Computer & Information Security (372-1-460-1)

Cryptographic Algorithms

Dept. of Software and Information Systems Engineering, Ben-Gurion University

Prof. Asaf Shabtai

{elovici, shabtaia}@bgu.ac.il

Spring, 2020

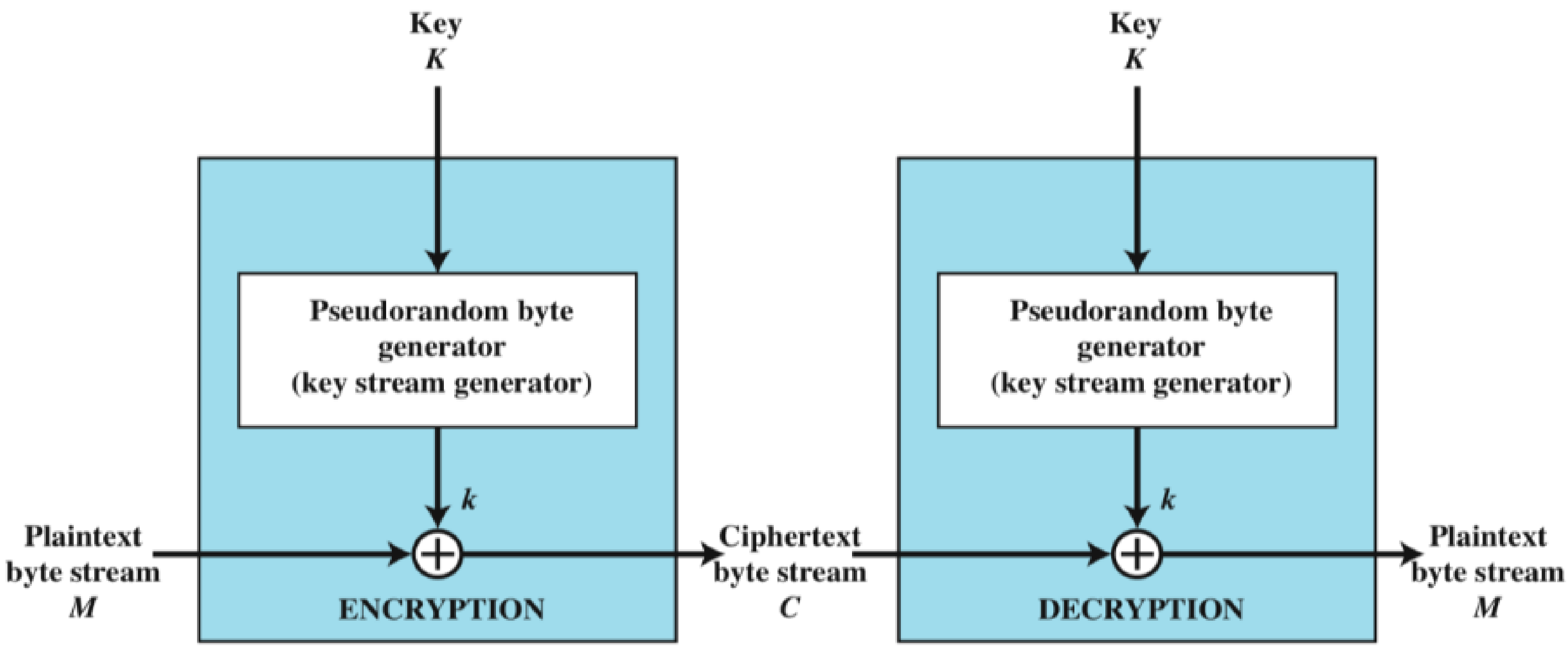
Stream Ciphers

- processes input elements continuously
- key input to a pseudorandom bit generator
  – produces stream of random like numbers using the key
  – unpredictable without knowing input key
  – XOR keystream output with plaintext bytes
- are faster and use far less code than Block-Cyphers

Stream Ciphers

- design considerations:
  - encryption sequence should have a large period - since it eventually repeats
  - keystream approximates random number properties 1s ~= 0s
  - uses a sufficiently long key to protect against brute force attack

