# Part I

## Message Integrity and Authenticated Encryption

## Message Integrity

Adversary is active, while messages are public.
So the adversary goal is modify messages (or forge new messages).
Example: protecting files on disk

## 1. Message Integrity and Authenticated Encryption

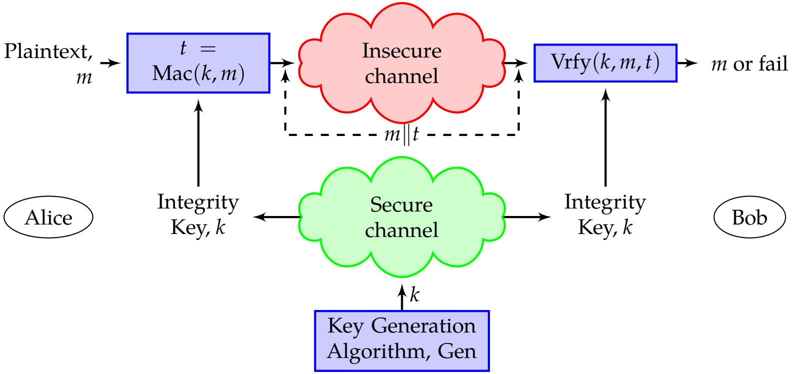
1 Message Authentication Codes (MACs)

2 MACs from PRFs and Block Ciphers

3 MACs from Hash Functions

4 Confidentiality and Message Authentication

## Integrity tags



# Message Authentication Code (MAC)

A MAC is a triple of efficient algorithms $(\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$.

| | |
|---|---|
| $\mathrm{Gen}()$ | generates a random symmetric key |
| $\mathrm{Mac}(k, m)$ | generates a fixed sized tag from the key and the message (can be randomized) |
| $\mathrm{Vrfy}(k, m, t)$ | outputs true (1) if $t$ is a valid tag for $k, m$, false (0) otherwise |

## Secure MACs

Given a random, unknown, key $k$.
Attacker model is Chosen Message Attack. The Adversary chooses $q$ messages and obtains the corresponding tags.
Attacker goal may be either:

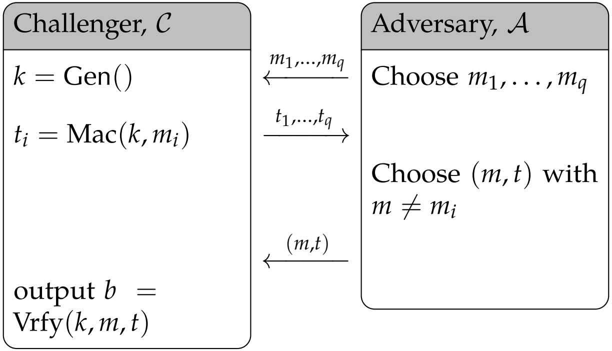
- Selective Forgery. Someone gives a new message $m$ to the Attacker. The Attacker wins if it finds a valid tag for $m$.
- Existential Forgery. The Attacker wins if it finds a new message and its valid tag.

## Secure MACs

- An attacker capable of Selective Forgery can also make an Existential Forgery
- A MAC secure against Existential Forgery is secure also against a Selective Forgery
- Classic error detection sequences (e.g. CRC) are designed to protect against random errors, not malicious errors. An intelligent attacker can use any efficient algorithm and recompute the tag.

## Secure MACs

### Formal Definition for non-adaptive $q$-query Adversary


Challenger, $\mathcal{C}$
$k = \text{Gen}()$
$t_i = \text{Mac}(k, m_i)$
output $b = \text{Vrfy}(k, m, t)$
$m_1,\ldots,m_q$
$t_1,\ldots,t_q$
$(m,t)$
Adversary, $\mathcal{A}$
Choose $m_1, \ldots, m_q$
Choose $(m, t)$ with $m \neq m_i$

A MAC is secure if $\Pr\{b = 1\}$ is negligible for any efficient $q$-query $\mathcal{A}$.

