Lecture 8 – Message Authentication

COSC- 466 Applied Cryptography

Adam O'Neill

Adapted from

http://cseweb.ucsd.edu/~mihir/cse107/

Setting the Stage

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

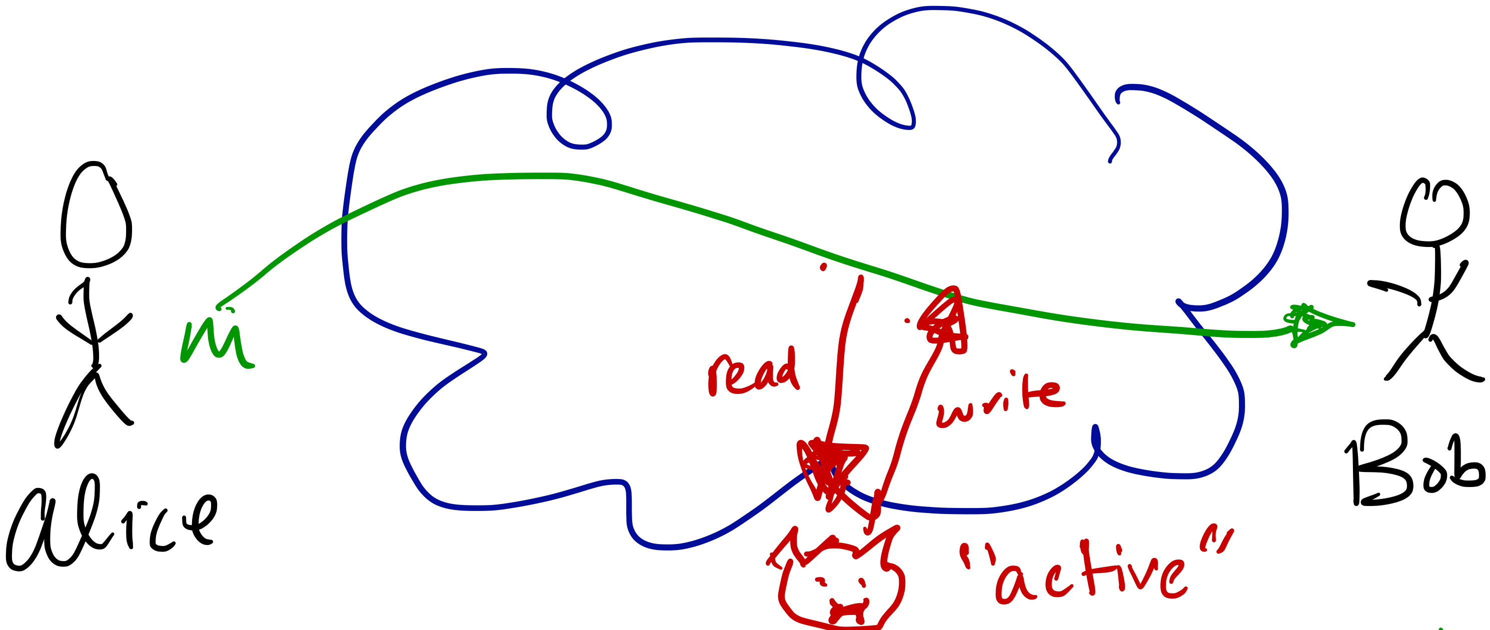
Setting the Stage

- 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.