Stream Ciphers



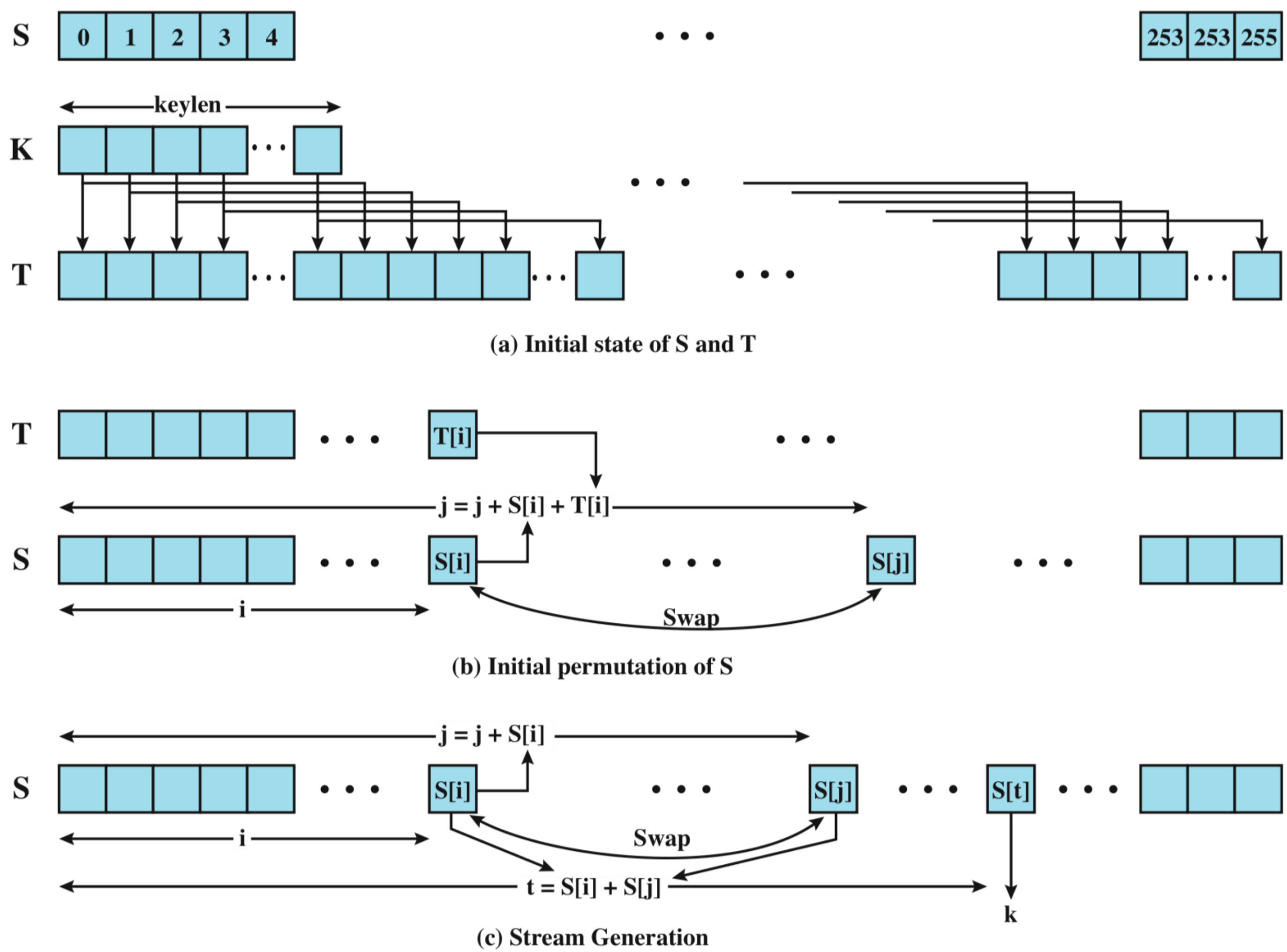
The RC4 Algorithm

- Designed in 1987 by Ron Rivest for RSA Security
- Stream cipher with byte-oriented operations
- Based on the use of a random permutation
- Can be expected to run very quickly in software
- Used in the SSL/TLS standards, WEP (Wired Equivalent Privacy) and WPA (WiFi Protected Access) protocol
- In September 1994 was anonymously posted on the Internet

RC4 Description

- Three main parts:
  - initialization of State Vector with the Symmetric Key
  - initial permutation = KSA (Key Scheduling Algorithm)
  - stream generation = PRGA (Pseudo Random Generation algorithm)
- Notation:
  - S = {0, 1, 2, … n-1} is the initial permutation
  - l = length of key

The RC4 Algorithm



RC4: Initialization of State Vector

- Two vectors of bytes:
  - $S[0], S[1], S[2], \ldots, S[255]$
  - $T[0], T[1], T[2], \ldots, T[255]$
- Key: variable length, from 1 to 256 bytes
- Initialization:
  1. $S[i] \leftarrow i, \quad \text{for } 0 \le i \le 255$
  2. $T[i] \leftarrow K[i \mod \text{key-length}], \quad \text{for } 0 \le i \le 255$

     (i.e., fill up $T[0..255]$ with the key K repeatedly.)

