Pseudorandom generators

Randomness is important!

- the key stream in the one-time pad
- the secret key used in ciphers
- the initialization vectors (IVs) used in ciphers

Pseudo-randomness

- we need an efficiently computable process that expands a short random string x into a long string f(x) that “appears” random
- not truly random in that
  - derived by a deterministic algorithm
  - output is dependent on initial values
- “anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin” – John von Neumann
- objectives
  - fast
  - secure

Types of PRGs

- Classical PRGs
  - linear congruential generator
- Cryptographically secure PRGs
  - e.g., Blum-Micali generator

Linear congruential generator – algorithm

Based on the linear recurrence: $x_i = a\,x_{i-1} + b \mod m$ $i \geq 1$ where

- x_0 is the **seed** or start value
- a is the multiplier
- b is the increment
- m is the modulus

Output

- $(x_1, x_2, \ldots, x_k)$
- $y_i = x_i \mod 2$
- $Y = (y_1 y_2 \ldots y_k)$ ← pseudo-random sequence of K bits

Linear congruential generator - example

Let $x_n = 3\,x_{n-1} + 5 \bmod 31$, $n \geq 1$, and $x_0 = 2$

- 3 and 31 are relatively prime, one-to-one
- 31 is prime, order is 30

We have the 30 residues in a cycle

2, 11, 7, 26, 21, 6, 23, 12, 10, 4, 17, 25, 18, 28, 27, 24, 15, 19, 0, 5, 20, 3, 14, 16, 22, 9, 1, 8, 29, 30

Pseudo-random sequences of 10 bits

- when $x_0 = 2$

  01101010001

- when $x_0 = 3$

  10001101001

Linear congruential generator – security

- Fast, but insecure
  - sensitive to the choice of parameters a, b, and m
  - correlation between successive values
  - short period, often $m=2^{32}$ or $m=2^{64}$

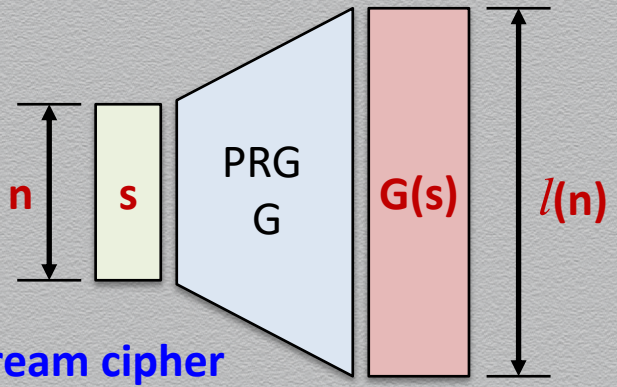
Linear congruential generator – application

- Used commonly in compilers
  - rand()
- Not suitable for high-quality randomness applications
- Not suitable for cryptographic applications
  - need to use cryptographically secure PRGs!

