Lecture 5.
Message authentication codes and authenticated encryption

Active adversaries, small shared secret

Bruno Grenet

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

Introduction to cryptology
Université Grenoble Alpes – IM²AG
M1 INFO, MOSIG & AM

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!

Contents

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 — e.g. $\kappa = 128$
- $\mathcal{M} = \{0,1\}^{<N}$: message space — e.g. $N = 2^{64}$
- $\mathcal{T} = \{0,1\}^{n}$: *tag* space — e.g. $n = 256$

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* — e.g. $s = 64$
- $\mathsf{Vrfy} : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{T}$

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

Two notions

Let $\mathsf{Mac}_k(\cdot)$ be a MAC with unknown key.

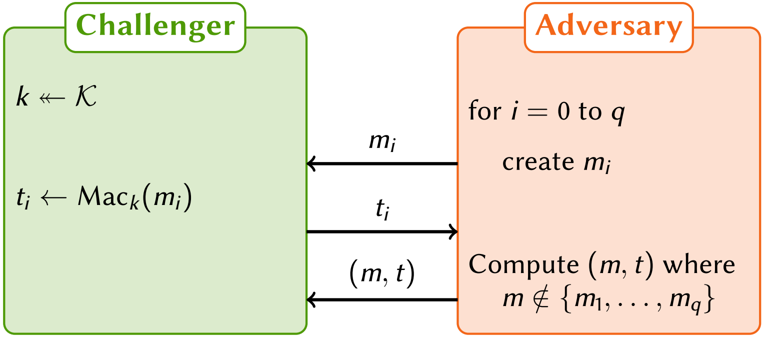
- Universal forgery: given m, *hard* to find t s.t. $\mathsf{Vrfy}_k(m, t) = 1$
- Existential forgery: *hard* to build a pair (m, t) s.t. $\mathsf{Vrfy}_k(m, t) = 1$

Remark: an additional notion

VIL-PRF security: *variable input-length pseudorandom function*

- *hard* to distinguish $\mathsf{Mac}_k(\cdot)$ from a random function $f : \mathcal{M} \to \mathcal{T}$
- VIL-PRF sec. $\Rightarrow$ Existential forgery sec. $\Rightarrow$ Universal forgery sec.

EUF-CMA: Existential UnForgeability under Chosen Message Attack



Challenger
$k \twoheadleftarrow \mathcal{K}$
$t_i \leftarrow \mathrm{Mac}_k(m_i)$
Adversary
for $i = 0$ to q
create m_i
m_i
t_i
(m, t)
Compute (m, t) where $m \notin \{m_1, \ldots, m_q\}$

Advantages

- Advantage of $\mathcal{A}$: $\mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{Mac}}(\mathcal{A}) = \Pr\left[\mathrm{Vrfy}_k(m, t) = 1\right]$
- Advantage function: $\mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{Mac}}(q, t) = \max_{\mathcal{A}_{q,t}} \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{Mac}}(\mathcal{A}_{q,t})$

  where $\mathcal{A}_{q,t}$ denotes an algorithm making $\leq q$ queries with running time $\leq t$

The replay attack

The attack

- An adversary observes a valid tag t for a message m
- The adversary can *replay* (m, t): m is still authenticated by t!

Workaround

- MACs are *not* designed to protect against this kind of attack
  - Still satisfies EUF-CMA security (or stronger notions)
- Solutions depend on the application. Examples:
  - Add a timestamp to the message: $t \leftarrow \mathrm{Mac}_k(m \| T)$ where T is the current time
  - Add a message counter: $t \leftarrow \mathrm{Mac}_k(m \| cpt)$
  - Use of a nonce-based MAC

Timing attack for universal forgery

Assumptions

- $\mathsf{Vrfy}_k(m,t)$ computes $t' \leftarrow \mathsf{Mac}_k(m)$ and checks whether $t = t'$
- The test $t = t'$ is performed byte per byte
- The test $t = t'$ stops as soon as $t_{[i]} \neq t'_{[i]}$ `strncmp`

Algorithm

Goal: Given a message m and oracle access to $\mathsf{Vrfy}_k(\cdot,\cdot)$, output a valid tag t for m

```
1. For $i = 1$ to $n/8$:
2.     For $j = 0$ to $2^8 - 1$:
3.         Call $\mathsf{Vrfy}_k(m, t')$ with $t' = t_1 \| \cdots \| t_{i-1} \| \langle j \rangle_2 \| 0 \| \cdots \| 0$
4.     $t_i \leftarrow \langle j \rangle_2$ where $j$ maximized the running time
5. Return $t = t_1 \| \cdots \| t_{n/8}$
```

