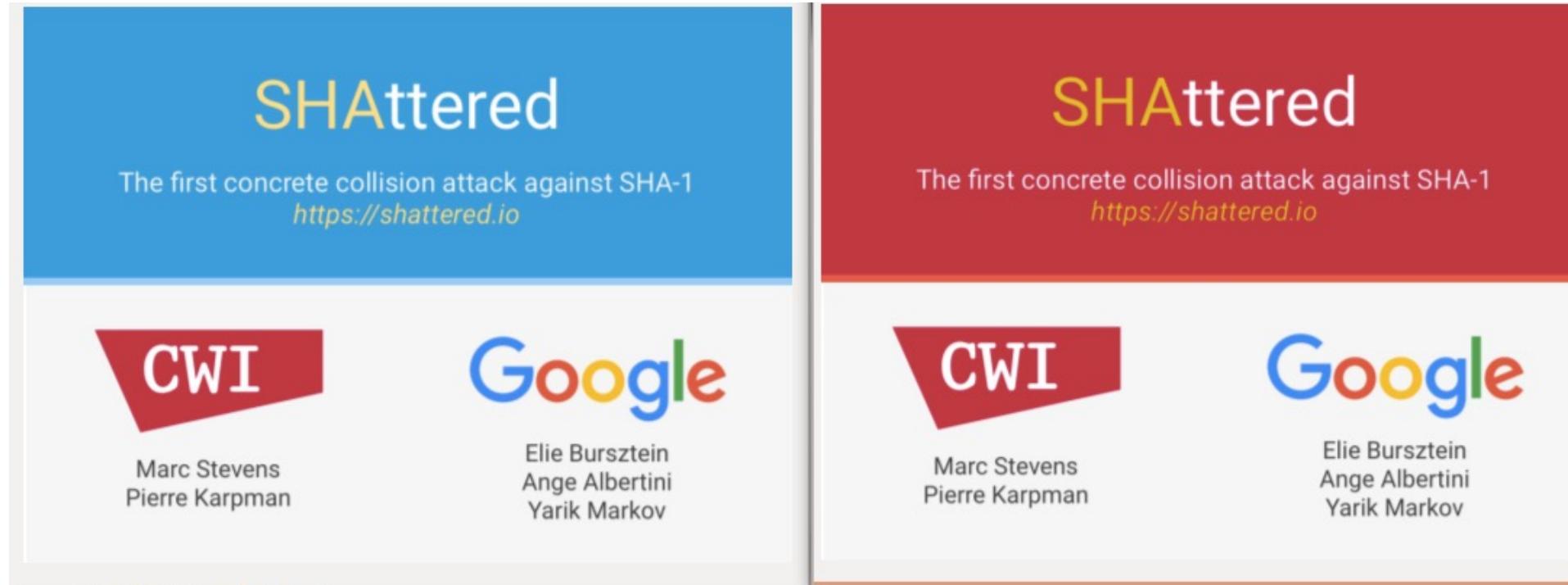
$$\text{SHA-1}(P \parallel M_1 \parallel M_2 \parallel S) = \text{SHA-1}(P \parallel M_1' \parallel M_2' \parallel S)$$


Referred to as an *identical-prefix collision attack*

How? Pick P, find  $M_1$  and  $M_1'$  that form *near-collision* on chaining variable. Then complete collision by finding  $M_2$  and  $M_2'$

They show how to extend to build colliding PDF files

# SHAttered attack



Required  $2^{63.1}$  SHA-1 compression function applications  
100,000x faster than birthday attack ( $2^{80}$ )

# Fallout of attack

SVN repositories can be broken (DoS attack)

- Checking in the two SHAttered PDFs corrupts repo

Marc Stevens & Dan Shumow (Microsoft) developed *counter-cryptanalysis*

Way to detect if a particular file is one half of colliding pair

Deployed at several large companies

Ongoing migration away from SHA-1 to SHA-256 / SHA-3

# Applications of hashing

- File comparison
- Digital signatures (coming up later)
- Password hashing and password-based encryption
- For message authentication codes
- Key derivation