Pseudorandom generators – PRGs

Deterministic PPT algorithm G that

on input a **seed** $s \in \{0,1\}^n$, outputs $G(s) \in \{0,1\}^{l(n)}$



a.k.a. **stream cipher**

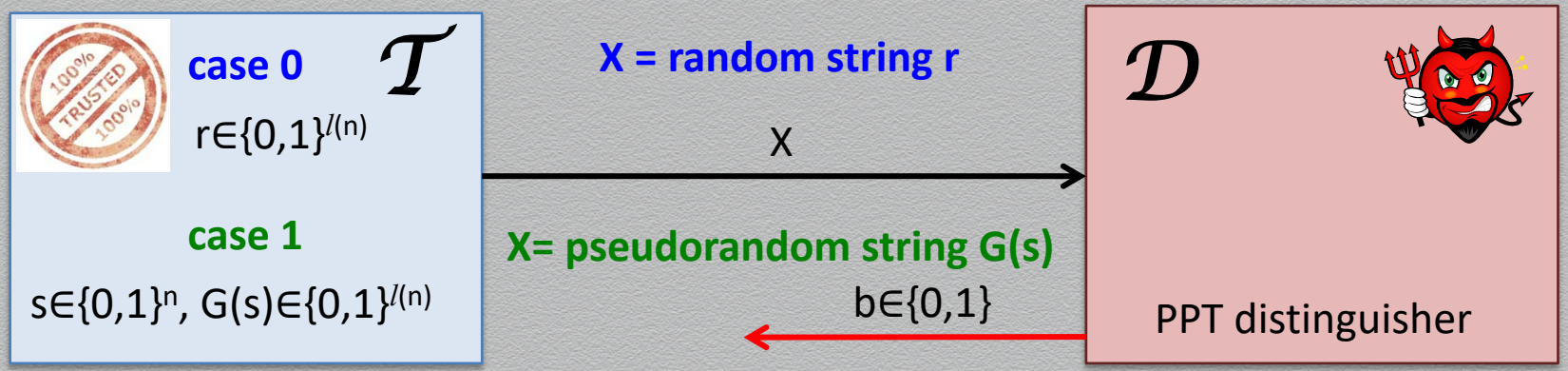
G is a PRG if:

- **expansion**
  - for polynomial *l*, it holds that for any n, *l*(n) > n
  - models the process of **extracting** randomness from a short random string
- **pseudorandomness**
  - **no efficient statistical test can tell apart** G(s) from a **truly random** string r

PRG – security

b = 0 when $\mathcal{D}$ thinks that its input X is random

b = 1 when $\mathcal{D}$ thinks that its input X is pseudorandom

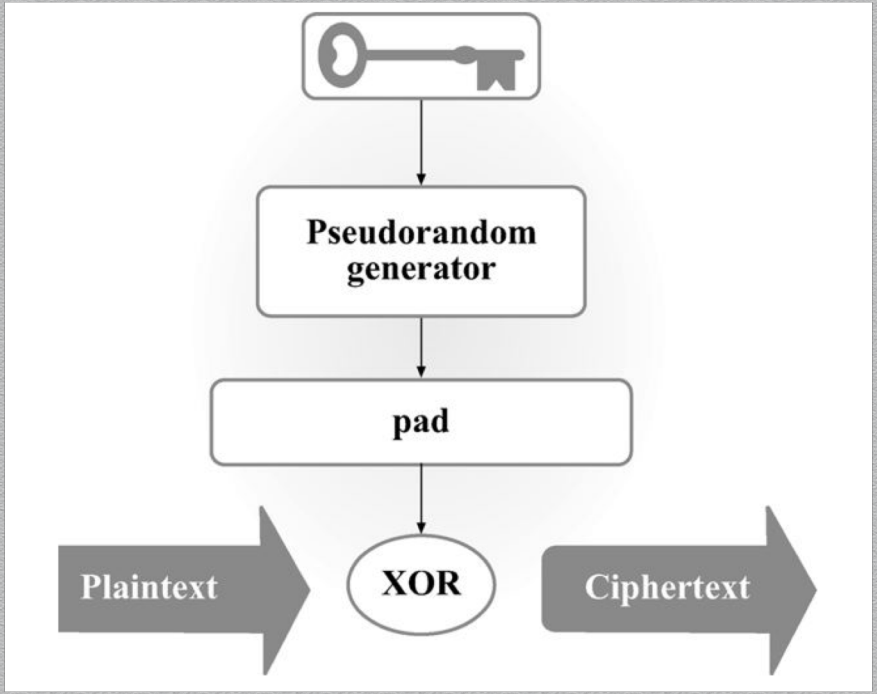


$$| \Pr[\mathcal{D}(G(s)) = 1] - \Pr[\mathcal{D}(r) = 1] | \leq negl(n)$$

$\mathcal{D}$ behaves the same no matter what its input is!