Remarks

- Used against updates verification of Xbox 360
- Workaround: time-independent string comparison "*constant time*"

Contents

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

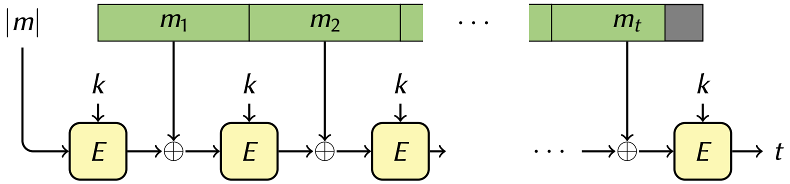
- Don't do $t_1 \leftarrow \mathsf{Mac}_k(m_1), \ldots, t_\ell \leftarrow \mathsf{Mac}_k(m_\ell)$! *cf.* ECB
- Pad the blocks with extra information
  - Block number — no reordering
  - Total message length ℓ — no shortening
  - Random identifier r — no recombination

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

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:
  - $\mathrm{Mac}'_k(m) = \mathrm{Mac}_k(H(m))$
  - $\mathrm{Vrfy}'_k(m, t) = \mathrm{Vrfy}_k(H(m), t)$
- Security: OK if Mac is secure and H is collision resistant

Direct constructions

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

Length-extension attack on PrefixMac

$$\mathrm{PrefixMac}_k(m) = H(k \| m)$$

Assumptions and remark

- H is a Merkle-Damgård hash function
  - $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is the compression function
  - $H(m) = F(IV, \mathrm{pad}(m))$ where $F(IV, x) = f(\cdots f(IV, x_1), \ldots, x_B)$
- $\mathrm{pad}(m) = m \| \mathrm{pad}_{|m|}$ where the extra bits only depend on the length of m
- k has length n

The attack

0. Choose any message $m \in \{0,1\}^n$
1. Request a tag $t \leftarrow \mathrm{PrefixMac}_k(m)$ $\quad = H(k\|m) = F(IV, k\|m\|\mathrm{pad}_{2n})$
2. Compute $t' \leftarrow f(t, \mathrm{pad}_{3n})$ $\quad = F(IV, k\|m\|\mathrm{pad}_{2n}\|\mathrm{pad}_{3n}) = H(k\|m\|\mathrm{pad}_{2n})$
3. Output $(m\|\mathrm{pad}_{2n}, t')$

This attack is a

Length-extension attack on PrefixMac

Ass

▶

▶

▶

The

0.

1. $\mathrm{d}_{2n})$

2. $\mathrm{d}_{2n})$

3.

This

Length-extension attack on PrefixMac

$$\mathrm{PrefixMac}_k(m) = H(k\|m)$$

Assumptions and remark

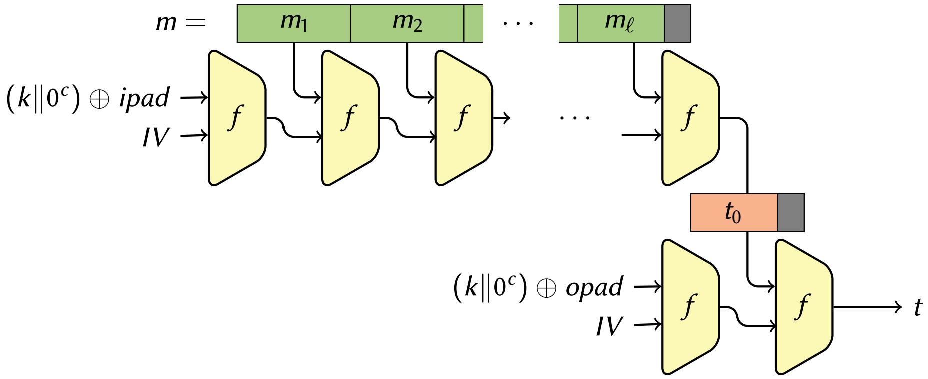
- H is a Merkle-Damgård hash function
  - $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is the compression function
  - $H(m) = F(IV, \mathrm{pad}(m))$ where $F(IV, x) = f(\cdots f(IV, x_1), \ldots, x_B)$
- $\mathrm{pad}(m) = m\| \mathrm{pad}_{|m|}$ where the extra bits only depend on the length of m
- k has length n