Setting and Goals

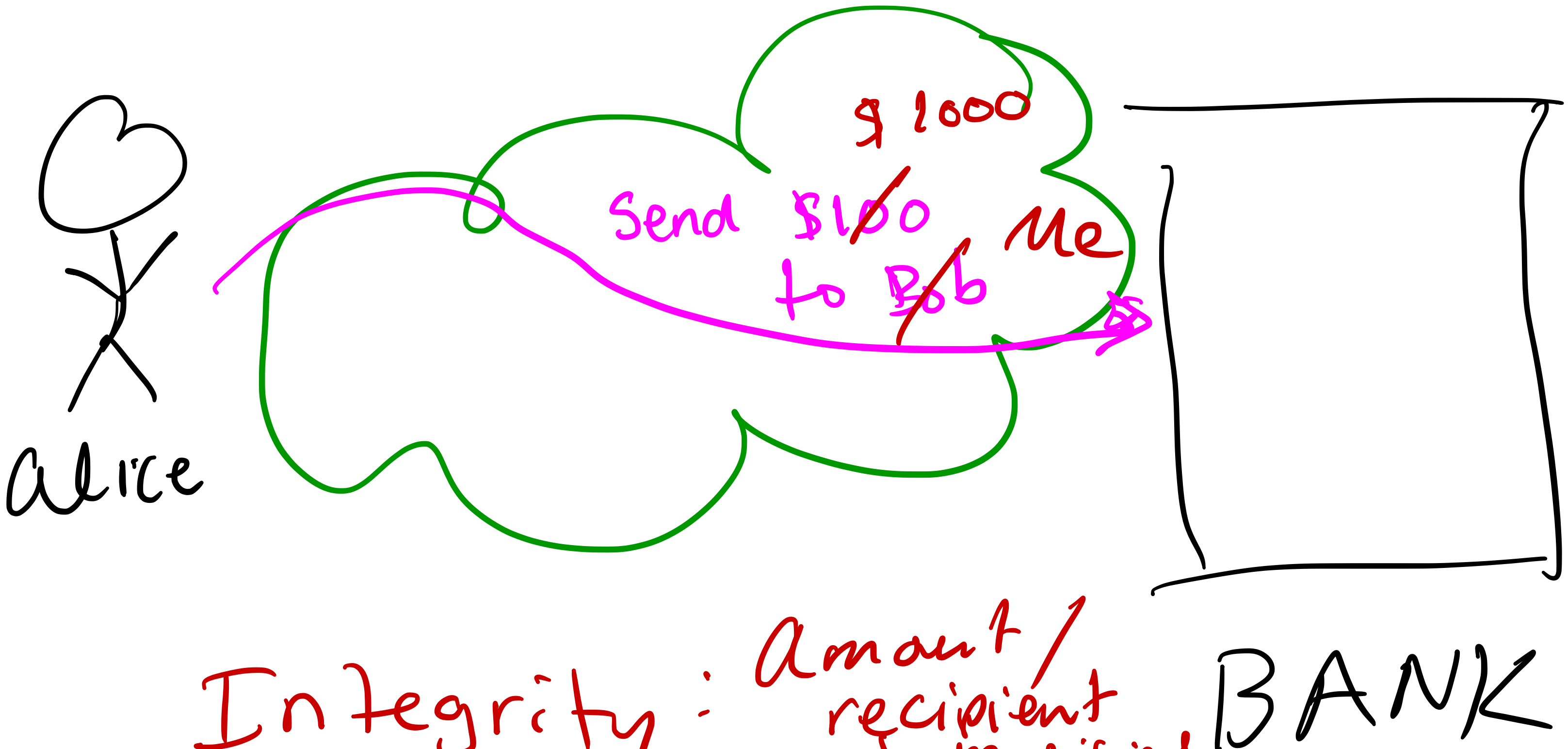


Alice wants to communicate M to Bob.

