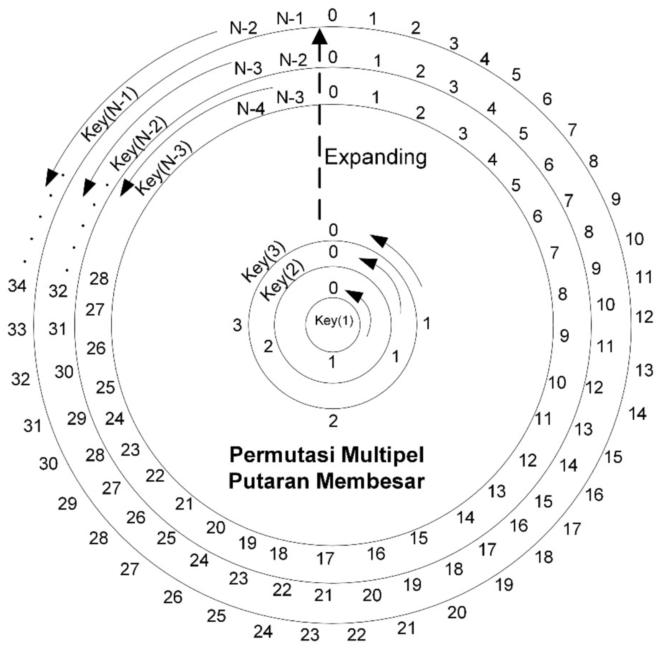


Dr. Yohan Suryanto, S.T, M.T

Semester Ganjul 2020/2021

KRIPTOGRAFI 1
Pendahuluan

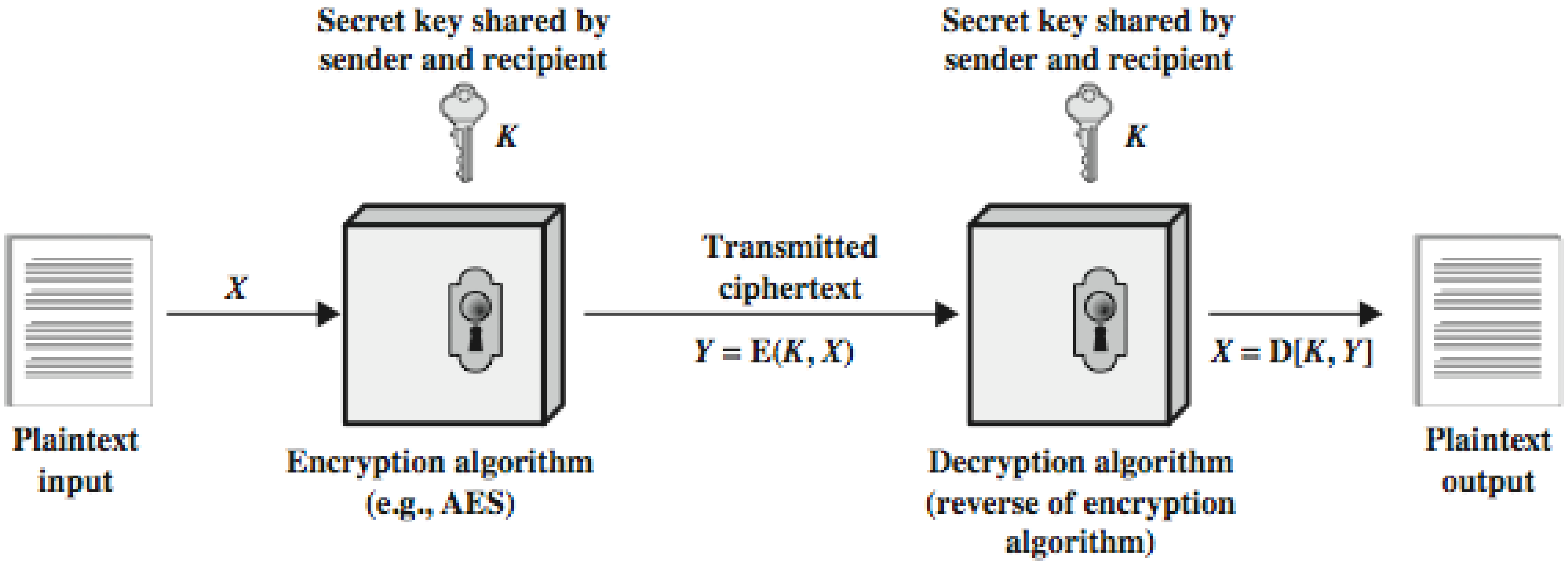
source:

