

## Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

## PART XII

## Construction Strategies

- **Stream ciphers:** For arbitrarily long messages (e.g., data streams).  
A key  $k \in \mathcal{K}$  is expanded to a **pseudorandom** stream,  $g(k)$ , used as a one-time pad.

$$\text{Enc}(k, m) = m \oplus g(k)$$

- **Block ciphers:** For messages with a fixed length.  
A **pseudorandom** permutation  $f_k : \mathcal{M}_\ell \rightarrow \mathcal{C}_\ell$ , used as a secret key, is applied to the message.

$$\text{Enc}(k, m) = f_k(m)$$

with a **chaining mode** can encrypt arbitrarily long messages

## Outline

- 1 Symmetric Encryption (II)
- 2 Message Authentication Codes

## Pseudorandom Generators (I)

$$g : \{0, 1\}^\kappa \rightarrow \{0, 1\}^\ell$$

For a random seed  $s \in \{0, 1\}^\kappa$ ,  $r = g(s)$  is **indistinguishable** from random

## Definition (Computational Indistinguishability)

Let  $V = \{V_\ell\}_{\ell \in \mathbb{Z}^+}$ ,  $W = \{W_\ell\}_{\ell \in \mathbb{Z}^+}$  be two families of probability distributions on a finite set family  $\mathcal{X} = \{\mathcal{X}_\ell\}_{\ell \in \mathbb{Z}^+}$ . We say that  $V$  and  $W$  are computationally indistinguishable if for any PPTM  $\mathcal{D}$ ,

$$\left| \Pr[\mathcal{D}(1^\ell, V_\ell) = 1] - \Pr[\mathcal{D}(1^\ell, W_\ell) = 1] \right| \in \text{negl}(\ell)$$

## Pseudorandom Generators (II)

NOTATION:  $V \approx_c W$  means  $V, W$  are computationally indistinguishable.

We abuse the notation writing  $V_\ell \approx_c W_\ell$ .

$U_\ell$  denotes the uniform distribution in  $\{0, 1\}^\ell$ .

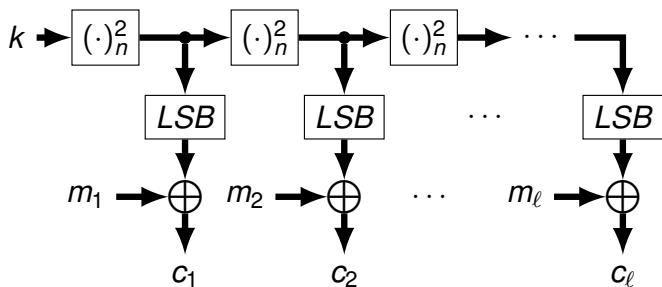
## Definition (Pseudorandom Generator (PRG))

An efficiently computable map family

$g = \{g_\ell : \{0, 1\}^{\kappa(\ell)} \rightarrow \{0, 1\}^\ell\}_{\ell \in \mathbb{Z}^+}$ , where  $\kappa(\ell) \leq \ell$ , is a pseudorandom generator if

$$g(U_{\kappa(\ell)}) \approx_c U_\ell$$

## SE From Blum-Blum-Shub PRG



## Examples

- **Blum-Blum-Shub generator (1986).** Very slow. Provably secure (under QR Assumption)  
 $g(s) = (r_1, r_2, \dots, r_\ell)$  where  $r_i = LSB(s^{2^i} \bmod n)$  for  $n = pq$ ,  $p = 2p' + 1$  and  $q = 2q' + 1$ ,  $p', q'$  random primes of binary length  $\kappa/2 - 1$ .
- **Non-Linear Feedback Shift Registers.** Very fast. Widely used (GSM, Bluetooth, ...). In general, they lead to weak symmetric encryption schemes.

## Pseudorandom Permutations

A permutation randomly selected from a given permutation family  $\mathcal{F}$  behaves like a truly random permutation.

