CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

Apr 21, 2022

Lecture 7b

Message Authentication Codes (we lagged behind a little bit)

MACs from block ciphers

MACs from hash functions

Plan

MACs from block ciphers

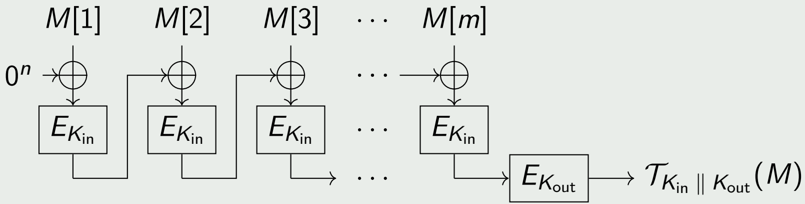
MACs from hash functions

ECBC MAC

Definition: ECBC-MAC

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

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

Security of ECBC

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

Theorem: Birthday attack is best possible

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 D against E such that

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

and D makes at most σ 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.

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 block cipher. Call such messages *full*. How do we deal with non-full messages?

$\boxed{M[1]}$ $\boxed{M[2]}$ $\boxed{M[3]}$

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

Non-full messages

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

$M[1]$	$M[2]$	$M[3]$

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

Padding with 0^*:

$M[1]$	$M[2]$	$M[3] \;\|\; 0^*$

Non-full messages

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

$M[1]$	$M[2]$	$M[3]$

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

Padding with 0^*:

$M[1]$	$M[2]$	$M[3] \;\|\; 0^*$

adversary A
$T \leftarrow \textbf{Tag}(1^n1^n0)$; Return $(1^n1^n00, T)$

This adversary has uf-cma advantage 1.

Non-full messages

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

$M[1]$	$M[2]$	$M[3] \quad \| \quad 10^*$

For a full message

$M[1]$	$M[2]$	$M[3]$	10^*

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

Bear in mind: padding for MACs is a tricky issues, and the padding methods that are given in the standards are here for a reason!

Plan

MACs from block ciphers

MACs from hash functions

MACing with hash functions

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

But such cryptographic hash functions are keyless.

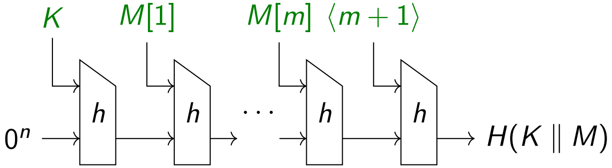
Question: How do we key hash functions to get MACs?

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

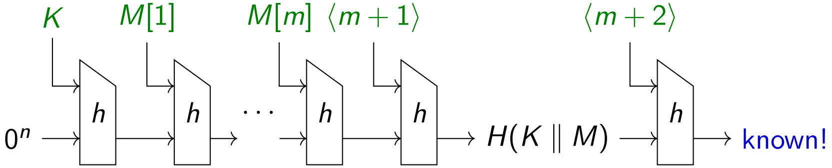
$$\mathcal{T}_K(M) = H(K \parallel M)$$

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'.

The length extension attack is a very important attack on MD-like constructions!

HMAC [BCK96]

Suppose H: $D \to \{0,1\}^n$ 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

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

HMAC [BCK96]

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

Alg $\mathrm{HMAC}(K, M)$

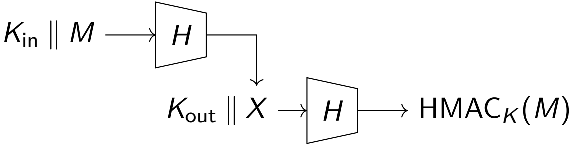
$K_{\mathsf{in}} \leftarrow \mathrm{ipad} \oplus K \,\|\, 0^{8B-n}$;

$K_{\mathsf{out}} \leftarrow \mathrm{opad} \oplus K \,\|\, 0^{8B-n}$

$X \leftarrow H(K_i \,\|\, M)$;

$Y \leftarrow H(K_o \,\|\, X)$

Return Y



HMAC

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, ...

HMAC Security

Theorem [BCK96]: HMAC is a secure PRF

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!

HMAC Security

Theorem [Be06]: HMAC is still a secure PRF

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.
- SHA3($K \parallel M$) *is* a secure MAC, and should definitely be used. (KMAC)

CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

April 21, 2022

Lecture 8

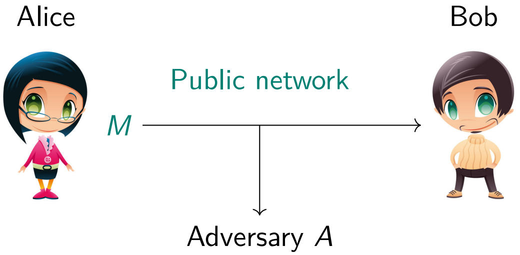
Authenticated Encryption (AE)

Security notions for AE

Generic composition

So many problems with basic CBC-MAC

So Far ...



We have looked at methods to provide privacy and authenticity separately:

Goal	Primitive	Security notion
Data privacy	symmetric encryption	IND-CPA
Data authenticity	MAC	UF-CMA

Authenticated Encryption

In practice we often want both privacy and authenticity.

