

Foundations of Applied Cryptography

Adam O'Neill

Based on <http://cseweb.ucsd.edu/~mihir/cse207/>



Setting the Stage

- We now have two lower-level primitives in our tool bag: **blockciphers** and **hash functions**.

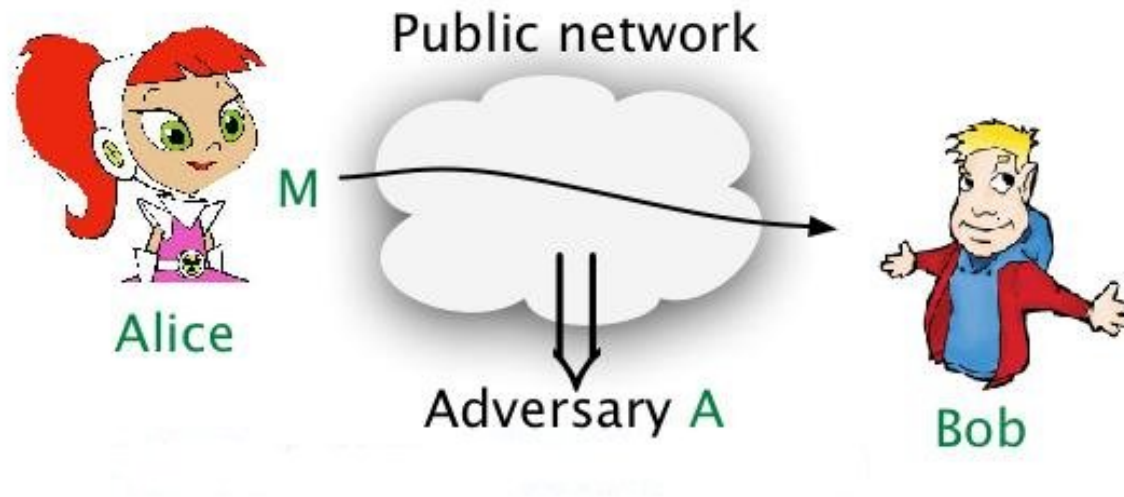
Setting the Stage

- We now have two lower-level primitives in our tool bag: **blockciphers** and **hash functions**.
- Today we study our second higher-level primitive, **message authentication codes**.

Setting the Stage

- We now have two lower-level primitives in our tool bag: **blockciphers** and **hash functions**.
- Today we study our second higher-level primitive, **message authentication codes**.
- Note that **authenticity** of data is arguably even more important than **privacy**.

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 ←

Example: Electronic Banking

Alice

Bob
(Bank)



Adversary Eve might

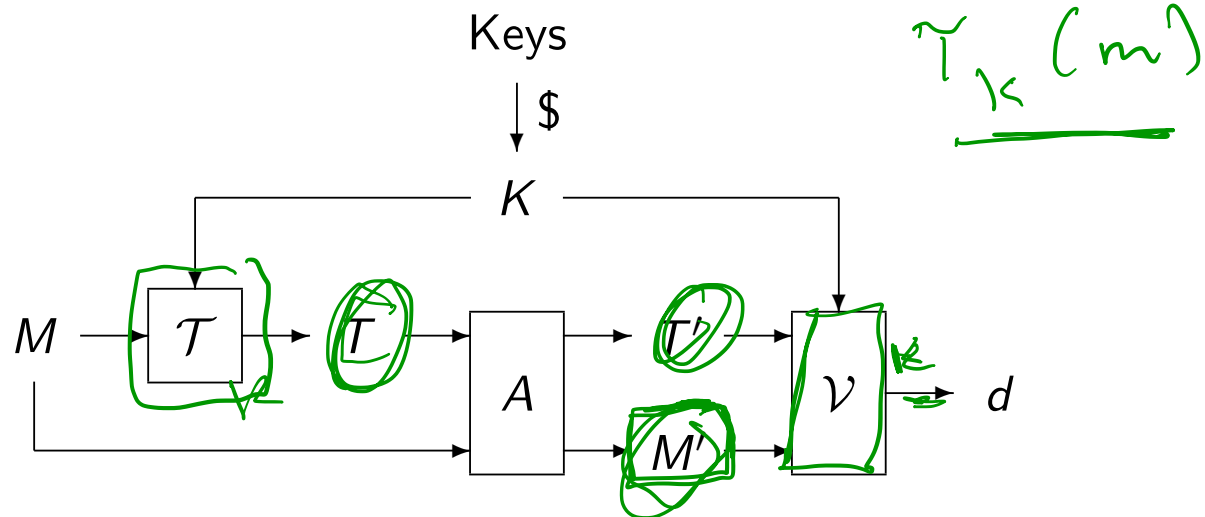
- Modify “Charlie” to “Eve”
- Modify “\$100” to “\$1000”

Integrity prevents such attacks.