PRG-based symmetric-key encryption scheme

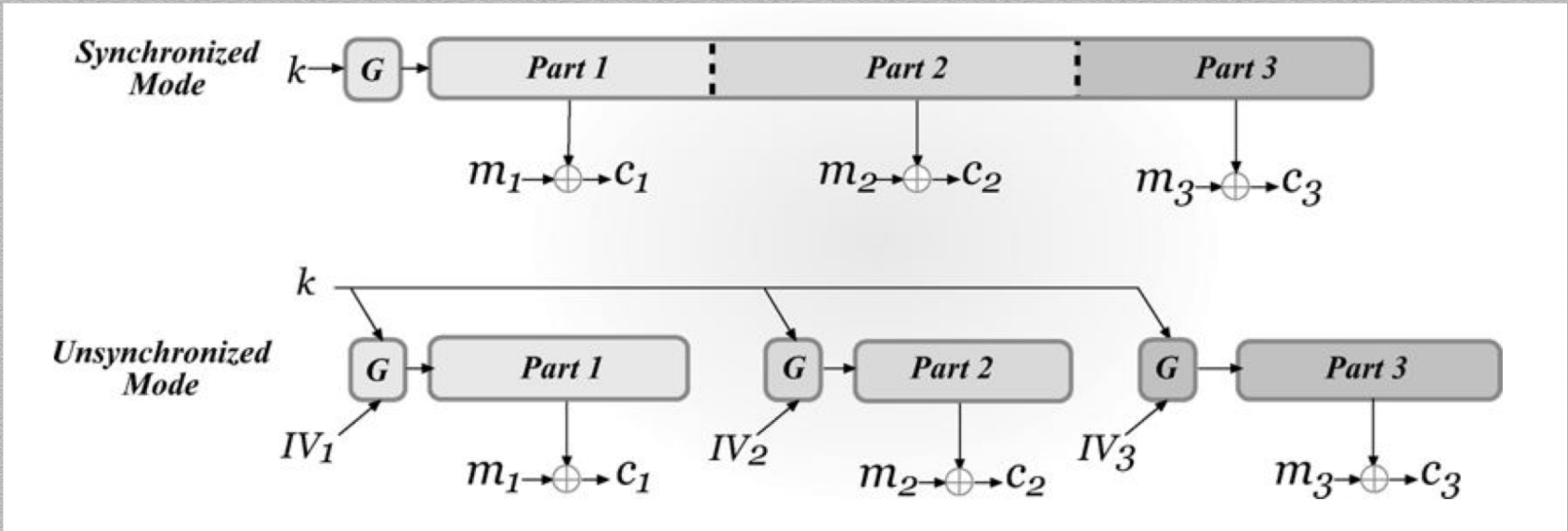
- Either fixed-length or arbitrary-length encryption scheme



encryption scheme is EAV-secure as long as the underlying PRG is secure

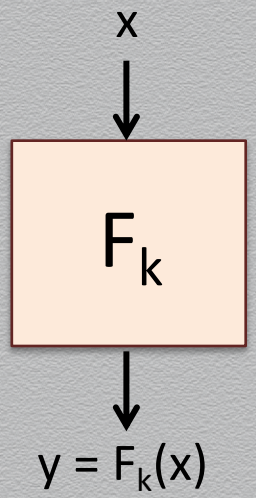
Modes of operation for stream ciphers

on-the-fly computation of new pseudorandom bits, no IV needed, EAV-security



random IV used for every new message is sent along with ciphertext, CPA-security

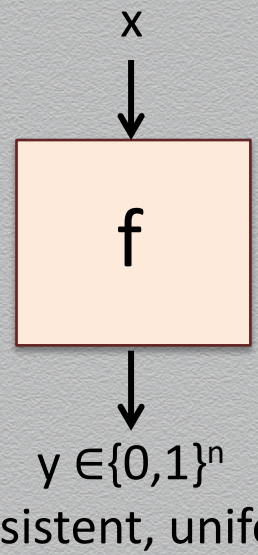
Pseudorandom functions – PRFs



a.k.a. **block cipher**

Generalize the concept of a PRG

- produce pseudorandom bits that also **depend** on specific input
- **keyed functions** of the form $F_k : \{0,1\}^n \rightarrow \{0,1\}^n$