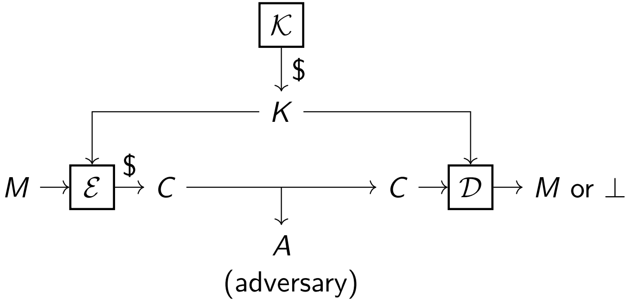
Example: A doctor wishes to send medical information M about Alice to the medical database. Then

- We want data privacy to ensure Alice's medical records remain confidential.
- We want authenticity to ensure the person sending the information is really the doctor and the information was not modified in transit.

We refer to this as authenticated encryption.

Authenticated Encryption Schemes

Syntactically, an authenticated encryption scheme is just a symmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where



Plan

Security notions for AE

Generic composition

So many problems with basic CBC-MAC

Privacy of Authenticated Encryption Schemes

The notion of privacy for symmetric encryption carries over, namely we want IND-CPA.

Integrity of Authenticated Encryption Schemes

Adversary's goal is to get the receiver to accept a "non-authentic" ciphertext C.

Integrity of ciphertexts: C is "non-authentic" if it was never transmitted by the sender.

INT-CTXT

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and A an adversary.

Game $\mathrm{INTCTXT}_{\mathcal{AE}}$

procedure Initialize
$K \xleftarrow{\$} \mathcal{K}$; $S \leftarrow \emptyset$

procedure Enc(M)
$C \xleftarrow{\$} \mathcal{E}_K(M)$
$S \leftarrow S \cup \{C\}$
Return C

procedure Finalize(C)
$M \leftarrow \mathcal{D}_K(C)$
if $(C \notin S \wedge M \neq \perp)$ then
  return true
Else return false

Definition: int-ctxt advantage

The int-ctxt advantage of A is

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{AE}}(A) = \Pr[\mathrm{INTCTXT}^{A}_{\mathcal{AE}} \Rightarrow \text{true}]$$

Integrity with privacy

The goal of authenticated encryption is to provide both integrity and privacy. We will be interested in IND-CPA + INT-CTXT.

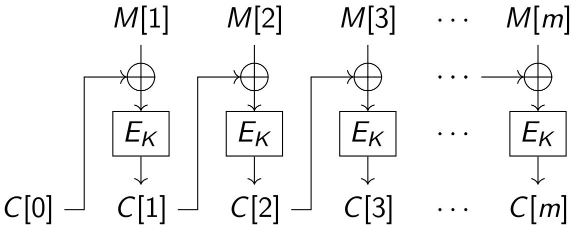
Plain Encryption Does Not Provide Integrity

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

$C[0] \xleftarrow{\$} \{0,1\}^n$
For $i = 1, \ldots, m$ do
  $C[i] \leftarrow \mathsf{E}_K(C[i-1] \oplus M[i])$
Return C

Alg $\mathcal{D}_K(C)$

For $i = 1, \ldots, m$ do
  $M[i] \leftarrow \mathsf{E}_K^{-1}(C[i]) \oplus C[i-1]$
Return M



Question: Is CBC\$ encryption INT-CTXT secure?

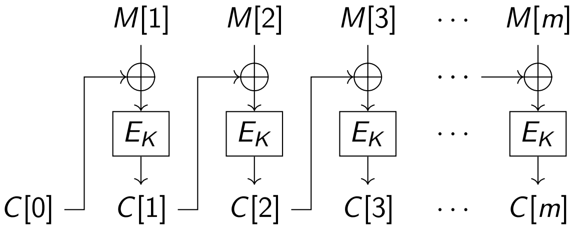
Plain Encryption Does Not Provide Integrity

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

$C[0] \xleftarrow{\$} \{0,1\}^n$
For $i = 1, \ldots, m$ do
  $C[i] \leftarrow \mathsf{E}_K(C[i-1] \oplus M[i])$
Return C

Alg $\mathcal{D}_K(C)$

For $i = 1, \ldots, m$ do
  $M[i] \leftarrow \mathsf{E}_K^{-1}(C[i]) \oplus C[i-1]$
Return M



Question: Is CBC\$ encryption INT-CTXT secure?

Answer: No, because any string $C[0]C[1]\ldots C[m]$ has a valid decryption.