1. Behrouz Foroyzan, The McGraw-Hill Companies
2. Stallings, William. "Cryptography and Network Security“
3. J. Wang and Z. Kissel. Introduction to Network Security: Theory and Practice. Wiley and HEP, 2015

Some Basic Terminology

- **plaintext** - original message
- **ciphertext** - coded message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - recovering ciphertext from plaintext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology** - field of both cryptography and cryptanalysis

Symmetric Cipher Model



Things to know

Any message written over a fixed set of symbols can be represented as a *binary string* (a sequence of 0's and 1's)

Binary digits 0 and 1 are called *bits*

To reduce computation overhead, encryption algorithms should only use operations that are easy to implement

Things to know

- For a binary string X:
  - The length of X, denoted by $|X|$, is the number of bits in X
  - If $|X| = l$, X is an l-bit binary string
  - Let a be a binary bit and k a non-negative integer. Denote by a^k a string consisting of k copies of a

  - Denote the concatenation of X and Y by XY or $X||Y$

What is Encryption?

- There are two approaches to network security
  - Crypto based: cryptographic algorithms and security protocols
  - System based: non-crypto
  - Combination of both forms a standard security structure
- Encryption
  - Make plain text messages unintelligible
  - The unintelligible text can be converted back to its original form

Common encryption methods

- Common encryption methods use secret keys and algorithms
  - Conventional encryption algorithms (a.k.a. symmetric-key encryption algorithms): Same key for encryption and decryption
  - Public-key encryption algorithms (a.k.a. asymmetric-key encryption algorithms): Different keys for encryption and decryption

ASCII CODE

- 7-bit binary strings
  - first and last 32 codes are control codes
  - 32 to 126 encode capital and lower-case English letters, decimal digits, punctuation marks, and arithmetic operation notations
- We often add an extra bit in front, making each character a byte
  - This allows us to either represent 128 extra characters, or have a parity bit for error detection
- The length of any binary string in ASCII is therefore divisible by 8
- The length of codes in other code sets, e.g. the Unicode, is divisible by 16
- Without loss of generality, assume the length of any plaintext string in binary is divisible by 8

Example: Substitution

- A one-to-one mapping of characters; e.g.

  substitute a with d, b with z, c with t, etc
- Unreadable to untrained eyes, this method maintains the statistical structure of the underlying language (e.g. character frequency)
- In English, the letter “e” appears most frequently of all single letters
- The letter with the highest frequency in the unintelligible text is likely the letter “e”
- The method can be applied to other letters and letter sequences to find the original message

CMO Map	G	Z	B	q	6	E	l	J	5	I	n	8	W	x	…	z
MAP A	5	I	W	E	6	q	l	J	G	Z	n	8	b	x	…	z
RCM	A	B	C	D	E	F	…	s	t	u	v	W	x	y	…	9

Character Substitution and Repeated Increment Shifting

XOR Encryption

- The exclusive-OR operation, denoted by $\oplus$ or XOR, is a simple binary operation used in encryption
- XOR encryption: Divide a string into blocks of equal length and encrypt each block with a secrete key of the same size of the block

- Block size of 8 (1 byte), on a two character (2 byte) string *M*
- An 8-bit Encryption key (such as: 1100 1010) on *M* twice:

$$\begin{array}{lcl} M: & & 1111\ 1111\ 0000\ 0000 \\ K: & \oplus & 1100\ 1010\ 1100\ 1010 \\ C: & & 0011\ 0101\ 1100\ 1010 \end{array}$$