## Definition (Pseudorandom Permutation (PRP))

A family of sets of efficiently computable bijective maps  $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell \in \mathbb{Z}^+}$ ,  $\mathcal{F}_\ell = \{f_k : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell\}_{k \in \mathcal{K}_\ell}$  is a family of pseudorandom permutations if for all oracle PPTM  $\mathcal{D}$ ,

$$\mathcal{D}^{\text{Eval}(1^\ell)} \approx_c \mathcal{D}^{\text{Rand}(1^\ell)}$$

where  $\text{Eval}(x) = f_k(x)$  for a randomly chosen  $f_k \in \mathcal{F}_\ell$  and  $\text{Rand}(x) = h(x)$  for  $h$  randomly chosen among all permutations of  $\{0, 1\}^\ell$ . ( $f_k$  and  $h$  are the same in all oracle queries.)

## Building Block Ciphers from PRP (I)

Let  $\mathcal{F}$  be a pseudorandom permutation family with efficiently computable inverses, and  $\Pi_{\mathcal{F}} = (\text{KeyGen}, \text{Enc}, \text{Dec})$  a symmetric encryption scheme such that

KeyGen( $\ell$ ):  
**output**  $k \leftarrow \mathcal{K}_{\ell}$ ;  
 Enc( $k, m$ ):  
**output**  $f_k(m)$ ;  
 Dec( $k, c$ ):  
**output**  $f_k^{-1}(c)$ ;

▶ proof...

## Theorem

$\Pi_{\mathcal{F}}$  is SE-LR<sub>1</sub> secure but not SE-LR<sub>2</sub> secure (i.e., it is secure if and only if  $q_{LR} \leq 1$ ).

## Proof (SE from PRP)

## Experiment

Exp-PRP-Rand( $\mathcal{F}, \mathcal{D}, \ell$ ):  
**table**  $Q \leftarrow \text{empty\_table}$ ;  
**output**  $\mathcal{D}^{\text{Rand}}(1^{\ell})$ ;

Oracle Rand( $m$ ):  
**find**  $c$  for  $(m, c)$  in  $Q$ ;  
**if found output**  $c$ ;  
**else repeat**  
 $c \leftarrow \{0, 1\}^{\ell}$ ;  
**find**  $c$  for  $(m', c)$  in  $Q$ ;  
**while found**;  
**insert**  $(m, c)$  in  $Q$ ;  
**output**  $c$ ;

$\mathcal{A}$ 's view is independent of  $b_*$

## Reduction:

$q_{LR} \leftarrow 0$ ;  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\text{OLR}}(1^{\ell})$ ;  
**if**  $b' = b_*$  **output** 1;  
**else output** 0;

Oracle  $\mathcal{O}_{LR}(m_0, m_1)$ :

**if**  $q_{LR} > 0$   
**output**  $\perp$ ;  
**else**  
 $q_{LR} \leftarrow 1$ ;  
**output** Rand( $m_{b_*}$ );

## Experiment

Exp-SE-LR<sub>1</sub>( $\Pi_{\mathcal{F}}, \mathcal{A}, \ell$ ):  
 $q_{LR} \leftarrow 0$ ;  
 $k \leftarrow \mathcal{K}_{\ell}$ ;  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\text{OLR}}(1^{\ell})$ ;  
**if**  $b' = b_*$  **output** 1;  
**else output** 0;

Oracle  $\mathcal{O}_{LR}(m_0, m_1)$ :

**if**  $q_{LR} > 0$   
**output**  $\perp$ ;  
**else**  
 $q_{LR} \leftarrow 1$ ;  
**output**  $f_k(m_{b_*})$ ;

◀ go back...