Plain Encryption Does Not Provide Integrity

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

$C[0] \xleftarrow{\$} \{0,1\}^n$
For $i = 1, \ldots, m$ do
  $C[i] \leftarrow \mathsf{E}_K(C[i-1] \oplus M[i])$
Return C

Alg $\mathcal{D}_K(C)$

For $i = 1, \ldots, m$ do
  $M[i] \leftarrow \mathsf{E}_K^{-1}(C[i]) \oplus C[i-1]$
Return M

adversary A
$C[0]C[1]C[2] \xleftarrow{\$} \{0,1\}^{3n}$
Return $C[0]C[1]C[2]$

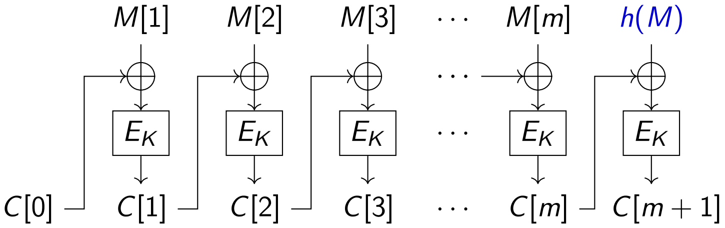
Then

$$\mathbf{Adv}^{\mathrm{int\text{-}ctxt}}_{\mathcal{SE}}(A) = 1$$

This violates INT-CTXT.

A scheme whose decryption algorithm never outputs $\perp$ cannot provide integrity!

Encryption with Redundancy

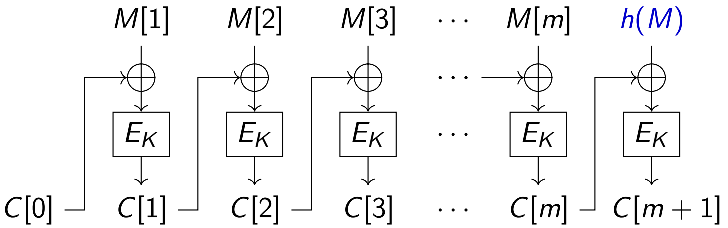


Here $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is our block cipher and $h\colon \{0,1\}^* \to \{0,1\}^n$ is a "redundancy" function, for example

- $h(M[1]\ldots M[m]) = 0^n$
- $h(M[1]\ldots M[m]) = M[1] \oplus \cdots \oplus M[m]$
- A CRC
- $h(M[1]\ldots M[m])$ is the first n bits of SHA$(M[1]\ldots M[m])$.

The redundancy is verified upon decryption.

Encryption with Redundancy



Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be our block cipher and $h\colon \{0,1\}^* \to \{0,1\}^n$ a redundancy function. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ be CBC$ encryption and define the encryption with redundancy scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ via

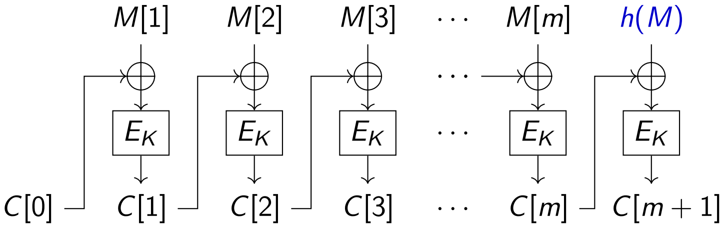
Alg $\mathcal{E}_K(M)$

$M[1] \ldots M[m] \leftarrow M$
$M[m+1] \leftarrow h(M)$
$C \xleftarrow{\$} \mathcal{E}'_K(M[1] \ldots M[m] M[m+1])$
return C

Alg $\mathcal{D}_K(C)$

$M[1] \ldots M[m] T \leftarrow \mathcal{D}'_K(C)$
if $(T = h(M[1] \,\|\, \ldots \,\|\, M[m]))$ then
    return M
else return $\perp$

Arguments in Favor of Encryption with Redundancy



The adversary will have a hard time producing the last enciphered block of a new message.

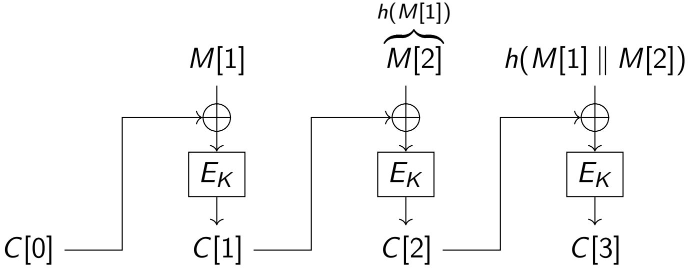
Encryption with Redundancy Fails

adversary A

$M[1] \stackrel{\$}{\leftarrow} \{0,1\}^n$; $M[2] \leftarrow h(M[1])$

$C[0]C[1]C[2]C[3] \stackrel{\$}{\leftarrow} \mathbf{Enc}(M[1]M[2])$

Return $C[0]C[1]C[2]$



This attack succeeds for any (not secret-key dependent) redundancy function h.

WEP Attack

A “real-life” rendition of this attack broke the 802.11 WEP protocol, which instantiated h as CRC and used a stream cipher for encryption [BGW].

What makes the attack easy to see is having a clear, strong and formal security model.

Plan

Security notions for AE

Generic composition

So many problems with basic CBC-MAC

Generic Composition

Definition: generic composition method (for AE)

A *generic composition method* **Comp** builds an authenticated encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}) = \textbf{Comp}[\mathcal{SE}, F]$ by combining

- a given IND-CPA symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$
- a given PRF $F : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$

	CBC\$-AES	CTR\$-AES	...
HMAC-SHA1			
CMAC			
ECBC			
⋮			

If **Comp** lives up to its claims then any entry works!