Operate essentially as a **random** function

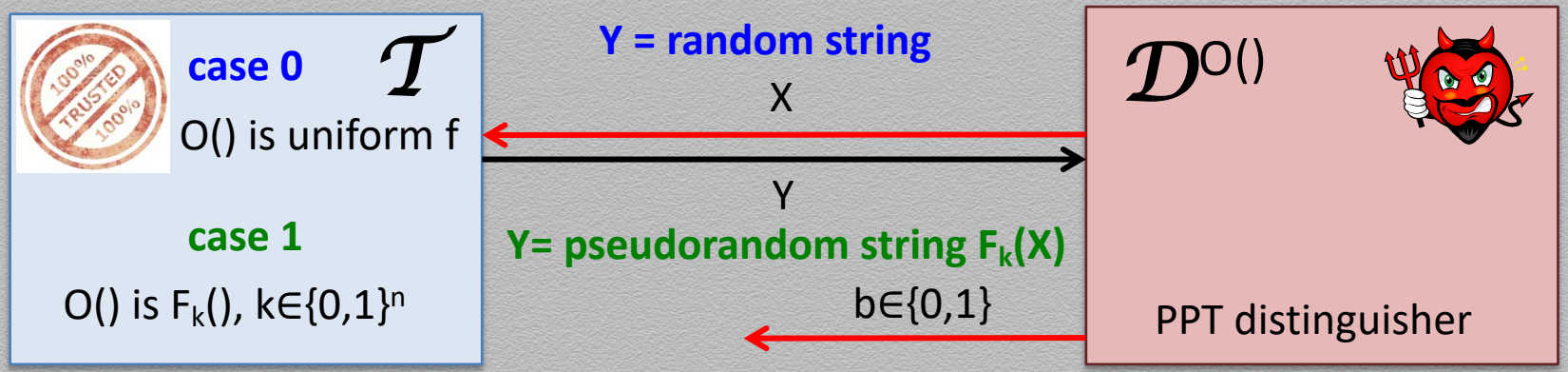
- F_k is PRF if it is **indistinguishable** from a truly random function f
  - e.g., f is an random permutation
- $f: \{0,1\}^n \rightarrow \{0,1\}^n$ is randomly selected for the set of all length-preserving functions mapping n-bit inputs to n-bit outputs

consistent, uniform

PRF – security

b = 0 when $\mathcal{D}$ thinks that its oracle is f()