## Proof (SE from PRP)

## Experiment

Exp-PRP-Real( $\mathcal{F}, \mathcal{D}, \ell$ ):  
 $k \leftarrow \mathcal{K}_{\ell}$ ;  
**output**  $\mathcal{D}^{\text{Eval}}(1^{\ell})$ ;

Oracle Eval( $m$ ):

**output**  $f_k(m)$ ;

## Reduction:

$q_{LR} \leftarrow 0$ ;  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\text{OLR}}(1^{\ell})$ ;  
**if**  $b' = b_*$  **output** 1;  
**else output** 0;

Oracle  $\mathcal{O}_{LR}(m_0, m_1)$ :

**if**  $q_{LR} > 0$   
**output**  $\perp$ ;  
**else**  
 $q_{LR} \leftarrow 1$ ;  
**output** Eval( $m_{b_*}$ );

## Experiment

Exp-SE-LR<sub>1</sub>( $\Pi_{\mathcal{F}}, \mathcal{A}, \ell$ ):  
 $q_{LR} \leftarrow 0$ ;  
 $k \leftarrow \mathcal{K}_{\ell}$ ;  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\text{OLR}}(1^{\ell})$ ;  
**if**  $b' = b_*$  **output** 1;  
**else output** 0;

Oracle  $\mathcal{O}_{LR}(m_0, m_1)$ :

**if**  $q_{LR} > 0$   
**output**  $\perp$ ;  
**else**  
 $q_{LR} \leftarrow 1$ ;  
**output**  $f_k(m_{b_*})$ ;

Perfect simulation!

## Building Block Ciphers from PRP (II)

Actually, no **deterministic stateless** symmetric encryption can be SE-LR secure for  $q_{LR} \geq 2$ ... but adding some unique padding to the message...

KeyGen( $\ell$ ):  
**output**  $k \leftarrow \mathcal{K}_{\ell}$ ;

Enc( $k, m$ ):  
 $r \leftarrow \{0, 1\}^{\kappa}$ ; //or instead, use a counter (stateful encryption)  
**output**  $f_k(m||r)$ ;

Dec( $k, c$ ):  
 $(m||r) \leftarrow f_k^{-1}(c)$ ;  
**output**  $m$ ;

▶ proof...

## Theorem

If  $2^{-\kappa} \in \text{negl}(\ell)$  then  $\Pi_{\mathcal{F}}$  is SE-LR secure

## Proof (SE from PRP with Padding)

**Experiment**  
Exp-PRP-Real( $\mathcal{F}, \mathcal{D}, \ell$ ):  
 $k \leftarrow \mathcal{K}_\ell$ ;  
output  $\mathcal{D}^{\text{Eval}}(1^\ell)$ ;

**Oracle** Eval( $x$ ):  
output  $f_k(x)$ ;

**Reduction:**  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{LR}}}(1^\ell)$ ;  
if  $b' = b_*$  output 1;  
else output 0;

**Oracle**  $\mathcal{O}_{\text{LR}}(m_0, m_1)$ :  
 $r \leftarrow \{0, 1\}^\kappa$ ;  
output Eval( $m_{b_*} \| r$ );

**Experiment**  
Exp-SE-LR( $\Pi_{\mathcal{F}}, \mathcal{A}, \ell$ ):  
 $k \leftarrow \mathcal{K}_\ell$ ;  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{LR}}}(1^\ell)$ ;  
if  $b' = b_*$  output 1;  
else output 0;

**Oracle**  $\mathcal{O}_{\text{LR}}(m_0, m_1)$ :  
 $r \leftarrow \{0, 1\}^\kappa$ ;  
output  $f_k(m_{b_*} \| r)$ ;

Perfect simulation!

## Practical Construction of Block Ciphers

Heuristic constructions based on:

- **confusion:** the plaintext/ciphertext pair depend on the key in an involved way
- **diffusion:** the ciphertext depends on the plaintext in an involved way
- **iteration:** the encryption procedure consists of a given number of rounds

GOAL: every input bit and every key bit affects all bits in the output

Structure:

- **key expansion:** a set of round keys is generated from the original key
- **mixing** of permutation (XORing, shifting...) and substitution (S-boxes) in each round.