The attack

0. Choose any message $m \in \{0,1\}^n$
1. Request a tag $t \leftarrow \mathrm{PrefixMac}_k(m)$ $\quad = H(k\|m) = F(IV, k\|m\| \mathrm{pad}_{2n})$
2. Compute $t' \leftarrow f(t, \mathrm{pad}_{3n})$ $\quad = F(IV, k\|m\| \mathrm{pad}_{2n} \| \mathrm{pad}_{3n}) = H(k\|m\| \mathrm{pad}_{2n})$
3. Output $(m\| \mathrm{pad}_{2n}, t')$

This attack is a

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



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

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 difference-unpredictable hash functions

Difference-unpredictable hash functions: *probabilistic* guarantees, no *adversarial* ones

Definition

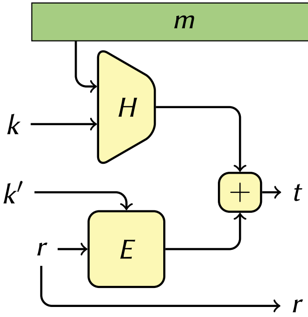
- A keyed hash function is a mapping $H : \mathcal{K} \times \mathcal{M} \to \mathcal{H}$
  - $\mathcal{K} = \{0,1\}^{\kappa}$: key space
  - $\mathcal{M} = \{0,1\}^{<N}$: message space
  - $\mathcal{H} = \{0,1\}^{n}$: digests
- A keyed hash function H is ε-DU if for each pair of message $m_0 \neq m_1$ and digest t,

$$\Pr_{k \twoheadleftarrow \mathcal{K}}\left[H_k(m_0) - H_k(m_1) = t\right] \leq \varepsilon$$

Remarks

- Requires that subtraction makes sense in $\mathcal{H}$ *abelian group*
- No security: usually easy to find 1st or 2nd preimages, and collisions
- Closer to hash functions used in algorithms and data structures

MACs from ε-DU hash functions: Carter-Wegman construction



$$\text{CW-Mac}_{k,k'}(m) = \langle r, H_k(m) + E(k', r) \rangle$$

- $H : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ is an ε-DU hash function
- $E : \mathcal{K}' \times \mathcal{R} \to \mathcal{T}$ is a block cipher
- $r \in \mathcal{R}$ is a random *nonce*

$$\text{CW-Mac} : (\mathcal{K} \times \mathcal{K}') \times \mathcal{M} \to \mathcal{R} \times \mathcal{T}$$

Security (informal)

If E is a PRF and H is ε-DU, then CW-Mac has strong EUF-CMA security

Intermission: Polynomials

Basic definitions

- *Ring* $\mathbb{K}[x]$ of polynomials over $\mathbb{K}$: $f = f_0 + f_1 x + \cdots + f_d x^d$ with $f_i \in \mathbb{K}$
  - d: degree of f (assuming $f_d \neq 0$)
  - Identify polynomials $\leftrightarrow$ vectors
- *Finite field* $\mathbb{K}$: finite set with $(+, -, \times, \div)$ operations
  - Prime fields: $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ = integers modulo a prime p
  - Extension fields: $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$ = polynomials modulo an irreducible polynomial
  - Binary fields: $\mathbb{F}_{2^n}$ = "*carry-less* integers"

Evaluation: polynomials as functions

- $f(\cdot) : k \mapsto f_0 + f_1 k + \cdots + f_d k^d$
- Horner scheme: evaluation in d additions and d multiplications by k
  i. $r \leftarrow f_d$
  ii. for i from $d-1$ to 0: $r \leftarrow r \times k + f_i$
- *Degree mantra*: a nonzero degree-d cannot vanish at more than d points
  - $\Pr_{k \twoheadleftarrow \mathbb{K}}[f(k) = 0] \leq \frac{d}{\#\mathbb{K}}$

Polynomial hash functions

Definition

For a field $\mathbb{K}$, the polynomial hash functions $H : \mathbb{K} \times \mathbb{K}^n \to \mathbb{K}$ is defined by $H_k(m) = k \times m(k)$, where $m = m_0 \| \cdots \| m_{n-1} \in \mathbb{K}^n$ and $m(k) = m_0 + \cdots + m_{n-1}k^{n-1}$