Integrity: Not modified in transit

Authenticity: Came from who it claims

Example: Electronic Banking



Integrity: amount / recipient not modified

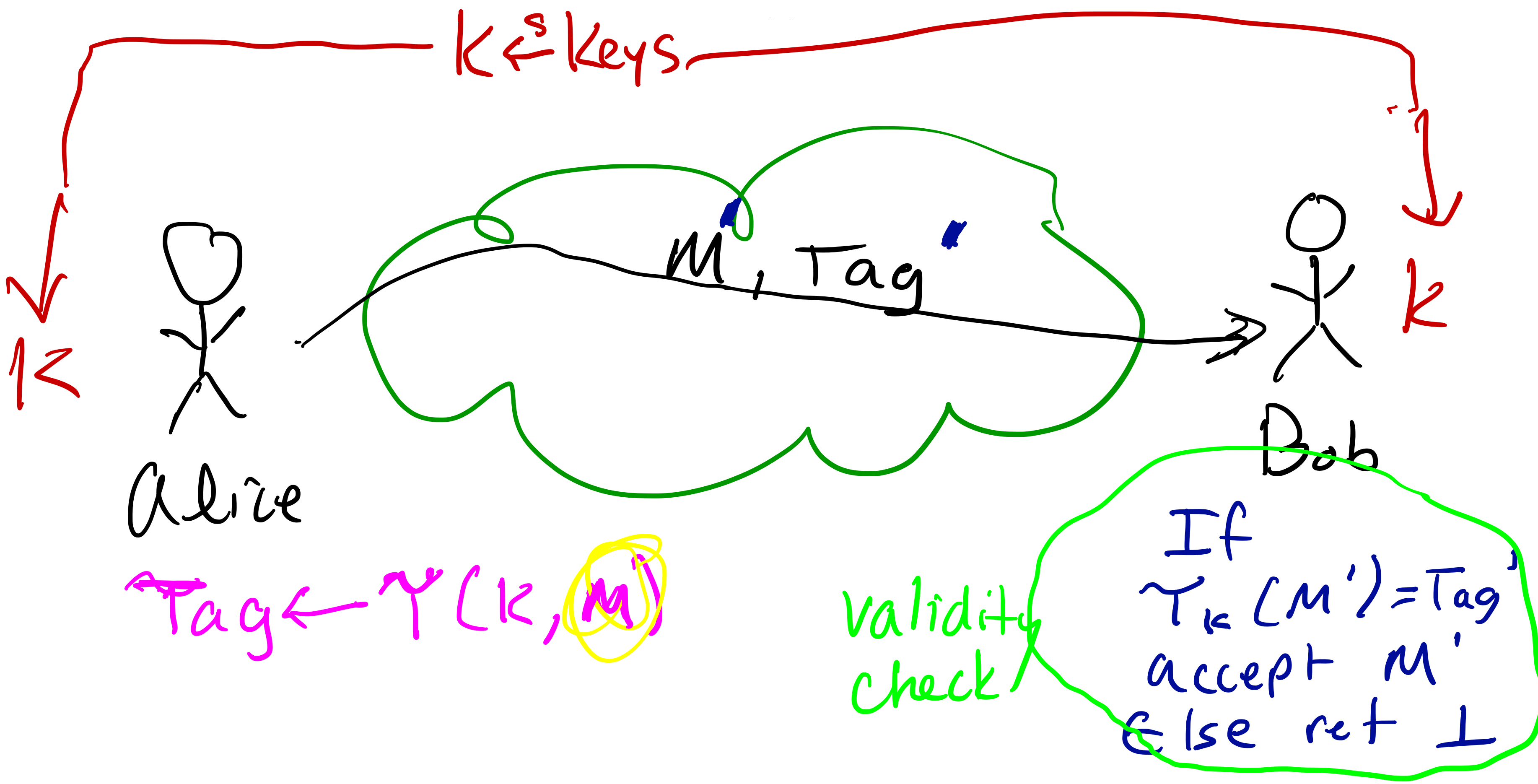
Authenticity: Really came from alice

MAC is a symmetric-key primitive

Syntax and Usage

tagging function

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



unforgeability under chosen message attack

UF-CMA

$\underbrace{\text{UF}}_{\text{goal}}$-$\underbrace{\text{CMA}}_{\text{capability}}$

$\underbrace{\text{IND}}_{\text{goal}}$-$\underbrace{\text{CPA}}_{\text{capability}}$

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

Game $\mathrm{UFCMA}_{\mathcal{T}}$	
procedure Initialize	**procedure Finalize**(M, T)
$K \xleftarrow{\$}$ Keys ; $S \leftarrow \emptyset$	If $M \in S$ then return false
procedure Tag(M)	If $M \notin D$ then return false
$T \leftarrow \mathcal{T}_K(M)$; $S \leftarrow S \cup \{M\}$	Return $(T = \mathcal{T}_K(M))$
return T	