## Proof (SE from PRP with Padding)

**Experiment**

Exp-PRP-Rand( $\mathcal{F}, \mathcal{D}, \ell$ ):  
table  $Q \leftarrow \text{empty\_table}$ ;  
output  $\mathcal{D}^{\text{Rand}}(1^\ell)$ ;

**Oracle** Rand( $x$ ):  
find  $c$  for  $(x, c)$  in  $Q$ ;  
if found output  $t$ ;  
else repeat  
   $c \leftarrow \{0, 1\}^\ell$ ;  
  find  $c$  for  $(x', c)$  in  $Q$ ;  
while found;  
insert  $(x, c)$  in  $Q$ ;  
output  $c$ ;

**Reduction:**  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{LR}}}(1^\ell)$ ;  
if  $b' = b_*$  output 1;  
else output 0;

**Oracle**  $\mathcal{O}_{\text{LR}}(m_0, m_1)$ :  
 $r \leftarrow \{0, 1\}^\kappa$ ;  
output Rand( $m_{b_*} \| r$ );

**Experiment**

Exp-SE-LR( $\Pi_{\mathcal{F}}, \mathcal{A}, \ell$ ):  
 $k \leftarrow \mathcal{K}_\ell$ ;  
 $b_* \leftarrow \{0, 1\}$ ;  
 $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{LR}}}(1^\ell)$ ;  
if  $b' = b_*$  output 1;  
else output 0;

**Oracle**  $\mathcal{O}_{\text{LR}}(m_0, m_1)$ :  
 $r \leftarrow \{0, 1\}^\kappa$ ;  
output  $f_k(m_{b_*} \| r)$ ;

$\mathcal{A}$ 's view is (nearly) independent of  $b_*$

← go back ...

## Example: AES256

Key size: 256 bits

Message and ciphertext block size: 128 bits ( $4 \times 4$  byte matrix)

Number of encryption rounds: 14

Key expansion: from 256 bits to 1920 bits (15 subkeys, each one a  $4 \times 4$  matrix of bytes) Adds 4 bytes in each iteration step

Encryption round:

- Apply a substitution box to each byte
- Perform rotation operations to rows
- Perform linear transformation to columns
- XOR with the round key

More details in

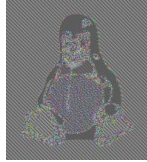
[http://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard)

## Block Chaining Modes

A strategy to encrypt messages larger than one block: Partition  $m \in \{0, 1\}^{n\ell}$  into  $n$  blocks  $m_1, \dots, m_n \in \{0, 1\}^\ell$  and encrypt each block separately.

ECB operation mode:

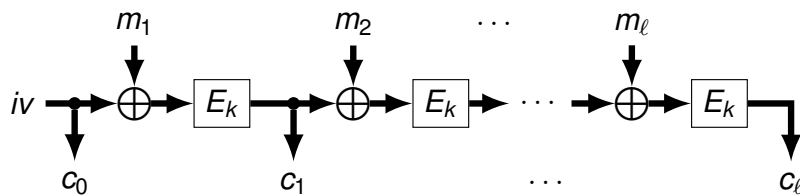
$$c_i = \text{Enc}(k, m_i)$$



(from Wikipedia)

Equal message blocks result in equal ciphertext blocks!

## CBC Mode



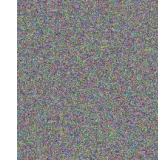
## Block Chaining Modes

A strategy to encrypt messages larger than one block: Partition  $m \in \{0, 1\}^{n\ell}$  into  $n$  blocks  $m_1, \dots, m_n \in \{0, 1\}^\ell$  and encrypt each block separately.