Generic Composition

Definition: generic composition method (for AE)

A *generic composition method* **Comp** builds an authenticated encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}) = \textbf{Comp}[\mathcal{SE}, F]$ by combining

- a given IND-CPA symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$
- a given PRF $F : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$

A key $K = K_e \| K_m$ for $\mathcal{AE}$ always consists of a key K_e for $\mathcal{SE}$ and a key K_m for F:

Alg $\mathcal{K}$

$K_e \xleftarrow{\$} \mathcal{K}'$; $K_m \xleftarrow{\$} \{0,1\}^k$
Return $K_e \| K_m$

Generic Composition Methods

The order in which the primitives are applied is important. Can consider

Method	Usage
Encrypt-and-MAC (E&M)	SSH
MAC-then-encrypt (MtE)	TLS 1.2
Encrypt-then-MAC (EtM)	IPSec

We study these following [BN].

Assessing **Comp**

Given a generic composition method **Comp**, and a security goal $\mathrm{X} \in \{\text{IND-CPA}, \text{INT-CTXT}\}$, we ask, does **Comp** provide X-security? There are two possible answers:

- **YES:** This means that FOR ALL secure choices of $\mathcal{SE}, F$, the authenticated encryption scheme $\mathcal{AE} = \textbf{Comp}[\mathcal{SE}, F]$ is X-secure.
- **NO:** This means that THERE EXIST secure choices of $\mathcal{SE}, F$ for which the authenticated encryption scheme $\mathcal{AE} = \textbf{Comp}[\mathcal{SE}, F]$ is NOT X-secure.

Above, secure choices of $\mathcal{SE}, F$ means these are IND-CPA-secure and PRF-secure, respectively.

So a NO does not mean **Comp** always fails to be X-secure, just that there are counter-example choices of IND-CPA $\mathcal{SE}$ and PRF F for which $\mathcal{AE}$ fails to be X-secure.

Encrypt-and-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e \| K_m}(M)$	**Alg** $\mathcal{D}_{K_e \| K_m}(C' \| T)$
$C' \xleftarrow{\$} \mathcal{E}'_{K_e}(M)$	$M \leftarrow \mathcal{D}'_{K_e}(C')$
$T \leftarrow F_{K_m}(M)$	If $(T = F_{K_m}(M))$ then return M
Return $C' \| T$	Else return $\perp$

Security	Achieved?
IND-CPA	
INT-CTXT	

Encrypt-and-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

$\textbf{Alg } \mathcal{E}_{K_e \| K_m}(M)$	$\textbf{Alg } \mathcal{D}_{K_e \| K_m}(C' \| T)$
$C' \xleftarrow{\$} \mathcal{E}'_{K_e}(M)$	$M \leftarrow \mathcal{D}'_{K_e}(C')$
$T \leftarrow F_{K_m}(M)$	If $(T = F_{K_m}(M))$ then return M
Return $C' \| T$	Else return $\perp$

Security	Achieved?
IND-CPA	NO
INT-CTXT	

Why? $T = F_{K_m}(M)$ is a deterministic function of M and allows detection of repeats.

Encrypt-and-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e||K_m}(M)$

$C' \xleftarrow{\$} \mathcal{E}'_{K_e}(M)$
$T \leftarrow F_{K_m}(M)$
Return $C'||T$

Alg $\mathcal{D}_{K_e||K_m}(C'||T)$

$M \leftarrow \mathcal{D}'_{K_e}(C')$
If $(T = F_{K_m}(M))$ then return M
Else return $\perp$

Security	Achieved?
IND-CPA	NO
INT-CTXT	

Encrypt-and-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e \| K_m}(M)$	**Alg** $\mathcal{D}_{K_e \| K_m}(C' \| T)$
$C' \xleftarrow{\$} \mathcal{E}'_{K_e}(M)$	$M \leftarrow \mathcal{D}'_{K_e}(C')$
$T \leftarrow F_{K_m}(M)$	If $(T = F_{K_m}(M))$ then return M
Return $C' \| T$	Else return $\perp$

Security	Achieved?
IND-CPA	NO
INT-CTXT	NO

Why? May be able to modify C' in such a way that its decryption is unchanged.

MAC-then-Encrypt

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e \| K_m}(M)$	**Alg** $\mathcal{D}_{K_e \| K_m}(C)$
$T \leftarrow F_{K_m}(M)$	$M \| T \leftarrow \mathcal{D}'_{K_e}(C)$
$C \xleftarrow{\$} \mathcal{E}'_{K_e}(M \| T)$	If $(T = F_{K_m}(M))$ then return M
Return C	Else return $\perp$

Security	Achieved?
IND-CPA	
INT-CTXT	

MAC-then-Encrypt

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

$\underline{\textbf{Alg } \mathcal{E}_{K_e \| K_m}(M)}$	$\underline{\textbf{Alg } \mathcal{D}_{K_e \| K_m}(C)}$
$T \leftarrow F_{K_m}(M)$	$M \| T \leftarrow \mathcal{D}'_{K_e}(C)$
$C \xleftarrow{\$} \mathcal{E}'_{K_e}(M \| T)$	If $(T = F_{K_m}(M))$ then return M
Return C	Else return $\perp$

Security	Achieved?
IND-CPA	YES
INT-CTXT	

Why? $\mathcal{SE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ is IND-CPA secure.

