# Message authentication codes – Authenticated encryption

## Crypto Engineering

Bruno Grenet

Université Grenoble-Alpes

https://membres-ljk.imag.fr/Bruno.Grenet/CryptoEng.html

# Introduction

## Crypto. is not *only* about encryption!

- Get access to a building, car, …
- Electronic signature for contracts, softwares, …
- Detect message tampering
- Detect "identity theft"
- …

$\Rightarrow$ require digital signatures and/or message authentication codes (MACs)

## Very important rule

Over a symmetric channel with potentially active adversaries

- It may be OK to only authenticate
- It is **never** OK to only encrypt

## Need both?

- Authenticated encryption!

1. MACs and their security

2. Designing MACs

3. Authenticated encryption

# Message authentication codes

## Definition

A message authentication code (MAC) is a mapping $\mathsf{Mac} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ with

- $\mathcal{K} = \{0,1\}^{\kappa}$: key space — for instance $\kappa = 128$
- $\mathcal{M} = \bigcup_{\ell < n} \{0,1\}^{\ell}$: message space — for instance $n = 2^{64}$
- $\mathcal{T} = \{0,1\}^{t}$: *tag* space — for instance $t = 256$

A MAC comes with a verification algorithm $\mathsf{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{0,1\}$

- $\mathsf{Vrfy}(k, m, t) = 1$ if the *tag* is valid, that is if $t \leftarrow \mathsf{Mac}(k, m)$

## Variant

A *nonce-based* MAC is a mapping $\mathsf{Mac} : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{T}$ with

- $\mathcal{N} = \{0,1\}^{s}$: *nonce space* — for instance $s = 64$
- $\mathsf{Vrfy} : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{T}$

The nonce is either deterministic or random, but publicly known and single-use

## Semantic

The *tag* authenticates the (sender of the) message

# MACs security

Informally, a MAC is secure if an adversay cannot compute *valid tags* without the key

## Three notions

Let $\mathrm{Mac}(k, \cdot)$ be a MAC with unknown key.

- Universal forgery: given $m$, *hard* to find $t$ s.t. $\mathrm{Vrfy}(k, m, t) = 1$
- Existential forgery: *hard* to build a pair $(m, t)$ s.t. $\mathrm{Vrfy}(k, m, t) = 1$
- VIL-PRF security: *hard* to distinguish $\mathrm{Mac}(k, \cdot)$ from a random function $f : \mathcal{M} \to \mathcal{T}$

(VIL-PRF stands for *variable input-length pseudorandom function*)

## Remarks

- The three notions can be defined using suitable *experiment* and *advantage*
- VIL-PRF sec. $\Rightarrow$ Existential forgery sec. $\Rightarrow$ Universal forgery sec.

1. MACs and their security

2. Designing MACs

3. Authenticated encryption

# MACs from block ciphers (*theory*)

## Case of fixed-length messages

Given $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, build

- $\mathsf{Mac}(k, m)$: compute $t \leftarrow E(k, m)$ and return $t$
- $\mathsf{Vrfy}(k, m, t)$: check whether $t = E(k, m)$

## Variable-length messages

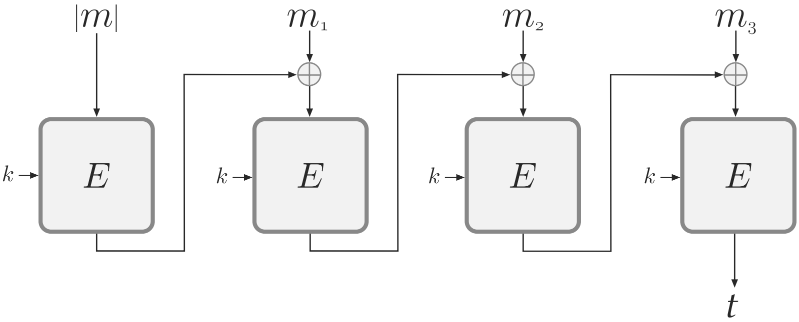
- Don't do $t_1 \leftarrow \mathsf{Mac}(k, m_1), \ldots, t_\ell \leftarrow \mathsf{Mac}(k, m_\ell)$! — *cf.* ECB
- Pad the blocks with extra information
  - Block number — no reordering
  - Total message length $\ell$ — no shortening
  - Random identifier $r$ — no recombination