CBC operation mode:

$$c_0 = iv$$

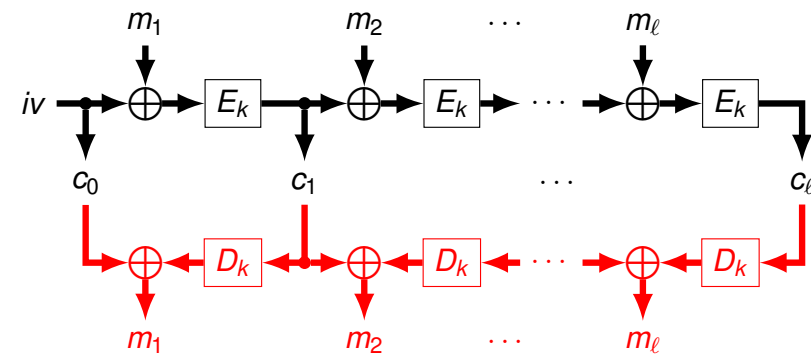
$$c_i = \text{Enc}(k, m_i \oplus c_{i-1})$$



(from Wikipedia)

The ciphertext has an extra random block  $c_0$ , the “initialization vector”

## CBC Mode



## Key Generation and Storage Issues

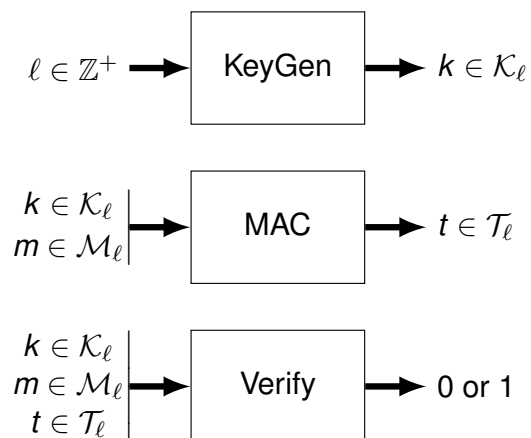
Key generation needs truly random bits ... hard to extract from "nature"

- low entropy sources (e.g., passwords) are not enough
- smoothing and entropy accumulation can be achieved by using heuristic techniques (diffusion, iteration) or dedicated hardware
- there are known attacks to real cryptosystems based on the lack of true randomness

Keys have to be stored securely ... new keys required!

- use hierarchical encryption: encrypt some keys under a common dedicated key
- can incur in **circularity**: plaintext depending on the key! It needs the stronger security notion SE-KDM

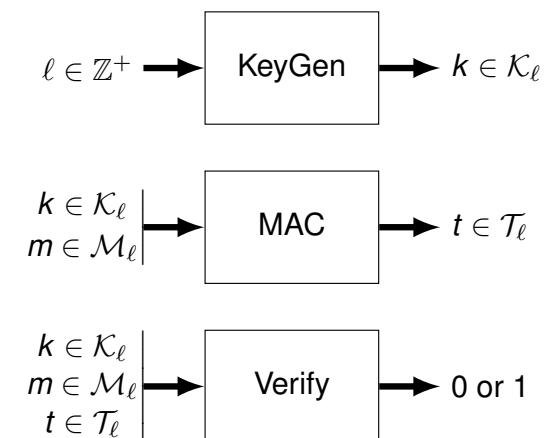
## Message Authentication Codes: Syntax



## Outline

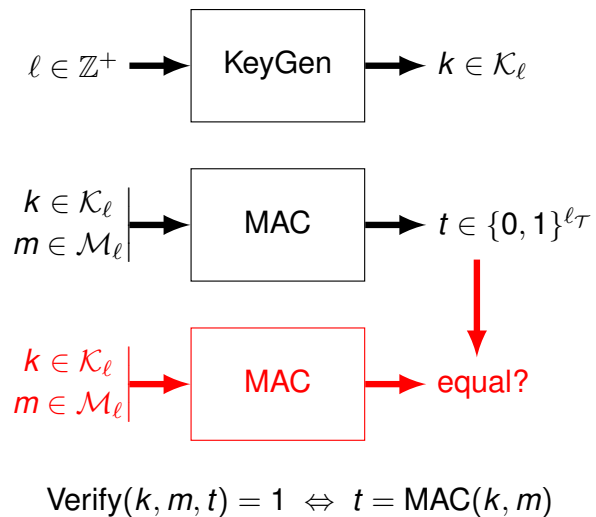
- 1 Symmetric Encryption (II)
- 2 Message Authentication Codes