## Examples

**Example**

Suppose that for some pair $m_0, m_1$, it holds that $\mathrm{Mac}(k, m_0) = \mathrm{Mac}(k, m_1)$ for half of the keys.

Then there exists and algorithm that wins (b=1) half of the times.

**Example**

Suppose that $\mathrm{Mac}(k, m)$ is 5 bits long.

Then there exists and algorithm that wins with probability $2^{-5}$ (non negligible).

## 1. Message Integrity and Authenticated Encryption

1 Message Authentication Codes (MACs)

2 MACs from PRFs and Block Ciphers

3 MACs from Hash Functions

4 Confidentiality and Message Authentication

## MACs from PRFs

Given a PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, define a $MAC_F$ as:

$k = \mathrm{Gen}()$: choose a random key

$t = \mathrm{Mac}(k, m)$: $t = F(k, m)$

$\mathrm{Vrfy}(k, m, t)$: output 1 if $t = F(k, m)$ otherwise output 0

**Theorem**

*If F is a PRF and $1/|\mathcal{Y}|$ is negligible, then the MAC derived from F is secure.*

This MAC works only for fixed-size messages. If size of MAC is too small, a random MAC is accepted with high probability.

## Small-MAC vs Big-MAC

The MAC $t = F(k, m)$ is a secure MAC for fixed-size messages. For example you can use AES as a MAC for 16-byte messages.

For variable length messages you need to construct a Big-MAC. Popular constructions:

| | |
|---|---|
| CBC-MAC: | uses a block cipher as a Small-MAC, typically found in lower layers of the stack (mobile, WiFi, ...) |
| HMAC: | uses a hash function as a Small-MAC, typically found in internet protocols (TLS, IPSec, SSH, web APIs, ...) |

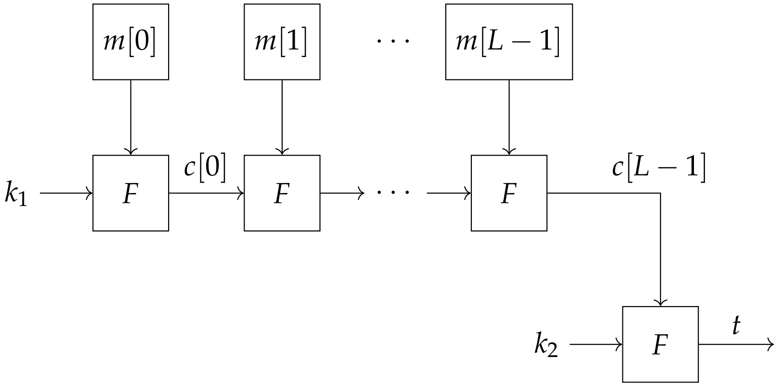
## Nested MAC (NMAC)

Given that $M = m[0]\|\ldots\|m[L-1]$, then

$$\begin{aligned} c[-1] &= k_1 \\ c[i] &= F(c[i-1], m[i]) \quad 0 \leq i < L \\ t &= F(k_2, c[L-1]) \end{aligned}$$

- Without the last encryption round it is easy to build the MAC tag of longer messages given the MAC of a prefix.
- The keys $k_1$ and $k_2$ must be different, but can be derived from a single key $k$.