The uf-cma advantage of adversary A is

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

Lower-Bound on Tag Length

Consider the adversary:

adversary A

Choose $m \in D$ arbitrarily

$T \xleftarrow{\$} R$

Output (m, T)

$$\mathrm{Adv}_{\mathcal{T}}^{\text{uf-cma}}(A) = \frac{1}{|R|}$$

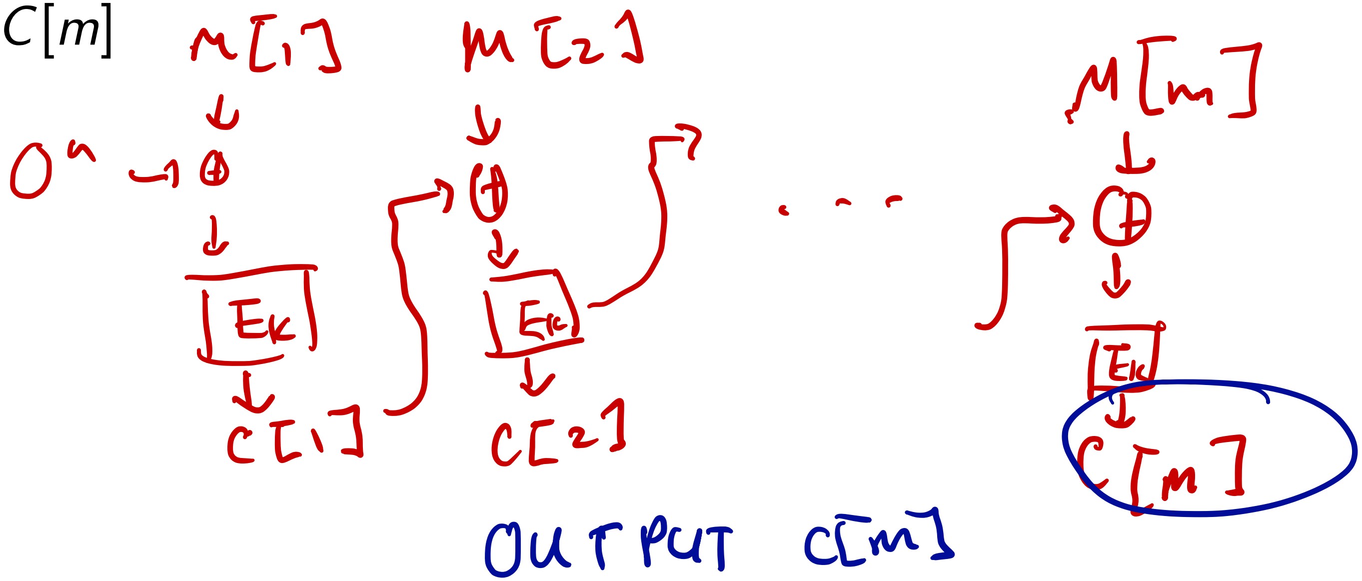
Basic CBC-MAC

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

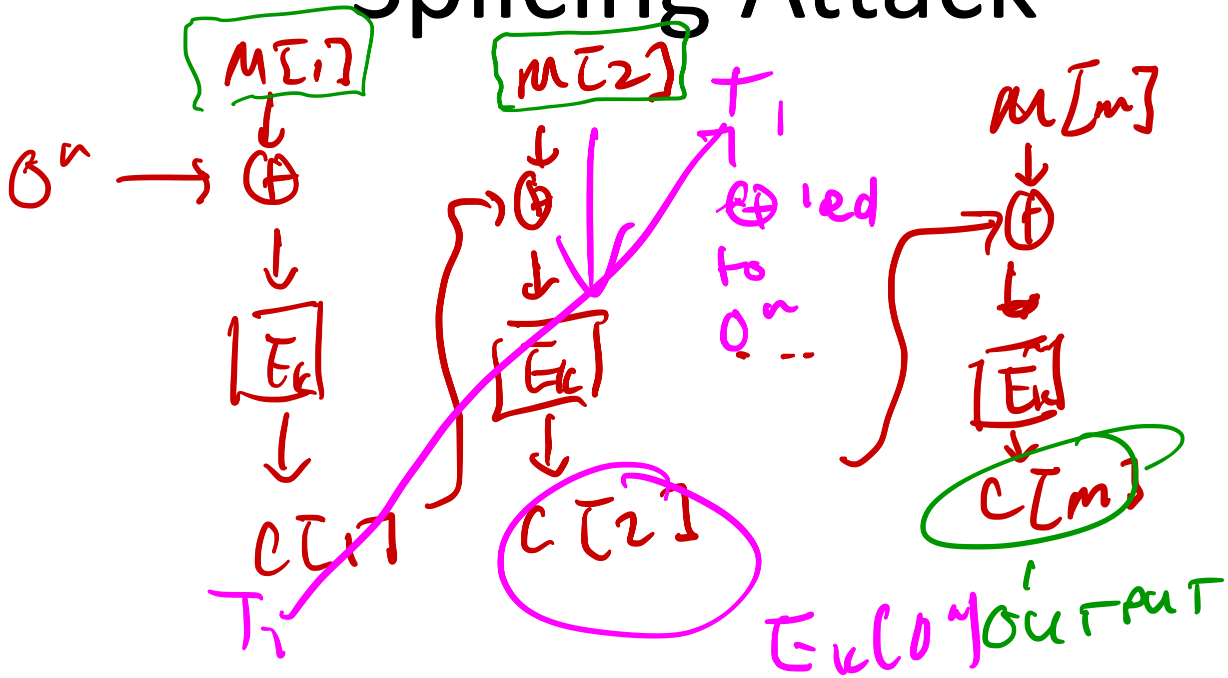
The basic CBC MAC $\mathcal{T} : \{0,1\}^k \times B^* \to B$ is defined by