b = 1 when $\mathcal{D}$ thinks that its oracle is $F_k()$

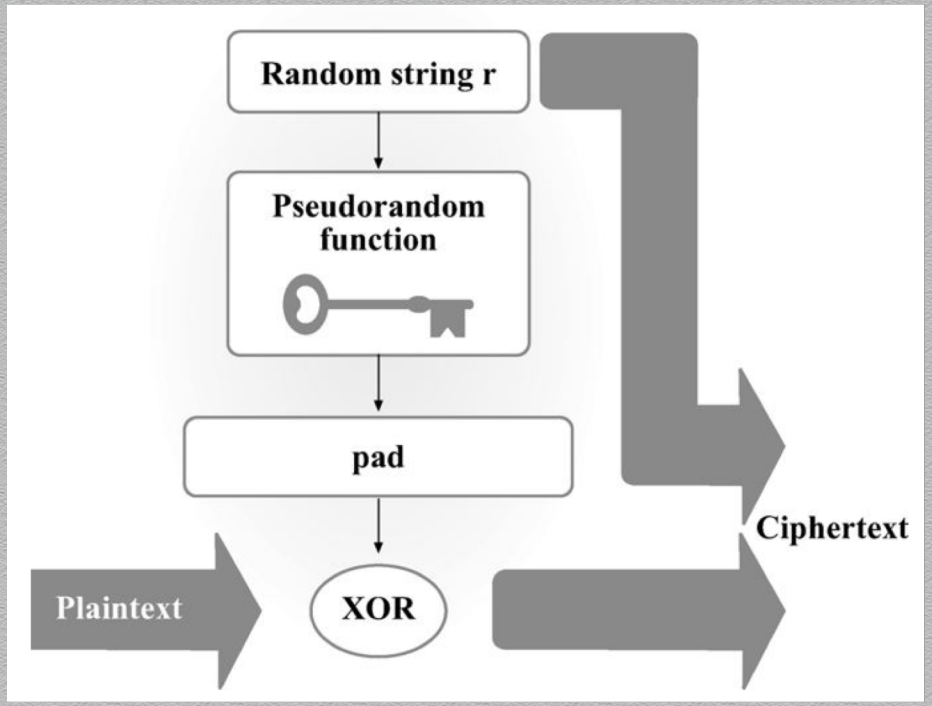


$$| \Pr[\mathcal{D}^{F(k,)}(1^n) = 1] - \Pr[\mathcal{D}^{f()}(1^n) = 1] | \leq \text{negl}(n)$$

$\mathcal{D}$ behaves the same no matter what its oracle is!

PRF-based symmetric-key encryption scheme

- Fixed-length encryption scheme

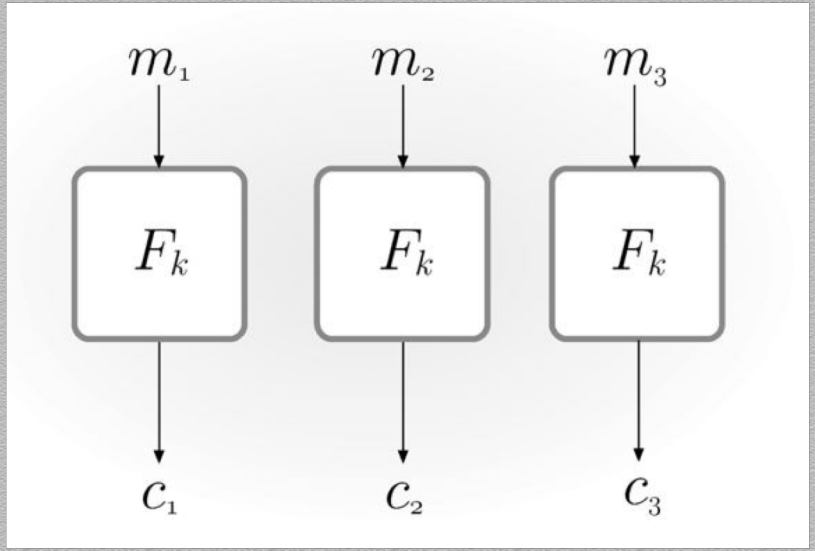


encryption scheme is CPA-secure as long as the underlying PRF is secure

Modes of operations

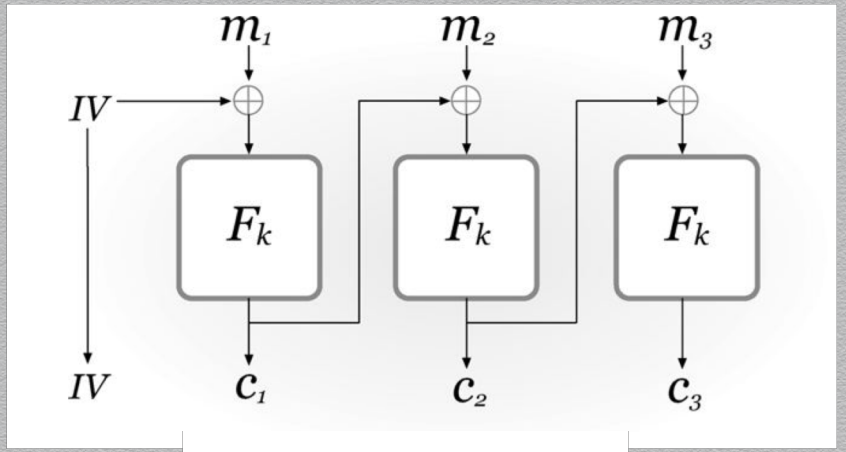
Modes of operation for block ciphers (I)

- ECB - electronic code book
  - insecure, of only historic value
  - deterministic, thus not CPA-secure
  - actually, not even EAV-secure

