

# HASH FUNCTIONS and MESSAGE AUTHENTICATION CODES

## Last Time: Hash Functions

Recall: SHA256:  $\{0, 1\}^{<2^{64}} \rightarrow \{0, 1\}^{256}$

Inputs are strings with any length less than  $2^{64}$ .

Outputs are 256 bits.

We saw last time SHA256 uses a block cipher-based construction as a “compression function”:  $\{0, 1\}^{512+256} \rightarrow \{0, 1\}^{256}$ .

How do you construct a hash function that can take arbitrary length inputs?

# Compression functions

A **compression function** is a family  $h : \{0, 1\}^k \times \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  of functions whose inputs are of a fixed size  $b + n$ , where  $b$  is called the block size.

E.g.  $b = 512$  and  $n = 256$ , in which case

$$h : \{0, 1\}^k \times \{0, 1\}^{768} \rightarrow \{0, 1\}^{256}$$

# The MD transform

Let  $h : \{0, 1\}^k \times \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  be a compression function with block length  $b$ . Let  $D$  be the set of all strings of at most  $2^b - 1$  blocks.

The **MD transform** builds from  $h$  a family of functions

$$H : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$$

such that: If  $h$  is CR, then so is  $H$ .

The problem of hashing long inputs has been reduced to the problem of hashing fixed-length inputs.

There is no need to try to attack  $H$ . You won't find a weakness in it unless  $h$  has one. That is,  $H$  is *guaranteed* to be secure *assuming*  $h$  is secure.

For this reason, MD is the design used in many hash functions, including the MD and SHA2 series. SHA3 uses a different paradigm.

**Given:** Compression function  $h : \{0, 1\}^k \times \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$ .

**Build:** Hash function  $H : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$ .

Since  $M \in D$ , its length  $\ell = |M|$  is a multiple of the block length  $b$ . We let  $\|M\|_b = |M|/b$  be the number of  $b$ -bit blocks in  $M$ , and parse as

$$M[1] \dots M[\ell] \leftarrow M .$$

Let  $\langle \ell \rangle$  denote the  $b$ -bit binary representation of  $\ell \in \{0, \dots, 2^b - 1\}$ .

# MD transform

**Given:** Compression function  $h : \{0, 1\}^k \times \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$ .

**Build:** Hash function  $H : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$ .

**Alg**  $H(K, M)$

$m \leftarrow \|M\|_b ; M[m+1] \leftarrow \langle m \rangle ; V[0] \leftarrow 0^n$

For  $i = 1, \dots, m+1$  do

$V[i] \leftarrow h_K(M[i] \| V[i-1])$

Return  $V[m+1]$

# MD preserves Collision Resistance

**Theorem:** Let  $h : \{0, 1\}^k \times \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  be a family of functions and let  $H : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$  be obtained from  $h$  via the MD transform. Given a cr-adversary  $A_H$  we can build a cr-adversary  $A_h$  such that

$$\text{Adv}_H^{\text{cr}}(A_H) \leq \text{Adv}_h^{\text{cr}}(A_h)$$

and the running time of  $A_h$  is that of  $A_H$  plus the time for computing  $H$  on the outputs of  $A_H$ .

**Implication:**

$$\begin{aligned} h \text{ CR} &\Rightarrow \text{Adv}_h^{\text{cr}}(A_h) \text{ small} \\ &\Rightarrow \text{Adv}_H^{\text{cr}}(A_H) \text{ small} \\ &\Rightarrow H \text{ CR} \end{aligned}$$