Properties

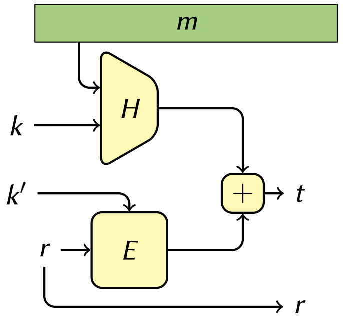
- H_k is linear: $H_k(a+b) = H_k(a) + H_k(b)$
- H is $\frac{n}{\#\mathbb{K}}$-difference-unpredictable: for any $a \neq b$ and $t \in \mathbb{K}$,

$$\Pr_k\left[H_k(a) - H_k(b) = t\right] = \Pr_k\left[k(a(k) - b(k)) - t = 0\right] = n/\#\mathbb{K}$$

Remarks

- Multiplication by k is needed for m_0 to “mix” with the key
- $\mathbb{K}$ must be large enough for $n/\#\mathbb{K}$ to be low
  - Ex.: $\#\mathbb{K} \simeq 2^{128}$ and $n = 32 \rightsquigarrow \Pr\left[H_k(a) - H_k(b) = t\right] \simeq 1/2^{123}$
  - Possible choices:
    - Prime field $\rightsquigarrow$ efficient floating-point arith. $\mathbb{F}_{2^{130}-5}$ in Poly1305
    - Binary field $\rightsquigarrow$ dedicated instr. (`pclmulqdq`) $\mathbb{F}_{2^{128}}$ in GMAC
    - Combination of different fields VMAC

MACs from polynomials: ex. of GMAC



$$\mathrm{GMac}_k(k', m) = \langle r, H_k(m) + E(k', r)\rangle$$

- $H_k(m) = m(k)$ with $m \in \mathbb{F}_{2^{128}}[x]$
- r is a random *nonce*
- E is a block cipher
- $+$ is addition in $\mathbb{F}_{2^{128}}$ (bit-wise XOR)

Contents

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 $c = \mathsf{Enc}_{k_E}(m)$ and $t = \mathsf{Mac}_{k_M}(m)$
  - Danger: t may reveal information on m
- Authenticate-then-encrypt:
  - $m \mapsto c$ where $c = \mathsf{Enc}_{k_E}(m \| t)$ and $t = \mathsf{Mac}_{k_M}(m)$
  - 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 $c = \mathsf{Enc}_{k_E}(m)$ and $t = \mathsf{Mac}_{k_M}(c)$
  - Danger: seems OK…

Need for a security definition that covers both encryption and authentication

Authenticated Encryption with Associated Data (AEAD)

Settings

- A *message* 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

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

where

- Enc 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)
- Dec decrypts and verifies: returns m if authentication is successful, $\perp$ otherwise
- $\mathrm{Dec}_k(\mathrm{Enc}_k(m, d, \nu), d, \nu) = m$ for all k, m, d and ν

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 $c_i = \mathrm{Enc}_k(m_i, d_i, \nu_i)$	oracle access to Enc_k
Adversary	tries to guess $(c, d, \nu) \notin \{(c_i, d_i, \nu_i)\}$ s.t. $\mathrm{Dec}_k(c, d, \nu) \neq \perp$	

$\rightarrow$ 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 $\mathrm{Dec}(k_E, c, d, \nu)$
  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 ν (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

Recap on *symmetric* cryptography

Symmetric cryptography: cryptography using a shared secret

Primitives

- Symmetric encryption: confidentiality against passive adversaries
  - Block cipher: building block   AES
  - Modes of operations: from block ciphers to encryption   CBC, CTR, …
- Message authentication codes: authenticity against active adversaries
  - From block ciphers, hash functions, polynomials, …   CBC-MAC, HMAC, GMAC
  - Combine both: Authenticated Encryption with Associated Data   GCM
- Hash functions: from large messages to small *digests*
  - From compression functions or permutations   SHA-2, SHA-3

Definitions of security

- Encryption: IND-CPA   IND-CCA1/2
- MACs: EUF-CMA   UUF-CMA
- Hash functions: 1$^{\text{st}}$ preimage, 2$^{\text{nd}}$ preimage, or collision resistance