MAC-then-Encrypt

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

$\underline{\textbf{Alg } \mathcal{E}_{K_e \| K_m}(M)}$	$\underline{\textbf{Alg } \mathcal{D}_{K_e \| K_m}(C)}$
$T \leftarrow F_{K_m}(M)$	$M \| T \leftarrow \mathcal{D}'_{K_e}(C)$
$C \xleftarrow{\$} \mathcal{E}'_{K_e}(M \| T)$	If $(T = F_{K_m}(M))$ then return M
Return C	Else return $\perp$

Security	Achieved?
IND-CPA	YES
INT-CTXT	

MAC-then-Encrypt

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e||K_m}(M)$

$T \leftarrow F_{K_m}(M)$
$C \stackrel{\$}{\leftarrow} \mathcal{E}'_{K_e}(M||T)$
Return C

Alg $\mathcal{D}_{K_e||K_m}(C)$

$M||T \leftarrow \mathcal{D}'_{K_e}(C)$
If $(T = F_{K_m}(M))$ then return M
Else return $\perp$

Security	Achieved?
IND-CPA	YES
INT-CTXT	NO

Why? May be able to modify C in such a way that its decryption is unchanged.

Encrypt-then-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e||K_m}(M)$

$C' \xleftarrow{\$} \mathcal{E}'_{K_e}(M)$
$T \leftarrow F_{K_m}(C')$
Return $C'||T$

Alg $\mathcal{D}_{K_e||K_m}(C'||T)$

$M \leftarrow \mathcal{D}'_{K_e}(C')$
If $(T = F_{K_m}(C'))$ then return M
Else return $\perp$

Security	Achieved?
IND-CPA	
INT-CTXT	

Encrypt-then-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e\|\|K_m}(M)$	**Alg** $\mathcal{D}_{K_e\|\|K_m}(C'\|\|T)$
$C' \stackrel{\$}{\leftarrow} \mathcal{E}'_{K_e}(M)$	$M \leftarrow \mathcal{D}'_{K_e}(C')$
$T \leftarrow F_{K_m}(C')$	If $(T = F_{K_m}(C'))$ then return M
Return $C'\|\|T$	Else return $\perp$

Security	Achieved?
IND-CPA	YES
INT-CTXT	

Why? $\mathcal{SE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ is IND-CPA secure.

Encrypt-then-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e||K_m}(M)$

$C' \xleftarrow{\$} \mathcal{E}'_{K_e}(M)$
$T \leftarrow F_{K_m}(C')$
Return $C'||T$

Alg $\mathcal{D}_{K_e||K_m}(C'||T)$

$M \leftarrow \mathcal{D}'_{K_e}(C')$
If $(T = F_{K_m}(C'))$ then return M
Else return $\perp$

Security	Achieved?
IND-CPA	YES
INT-CTXT	

Encrypt-then-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

Alg $\mathcal{E}_{K_e \| K_m}(M)$	**Alg** $\mathcal{D}_{K_e \| K_m}(C' \| T)$
$C' \xleftarrow{\$} \mathcal{E}'_{K_e}(M)$	$M \leftarrow \mathcal{D}'_{K_e}(C')$
$T \leftarrow F_{K_m}(C')$	If $(T = F_{K_m}(C'))$ then return M
Return $C' \| T$	Else return $\perp$

Security	Achieved?
IND-CPA	YES
INT-CTXT	YES

Why? If $C \| T$ is new then T will be wrong.

Two keys or one?

We have used separate keys K_e, K_m for the encryption and message authentication. However, these can be derived from a single key K via $K_e = F_K(0)$ and $K_m = F_K(1)$, where F is a PRF such as a block cipher, the CBC-MAC or HMAC.

Trying to directly use the same key for the encryption and message authentication is error-prone, but works if done correctly.

Plan

Security notions for AE

Generic composition

So many problems with basic CBC-MAC

Basic CBC-MAC

Basic CBC-MAC is:

- Very simple;
- as we saw, **NOT** a secure MAC with variable-length messages;
- still, very popular and very well known/widespread;
- often, very often badly implemented.

We do have alternatives (ECBC-MAC, or EtM), but let us study the various ways to fail with basic CBC-MAC.

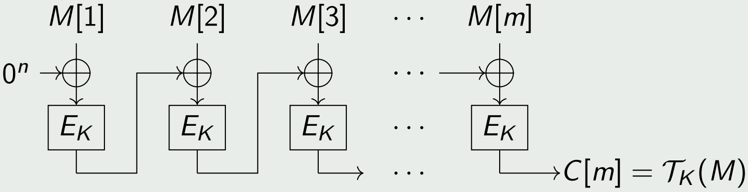
Example: Basic CBC MAC

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, $M = M[1] \dots M[m]$.