## A candidate compression function

Let  $E : \{0, 1\}^b \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a block cipher. Let us define keyless compression function  $h : \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  by

$$h(x\|v) = E_x(v) .$$

**Question:** Is  $h$  collision resistant?

## A candidate compression function

Let  $E : \{0, 1\}^b \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a block cipher. Let us define keyless compression function  $h : \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  by

$$h(x\|v) = E_x(v) .$$

**Question:** Is  $h$  collision resistant?

We seek an adversary that outputs distinct  $x_1\|v_1, x_2\|v_2$  satisfying

$$E_{x_1}(v_1) = E_{x_2}(v_2) .$$

## A candidate compression function

Let  $E : \{0, 1\}^b \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a block cipher. Let us define keyless compression function  $h : \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  by

$$h(x\|v) = E_x(v) .$$

**Question:** Is  $h$  collision resistant?

We seek an adversary that outputs distinct  $x_1\|v_1, x_2\|v_2$  satisfying

$$E_{x_1}(v_1) = E_{x_2}(v_2) .$$

**Answer:** **NO**,  $h$  is NOT collision-resistant, because the following adversary  $A$  has  $\text{Adv}_h^{\text{cr}}(A) = 1$ :

adversary  $A$

$x_1 \leftarrow 0^b ; x_2 \leftarrow 1^b ; v_1 \leftarrow 0^n ; y \leftarrow E_{x_1}(v_1) ; v_2 \leftarrow E_{x_2}^{-1}(y)$

Return  $x_1\|v_1, x_2\|v_2$

# The Davies-Meyer compression function

Let  $E : \{0, 1\}^b \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a block cipher. Let us define keyless compression function  $h : \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  by

$$h(x\|v) = E_x(v) \oplus v.$$

**Question:** Is  $h$  collision resistant?

# The Davies-Meyer compression function

Let  $E : \{0, 1\}^b \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a block cipher. Let us define keyless compression function  $h : \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  by

$$h(x\|v) = E_x(v) \oplus v .$$

**Question:** Is  $h$  collision resistant?

We seek an adversary that outputs distinct  $x_1\|v_1, x_2\|v_2$  satisfying

$$E_{x_1}(v_1) \oplus v_1 = E_{x_2}(v_2) \oplus v_2 .$$

**Answer:** Unclear how to solve this equation, even though we can pick all four variables.

# The Davies-Meyer compression function

Let  $E : \{0, 1\}^b \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a block cipher. Let us define keyless compression function  $h : \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$  by

$$h(x\|v) = E_x(v) \oplus v .$$

This is called the Davies-Meyer method and is used in the MD and SHA2 series of hash functions, modulo that the  $\oplus$  may be replaced by addition.

In particular the compression function sha256 of SHA256 is underlain in this way by the block cipher  $E^{\text{sha256}} : \{0, 1\}^{512} \times \{0, 1\}^{256} \rightarrow \{0, 1\}^{256}$  that we saw earlier, with the  $\oplus$  being replaced by component-wise addition modulo  $2^{32}$ .

So far we have looked at attacks that do not attempt to exploit the structure of  $h$ .

Can we get better attacks if we *do* exploit the structure?

Ideally not, but hash functions have fallen short!

# Cryptanalytic attacks against hash functions

When	Against	Time	Who
1993,1996	md5	$2^{16}$	[dBBo,Do]
2004	MD5	1 hour	[WaFeLaYu]
2005,2006	MD5	1 minute	[LeWadW,Kl]
2005	SHA1	$2^{69}$	[WaYiYu]
2017	SHA1	$2^{63.1}$	[SBKAM]

Collisions found in compression function md5 of MD5 did not yield collisions for MD5, but collisions for MD5 are now easy.

2017: Google, Microsoft and Mozilla browsers stop accepting SHA1-based certificates.

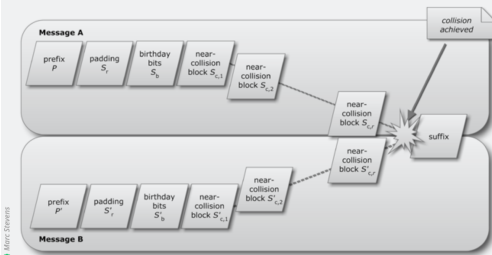
The SHA256 and SHA512 hash functions are still viewed as secure, meaning the best known attack is the birthday attack.

# Flame malware exploited an MD5 collision

## Crypto breakthrough shows Flame was designed by world-class scientists

The spy malware achieved an attack unlike any cryptographers have seen before.