We can decrypt *C* using the same key; i.e., we simply XOR *C* with *K* to get *M*:

$$\begin{array}{lcl} C: & & 0011\ 0101\ 1100\ 1010 \\ K: & \oplus & 1100\ 1010\ 1100\ 1010 \\ M: & & 1111\ 1111\ 0000\ 0000 \end{array}$$

- This is simple and easy to implement
- But it is not secure, for knowing any one pair (M_i, C_i) will reveal K:

$$M_i \oplus C_i = M_i \oplus (M_i \oplus K) = K$$

Permutation

- Circular Permutation
- Prime modulus multiplicative linear congruential generator (PMMLCG)
- Shrinking and Expanding Multiple circular Permutation

Criteria of Data Encryptions

- XOR encryption is secure if a key is only used once, but it’s unpractical
- How about keeping encryption algorithms private?
- To study the security of encryption algorithms, we assume that everything except the encryption keys are publicly disclosed, and the keys are reusable
- Good encryption algorithms must satisfy the following criteria:

  -Efficiency

  -Resistance to Statistical Analysis

  -Resistance to Brute-Force Attacks

  -Resistance to Mathematical Analysis Attacks

Efficiency

- Operations used in the algorithms must be easy to implement on hardware and software
- Execution of the algorithms should consume only moderate resources
- Time complexity and space complexity must be kept within a small constant factor of the input size
- Common operations:
  - XOR
  - Permutations: one-to-one mapping
  - Substitution: many-to-one mapping
  - Circular shift: a special form of permutation
  - Operations on finite fields

Resistance to Statistical Analysis

- Analyzing the frequencies of characters in C, one can find out the original characters in M they correspond to
- *Diffusion* and *confusion* are standard methods to flatten statistical structure
  - Diffusion: Each bit in C should depend on multiple bits (as evenly as possible) in M
    - Diffusion can be obtained by executing a fixed sequence of operations for a fixed number of rounds on strings generated from the previous round
  - Confusion: Each bit in C should depend on multiple bits (as evenly as possible) in the secrete key K
    - Confusion can be obtained by generating sub-keys from K and using different sub-keys in different rounds

Resistance to Brute-Force Attacks

- The strength of an encryption algorithm depends on its operations and the key length
- Suppose the encryption key is l-bit long, with 2^l possible keys
- If Eve the eavesdropper attains a ciphertext message C and knows the algorithm used to encrypt it, she can try all keys one at a time until she decrypts the message into something makes sense
- Thus, the time complexity of a brute-force attack is in the order of 2^l
- Under current technologies, it is believed that l = 128 would be sufficient
- The time complexity of a brute-force attack is often used as the benchmark for other cryptanalysis attacks: If an attack with a time complexity substantially less than 2^l is found, the attack is considered useful

Resistance to Other Attacks

- Other common attacks: *chosen-plaintext attacks* and *mathematical attacks*
- Chosen-plaintext Attacks:
  - Obtain a specific M encrypted to C
  - Use this pair (M, C) to find out the key used
  - Example: XOR encryption

    If Eve knows (M, C) she can find K easily:

    $$C = (M \oplus K)$$
    $$M \oplus C = M \oplus (M \oplus K)$$
    $$M \oplus C = K!$$
- Mathematical Attacks:
  - Use mathematical methods to decipher encrypted messages
    - Differential Cryptanalysis, Linear Cryptanalysis, Algebraic Cryptanalysis.
    - Require sophisticated mathematics

Implementation Criteria

- Implementations of encryption algorithms must resist *side channel attacks* (SCA)
- SCA explores loopholes in the implementation environments
  - Timing Attacks: Attacker analyzes the computing time of certain operations
  - Useful if the run-time of certain operations varies when the key has different bit values
- Combating Timing Attacks:
  - Flatten computation time differences by adding redundant operations on instructions that take less time to execute

