

Last time:

Encryption schemes:

Defined (many-time) security

Saw a construction assuming a PRF:

$$\text{Enc}(K,m) = (r, m \oplus F(k,r))$$

Today:

AES (Advanced Encryption Standard): the PRF we use in practice

Authentication

AES (Advanced Encryption Standard)

AES is a block cipher

A block cipher is a deterministic encryption scheme that encrypts messages of fixed length (called a block).

Historically, block ciphers were designed to be used as encryption schemes.

Today, however, we know that a secure encryption scheme must be randomized (and block ciphers are deterministic).

Hence, today they are most often used as a PRF building block within cryptographic constructions (and are the "bread and butter" of cryptography).

NIST (National Institute of Standards and Technology):

A body that standardizes our constructions, so that people know which constructions to use!

The first block cipher that was standardized by NIST was DES (Data Encryption Standard) in 1977.

It has only 56 bit keys, and hence is no longer secure in practice.

In 2001, DES was replaced with a new standard: AES

NIST announced a block-cipher competition, and the winners were two Belgian cryptographers, Daemen and Rijmen.

AES is a substitution-permutation network,

an idea that dates back to the seminal work of Shannon in 49.

It proceeds in rounds: Each round has two steps:

1. Substitution (confusion)

2. Permutation (diffusion)

The AES encryption algorithm has a 128 bit block.

Namely, $\text{AES}(k): \{0,1\}^{128} \rightarrow \{0,1\}^{128}$.

The key can be either 128 bits, 192 bits, or 256 bits.

Note that since AES is a permutation it can be distinguished from random after seeing $\sim 2^{64}$ hash values, due to the birthday paradox.

It consists of 10 rounds for 128 bit key, 12 rounds for 192 bit key.

and 14 rounds for 256 bit keys

These exact numbers were chosen to optimize efficiency and security.

The structure of AES:

The 128 bit plaintext is thought of 16 bytes

Each byte is thought of as an element in a finite

field with 2 elements, denoted by $GF[2]$

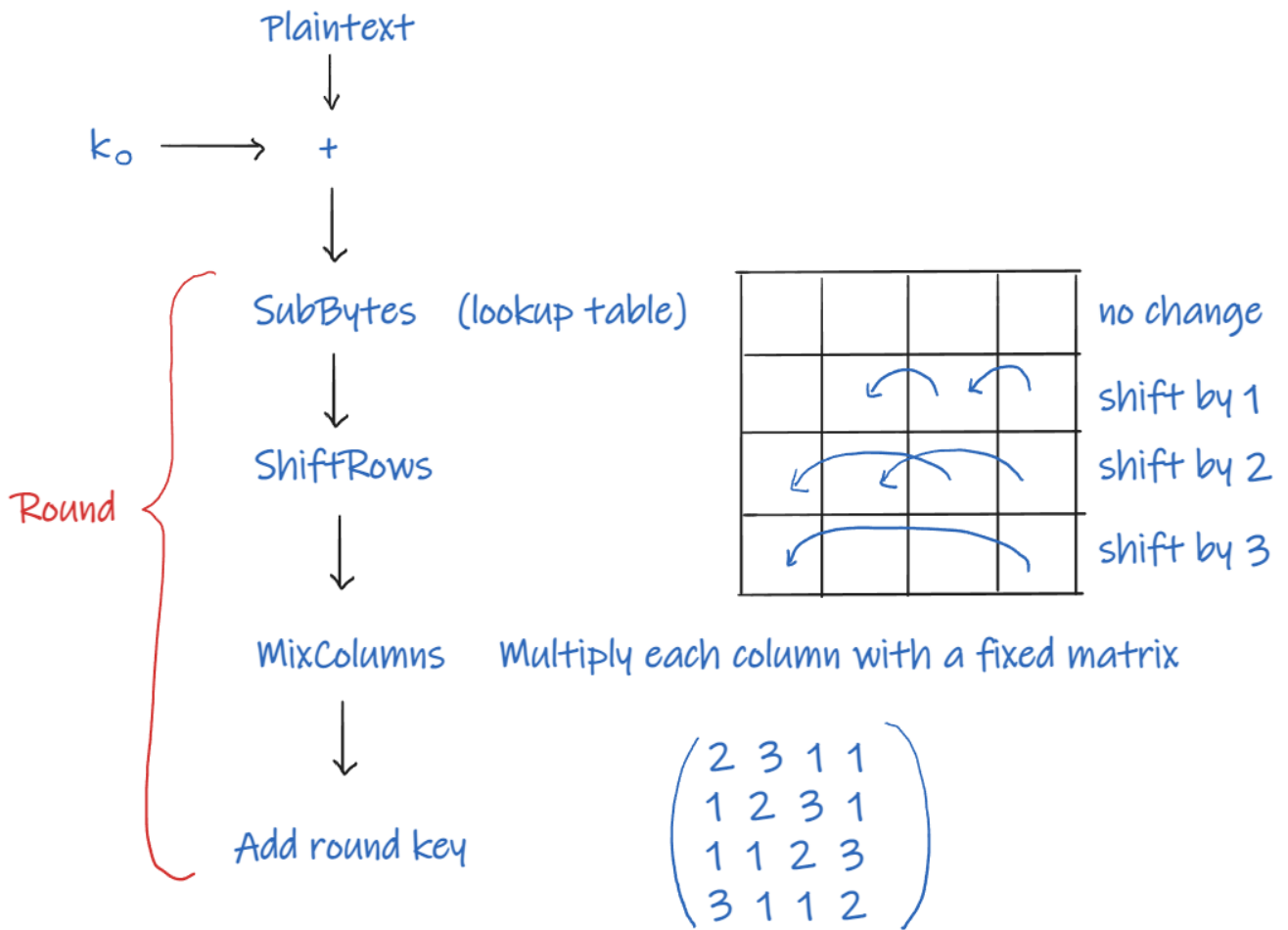
A finite field is called a Galois Field.