DAN GOODIN - 6/7/2012, 11:20 AM



**Enlarge** / An overview of a chosen-prefix collision. A similar technique was used by the Flame espionage malware that targeted Iran. The scientific novelty of the malware underscored the sophistication of malware sponsored by wealthy nation states.



The Flame espionage malware that infected computers in Iran achieved mathematic breakthroughs that could only have been accomplished by world-class cryptographers, two of the world's foremost cryptography experts said.



"We have confirmed that Flame uses a yet unknown MD5 chosen-prefix collision attack," Marc Stevens wrote in an [e-mail posted to a cryptography discussion group](#) earlier this week. "The collision attack itself is very interesting from a scientific viewpoint, and there are already some practical implications." Benne de Weger, a Stevens colleague and another expert in cryptographic collision

### Flame

**Revealed: Stuxnet "beta's" devious alternate attack on Iran nuke program**

**Massive espionage malware targeting governments undetected for 5 years**

**Iranian computers targeted by new malicious data wiper program**

**New in-the-wild malware**

# SHA1 collision: <https://shattered.io/>

Here are two PDF files that display different content, yet have the same SHA-1 digest.

**SHattered**  
The first concrete collision attack against SHA-1  
<https://shattered.io>

**CWI** **Google**

Marc Stevens  
Pierre Karpman

Elie Bursztein  
Ange Albertini  
Yarik Markov

**SHattered**  
The first concrete collision attack against SHA-1  
<https://shattered.io>

**CWI** **Google**

Marc Stevens  
Pierre Karpman

Elie Bursztein  
Ange Albertini  
Yarik Markov

```
└─ sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccb7f0a 1.pdf
38762cf7f55934b34d179ae6a4c80cadccb7f0a 2.pdf
└─ /tmp/sha1
└─ sha256sum *.pdf
2bb787a73e37352f92383abe7e2902936d1059ad9f1ba6daaa9c1e58ee6970d0 1.pdf
d4488775d29bdef7993367d541064dbdda50d383f89f0aa13a6ff2e0894ba5ff 2.pdf
```

0.64G 3-11h

# Academic cryptography vs. real-world security

MD5 was known to have weaknesses in the 1990s.

A full collision was computed in 2004.

People are still using MD5 now.

SHA-1's flaws have been known since 2005.

A full collision was computed in 2017.

Deprecation of SHA-1 has been slowed by intense resistance.

Linus Torvalds on Git's use of SHA-1:

```
I doubt the sky is falling for git as a source
control management tool. Do we want to migrate to another hash? Yes.
Is it "game over" for SHA1 like people want to say? Probably not.
```

```
I haven't seen the attack details, but I bet
```

```
(a) the fact that we have a separate size encoding makes it much
harder to do on git objects in the first place
```

```
(b) we can probably easily add some extra sanity checks to the opaque
data we do have, to make it much harder to do the hiding of random
data that these attacks pretty much always depend on.
```

# Academic cryptography vs. real-world security

Problem: Deprecating weak algorithms or parameters breaks backwards compatibility.

Problem: Many people think they understand cryptography and can make their own security choices.

Problem: Cryptography is hard.

General Principle: Attacks get better. An “academic” break violating a theoretical definition of security may lead later on to a “real-world” vulnerability.

National Institute for Standards and Technology (NIST) held a world-wide competition to develop a new hash function standard.

Contest webpage:

<http://csrc.nist.gov/groups/ST/hash/index.html>

Requested parameters:

- Design: Family of functions with 224, 256, 384, 512 bit output sizes
- Security: CR, one-wayness, near-collision resistance, others...
- Efficiency: as fast or faster than SHA2-256