## Message Authentication Codes: Correctness



$$\forall m \in \mathcal{M}_\ell, \forall k \in \mathcal{K}_\ell, \text{Verify}(k, m, \text{MAC}(k, m)) = 1$$

## Typical Verification Procedure



## Perfect One-Time MAC Construction

## Definition (Pairwise Independent Function Family)

$\mathcal{F} = \{f_k : \mathcal{X} \rightarrow \mathcal{Y}\}_{k \in \mathcal{K}}$  is a **pairwise independent** function family if for all different  $x, x' \in \mathcal{X}$  the random variables  $f_{U_{\mathcal{K}}}(x)$  and  $f_{U_{\mathcal{K}}}(x')$  are independent, where  $U_{\mathcal{K}}$  stands for a uniformly distributed random variable in  $\mathcal{K}$ .

**Example:**  $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_q$ ,  $\mathcal{K} = \mathbb{Z}_q^{\times} \times \mathbb{Z}_q$ , for a  $\ell$ -bits long prime  $q$ , and  $f_{a,b}(x) = ax + b \pmod q$

**Construction:**

KeyGen :

**output**  $k \leftarrow \mathcal{K}$ ;MAC( $k, m$ ) :**output**  $f_k(m)$ ;

## MAC: Perfect Unforgeability

Informal definition:

“Impossible to find a new pair  $(m', t')$  from  $(m, t)$  without  $k$ ”

## Definition (Perfect One-Time Unforgeability)

For a uniformly distributed  $K \in \mathcal{K}_{\ell}$  and for any different  $m, m' \in \mathcal{M}_{\ell}$ , **the random variables**  $\text{MAC}(K, m)$  and  $\text{MAC}(K, m')$  **are independent.**

Definition (Perfect  $n$ -Times Unforgeability)

For a uniformly distributed  $K \in \mathcal{K}_{\ell}$  and for any different  $m_1, m_2, \dots, m_n, m' \in \mathcal{M}_{\ell}$ , **the random variables**  $T_i = \text{MAC}(K, m_i)$ ,  $i = 1, \dots, n$  and  $T' = \text{MAC}(K, m')$  **are independent.**

## MAC: Computational Unforgeability

 $\Pi = (\text{KeyGen}, \text{MAC})$  for spaces  $\mathcal{M}, \mathcal{C}, \mathcal{T}$ .**Experiment** Exp-MAC-EU-CMA( $\Pi, \mathcal{A}, \ell$ ) : $k \leftarrow \text{KeyGen}(\ell)$ ;**table**  $Q \leftarrow \text{empty\_table}$ ; $(m', t') \leftarrow \mathcal{A}^{\mathcal{O}_{\text{MAC}}(1^{\ell})}$ ;**if**  $m' \notin Q$  **and**  $t' = \text{MAC}(k, m')$  **output** 1; //  $\mathcal{A}$  wins**else output** 0;**Oracle**  $\mathcal{O}_{\text{MAC}}(m)$  :**insert**  $m$  **in**  $Q$ ;**output**  $\text{MAC}(k, m)$ ;

## Definition (MAC-EU-CMA)

The MAC  $\Pi$  is MAC-EU-CMA secure if for all Oracle PPTM,  $\mathcal{A}$ ,

$$\Pr[\text{Exp-MAC-EU-CMA}(\Pi, \mathcal{A}, \ell) = 1] \in \text{negl}(\ell)$$

# Computationally Secure Constructions

Known efficient (constant length MAC) constructions based on

- hash functions (compression functions with fixed length output and hard to invert, find collisions, ...)
- block ciphers
- ...

## Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

**END OF PART XII**