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

$C[0] \leftarrow 0^n$
for $i = 1, \ldots, m$ do $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$
return $C[m]$



Splicing Attack



Suppose you query some $M = M[1]$

- get $T_1 = E_k(M[1])$

query 0^n, get $E_k(0^n) = T_2$

Now we can forge:

Message $M[1] \| T[1]$ has tag T_2

Replay Attacks

- Refers to a real-life adversary being able to capture and re-transmit a message and tag.

Replay Attacks

- Refers to a real-life adversary being able to capture and re-transmit a message and tag.
- Not captured by UF-CMA.

Replay Attacks

- Refers to a real-life adversary being able to capture and re-transmit a message and tag.
- Not captured by UF-CMA.
- Best dealt with 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. $\mathcal{T}_K(M \| Time_A)$ Fix: Add $Time_A$ to authenticated material

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

Does this work?

No! When adversary intercepts $(M, \mathcal{T}_K(M), Time_A)$ it can change $Time_A$ to some $Time_A + \alpha$ and retransmit ~~at~~ time $Time_A + \alpha$.

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.

- Counter maintained locally by each party, and is part of authenticated material
- Now adversary cannot replay a message because it won't have the right counter.

PRF-as-a-MAC

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

Theorem [GGM86,BKR96]: Let $F : \{0,1\}^k \times D \to \{0,1\}^n$ 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

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

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

Proof Intuition

Proof by reduction

Given adversary A against UF-CMA we construct an adversary B against PRF:

- Adversary B runs A
  When A queries its oracle, B uses its own oracle to answer.
- If A produces a valid forgery, B guesses REAL.

PRF Domain Extension

- We have blockciphers that are good PRFs but are fixed-input-length (FIL).

PRF Domain Extension

- We have blockciphers that are good PRFs but are fixed-input-length (FIL).
- Want a MAC that is variable-input-length (VIL).

PRF Domain Extension

- We have blockciphers that are good PRFs but are fixed-input-length (FIL).
- Want a MAC that is variable-input-length (VIL).
- By prior result this reduces to building a VIL-PRF from a FIL-PRF (aka. PRF domain extension).

