

GCM encryption: encrypt message with AES in counter mode

compute Carter-Wegman MAC on resulting message using GHASH as the underlying hash function and the block cipher as underlying PRF

← Galois Hash

← key derived from PRF evaluation at 0^n

↑ GHASH operates on blocks of 128-bits

operations can be expressed as operations over

$GF(2^{128})$ — Galois field with 2^{128} elements

implemented in hardware — very fast!

Typically, use AES-GCM for authenticated encryption

$GF(2^{128})$ is defined by the polynomial $g(x) = x^{128} + x^7 + x^2 + x + 1$

↳ elements are polynomials over $\mathbb{F}_2$ with degree less than 128 [e.g. $x^{127} + x^{52} + x^2 + x + 1$]

(can be represented by 128-bit string: each bit is coefficient of polynomial)

↳ can add elements (xor) and multiply them (as polynomials) — implemented in hardware

(also used for evaluating the AES round function)

↓ $(m[1], m[2], \dots, m[l])$

↳ $GHASH(k, m) := m[1]k^l + m[2]k^{l-1} + \dots + m[l]k$ [values $m[1], \dots, m[l]$ give coefficients of polynomial, evaluate at point k]

↑ same as one-time MAC from above

Oftentimes, only part of the payload needs to be hidden, but still needs to be authenticated

↳ e.g., sending packets over a network: desire confidentiality for packet body, but only integrity for packet headers (otherwise, cannot route!)

AEAD: authenticated encryption with associated data

↳ augment encryption scheme with additional plaintext input; resulting ciphertext ensures integrity for associated data, but not confidentiality (will not define formally here but follows straightforwardly from AE definitions)

↳ can construct directly via "encrypt-then-MAC": namely, encrypt payload and MAC the ciphertext + associated data

↳ AES-GCM is an AEAD scheme

Constructing block ciphers: typically, relies on an "iterated cipher"

Difficult to design! *Never invent your own crypto - use well-studied, standardized constructions and implementations!*

We will look at two classic designs:

- DES / 3DES (Data Encryption Standard) 1977 (developed at IBM)
- AES (Advanced Encryption Standard) 2002 [most widely used block cipher, implemented in hardware in Intel processors]

on modern Intel processors,
(with AES-NI), ~4 cycles/
round

DES design uses 56-bit keys (and 64-bit blocks)

56-bit keys was a compromise between 40-bit keys (NIST/NSA) and 64-bit keys (cryptographers - notably Hellman)

↳ turned out to be insufficient

- 1997: DES challenge solved in 96 days (massive distributed effort)
- 1998: with dedicated hardware, DES can be broken in just 56 hours → not secure enough!
- 2007: using off-the-shelf FPGAs (20), can break DES in just 2.8 days → anyone can now break DES!

↳ 2-DES: apply DES twice (keys now 112-bits)

↳ meet-in-the-middle attack gives no advantage (though space usage is high)

↳ 3-DES: apply DES three times [3DES($(k_1, k_2, k_3), x$) := DES(k_3 , DES⁻¹(k_2 , DES(k_1 , x)))]

↳ 168-bit keys - standardized in 1998 after brute force attacks on DES shown to be feasible

AES (2002 - most common block cipher in use today):

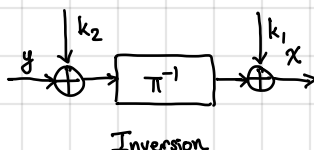
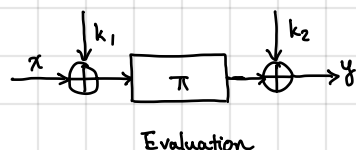
- 3DES is slow (3x slower than DES)
- 64-bit block size not ideal (recall that block size determines adversary's advantage when block cipher used for encryption)

also have 192-bit and 256-bit variants

AES block cipher has 128-bit blocks (and 128-bit keys) (but block size always 2^{128})

