CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

January 29 2018

Review

- PRGs can be used to design EAV secure encryption
  - Reduction proof
- In practice, PRGs are implemented with stream ciphers
- Examples of insecure constructions (LFSR, RC4) and “secure” ciphers (e.g., Salsa20)
- Attacks on protocol implementations
  - Two-time pad in MS PPTP
  - Related keys in WEP

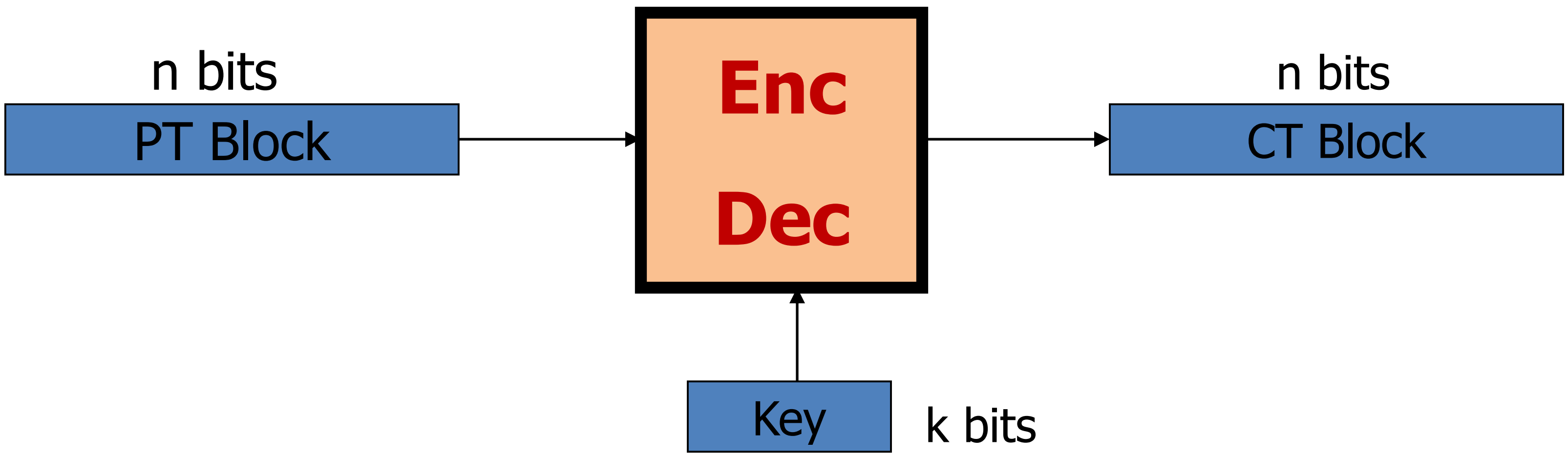
Outline

- Block ciphers vs stream ciphers
- Pseudorandom functions
  - Definitions
  - Examples
- Connections between PRF and PRG
  - Construct PRG from PRF
  - Construct PRF from PRG (GGM construction)
- Pseudorandom permutations
- Stronger notions of security

Stream ciphers vs Block ciphers

- Stream ciphers
  - Encrypt variable-length messages to variable-length ciphertexts
  - Used in practice to instantiate PRG
  - Encrypt messages on demand
  - Faster, but more security vulnerabilities
- Block ciphers
  - Map n-bit plaintext to n-bit ciphertext
  - Output is indistinguishable from random permutation
  - Fixed length
  - More secure in general (e.g., AES)