Data Encryption Standard (DES)

- Published by the US National Bureau of Standards (NBS) in 1977
- A concrete implementation of the Feistel Cipher Scheme (FCS), invented by Horst Feistel
- Symmetrical encryption and decryption structures
- Use four basic operations: XOR, permutations, substitution, and circular shift
- Widely used from mid-70’s to early-2000’s.
- Phased out by AES and other better encryption algorithms

3DES/2, 2DES and 3DES/3

- DES is not a group!
  - No two encryptions are the same as a single one: $E_K(M)$!= $E_{K1}(E_{K2}(M)$
- We can use Multiple DES
  - Take X keys and apply DES Y times to get YDES/X
  - We have, e.g., 2DES/2, 3DES/2, 3DES/3
  - Can effectively extend the length of encryption keys using existing DES
  - Can resist brute-force attacks

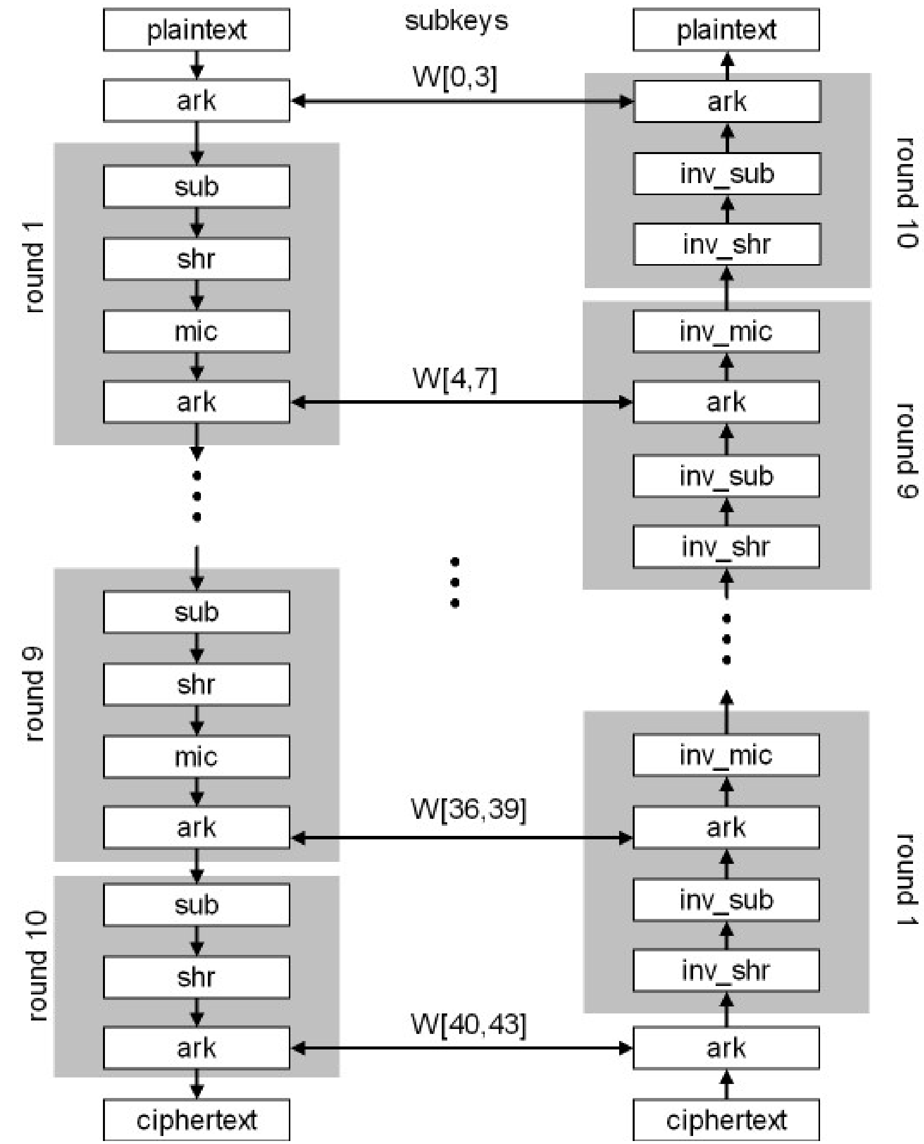
AES

- Advanced Encryption Standard competition began in 1997
- Rijndael was selected to be the new AES in 2001
- AES basic structures:
  - block cipher, but not Feistel cipher
  - encryption and decryption are similar, but not symmetrical
  - basic unit: byte, not bit
  - block size: 16-bytes (128 bits)
  - three different key lengths: 128, 192, 256 bits
    - AES-128, AES-192, AES-256
  - each 16-byte block is represented as a 4 x 4 square matrix, called the *state matrix*
  - the number of rounds depends on key lengths
  - 4 simple operations on the state matrix every round (except the last round)

The Four Simple Operations

- **substitute-bytes** (sub)
  - Non-linear operation based on a defined substitution box
  - Used to resist cryptanalysis and other mathematical attacks
- **shift-rows** (shr)
  - Linear operation for producing diffusion
- **mix-columns** (mic)
  - Elementary operation also for producing diffusion
- **add-round-key** (ark)
  - Simple set of XOR operations on state matrices
  - Linear operation
  - Produces confusion

AES 128



plaintext
subkeys
plaintext
ark
W[0,3]
ark
round 1
sub
shr
mic
ark
W[4,7]
round 10
inv_sub