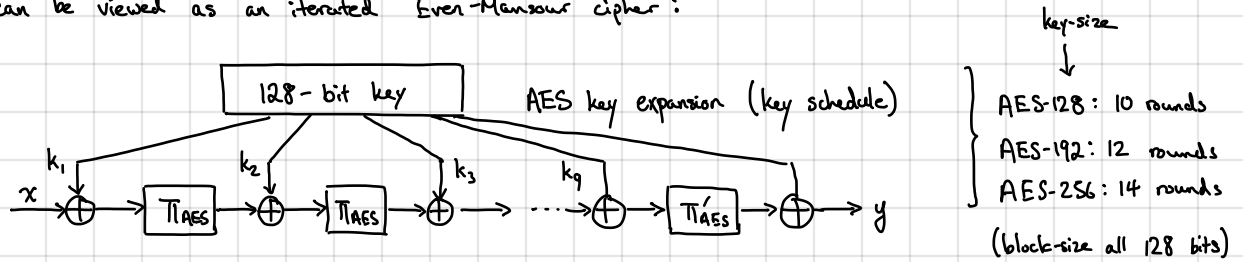
↳ follows another classic design paradigm: iterated Even-Mansour (also called alternating key ciphers)

Even-Mansour block cipher: keys (k_1, k_2) , input x :



Theorem (Even-Mansour): If π is modeled as a random permutation, then the Even-Mansour block cipher is secure (i.e., it is a secure PRP).

The AES block cipher can be viewed as an iterated Even-Mansour cipher:



Permutations π_{AES} and π'_{AES} are fixed permutations and cannot be ideal permutations

↳ Cannot appeal to security of Even-Mansour for security

↳ cannot write down random permutation over $\{0,1\}^{128}$

↳ But still provides evidence that this design strategy is viable [similar to DES and Luby-Rackoff]

AES round permutation: composed of three invertible operations that each operate on a 128-bit block

a_0	a_1	a_2	a_3
a_4	a_5	a_6	a_7
a_8	a_9	a_{10}	a_{11}
a_{12}	a_{13}	a_{14}	a_{15}

128 bits arranged in 4-by-4 grid of bytes ($\{0,1\}^8$)

SubBytes: apply a fixed permutation $S: \{0,1\}^8 \rightarrow \{0,1\}^8$ to each cell

↳ hard coded in the AES standard (similar to S-box)

(chosen very carefully to resist attacks)

ShiftRows: cyclic shift the rows of the matrix

- 1st row unchanged
- 2nd row shifted left by 1
- 3rd row shifted left by 2
- 4th row shifted left by 3

elements are polynomials over $GF(2)$ modulo the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$

MixColumns: the matrix is interpreted as a 4-by-4 matrix over $GF(2^8)$ and multiplied by a fixed invertible matrix (also carefully chosen and hard-coded into the standard)

Observe: Every operation is invertible, so composition is also invertible

π_{AES} : SubBytes; ShiftRows; MixColumns

π'_{AES} : SubBytes; ShiftRows No MixColumns for the last round [done so AES decryption circuit better resembles AES encryption]

Security of AES: Brute-force attack: 2^{128}

Best-known key recovery attack: $2^{126.1}$ time — only 4x better than brute force!

What does 2^{128} -time look like?

- Suppose we can try 2^{40} keys a second.

↳ 2^{88} seconds to break 1 AES key $\sim 10^{19}$ years (710 million times larger than age of the universe!)

- Total computing power on Earth (circa 2015)

↳ estimated to be $\sim 2^{70}$ operations/second (currently, bitcoin mining computes $\sim 2^{66}$ hashes/second)

Let's say we can do 2^{80} operations/second

↳ still require 2^{48} seconds to break AES ~ 9 million years of compute

If we move to 256-bit keys, best brute force attack takes $2^{254.2}$ time (on AES-256)

In well-implemented systems, the cryptography is not the weak point — breaking the crypto requires new algorithmic techniques

↳ But side channels/bad implementations can compromise crypto

↙ e.g., quantum computers