Syntax and Usage

(K, τ)

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


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

Usage

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 $\mathcal{T}_K(M') = T'$.


canonical
ver. alg.

unforgeability under chosen
msg.
attack

UF-CMA

existential unforgeability

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

<p>Game $\text{UFCMA}_{\mathcal{T}}$</p> <p>procedure Initialize $K \xleftarrow{\\$} \text{Keys}; S \leftarrow \emptyset$</p> <p>procedure Tag(M) $\rightarrow m^*$ $T \leftarrow \mathcal{T}_K(M); S \leftarrow S \cup \{M\}$ return T</p>	<p>procedure Finalize(M, T) If $M \in S$ then return false If $M \notin D$ then return false Return ($T = \mathcal{T}_K(M)$)</p>
--	--

The uf-cma advantage of adversary A is

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

Explanation

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.

Lower-Bound on Tag Length

Let $\mathcal{T}: \text{Keys} \times D \rightarrow \{0, 1\}^\ell$ be a message authentication code. Specify in pseudocode an efficient adversary A making zero **Tag** queries and achieving $\text{Adv}_{\mathcal{T}}^{\text{uf-cma}}(A) = 2^{-\ell}$.

Conclusion: Tags have to be long enough.

For 80 bit security, tags have to be at least 80 bits.

attack



Basic CBC-MAC

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

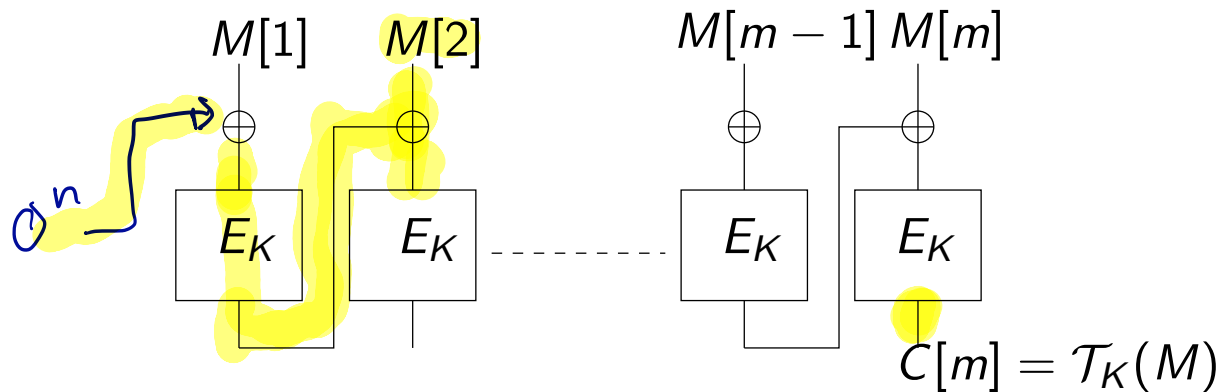
The basic CBC MAC $\mathcal{T} : \{0, 1\}^k \times B^* \rightarrow B$ is defined by

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

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

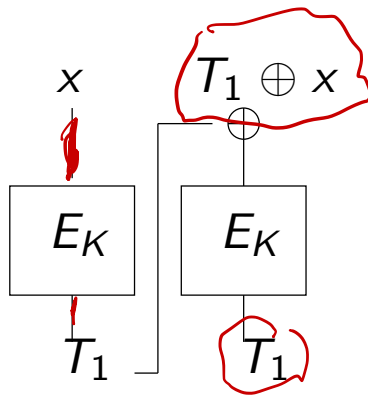
Let $x \in \{0, 1\}^n$

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

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

Return M, T_1

Then,



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

Replay Attacks

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.

Using Timestamps

Let $Time_A$ be the time as per Alice's local clock and $Time_B$ the time as per Bob's local clock.

- Alice sends $(M, \mathcal{T}_K(M), Time_A)$
- Bob receives $(M, T, Time)$ and accepts iff $T = \mathcal{T}_K(M)$ and $|Time_B - Time| \leq \Delta$ where Δ is a small threshold.

Does this work?

Using Timestamps

Let $Time_A$ be the time as per Alice's local clock and $Time_B$ the time as per Bob's local clock.

- Alice sends $(M, \mathcal{T}_K(M\|Time_A), Time_A)$
- Bob receives $(M, T, Time)$ and accepts iff $\mathcal{T}_K(M\|Time) = T$ and $|Time_B - Time| \leq \Delta$ where Δ is a small threshold.

Using Counters

Alice maintains a counter ctr_A and Bob maintains a counter ctr_B . Initially both are zero.

- Alice sends $(M, \mathcal{T}_K(M||ctr_A))$ and then increments ctr_A
- Bob receives (M, T) . If $\mathcal{T}_K(M||ctr_B) = T$ then Bob accepts and increments ctr_B .