Definition: Basic CBC-MAC

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 $C[m]$



Plan

So many problems with basic CBC-MAC

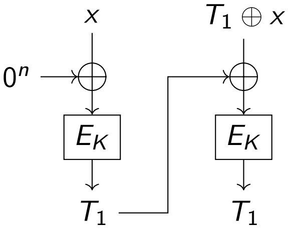
- Basic CBC-MAC and variable input length
- Basic CBC-MAC with random IV
- Same key for MAC and encryption

Splicing attack on basic CBC MAC

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 $C[m]$

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,

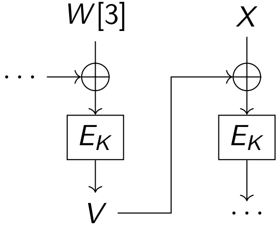


$$
\begin{aligned}
\mathcal{T}_K(M) &= E_K(E_K(x) \oplus T_1 \oplus x) \\
&= E_K(T_1 \oplus T_1 \oplus x) \\
&= E_K(x) \\
&= T_1
\end{aligned}
$$

Splicing attack: even worse

The splicing attack is not limited to a small insertion. For example:

- Adversary does three queries, gets three tags:
  - $T = \mathbf{Tag}(M[1]M[2]M[3]M[4]M[5]M[6])$.
  - $U = \mathbf{Tag}(M[1]M[2]M[3])$.
  - $V = \mathbf{Tag}(W[1]W[2]W[3])$.
- Now let $X = V \oplus U \oplus M[4]$.
  Then T is a valid MAC for the message $W[1]W[2]W[3]XM[5]M[6]$.



Because $V \oplus (\underbrace{V \oplus U \oplus M[4]}_{X}) = U \oplus M[4]$, the CBC-MAC chain continues as in the computation of the first MAC!

Fixing Basic CBC-MAC with length information

Remember: for hash functions, the Merkle-Damgård (MD) transform was adding some length information, and that was useful to prove a theorem about collision resistance.

- Sure, but it was meant for a theorem about CR. Not the same context at all.
- Still, it might seem natural to ask whether such a thing does anything good.

Fixing Basic CBC-MAC with length information

Remember: for hash functions, the Merkle-Damgård (MD) transform was adding some length information, and that was useful to prove a theorem about collision resistance.

- Sure, but it was meant for a theorem about CR. Not the same context at all.
- Still, it might seem natural to ask whether such a thing does anything good.

Bottom line:

- Appending the length DOES NOT WORK, and fails pretty much in the same way as in the previous example. (see this link).
- Prepending with the length does work. (but some implementations don't like this because the length is not necessarily known in advance...).
- This is an extra step that lazy implementations think they can do away with. Bad idea.

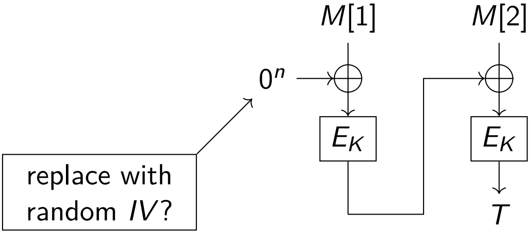
Plan

So many problems with basic CBC-MAC

- Basic CBC-MAC and variable input length
- Basic CBC-MAC with random IV
- Same key for MAC and encryption

Wouldn't a random IV be better than 0^n?

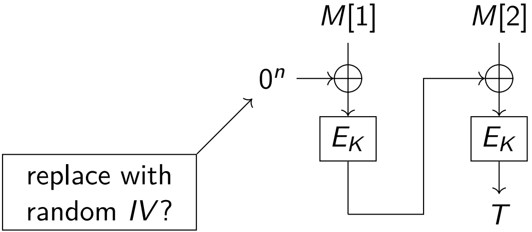
Basic CBC-MAC uses 0^n as an IV.



What if we "improve" it to a random value?

Wouldn't a random IV be better than 0^n?

Basic CBC-MAC uses 0^n as an IV.

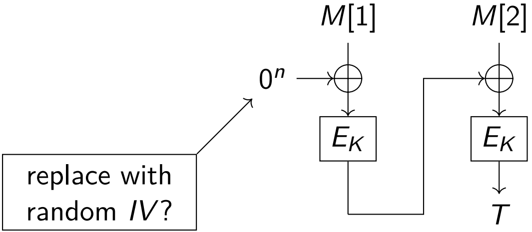


What if we "improve" it to a random value?

- Problem: the IV must be somewhere. And in many cases, that means that it might be controlled by the adversary.

Wouldn't a random IV be better than 0^n?

Basic CBC-MAC uses 0^n as an IV.



What if we "improve" it to a random value?

- Problem: the IV must be somewhere. And in many cases, that means that it might be controlled by the adversary.
- This improvement turns out to be a very bad idea: the adversary can control the first block of the message.

Plan

So many problems with basic CBC-MAC

- Basic CBC-MAC and variable input length
- Basic CBC-MAC with random IV
- Same key for MAC and encryption

K_m and K_e

Using the same key is bad practice.

What if we do this with CBC-MAC?

Exercise

Let $E = AES$. Let $\mathcal{K}$ return a random 128-bit AES key K. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where $\mathcal{E}$, $\mathcal{D}$ are below. Here, $X[i]$ denotes the i-th 128-bit block of a string whose length is a multiple of 128.