**Submissions:** 64

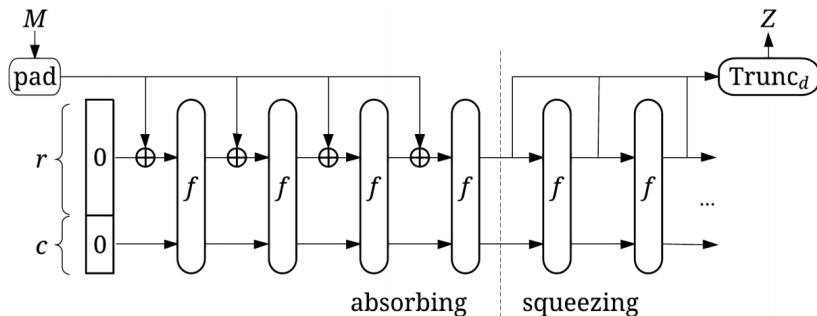
**Round 1:** 51

**Round 2:** 14: BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein.

**Finalists:** 5: BLAKE, Grostl, JH, Keccak, Skein.

**SHA3:** 1: Keccak

# SHA3: The Sponge construction



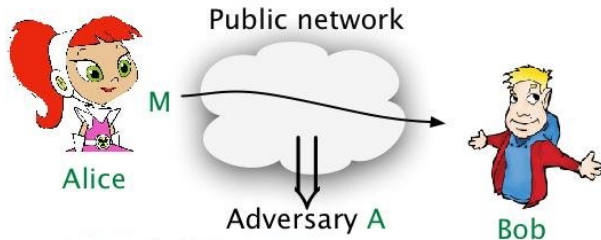
$f: \{0, 1\}^{r+c} \rightarrow \{0, 1\}^{r+c}$  is a (public, invertible!) permutation.

$d$  is the number of output bits, and  $c = 2d$ .

SHA3 does not use the MD paradigm used by the MD and SHA2 series.

Shake( $M, d$ )— Extendable-output function, returning any given number  $d$  of bits.

# Integrity and authenticity



The goal is to ensure that

- $M$  really originates with Alice and not someone else
- $M$  has not been modified in transit

# Integrity and authenticity example

Alice

Bob  
(Bank)



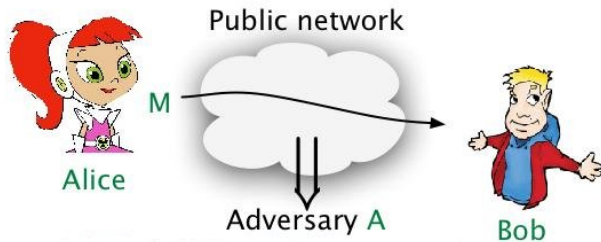
Alice  
Pay \$100 to Charlie

Adversary Eve might

- Modify "Charlie" to "Eve"
- Modify "\$100" to "\$1000"

Integrity prevents such attacks.

# Is a hash function a good cryptographic integrity check?



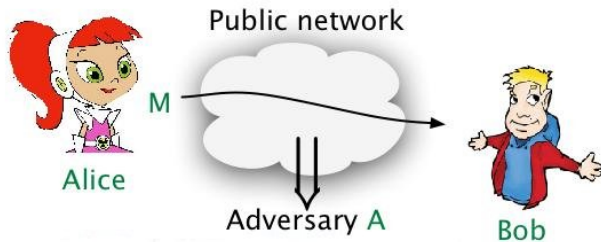
Proposal:

- Alice sends  $(M, T = H(M))$  using a collision-resistant hash function like SHA256.
- Bob receives  $(M', T')$  and checks that  $T' = H(M')$ .

Assume the adversary  $A$  can read and modify messages in transit.

Does this ensure the integrity of  $M$ ?

# Is a hash function a good cryptographic integrity check?



Proposal:

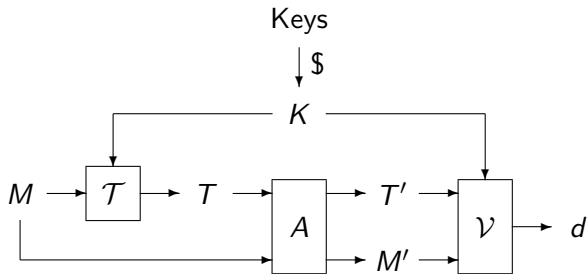
- Alice sends  $(M, T = H(M))$  using a collision-resistant hash function like SHA256.
- Bob receives  $(M', T')$  and checks that  $T' = H(M')$ .

Assume the adversary  $A$  can read and modify messages in transit.