b_0			b_{12}
b_1			b_{13}
b_2			b_{14}
b_3			b_{15}

A finite field as operations: addition, mult, inverses.

AES uses operations in this field.

Key expansion: Expands the initial key into round keys



Authentication:



Goal: Bob wants to know that the message indeed came from Alice



How does the server know the instruction came from Alice?

Message Authentication Code:

Assumes the communicating parties share a random secret key K .




Impossible unless Alice knows a secret that the adv doesn't know, and that Bob can somehow verify

It consists of two functions:

A signing function $S(K,M)$ that produces a "tag" for the message M .

A verification function $V(K,M,tag)$ and outputs 0/1.

Often V checks tag by recomputing $S(K,M)$,
in which case we can define a single alg, often
referred to as $MAC(K,M)$



Correctness:

For every K in the key space, and every M in the message space:

$$V(K,M,S(K,M))=1.$$

Security: ??

Attacker Power:

Chosen message attack:

Attacker can obtain valid tags for any messages of his choice: M_1, M_2, \dots

A common real world attack: The attacker sends Alice emails of his choice.

Alice will store these emails on her disc, but will tag them first.

Then the attacker can steal her disc.

Attacker Goal:

Existential forgery:

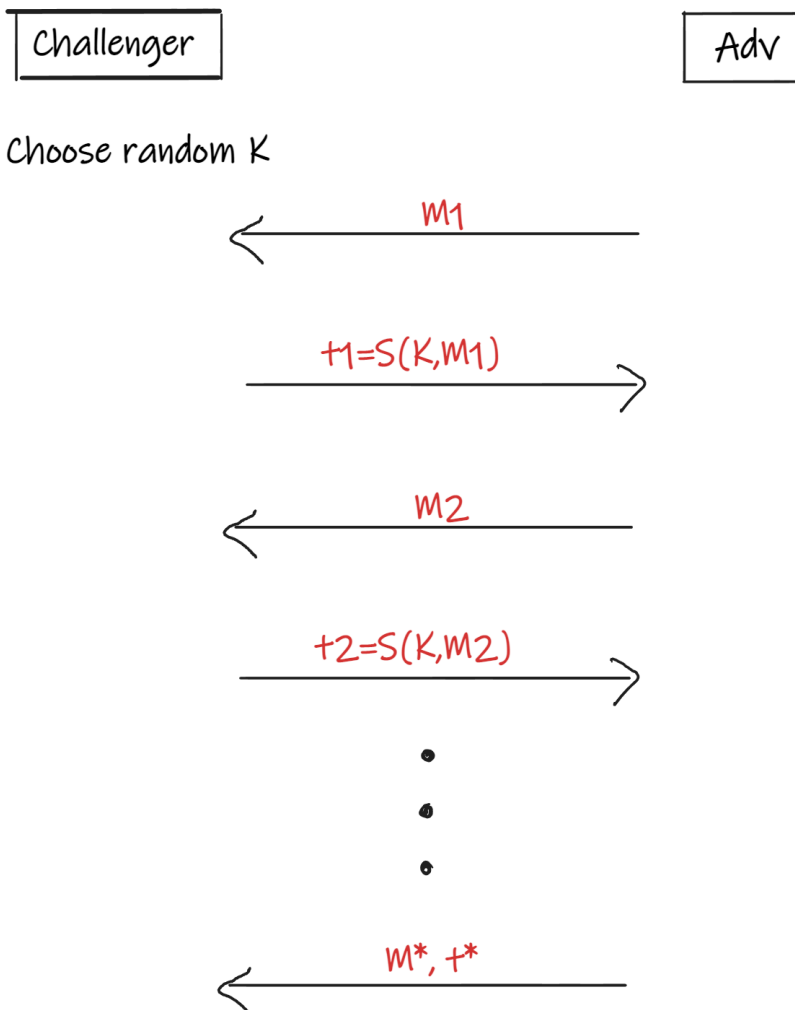
An attacker who is given tags $t_1=S(K,M_1)$, $t_2=S(k,M_2)$,...

for messages M_1, M_2, \dots of his choice,

cannot produce a valid tag for any new message M^* .