  $\Rightarrow t_i \leftarrow \mathsf{Mac}(k, r \| \ell \| i \| m_i)$

## Properties

- If $E$ is a good **PRF**, Mac has good security properties
- Not efficient for variable-length messages: small, thereby numerous, blocks

# MACs from block ciphers (*practice*): ex. of CBC-MAC




## Properties

- Security proofs in the PRF model
- Only requires a block cipher
- Not very efficient

# MACs from hash functions (*theory*)

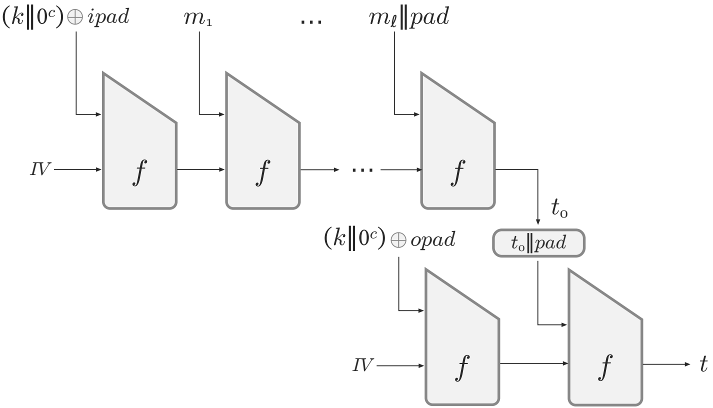
## *Hash-and-MAC*

- Given:
  - A secure Mac for fixed-length messages (with Vrfy)
  - A good hash function $H$
- Build:
  - $\mathsf{Mac}'(k, m) = \mathsf{Mac}(k, H(m))$
  - $\mathsf{Vrfy}'(k, m, t) = \mathsf{Vrfy}(k, H(m), t)$
- Security: OK if Mac is secure and $H$ is collision resistant

## Direct constructions

- Given a hash function $H$, several possibilities:
  - $\mathsf{PrefixMac}(k, m) = H(k \| m)$ — length-extension attack
  - $\mathsf{SuffixMac}(k, m) = H(m \| k)$ — collision attack
  - $\mathsf{SandwichMac}(k_1 \| k_2, m) = H(k_1 \| m \| k_2)$ — also problems
- Yet, one good solution is a variant of SandwichMac

# MACs from hash functions (*practice*): ex. of HMAC




- $\text{HMac}(k, m) = H\Big((k\|0^c) \oplus opad \,\Big\|\, H\big((k\|0^c) \oplus ipad \,\|\, m\big)\Big)$
  - $H$ is a Merkle-Damgård construction
  - $opad = (0\text{x}36)^{b/8} = 00110110 \;\; 00110110 \;\; \dots \;\; 00110110$
  - $ipad \;= (0\text{x}5\text{C})^{b/8} = 01011100 \;\; 01011100 \;\; \dots \;\; 01011100$

# HMAC properties – comparison with CBC-MAC

## HMAC properties

- Secure up to the birthday bound of $H$
- Only *black-box* calls to $H$
  - Easy implementation
  - With *white-box* access: NMAC — slightly more efficient
- Widespread use — e.g. in TLS

## Block cipher vs. Hash-based MACs

- Block cipher: usually smallish block size $\rightarrow$ limited generic security
- Hash functions: faster to process large data

  $\Rightarrow$ Hash-based constructions more used than block-cipher-based
- But one can do even better!
  - Polynomial MACs — e.g. VMAC
  - Dedicated constructions — PelicanMAC

# MACs from polynomials: polynomial hash functions

## Reminder: polynomials

- Degree-$(n-1)$ polynomial over $\mathbb{K}$: $M(X) = m_0 + m_1X + \cdots + m_{n-1}X^{n-1}$ with $m_i \in \mathbb{K}$
- Evaluation: $M(\cdot) : k \mapsto m_0 + m_1k + \cdots + m_{n-1}k^{n-1}$