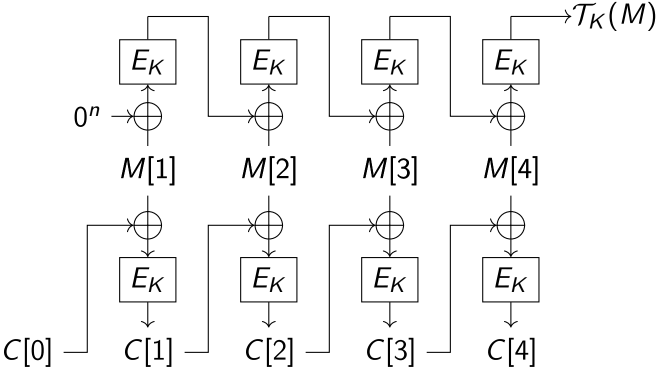
Alg $\mathcal{E}_K(M)$

if $|M| \neq 512$ then return $\perp$
$M[1] \ldots M[4] \leftarrow M$
$C_e[0] \xleftarrow{\$} \{0,1\}^{128}$; $C_m[0] \leftarrow 0^{128}$
for $i = 1, \ldots, 4$ do
  $C_e[i] \leftarrow E_K(C_e[i-1] \oplus M[i])$
  $C_m[i] \leftarrow E_K(C_m[i-1] \oplus M[i])$
$C_e \leftarrow C_e[0] C_e[1] C_e[2] C_e[3] C_e[4]$
$T \leftarrow C_m[4]$; return (C_e, T)

Alg $\mathcal{D}_K((C_e, T))$

if $|C_e| \neq 640$ then return $\perp$
$C_m[0] \leftarrow 0^{128}$
for $i = 1, \ldots, 4$ do
  $M[i] \leftarrow E_K^{-1}(C_e[i]) \oplus C_e[i-1]$
  $C_m[i] \leftarrow E_K(C_m[i-1] \oplus M[i])$
if $C_m[4] \neq T$ then return $\perp$
return M

Graphically



We want to know whether it is a good idea.

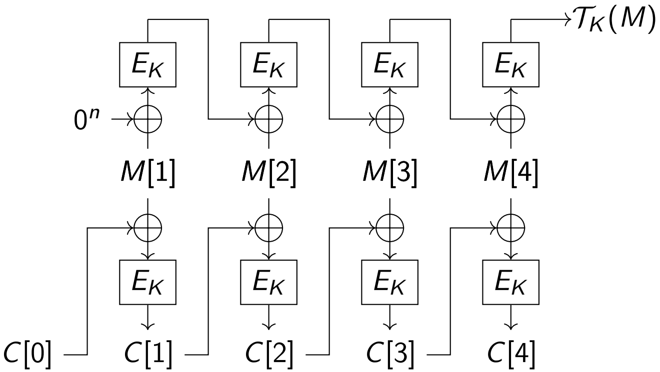
In Python/PlayCrypt

```
def Encrypt(K, M):
    assert len(M) % n_bytes == 0
    T = int_to_string(0, n_bytes)
    C = random_string(n_bytes)
    c = C
    for m in split(M, n_bytes):
        c = E(K, xor_strings(c, m))
        C += c
        T = E(K, xor_strings(T, m))
    C += T
    return C
```

In Python/PlayCrypt

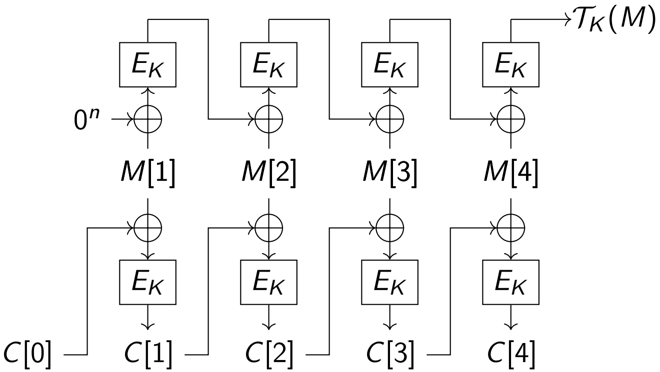
```
def Decrypt(K, C):
    assert len(C) % n_bytes == 0
    C = split(C, n_bytes)
    received_T = C[-1]
    c = C[0]
    T = int_to_string(0, n_bytes)
    M = ""
    for x in C[1:-1]:
        m = xor_strings(c, E_I(K, x))
        c = x
        T = E(K, xor_strings(T, m))
        M += m
    if T == received_T:
        return M
    else:
        return None
```

Exercise



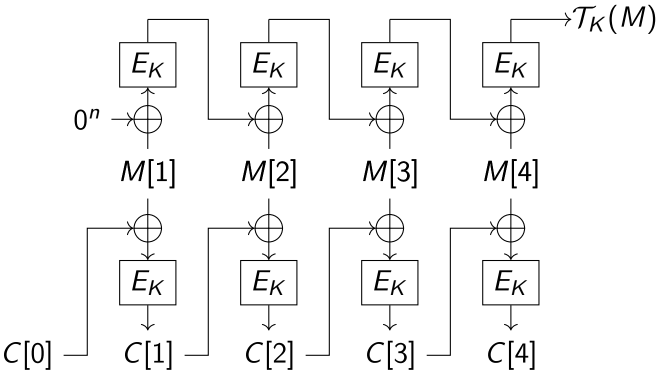
- Is $\mathcal{SE}$ IND-CPA-secure? Why or why not?