## Nested MAC



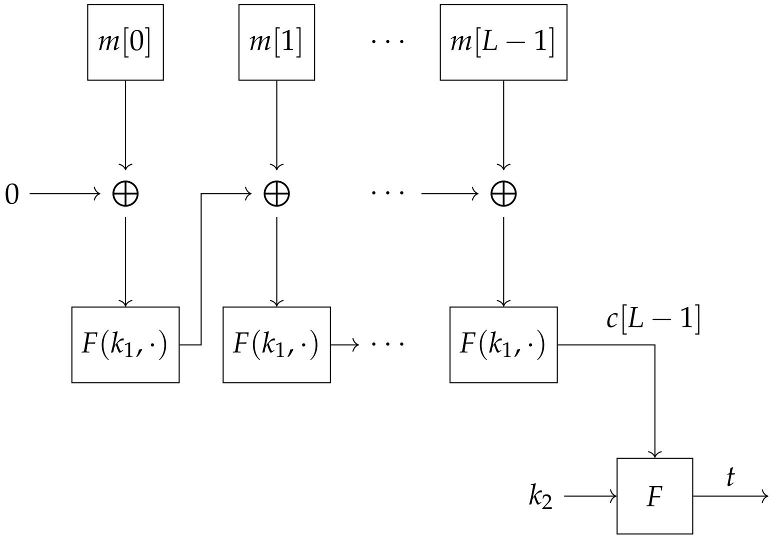
## Encrypted CBC-MAC

Given that $M = m[0] \| \ldots \| m[L-1]$, then

$$\begin{aligned} c[-1] &= 0 \\ c[i] &= F(k_1, m[i] \oplus c[i-1]) \quad 0 \le i < L \\ t &= F(k_2, c[L-1]) \end{aligned}$$

- $F$ can be a block cipher (remember PRF switching lemma)
- The keys $k_1$ and $k_2$ must be different, but can be derived from a single key $k$.

## Encrypted CBC-MAC



## Last encryption in ECBC-MAC

Consider the MAC without last round (in this MAC the tag is $c[L]$). Consider the attack:

1. Choose a one-block message $m$.
2. Obtain $c[0] = F(k_1, m)$.
3. Choose $M = m\|(m \oplus c[0])$
4. Output the message $M$ with MAC tag $t$.

Indeed:

$$c[1] = F(k_1, c[0] \oplus m[1]) = F(k_1, c[0] \oplus m \oplus c[0]) = t$$

## MAC Security Analysis

Formal

**Theorem**

*For any L-block message and every q-query adversary $\mathcal{A}$*

- *ECBC-MAC is secure as long as $2q^2/|\mathcal{X}|$ is negligible*
- *NMAC is secure as long as $q^2/2|\mathcal{K}|$ is negligible*

If we want $2q^2/|\mathcal{X}| < 2^{-32}$ then, with AES we must change the key every $2^{128/2-32} = 2^{48}$ messages. With DES every $2^{16}$ messages.

## MAC Security Attacks

### Intuition

For both CBC and NMAC, if: $\text{Mac}(k, x) = \text{Mac}(k, y)$, then $\text{Mac}(k, x\|w) = \text{Mac}(k, y\|w)$. So, if the $q$-Adversary makes so many queries that a collision is non-negligible, then the Adversary can forge a MAC.
Collisions are likely after:

- $\sqrt{|\mathcal{X}|}$ messages in ECBC-MAC
- $\sqrt{|\mathcal{K}|}$ messages in NMAC

## 1. Message Integrity and Authenticated Encryption

1. Message Authentication Codes (MACs)
2. MACs from PRFs and Block Ciphers
3. MACs from Hash Functions
4. Confidentiality and Message Authentication

## Using hash functions to buld MACs

### The HMAC construction

MACs are a kind of *keyed* hash functions, so it is natural to use hash functions for building a MAC. Since hashes are generally faster than block cipers, these MACs are also generally faster.

When the hash function is obtained iteratively (e.g. using Merkle-Damgård) special precautions must be taken.

HMAC is a construction that provides a secure MAC provided that the compression function underlying $h$ has certain pseudorandom properties. This assumption is believed to be true for the most popular hash functions.

## The HMAC construction

### Preliminaries

Let compress be a collision resistant compression function.
Let $h$ be a hash function obtained from compress using Merkle-Damgård with $z_0 = IV$.
$IV$, ipad and opad are constants of length $\ell$. Generally

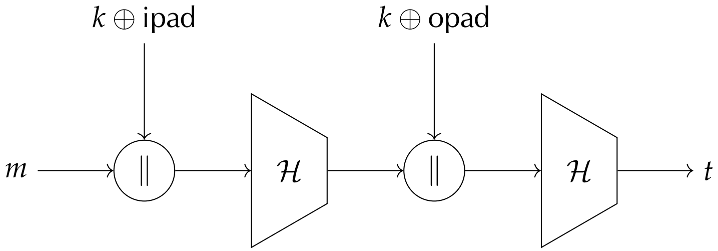
- ipad $=$ `0x5C` repeated many times
- opad $=$ `0x36` repeated many times