## Definition

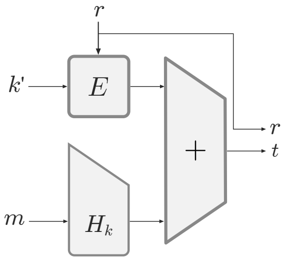
The polynomial hash functions $H_k$ (for $k \in \mathbb{K}$) are (*keyed*) hash functions defined by $H_k(m) = k \times M(k)$, where

- $m = m_0 \| \cdots \| m_{n-1} \in \mathbb{K}^n$
- $M(X) = m_0 + m_1X + \cdots + m_{n-1}X^{n-1}$

## Properties and remarks

- Multiplication by $k$ is needed for $m_0$ to "mix" with the key
- $H_k$ is linear: $H_k(a+b) = H_k(a) + H_k(b)$
- For any $a \neq b$, $\Pr_{k \twoheadleftarrow \mathbb{K}}[H_k(a) = H_k(b)] = \Pr_{k \twoheadleftarrow \mathbb{K}}[k(A(k) - B(k)) = 0] \leq \frac{n}{\#\mathbb{K}}$
  - Ex.: $\#\mathbb{K} \simeq 2^{128}$ and $n = 32 \rightsquigarrow$ prob. $\simeq 1/2^{-96}$ *optimal*

# MACs from polynomials: ex. of GMAC




- $\mathrm{GMac}(k, k', m) = \langle r, H_k(m) + E(k', r)\rangle$ with
  - $H_k(m) = M(k)$
  - $r$ a random *nonce*
  - $E$ a block cipher

# MACs from polynomials: implementation issues

## Which field $\mathbb{K}$?

- $\mathbb{K}$ must be large enough for collision prob. to be low — e.g. $\#\mathbb{K} \simeq 2^{128}$
- Two standard choices:
  - Prime field: integers modulo a prime number $\rightsquigarrow$ efficient floating-point arith. — $\mathbb{F}_{2^{130}-5}$ in Poly1305
  - Binary field: "*carry-less* integers" $\rightsquigarrow$ dedicated instr. (`pclmulqdq`) — $\mathbb{F}_{2^{128}}$ in GMAC
  - Combination of different fields — VMAC

## Evaluation

- Given $M = m_0 + \cdots + m_{n-1}X^{n-1}$ and $k$, compute $M(k)$
- Horner scheme:
  i. $r \leftarrow m_{n-1}$
  ii. for $i$ from $n-2$ to 0: $r \leftarrow r \times k + m_i$

$\rightsquigarrow$ $n-1$ additions, $n-1$ mutliplications *by the constant $k$*

1. MACs and their security

2. Designing MACs

3. Authenticated encryption

# What do we want to achieve?

We can encrypt and authenticate messages: can we do both?

## Why is there a question?

- Encrypt-and-authenticate:
  - $m \mapsto (c, t)$ where $c = \mathrm{Enc}_{k_E}(m)$ and $t = \mathrm{Mac}_{k_M}(m)$
  - Danger: $t$ may reveal information on $m$
- Authenticate-then-encrypt:
  - $m \mapsto c$ where $c = \mathrm{Enc}_{k_E}(m \| t)$ and $t = \mathrm{Mac}_{k_M}(t)$
  - Danger: the decryption can fail for two reasons (bad padding or invalid tag)

    $\rightsquigarrow$ *bad padding attack*
- Encrypt-then-authenticate:
  - $m \mapsto (c, t)$ where $c = \mathrm{Enc}_{k_E}(m)$ and $t = \mathrm{Mac}_{k_M}(c)$
  - Danger: seems OK…

Need for a security definition that cover both encryption and authentication

# Authenticated Encryption with Associated Data (AEAD)

## Settings

- A *plaintext* is sent encrypted
- Some *associated data* is sent unencrypted
- Both are authenticated

$\rightarrow$ Example: IP packets (associated data = headers)