Does this ensure the integrity of  $M$ ? **No.**

# Message authentication codes

A message authentication code  $\mathcal{T} : \text{Keys} \times D \rightarrow R$  is a family of functions. The envisaged usage is shown below, where  $A$  is the adversary:



We refer to  $T$  as the MAC or tag. We have defined

Algorithm  $\mathcal{V}_K(M', T')$

If  $\mathcal{T}_K(M') = T'$  then return 1 else return 0

Sender and receiver share key  $K$ .

To authenticate  $M$ , sender transmits  $(M, T)$  where  $T = \mathcal{T}_K(M)$ .

Upon receiving  $(M', T')$ , the receiver accepts  $M'$  as authentic iff  $\mathcal{V}_K(M', T') = 1$ , or, equivalently, iff  $\mathcal{T}_K(M') = T'$ .

Let  $\mathcal{T}: \text{Keys} \times D \rightarrow R$  be a message authentication code. Let  $A$  be an adversary.

Game  $\text{UFCMA}_{\mathcal{T}}$

**procedure** Initialize

$K \xleftarrow{\$} \text{Keys}; S \leftarrow \emptyset$

**procedure** Tag( $M$ )

$T \leftarrow \mathcal{T}_K(M); S \leftarrow S \cup \{M\}$   
return  $T$

**procedure** Finalize( $M, T$ )

If  $M \in S$  then return false

If  $M \notin D$  then return false

Return  $(T = \mathcal{T}_K(M))$

The uf-cma advantage of adversary  $A$  is

$$\text{Adv}_{\mathcal{T}}^{\text{uf-cma}}(A) = \Pr \left[ \text{UFCMA}_{\mathcal{T}}^A \Rightarrow \text{true} \right]$$

Adversary  $A$  does not get the key  $K$ .

It can call **Tag** with any message  $M$  of its choice to get back the correct tag  $T = \mathcal{T}_K(M)$ .

To win, the adversary  $A$  must output a message  $M \in D$  and a tag  $T$  that are

- Correct:  $T = \mathcal{T}_K(M)$
- New:  $M \notin S$ , meaning  $M$  was not a query to **Tag**

**Interpretation:** **Tag** represents the sender and Finalize represents the receiver. Security means that the adversary can't get the receiver to accept a message that is not authentic, meaning was not already transmitted by the sender.

## Example: Basic CBC MAC

Let  $E : \{0, 1\}^k \times B \rightarrow B$  be a block cipher, where  $B = \{0, 1\}^n$ . View a message  $M \in B^*$  as a sequence of  $n$ -bit blocks,  $M = M[1] \dots M[m]$ .

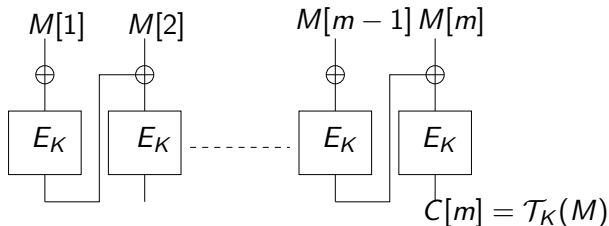
The basic CBC MAC  $\mathcal{T} : \{0, 1\}^k \times B^* \rightarrow B$  is defined by

**Alg**  $\mathcal{T}_K(M)$

$C[0] \leftarrow 0^n$

for  $i = 1, \dots, m$  do  $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$

return  $C[m]$



# Splicing attack on basic CBC MAC

**Alg**  $\mathcal{T}_K(M)$

$C[0] \leftarrow 0^n$

for  $i = 1, \dots, m$  do

$C[i] \leftarrow E_K(C[i-1] \oplus M[i])$

return  $C[m]$

**adversary**  $A$

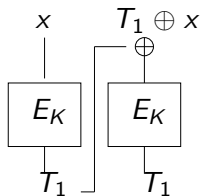
Let  $x \in \{0, 1\}^n$

$T_1 \leftarrow \mathbf{Tag}(x)$

$M \leftarrow x || T_1 \oplus x$

Return  $M, T_1$

Then,



$$\begin{aligned}\mathcal{T}_K(M) &= E_K(E_K(x) \oplus T_1 \oplus x) \\ &= E_K(T_1 \oplus T_1 \oplus x) \\ &= E_K(x) \\ &= T_1\end{aligned}$$