Note: Adv wins even if M^* is gibberish. This can still be devastating, since sometimes parties MAC a secret key, which is gibberish.

Security as a game:



Adv wins if M^* is different from M_1, M_2, \dots and $V(K, M^*, t^*)=1$

Strong security: Adv wins if (M^*, t^*) is different from (M_i, t_i) for every i , and $V(K, M^*, t^*) = 1$

(strong)

Def: The MAC is secure (existentially secure against adaptive chosen message attacks)

if any efficient adversary wins in this game with negligible probability

How do we construct a MAC???

May seem impossible!

How can we use a single fixed size secret K ,
to generate more and more unpredictable tags?

Moreover, the MACs used in practice generate random looking tags!

How can we take a fixed size random secret K ,
and deterministically generate more and more randomness???

Pseudo Random Function!

$$S(K, m) = F(K, m)$$

Question: Is every PRF F with domain D a secure MAC for messages from D ,

where the tag of M is $F(K, M)$?

No!

Note: tag cannot be too small. If tag is only 4 bits, the MAC cannot be secure!

If we think of 2^{-128} as negligible, then tag needs to be at least 128 bits long.

Theorem: Every PRF with domain D and range R where $1/|R|$ is "negligible" is a MAC for messages from D .

Corollary: AES is a secure MAC for messages of length 128 bits.

Question: How can we MAC messages of arbitrary length?

Going from small MAC to big MAC:

One approach: Use a collision resistant hash function (CRHF)

H is a CRHF if it is hard to find distinct m, m' s.t. $H(m) = H(m')$

$$S(K, m) = F(K, H(m))$$

where F is a PRF (such as AES) and H is a CRHF

This is secure!

Problem: In practice F is AES which outputs 128 bits, and we can find collisions in time 2^{64}

This is due to the birthday paradox, which allows finding collisions in time $\sqrt{\text{range size}}$.

Two MAC constructions standardized by NIST:

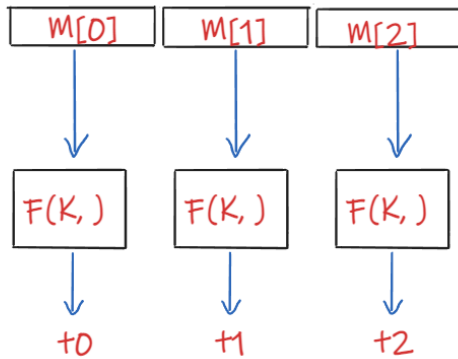
One based on AES (CMAC) and one based on SHA2 (HMAC).

Today! 

Next class we will see a different construction of authenticated encryption AES-GCM (Galois Counter Mode)

AES-based MAC (There is also a hash-based MAC called HMAC)

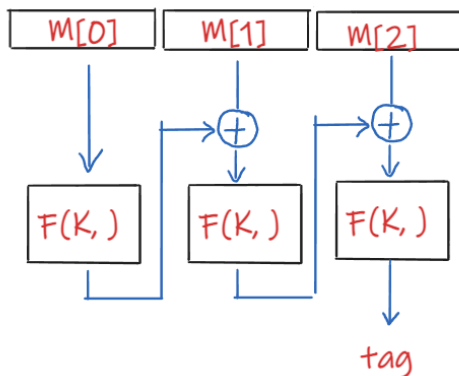
Try 1:



Insecure! Mix and match attack

Adversary can use the tag for message $(M[0], M[1])$ to tag the message $(M[1], M[0])$.

Try 2:

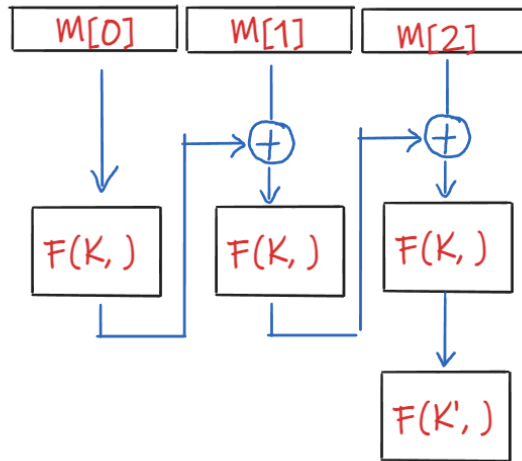


Insecure! Extension attack

Adversary can use the tag for message $(M[0], M[1], M[2])$ and tag' for message $m'[0]$, to tag the message $(M[0], M[1], M[2], \text{tag} \text{ xor } m'[0])$.

Final try:

Cipher Block Chaining
↑
CBC MAC:



The secret key is (K, K')

This additional secret key prohibits these "extension attacks"

Similar to why adding the message length in the construction of a collision resistant hash function is needed to make it secure.

Note: Need to pad the message so that its length will be a multiple of 128.

This is a HW problem (in Pset 1).

The standardized version of CBC MAC is called CMAC