Exercise



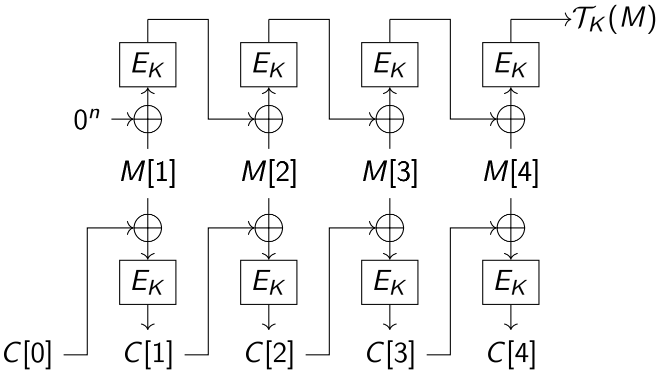
- Is $\mathcal{SE}$ IND-CPA-secure? Why or why not?
  The MAC part is deterministic. We cannot have IND-CPA in this case!

Exercise



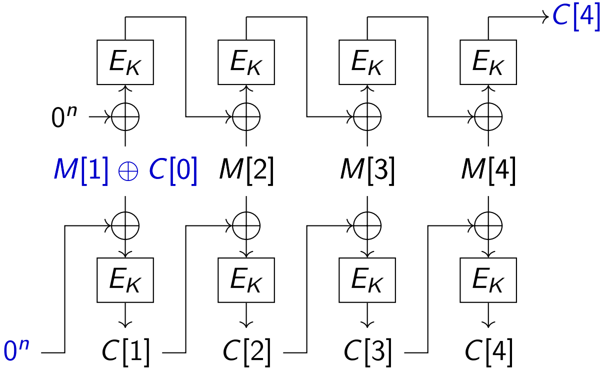
- Is $\mathcal{SE}$ INT-CTXT-secure? Why or why not?

Exercise



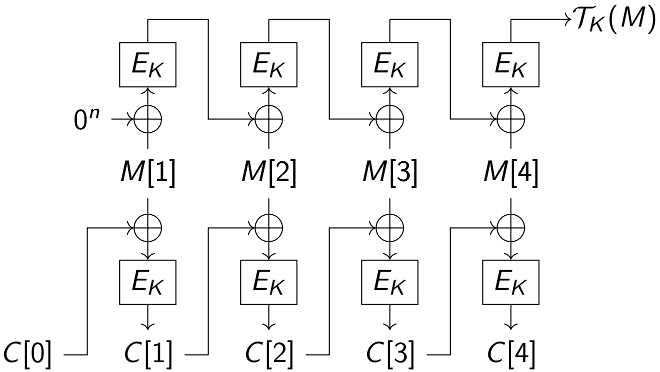
- Is $\mathcal{SE}$ INT-CTXT-secure? Why or why not?
  This is trickier.

Exercise

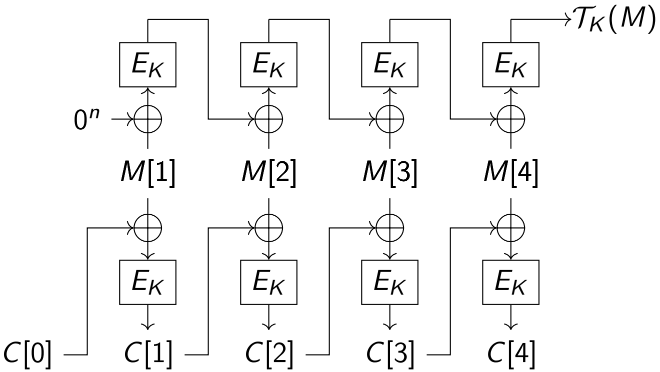


- Is $\mathcal{SE}$ INT-CTXT-secure? Why or why not?
  This is trickier.
  Exercise: show that $(0^n C[2] C[3], C[3])$ is also valid.

Exercise



- Is $\mathcal{SE}$ an Encrypt-and-MAC construction? Justify your answer.



- Is $\mathcal{SE}$ an Encrypt-and-MAC construction? Justify your answer. The generic composition mechanism assumes that K_e and K_m are distinct. If they match, we have the flaws of E&M, and more!

Comclusion about CBC-MAC

Basic CBC-MAC has so many possible misuses (including very very bad ones like Basic CBC-MAC + CTR mode).

- There are ways to do it right.
- It's also dangerously close to total blunders.
- Never code this sort of "simple thing" on your own. Good AE modes are here for that.

Authenticated encryption today

- Dedicated schemes: OCB, OCBx (x=1,2,3), GCM, CCM, EAX
- TLS uses GCM
- CAESAR competition to standardize new schemes: `http://competitions.cr.yp.to/caesar.html`