# Insecurity of basic CBC MAC

**Alg**  $\mathcal{T}_K(M)$

$C[0] \leftarrow 0^n$

for  $i = 1, \dots, m$  do

$C[i] \leftarrow E_K(C[i-1] \oplus M[i])$

return  $C[m]$

**adversary**  $A$

Let  $x \in \{0, 1\}^n$

$T_1 \leftarrow \mathbf{Tag}(x)$

$M \leftarrow x || T_1 \oplus x$

Return  $M, T_1$

Then  $\text{Adv}_{\mathcal{T}}^{\text{uf-cma}}(A) = 1$  and  $A$  is efficient, so the basic CBC MAC is not UF-CMA secure.

Suppose Alice transmits  $(M_1, T_1)$  to Bank where  $M_1 = \text{"Pay \$100 to Bob"}$ . Adversary

- Captures  $(M_1, T_1)$
- Keeps re-transmitting it to bank

Result: Bob gets \$100, \$200, \$300,...

Our UF-CMA notion of security does not ask for protection against replay, because  $A$  will not win if it outputs  $M, T$  with  $M \in S$ , even if  $T = \mathcal{T}_K(M)$  is the correct tag.

**Question:** Why not?

**Answer:** Replay is best addressed as an add-on to standard message authentication. This can be done using timestamps or synchronized counters.

# Any PRF is a MAC

If  $F$  is PRF-secure then it is also UF-CMA-secure:

**Theorem [GGM86,BKR96]:** Let  $F : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$  be a family of functions. Let  $A$  be a uf-cma adversary making  $q$  **Tag** queries and having running time  $t$ . Then there is a prf-adversary  $B$  such that

$$\text{Adv}_F^{\text{uf-cma}}(A) \leq \text{Adv}_F^{\text{prf}}(B) + \frac{1}{2^n}.$$

Adversary  $B$  makes  $q + 1$  queries to its Fn oracle and has running time  $t$  plus some overhead.

We do not prove this here, but we give a little intuition.

# Intuition for why PRFs are UF-CMA-secure

- 1 Random functions make good (UF-CMA) MACs
- 2 PRFs are pretty much as good as random functions

For (1), suppose  $F_n : D \rightarrow \{0, 1\}^n$  is random and consider  $A$  who

- Can query  $F_n$  at any points  $x_1, \dots, x_q \in D$  it likes
- To win, must output  $x, T$  such that  $x \notin \{x_1, \dots, x_q\}$  but  $T = F_n(x)$

Then,

$$\Pr[A \text{ wins}] =$$

# Intuition for why PRFs are UF-CMA-secure

- 1 Random functions make good (UF-CMA) MACs
- 2 PRFs are pretty much as good as random functions

For (1), suppose  $F_n : D \rightarrow \{0, 1\}^n$  is random and consider  $A$  who

- Can query  $F_n$  at any points  $x_1, \dots, x_q \in D$  it likes
- To win, must output  $x, T$  such that  $x \notin \{x_1, \dots, x_q\}$  but  $T = F_n(x)$

Then,

$$\Pr[A \text{ wins}] = \frac{1}{2^n}$$

because  $A$  did not query  $F_n(x)$ .

# Intuition for why PRFs are UF-CMA-secure

- 1 Random functions make good (UF-CMA) MACs
- 2 PRFs are pretty much as good as random functions

For (2), consider  $A$  who

- Can query  $F_K$  at any points  $x_1, \dots, x_q \in D$  it likes
- To win, must output  $x, T$  such that  $x \notin \{x_1, \dots, x_q\}$  but  $T = F_K(x)$

If  $\Pr[A \text{ wins}]$  is significantly more than  $2^{-n}$  then we are detecting a difference between  $F_K$  and a random function.