encrypted

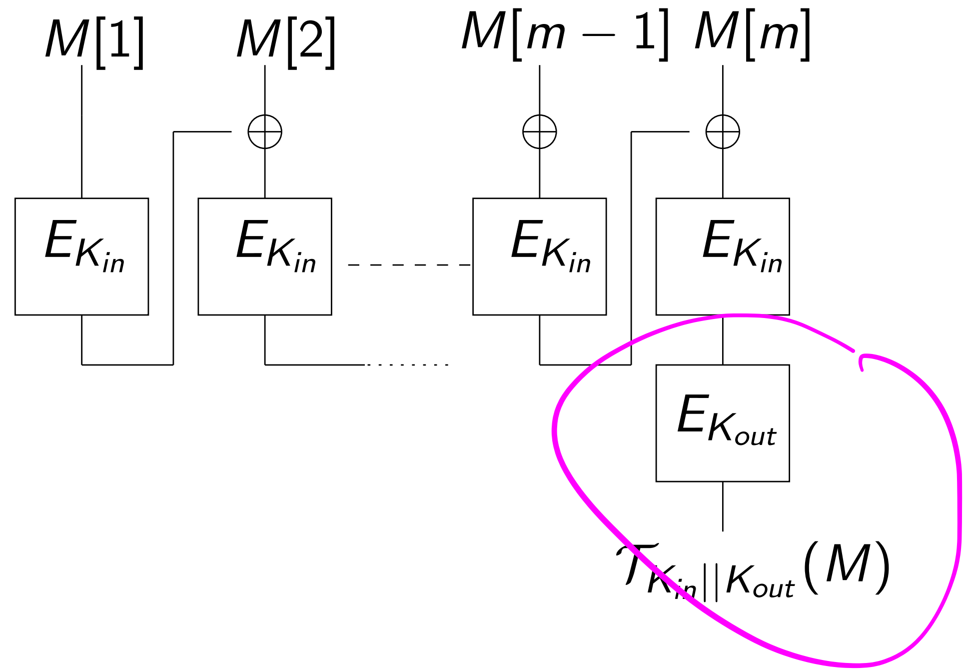
ECBC-MAC

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

two keys for underlying block cipher.

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

$C[0] \leftarrow 0^n$
for $i = 1, ..., m$ do
  $C[i] \leftarrow E_{K_{in}}(C[i-1] \oplus M[i])$
$T \leftarrow E_{K_{out}}(C[m])$
return T



Birthday Attacks

- A random function will have collisions with some probability. (we are trying to show a MAC is a VIL-PRF)
  - A MAC by definition does not have collisions

Birthday attack after q attempts succeeds with probability $\approx \frac{q(q-1)}{|R|}$

Theorem

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

$\underline{\textbf{Alg } F_{K_{in}||K_{out}}(M)}$
$C[0] \leftarrow 0^n$
for $i = 1, ..., m$ 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

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

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

Proof Intuition

- Perform a birthday attack on the underlying block cipher to break ECBC.
- Keep querying messages and hope there's a collision in inputs to the underlying blockcipher
- If not, the output always looks random

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 $n = 128$.

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 $H : D \to \{0,1\}^n$ represent the hash function and set

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

Is this UF-CMA / PRF secure?

Length-Extension Attack

HMAC [BCK'96]

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

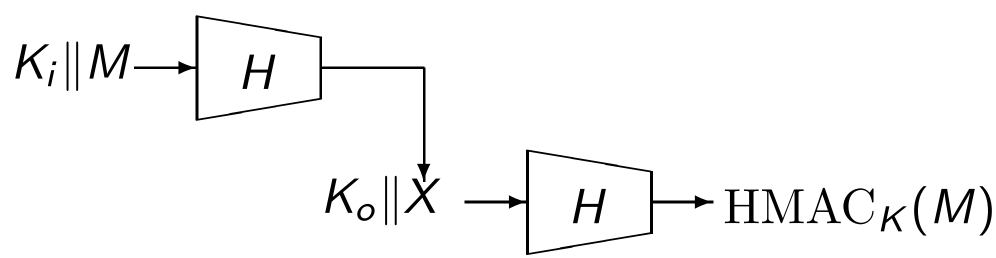
$$K_o = \text{opad} \oplus K||0^{352} \text{ and } K_i = \text{ipad} \oplus K||0^{352}$$

where

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

in HEX. Then

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



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.

Recommendations

- Don't use HMAC-MD5.

Recommendations

- Don't use HMAC-MD5.
- No immediate need to remove HMAC-SHA1.

Recommendations

- Don't use HMAC-MD5.
- No immediate need to remove HMAC-SHA1.
- But for new applications best to use HMAC-SHA2-d (for d = 256,512) or HMAC-SHA3.

Carter-Wegman MACs