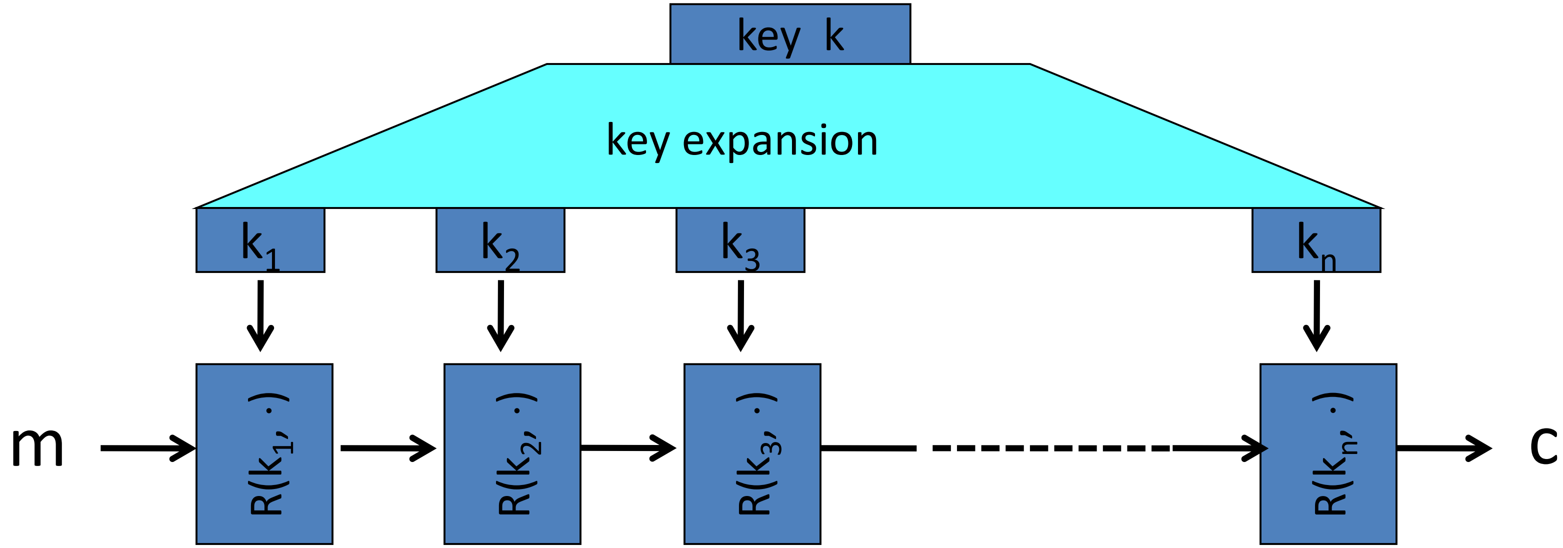
Block ciphers: crypto work horse



Canonical examples:

1. 3DES: n= 64 bits, k = 168 bits
2. AES: n=128 bits, k = 128, 192, 256 bits

Block Ciphers Built by Iteration



R(k,m) is called a round function

for 3DES (n=48), for AES-128 (n=10)

Performance:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	Cipher	Block/key size	Speed (MB/sec)
stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
block	3DES	64/168	13
	AES-128	128/128	109

Encryption in Practice

stream ciphers ≈ pseudorandom generators

block ciphers ≈ pseudorandom functions /permutations

- Practical encryption
  - Good **block ciphers** that withstood the test of time (3DES, AES)
    - Widely used in many practical applications
    - More scrutiny from the community
  - Several recent constructions of **stream ciphers** (eStream)

Tool: Pseudorandom Function

- **PRG:** have short n-bit "seed" s that describes a "random-looking" longer ℓ-bit string r=G(s).

- **PRF:** have short n-bit "seed" k that describes a "random-looking" function

$$F_k : \{0,1\}^u \rightarrow \{0,1\}^v$$

  – Seeing $F_k(x)$ for various inputs x, looks like seeing uniformly random outputs

Pseudorandom Functions

- Syntax: For each security parameter n and each "seed" $k \in \{0,1\}^n$ there is a function

  $F_k : \{0,1\}^u \rightarrow \{0,1\}^v$

- Efficiency: Given k, x compute $F_k(x)$ in poly(n) time.

- How do we define security?

Scenario 1

security parameter
n

distinguisher D

challenger
chooses a random $\mathbf{k} \in \{0,1\}^n$.

$x_1 \in \{0,1\}^u$ →

← $F_k(x_1)$

$x_2 \in \{0,1\}^u$ →

← $F_k(x_2)$

⋮

$x_t \in \{0,1\}^u$ →

← $F_k(x_t)$

Scenario 0

security parameter
n

distinguisher D

$x_1 \in \{0,1\}^u$ →

← $f(x_1)$

$x_2 \in \{0,1\}^u$ →

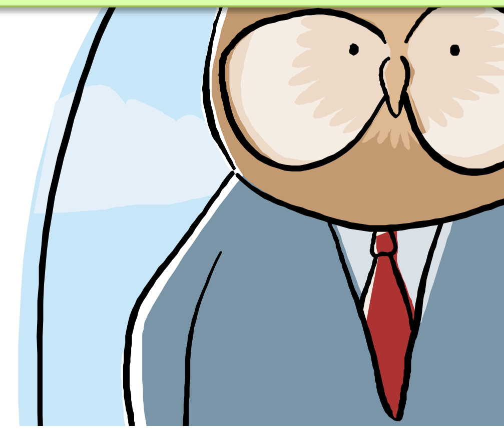
← $f(x_2)$

⋮

$x_t \in \{0,1\}^u$ →

← $f(x_t)$

challenger
chooses a random function
$f : \{0,1\}^u \rightarrow \{0,1\}^v$



outputs $b \in \{0,1\}$

Pseudorandom Functions (definition)

- We say that F is a **pseudorandom function (PRF) family** if for all **PPT distinguisher D** the probability to correctly distinguish **scenario 0** from **scenario 1** is **negligible**.