PRF-as-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

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

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

tight
reduction

$n \geq 128$ AES

PRF Domain Extension

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

- **FIL** (Fixed-input-length) if $D = \{0, 1\}^\ell$ for some ℓ
- **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 blockciphers and compression functions, but they are FIL.

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

PRF Domain Extension

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. The exercises explored other solutions.

We will see solutions that work including

- ECBC: The encrypted CBC-MAC *2-key*
- CMAC: A NIST standard *1-key*
- HMAC: A highly standardized and used hash-function based MAC

encrypted

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

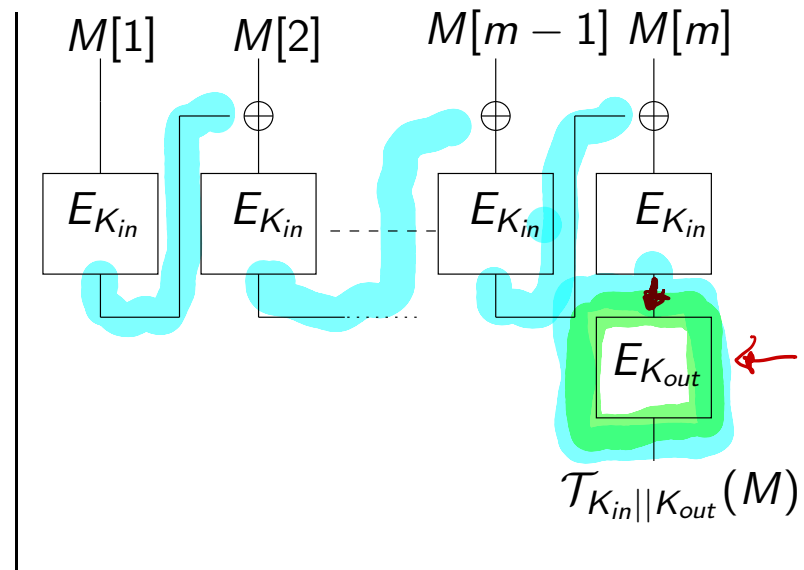
$C[0] \leftarrow 0^n$

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

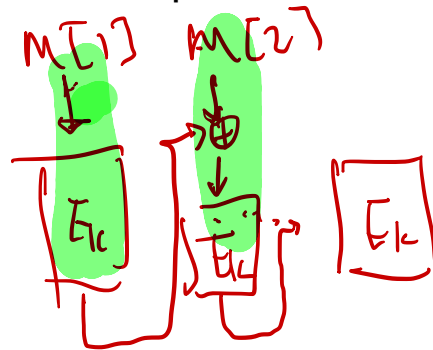


Birthday Attacks

There is a large class of MACs, including ECBC MAC, CMAC, 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 blockcipher or compression function.



Security of ECBC

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 σ blocks, and has running time t . Then there is a prf-adversary B against E such that

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

2^{-128}
 $n = 128$

and B makes at most σ oracle queries and has running time about t .

$$\frac{\sigma^2}{2^n}$$

Implications

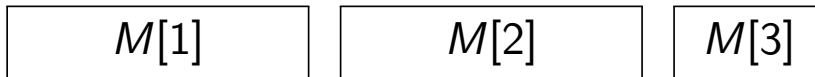
The number q of m -block messages that can be safely authenticated is about $2^{n/2}/m$, where n is the block-length of the blockcipher, or the length of the chaining input of the compression function.

MAC	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 $n = 128$.

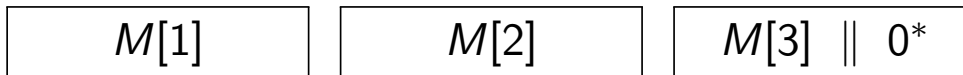
Non-Full Messages

So far we assumed messages have length a multiple of the block-length of the blockcipher. Call such messages *full*. How do we deal with non-full messages?



The obvious approach is padding. But how we pad matters.

Padding with 0^* :



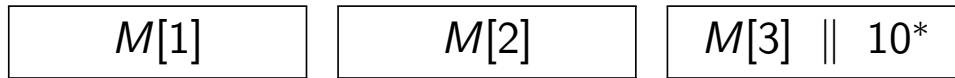
adversary A

$T \leftarrow \mathbf{Tag}(1^n 1^n 0)$; Return $(1^n 1^n 00, T)$

This adversary has uf-cma advantage 1.

Non-Full Messages

Padding with 10^* : For a non-full message



For a full message



This works, but if M was full, an extra block is needed leading to an extra blockcipher operation.