## Definition

An AEAD scheme is a pair of mappings

- $E : \mathcal{K} \times \mathcal{M} \times \mathcal{D} \times \mathcal{N} \rightarrow \mathcal{C}$
- $D : \mathcal{K} \times \mathcal{C} \times \mathcal{D} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

where

- $E$ encrypts $m \in \mathcal{M}$ with $k \in \mathcal{K}$ and $\nu \in \mathcal{N}$ (*nonce*), and authenticates it together with $d \in \mathcal{D}$ (associated data)
- $D$ decrypts and verifies: returns $m$ if authentication is successful, $\perp$ otherwise
- $D(k, E(k, m, d, \nu), d, \nu) = m$ for all $k$, $m$, $d$ and $\nu$

# Security notions

## CPA security

Similar to CPA-security for encryption schemes, with two caveats:

- requests to the challenger include associated data and a nonce
- each nonce should be used only once

## Ciphertext integrity – INT-CTXT

Challenger draws $k \twoheadleftarrow \mathcal{K}$

Adversary requests several $c_i = E(k, m_i, d_i, \nu_i)$ (without knowing $k$)

Adversary tries to guess $(c, d, \nu) \notin \{(c_i, d_i, \nu_i)\}$ s.t. $D(k, c, d, \nu) \neq \perp$

$\rightarrow$ INT-CTXT advantage = probability of success of the adversary

## AEAD security

An AEAD scheme is secure if it is both IND-CPA and INT-CTXT secure

# Building AEAD schemes (*theory*)

## Encrypt-then-authenticate

- Given (nonce-based) encryption scheme (Enc, Dec) and MAC (Mac, Vrfy)
- We build an AEAD scheme $(E, D)$ where

$E((k_E, k_M), m, d, \nu)$:

1. $c \leftarrow \mathrm{Enc}(k_E, m, \nu)$
2. $t \leftarrow \mathrm{Mac}(k_M, (c, d), \nu)$
3. Output $(c, t)$

$D((k_E, k_M), (c, t), d, \nu)$:

1. If $\mathrm{Vrfy}(k_M, (c, d), t, \nu)$:
2. Return $D(k_E, c, d, \nu)$
3. Else: return $\perp$

## Security

The AEAD scheme $(E, D)$ is secure if both the encryption scheme and the MAC are secure

# Building AEAD schemes (*practice*): ex. of GCM

## Galois Counter Mode (GCM)

- Standardized by NIST (2007)
- Based on GMAC and AES (used in CTR mode for encryption and in GMAC)

## Encryption - authentication

Inputs: key $k$, message $m$, associated data $d$, nonce $\nu$ ($E$ is the block cipher)

1. $k_m \leftarrow E(k, 0^{128})$ // *Key for GMAC*
2. $x \leftarrow (\nu \| 0^{31} 1) + 1$ // Initial counter value for CTR
3. $c \leftarrow$ encryption of $m$ using $E$ in CTR mode with initial counter value $x$
4. $(c', d') \leftarrow$ pad $c$ and $d$ with zeroes, to length multiple of 128
5. $h \leftarrow H_{k_m}(d' \| c' \| \text{length}(d) \| \text{length}(c))$ // $H_k(m) = M(k)$
6. $t \leftarrow h \oplus E(k, x)$
7. Output $(c, t)$

# About GCM

## Properties

- Very fast and parallelizable
- Security:
  - Proven secure if $E$ is a good PRP
  - Proven secure when $E$ is AES

$\rightarrow$ Only one assumption for both IND-CPA and INT-CTXT security

## Use

- SSH
- TLS 1.2 & 1.3
- OpenVPN 2.4+
- …

# Conclusion

### Authentication is essential!

- Authentication without encryption may be useful
- Encryption without authentication is (almost) never useful

### But encryption is most of the time needed too!

- Combination of both can lead to nasty surprises…
- Modern view: do both at the same time $\rightarrow$ AEAD

### Good authenticated encryption is hard

- Theoretical definitions are complicated, though intuitive
- Still an active area of research https://competitions.cr.yp.to/caesar.html

### A non-exhaustive list of MACs

AMAC, BMAC, CMAC, DMAC, EMAC, FMAC, GMAC, HMAC, IMAC, JMAC, KMAC, LMAC, MMAC, NMAC, OMAC, PMAC, QMAC, RMAC, SMAC, TMAC, UMAC, VMAC, WMAC, XMAC, YMAC, ZMAC, PelicanMAC, SandwichMAC