inv_shr
inv_mic
ark
inv_sub
inv_shr
round 9
round 9
sub
shr
mic
ark
W[36,39]
inv_mic
ark
inv_sub
inv_shr
round 1
round 10
sub
shr
ark
W[40,43]
ark
ciphertext
ciphertext

Modes of Operations

- Let l be the block size of a given block cipher;

  $l = 64$ in DES, $l = 128$ in AES

- Let M be a plaintext string. Divide M into a sequence of blocks:

$$M = M_1 M_2 \ldots M_k,$$

- such that the size of each block M_i is l (padding the last block if necessary)

- There are several methods to encrypt M, where are referred to as block-cipher modes of operations

Standard Modes of Operations

Standard block-cipher modes of operations:

- electronic-codebook mode (ECB)
- cipher-block-chaining mode (CBC)
- cipher-feedback mode (CFB)
- output-feedback mode (OFB)
- counter mode (CTR)

Electronic-Codebook Mode (ECB)

ECB encrypts each plaintext block independently.

ECB Encryption Steps	ECB Decryption Steps
$C_i = E_k(M_i),$ $i = 1, 2, \cdots, k$	$M_i = D_k(C_i),$ $i = 1, 2, \cdots, k$

Easy and straightforward. ECB is often used to encrypt short plaintext messages

However, if we break up our string into blocks, there could be a chance that two different blocks are identical.

This provides the attacker with some information about the original text

Other Block-Cipher Modes deal with this in different ways

Cipher-Block-Chaining Mode (CBC)

- When the plaintext message M is long, the possibility that some blocks may repeat will increase
- CBC can overcome the weakness of ECB
- In CBC, the previous ciphertext block is used to encrypt the current plaintext block
- CBC uses an initial l-bit block C_0, referred to as *initial vector*

CBC Encryption Steps	CBC Decryption Steps
$C_i = E_k(C_{i-1} \oplus M_i)$, $i = 1, 2, \cdots, k$	$M_i = D_k(C_i) \oplus C_{i-1}$, $i = 1, 2, \cdots, k$

- What if a bit error occurs in a ciphertext block during transmission?
  - One bit change in C_i during transmission affects the decryption for M_i and M_{i+1}

Cipher-Feedback Mode (CFB)