MACing with Hash Function

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

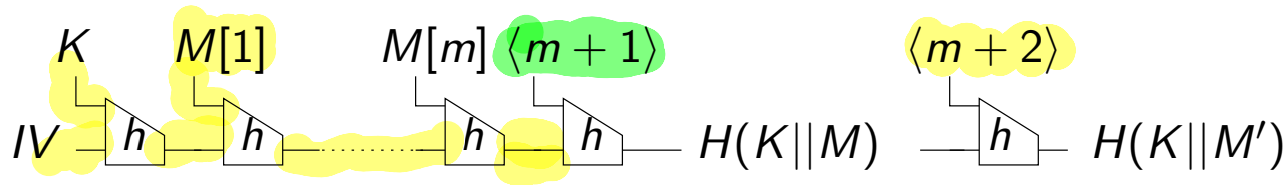
$$T_K(M) = \underbrace{H(K || M)}$$

$$H(m || k)$$

$$H(k || m || k)$$

Is this UF-CMA / PRF secure?

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' .

Exercise: Specify in pseudocode an adversary mounting the above attack to achieve uf-cma advantage 1 using 1 **Tag** query.

HMAC (BCK'96)

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, SHA512)

Define the following constants

- ipad : The byte 36 repeated B times
- opad : The byte 5C repeated B times

512 512 512
 K_0 $M_1 \dots M_m$

512

HMAC - SHA1

Suppose $H : D \rightarrow \{0, 1\}^{160}$ is the hash function. HMAC has a 160-bit key K . Let

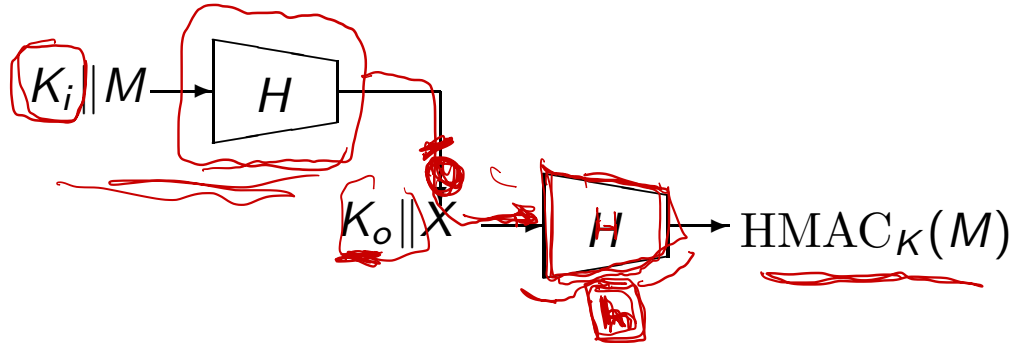
$$K_o = \text{opad} \oplus K \parallel 0^{352} \text{ and } K_i = \text{ipad} \oplus K \parallel 0^{352}$$

where

$$\text{opad} = 5D \text{ and } \text{ipad} = 36$$

in HEX. Then

$$\text{HMAC}_K(M) = H(K_o \parallel H(K_i \parallel M))$$



HMAC

Features:

- Blackbox 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, ...



Security Results

Theorem: [BCK96] HMAC is a secure PRF assuming

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

But recent attacks show MD5 is **not** CR and SHA1 may not be either.

So are HMAC-MD5 and HMAC-SHA1 secure?

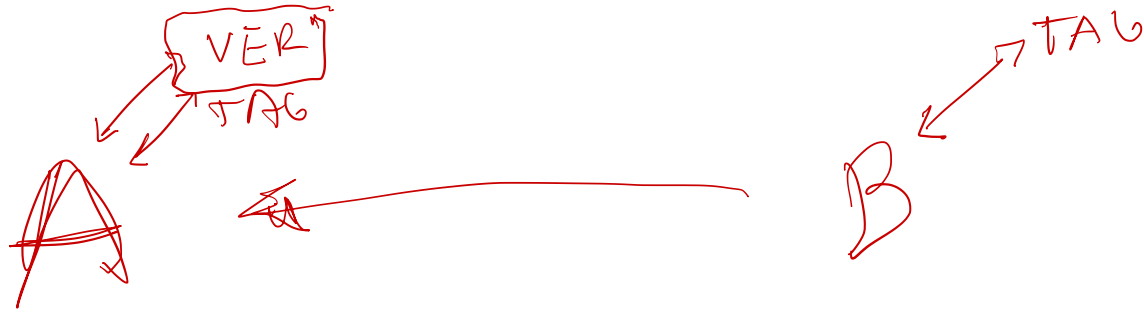
- No attacks so far, but
- Proof becomes vacuous!

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

- **The compression function is a PRF**

Current attacks do not contradict this assumption. This new result may explain why HMAC-MD5 is standing even though MD5 is broken with regard to collision resistance.

Randomized MACS, Strong
Unforgeability, verification queries



~~T~~ \leftarrow Tag(K, M)

Carter-Wegman