## The HMAC construction

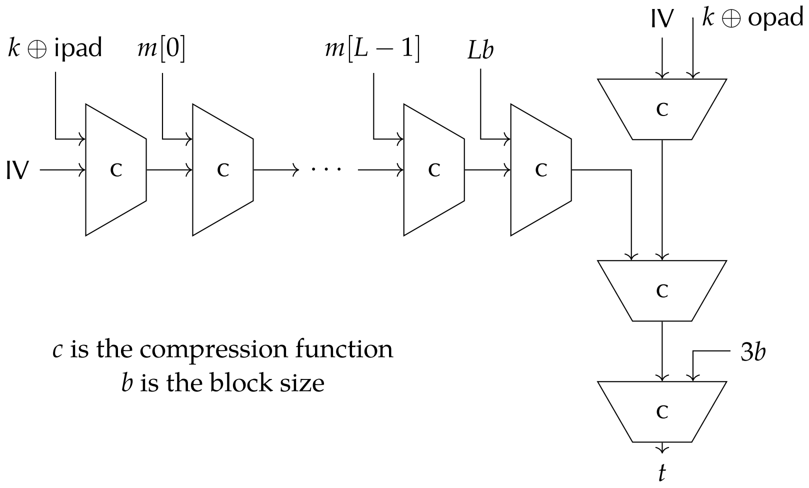
### Algorithms

| | |
|---|---|
| Gen | choose the key $k \leftarrow \{0,1\}^\ell$ randomly. |
| Mac(k,m) | output the tag $$t := h\big((k \oplus \text{opad})\|h((k \oplus \text{ipad}\|m)\big)$$ |
| Vrfy(k,m,t) | Output 1 if Mac(k,m)=t. Output 0 otherwise. |

## HMAC

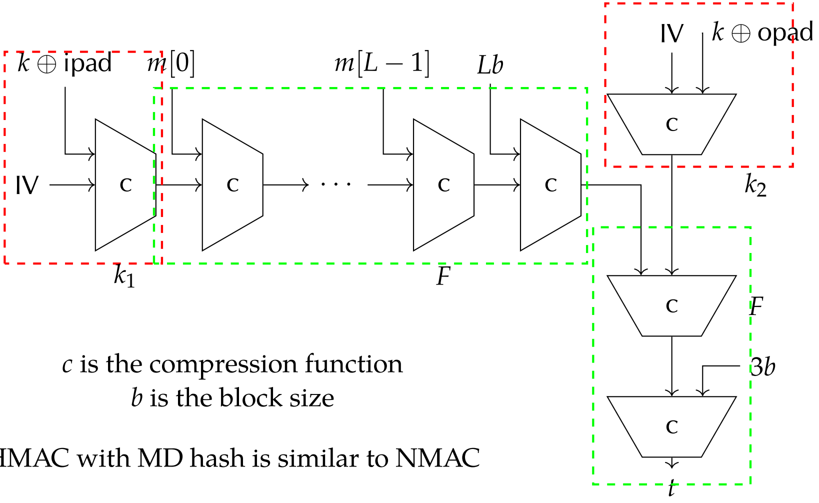


## HMAC with MD Hash Functions



$c$ is the compression function
$b$ is the block size

## HMAC with MD Hash Functions



$c$ is the compression function
$b$ is the block size

HMAC with MD hash is similar to NMAC

## Security of HMAC

HMAC can be proved to be an existentially unforgeable MAC under adaptive chosen-message attack if the following assumptions are true:

1. The internal and external prexifes look like independent and random. Formally if $k$ is random and

$$G(k) = \text{compress}(k \oplus \text{opad}) \| \text{compress}(k \oplus \text{ipad})$$

   is a secure pseudorandom generator.
2. $h$ is obtained using Merkle-Damgård on compress
3. $h$ is collision resistant when the input is a given random prefix.
   - This is a weaker condition than compress is collision resistant
4. compress is a secure fixed-length MAC
   - This may or may not be true independently on compress being collision resistant