Formally: For all PPT distinguisher **D**:

| Pr[D outputs "1" in scenario 0] – Pr[D outputs "1" in scenario 1] |

is negligible in **n**

$$|Pr[D^{F_k(\cdot)}(n) = 1] - Pr[D^{f(\cdot)}(n) = 1]| \leq negl(n)$$

Polynomial number of queries to oracle

Example 1

Let $F: K \times X \rightarrow \{0,1\}^{128}$ be a secure PRF.

Is the following G a secure PRF?

$$G(k, x) = \begin{cases} 0^{128} & \text{if } x=0 \\ F(k,x) & \text{otherwise} \end{cases}$$

→ No, it is easy to distinguish G from a random function

Yes, an attack on G would also break F

It depends on F

Example 2

Let F: $\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be defined as

$$F_k(x) = k \oplus x$$

Is F a secure PRF?

Yes, F is a PRF

→ No, F is not a PRF

Build D - distinguisher for F

- D has access to oracle O
- D chooses x_1, x_2 and gets back $y_1 = O(x_1); y_2 = O(x_2)$
- D outputs 1 if $x_1 \oplus x_2 = y_1 \oplus y_2$

Connection between PRF and PRG

Cryptographic PRG

outputs:

a random string **r**

or

G(s) (where **s** random) →

0 if he thinks it's **r**

1 if he thinks it's **G(s)**

Should not be able to distinguish...

Definition

n – a parameter

S – a variable distributed uniformly over $\{0,1\}^n$

r – a variable distributed uniformly over $\{0,1\}^{\ell(n)}$

Definition: **G** is a **cryptographic PRG** if for every PPT algorithm **D** we have:

$$| \; P[\, D(G(s)) = 1 \,] - P[\, D(r) = 1 \,] \; |$$

is negligible in **n**.

An easy application: PRF $\Rightarrow$ PRG

Let $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRF.

Then the following $G: K \rightarrow \{0,1\}^{nt}$ is a secure PRG:

$$G(k) = F(k,1) \parallel F(k,2) \parallel \cdots \parallel F(k,t)$$

Key property: parallelizable

Security from PRF property: $F(k, \cdot)$ indist. from random function $f(\cdot)$

Reduction proof

- Assume, by contradiction, that G is not a secure PRG. There exists a distinguisher D such that:

  | Pr[D(r) = 1] – Pr[D(G(s)) = 1] | $= \epsilon(\mathrm{n})$
- We build A a distinguisher for F
- A is given access to oracle function O ($O = F_k(\cdot)$ in world 0 and $O = f\ (\cdot)$ in world 1)
- A queries O on inputs 1,…,t and computes $y_i = O(i)$
- A runs D on input $y_1 \dots y_t$
- A outputs what D outputs

Reduction proof

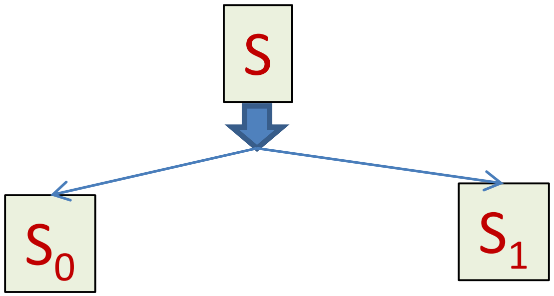
- Assume, by contradiction, that G is not secure PRG. There exists a distinguisher D such that:

  | Pr[D(r) = 1] – Pr[D(G(s)) = 1] | $= \epsilon(\mathrm{n})$
- We build A a distinguisher for F
- In world 1, $O = F_k(\cdot)$ and Pr[$A^{F_k(\cdot)}$= 1] = Pr[D(F(k,1) ll F(k,2) ll ⋯ ll F(k,t)) = 1] = Pr[D(G(k)) =1]
- In world 0, $O = f\ (\cdot)$ random function and Pr[$A^{f\,(\cdot)}$= 1] = Pr[D(r) = 1]

|Pr[$A^{F_k(\cdot)}$= 1] – Pr[$A^{f\ (\cdot)}$= 1] | = | Pr[D(r) = 1] – Pr[D(G(s)) = 1] | $= \epsilon(\mathrm{n})$

Constructing a 1-bit PRF from PRG

- Let $G : \{0,1\}^{n} \rightarrow \{0,1\}^{2n}$ be a PRG.



$(S_0, S_1) = G(S)$

- Define PRF: $F_s(x) = S_x$

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/