- CFB turns block ciphers to stream ciphers
- $M = w_1 w_2 \ldots w_m$, where w_i is s-bit long
- Encrypts an s-bit block one at a time:
  - s=8: stream cipher in ASCII
  - s=16: unicode stream cipher
- Also has an l-bit initial vector V_0

$$pfx_s(U) = s \text{ bits prefix of U}$$
$$sfx_s(U) = s \text{ bits subfix of U}$$

CFB Encryption Steps	CFB Decryption Steps
$U_i = E_k(V_{i-1})$ $C_i = w_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})C_i$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $C_m = w_m \oplus pfx_s(U_m)$	$U_i = E_k(V_{i-1})$ $w_i = C_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})C_i$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $w_m = C_m \oplus pfx_s(U_m)$

Output-Feedback Mode (OFB)

- OFB also turns block ciphers to stream ciphers
- The only difference between CFB and OFB is that OFB does not place C_i in V_i .
- Feedback is independent of the message
- Used in error-prone environment

OFB Encryption Steps	OFB Decryption Steps
$U_i = E_k(V_{i-1})$ $C_i = w_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})pfx_s(U_i)$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $C_m = w_m \oplus pfx_s(U_m)$	$U_i = E_k(V_{i-1})$ $w_i = C_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})pfx_s(U_i)$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $w_m = C_m \oplus pfx_s(U_m)$

Counter Mode (CTR)

- CTR is block cipher mode.
- An l-bit counter Ctr, starting from an initial value and increases by 1 each time
- Used in applications requiring faster encryption speed

CTR Encryption Steps	CTR Decryption Steps
$Ctr = Ctr_0$ $C_i = E_k(Ctr^{++}) \oplus M_i$ $i = 1,2,\cdots,k$	$Ctr = Ctr_0$ $M_i = E_k(Ctr^{++}) \oplus C_i$ $i = 1,2,\cdots,k$

Stream Ciphers

- Stream ciphers encrypts the message one byte (or other small blocks of bits) at a time
- Any block ciphers can be converted into a stream cipher (using, e.g. CFB and OFB) with extra computation overhead
- How to obtain light-weight stream ciphers?

RC4

- RC4, designed by Rivest for RSA Security, is a light-weight stream cipher
  - It is a major component in WEP, part of the IEEE 802.11b standard.
  - It has variable key length: ranging from 1 byte to 256 bytes
  - It uses three operations: substitution, modular addition, and XORs.

Key Generation

- Secret keys are the most critical components of encryption algorithms
- Best way: random generation
  - Generate pseudorandom strings using deterministic algorithms (pseudorandom number generators “PRNG”); e.g.
    - ANSI X9.17 PRNG
    - BBS Pseudorandom Bit Generator

ANSI X9.17 PRNG

- Published in 1985 by the American National Standard Institute (ANSI) for financial institution key management
- Based on 3DES/2 with two initial keys K_1 and K_2, and an initial vector V_0
- Two special 64-bit binary strings T_i and V_i:
  - T_i represents the current date and time, updated before each round
  - V_i is called a seed and determined as follows:

$$R_i = EDE_{K_1, K_2}(V_i \oplus EDE_{K_1, K_2}(T_i)),$$
$$V_{i+1} = EDE_{K_1, K_2}(R_i \oplus EDE_{K_1, K_2}(T_i)),$$
$$i = 0, 1, \ldots$$

BBS Pseudorandom Bit Generator

- It generates a pseudorandom bit in each round of computation.
- Let p and q be two large prime numbers satisfying

$$p \bmod 4 = q \bmod 4 = 3$$

- Let $n = p$ X q and s be a positive number, where
  - s and p are relatively prime; i.e. $\gcd(s,p) = 1$
  - s and q are relatively prime; i.e. $\gcd(s,q) = 1$
- BBS pseudorandom bit generation:

Lifelong Learning

THANKS YOU

Sources

- *Forouzan, Behrouz A., and Debdeep Mukhopadhyay. Cryptography and Network Security (Sie). McGraw-Hill Education, 2011.*
- *Stallings, William. "Cryptography and Network Security. 2005." ISBN: 0-13-187316-4.*