A family of functions  $F: \text{Keys} \times D \rightarrow R$  is

- FIL (Fixed-input-length) if  $D = \{0, 1\}^\ell$  for some  $\ell$
- VIL (Variable-input-length) if  $D$  is a “large” set like  $D = \{0, 1\}^*$  or  $D = \{ M \in \{0, 1\}^* : 0 < |M| < n2^n \text{ and } |M| \bmod n = 0 \}$  .  
for some  $n \geq 1$  or ...

We have families we are willing to assume are PRFs, namely block ciphers and compression functions, but they are FIL.

**PRF Domain Extension Problem:** Given a FIL PRF, construct a VIL PRF.

**PRF Domain Extension Problem:** Given a FIL PRF, construct a VIL PRF.

The basic CBC MAC is a candidate construction but we saw above that it fails to be UF-CMA and thus also fails to be a PRF.

We will see solutions that work including

- ECBC: The encrypted CBC-MAC
- HMAC: A highly standardized and used hash-function based MAC

# ECBC MAC

Let  $E : \{0, 1\}^k \times B \rightarrow B$  be a block cipher, where  $B = \{0, 1\}^n$ . The encrypted CBC (ECBC) MAC  $\mathcal{T} : \{0, 1\}^{2k} \times B^* \rightarrow B$  is defined by

**Alg**  $\mathcal{T}_{K_{in}||K_{out}}(M)$

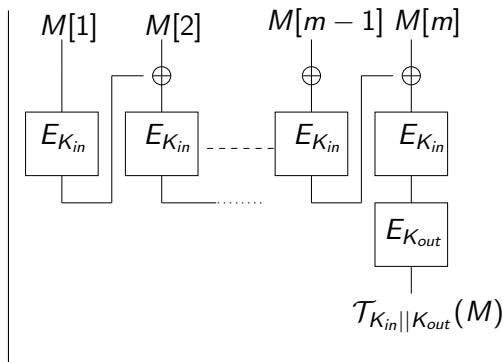
$C[0] \leftarrow 0^n$

for  $i = 1, \dots, m$  do

$C[i] \leftarrow E_{K_{in}}(C[i-1] \oplus M[i])$

$T \leftarrow E_{K_{out}}(C[m])$

return  $T$



# Birthday attacks on MACs

There is a large class of MACs, including ECBC MAC, HMAC, ... which are subject to a **birthday attack** that violates UF-CMA using about  $q \approx 2^{n/2}$  **Tag** queries, where  $n$  is the tag (output) length of the MAC.

Furthermore, we can typically show this is best possible, so the birthday bound is the “true” indication of security.

The class of MACs in question are called iterated-MACs and work by iterating some lower level primitive such as a block cipher or compression function.

Birthday attack is best possible:

**Theorem:** Let  $E : \{0, 1\}^k \times B \rightarrow B$  be a family of functions, where  $B = \{0, 1\}^n$ . Define  $F : \{0, 1\}^{2k} \times B^* \rightarrow \{0, 1\}^n$  by

**Alg**  $F_{K_{in}||K_{out}}(M)$

$C[0] \leftarrow 0^n$

for  $i = 1, \dots, m$  do  $C[i] \leftarrow E_{K_{in}}(C[i-1] \oplus M[i])$

$T \leftarrow E_{K_{out}}(C[m]);$  return  $T$

Let  $A$  be a prf-adversary against  $F$  that makes at most  $q$  oracle queries, these totalling at most  $\sigma$  blocks, and has running time  $t$ . Then there is a prf-adversary  $D$  against  $E$  such that

$$\text{Adv}_F^{\text{prf}}(A) \leq \text{Adv}_E^{\text{prf}}(D) + \frac{\sigma^2}{2^n}$$

and  $D$  makes at most  $\sigma$  oracle queries and has running time about  $t$ .

# Security of iterated MACs

The number  $q$  of  $m$ -block messages that can be safely authenticated is about  $2^{n/2}/m$ , where  $n$  is the block-length of the block cipher, or the length of the chaining input of the compression function.