## 1. Message Integrity and Authenticated Encryption

1 Message Authentication Codes (MACs)

2 MACs from PRFs and Block Ciphers

3 MACs from Hash Functions

4 Confidentiality and Message Authentication

## Authenticated Encryption

We know how to build an encryption scheme for confidentiality and a MAC for authentication. We want a scheme that provides confidentiality and integrity against an active attacker.
In general a random combination of an encryption scheme and a MAC is not secure, so one should use an Authenticated Encryption scheme, which provides:

- semantic security against active attacker
- ciphertext integrity (attacker cannot create new ciphertext that the receiver accepts as good)

## Authenticated Encryption (AE)

CTR and CBC modes do not provide ciphertext integrity because the receiver always deliver the decrypted ciphertext.
Implications of AE:

1. authenticity: accepted messages come from someone who knows the key (or are a replay)
2. the scheme is semantically secure against Chosen Ciphertext Attacks (CCA). A CCA is an attack in which the attacker can obtain decryption of $q$ chosen ciphetexts.

## Authenticated Encryption (AE)

Assume we have two APIs:

- an encryption scheme $\text{Enc}(k_1, m)$
- a MAC $\text{Mac}(k_2, m)$

Note that $k_1$ and $k_2$ are independent keys. Never use the same key, unless there is a specific proof of security.
Until recently, AE was not well understood, so Enc and Mac are combined in various, not always secure, ways.

- Encrypt-and-Authenticate (used in SSH)
- Authenticate-then-Encrypt (used in TLS)
- Encrypt-then-Authenticate (used in IPSec)

## Encrypt-and-Authenticate

### Definition

The transmitter calculates independently

$$c = \mathsf{Enc}(k_1, m) \quad t = \mathsf{Mac}(k_2, m)$$

and sends the pair $(c, t)$.
The receiver decrypts $m$ and verifies the tag. In case of success returns $m$, otherwise ignores the message.

### Security

Even the most secure MAC does not imply any privacy. In particular, any deterministic MAC not provide message indistingushability.
This approach must not be used.

## Authenticate-then-Encrypt

### Definition

The transmitter calculates the tag and then encrypts the tag together with the message

$$t = \mathsf{Mac}(k_2, m) \quad c = \mathsf{Enc}(k_1, m \| t)$$

and sends the pair $(c, t)$.
The receiver decripts $m$ and $t$ and verifies the tag. In case of success returns $m$, otherwise ignores the message.

### Security

The receiver decrypts the message even if it is forged (CCA). By testing whether it accepts or rejects a message an attacker can obtain information on the encryption key.
In general it can be unsecure. It can be used if Enc is rand-CTR or rand-CBC.

## Encrypt-then-Authenticate

### Definition

The transmitter encrypts the message and calculates the tag over the encrypted message

$$c = \mathsf{Enc}(k_1, m) \quad t = \mathsf{Mac}(k_2, c)$$

and sends the pair $(c, t)$.
The receiver verifies the tag. In case of success it deciphers and returns $m$, otherwise ignores the message.

### Security

It is proved secure in the general case. Beware that the MAC should protect also additional unencrypted data such as the IV and the identifiers of the used algorithms. So care must be used because it is easy to make mistakes.

## AEAD schemes

There are a few schemes that provide AE with Associated Data (AEAD). The Associated Data is authenticated but not encrypted. They are all nonce-based:

- CCM: Counter Mode encryption plus CBC-MAC using Authenticate-then-Encrypt. The same encryption key can be used for both, provided that the counter values used in the encryption do not collide with the (pre-)initialization vector used in the authentication.
- EAX: Counter Mode encryption plus CMAC (a variant of CBC-MAC) using Encrypt-then-Authenticate. Solves some efficiency problems of CCM.
- GCM (Galois Counter Mode): Counter Mode plus an ad-hoc MAC scheme based on multiplications in $GF(2^{128})$ using Encrypt-then-Authenticate. Much faster than CCM and especially suited for hardware implementation. Intel has introduced the instruction PCLMULQDQ especially for accelerating GCM.