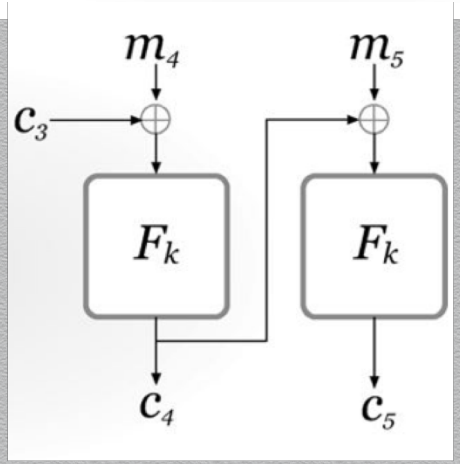


Modes of operation for block ciphers (II)

- CBC – cipher block chaining
  - CPA-secure if F_k a permutation
  - uniform IV
    - otherwise security breaks

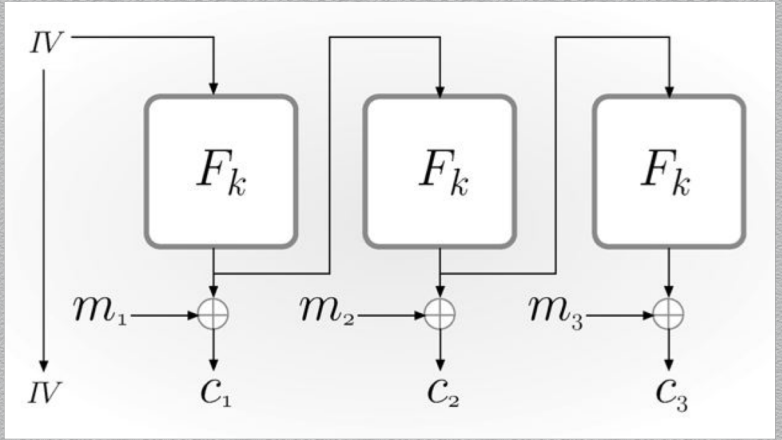


- Chained CBC
  - use last block ciphertext of current message as IV of next message
  - saves bandwidth but not CPA-secure



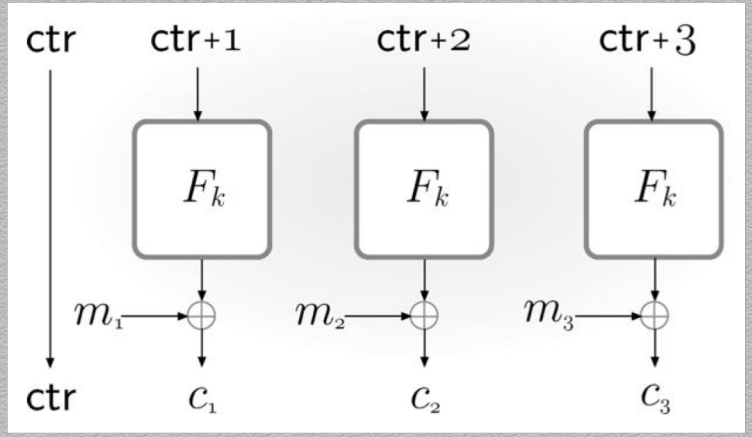
Modes of operation for block ciphers (III)

- OFB – output feedback
  - IV uniform
  - no need message length to be multiple of n
  - resembles synchronized stream-cipher mode
  - stateful variant (chaining) is secure
  - CPA-secure if F_k is PRF



Modes of operation for block ciphers (IV)

- CTR – counter mode
  - ctr uniform
  - no need message length to be multiple of n
  - resembles synchronized stream-cipher mode
  - CPA-secure if F_k is PRF
  - no need for Fk to be invertible
  - parallelizable



Notes on modes of operation

- block length matters
  - if small, IV or ctr can be “recycled”
- IV are often misused
  - e.g., reused or not uniformly random
  - in this case, CBC is a better option than OFB/CTR