<b>MAC</b>	$n$	$m$	$q$
DES-ECBC-MAC	64	1024	$2^{22}$
AES-ECBC-MAC	128	1024	$2^{54}$
AES-ECBC-MAC	128	$10^6$	$2^{44}$
HMAC-SHA1	160	$10^6$	$2^{60}$
HMAC-SHA256	256	$10^6$	$2^{108}$

$m = 10^6$  means message length 16Mbytes when  $n = 128$ .

# MACing with hash functions

The software speed of hash functions (MD5, SHA1) lead people in 1990s to ask whether they could be used to MAC.

But such cryptographic hash functions are **keyless**.

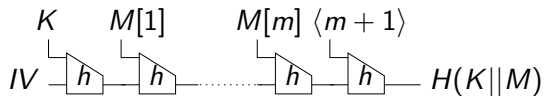
**Question:** How do we key hash functions to get MACs?

**Proposal:** Let  $H : D \rightarrow \{0, 1\}^n$  represent the hash function and set

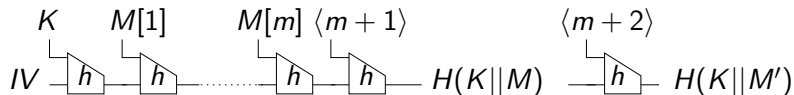
$$\mathcal{T}_K(M) = H(K||M)$$

Is this UF-CMA / PRF secure?

# Length extension attack



# Length extension attack



Let  $M' = M||\langle m+1 \rangle$ . Then

$$H(K||M') = h(\langle m+2 \rangle || H(K||M))$$

so given the MAC  $H(K||M)$  of  $M$  we can easily forge the MAC of  $M'$ .

Suppose  $H: D \rightarrow \{0,1\}^n$  is the hash function, built from an underlying compression function via the MD transform.

Let  $B \geq n/8$  denote the byte-length of a message block ( $B = 64$  for MD5, SHA1, SHA256)

Define the following constants

- $\text{ipad}$  : The byte 36 repeated  $B$  times
- $\text{opad}$  : The byte 5C repeated  $B$  times

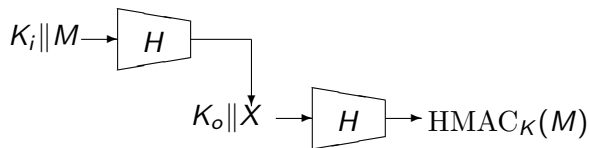
HMAC:  $\{0, 1\}^n \times D \rightarrow \{0, 1\}^n$  is defined as follows:

**Alg** HMAC( $K, M$ )

$K_i \leftarrow \text{ipad} \oplus K \parallel 0^{8B-n}$  ;  $K_o \leftarrow \text{opad} \oplus K \parallel 0^{8B-n}$

$X \leftarrow H(K_i \parallel M)$  ;  $Y \leftarrow H(K_o \parallel X)$

Return  $Y$



## Features:

- Black box use of the hash function, easy to implement
- Fast in software

## Usage:

- As a MAC for message authentication
- As a PRF for key derivation

## Security:

- Subject to a birthday attack
- Security proof shows there is no better attack [BCK96,Be06]

**Adoption and Deployment:** HMAC is one of the most widely standardized and used cryptographic constructs: SSL/TLS, SSH, IPsec, FIPS 198, IEEE 802.11, IEEE 802.11b, ...

**Theorem:** [BCK96] HMAC is a secure PRF assuming

- The compression function is a PRF
- The hash function is collision-resistant (CR)

But attacks show MD5 and SHA1 are **not** CR.

So are HMAC-MD5 and HMAC-SHA1 secure?

- No attacks so far, but
- Proof becomes vacuous!

**Theorem:** [Be06] HMAC is a secure PRF assuming **only**

- The compression function is a PRF

Current attacks do not contradict this assumption. This result may explain why HMAC-MD5 and HMAC-SHA1 are standing even though the hash functions are broken with regard to collision resistance.

# HMAC Recommendations

- Don't use HMAC-MD5
- No immediate need to remove HMAC-SHA1
- HMAC-SHA256, HMAC-SHA512 are fine choices.
  
- SHA3 is not vulnerable to length extension attacks.
- $\text{SHA3}(K||M)$  is a secure MAC. (KMAC)