RC4: Initial Permutation (KSA)

- Initial Permutation of S:

  $j \leftarrow 0$

  for $i \leftarrow 0$ to 255 do

  $\quad j \leftarrow (j + S[i] + T[i]) \bmod 256$

  $\quad$ Swap $S[i], S[j]$

- This part of RC4 is generally known as the Key Scheduling Algorithm (KSA).
- After KSA, the input key and the temporary vector T will no longer be used.

RC4: Key Stream Generation

- Key stream generation:

$$
\begin{aligned}
& i, j \leftarrow 0 \\
& \text{while (true)} \\
& \qquad i \leftarrow (i + 1) \bmod 256 \\
& \qquad j \leftarrow (j + S[i]) \bmod 256 \\
& \qquad \text{Swap } S[i], S[j] \\
& \qquad t \leftarrow (S[i] + S[j]) \bmod 256 \\
& \qquad k \leftarrow S[t] \\
& \qquad \text{output } k
\end{aligned}
$$

RC4 Example

- Simple 4-byte example
- $S = \{0, 1, 2, 3\}$
- $K = \{1, 7, 1, 7\}$
- Set $i = j = 0$

KSA

- First Iteration (i = 0, j = 0, S = {0, 1, 2, 3}):
- j = (j + S[i] + K[i]) = (0 + 0 + 1) = 1 (1mod 4)
- Swap S[i] with S[j]: Swap S[0] with S[1]: S = {1, 0, 2, 3}

- Second Iteration (i = 1, j = 1, S = {1, 0, 2, 3}):
- j = (j + S[i] + K[i]) = (1 + 0 + 7) = 0 (8mod 4)
- Swap S[i] with S[j]: S = {0, 1, 2, 3}

- K = {1, 7, 1, 7}

KSA

Third Iteration ($i = 2, j = 0, S = \{0, 1, 2, 3\}$):
$j = (j + S[i] + K[i]) = (0 + 2 + 1) = 3$ (3mod 4)
Swap S[i] with S[j]: $S = \{0, 1, 3, 2\}$

Fourth Iteration ($i = 3, j = 3, S = \{0, 1, 3, 2\}$):
$j = (j + S[i] + K[i]) = (3 + 2 + 7) = 0$ (12 mod 4)
Swap S[i] with S[j]: $S = \{2, 1, 3, 0\}$

$K = \{1, 7, 1, 7\}$

PRGA (Pseudo Random Generation algorithm)

- Reset i = j = 0, Recall S = {2, 1, 3, 0}
- i = i + 1 = 1 (1 mod 4)
- j = j + S[i] = 0 + 1 = 1 (1 mod 4)
- Swap S[i] and S[j]: S = {2, 1, 3, 0}
- t= (S[i] + S[j]) mod 4 = 1+1=2 (2 mod 4)
- Output k = S[t] = S[2] = 3

The RC4 Algorithm

- Does not use IV (nonce)
- Same key on the same plaintext will result in the same cypher
- Weakness in the random number generator
- WEP was hacked in 2007

Risks in using stream ciphers

"Two time pad" is insecure:

$$\begin{cases} C_1 \leftarrow m_1 \oplus PRG(k) \\ C_2 \leftarrow m_2 \oplus PRG(k) \end{cases}$$

Eavesdropper does:

$$C_1 \oplus C_2 \quad \rightarrow \quad m_1 \oplus m_2$$

Enough redundant information in English that:

$$m_1 \oplus m_2 \rightarrow \quad m_1 , m_2$$

Risks in using stream ciphers

- **Short Cycle Length** key-streams generated by pseudorandom generators are cyclic. True random are unbreakable.
- **Correlation Attack** statistical analysis where parts of the contents of the two messages could be identified as equal → leads to the key, or parts of the key.

Risks in using stream ciphers

- **Substitution *Attack*** type of man-in-the-middle attack: In structure messages specific part my be substituted → cause confusion or misbehavior of the system even if the information is protected by a strong stream cipher.
- **Reused-Key *Attack*** Attack known from Wired Equivalent Privacy (WEP) : Example: long term key plus 24 bits changing as IV: Chance of finding reused key is high: Breaking the system in short time is likely.

Message Authentication

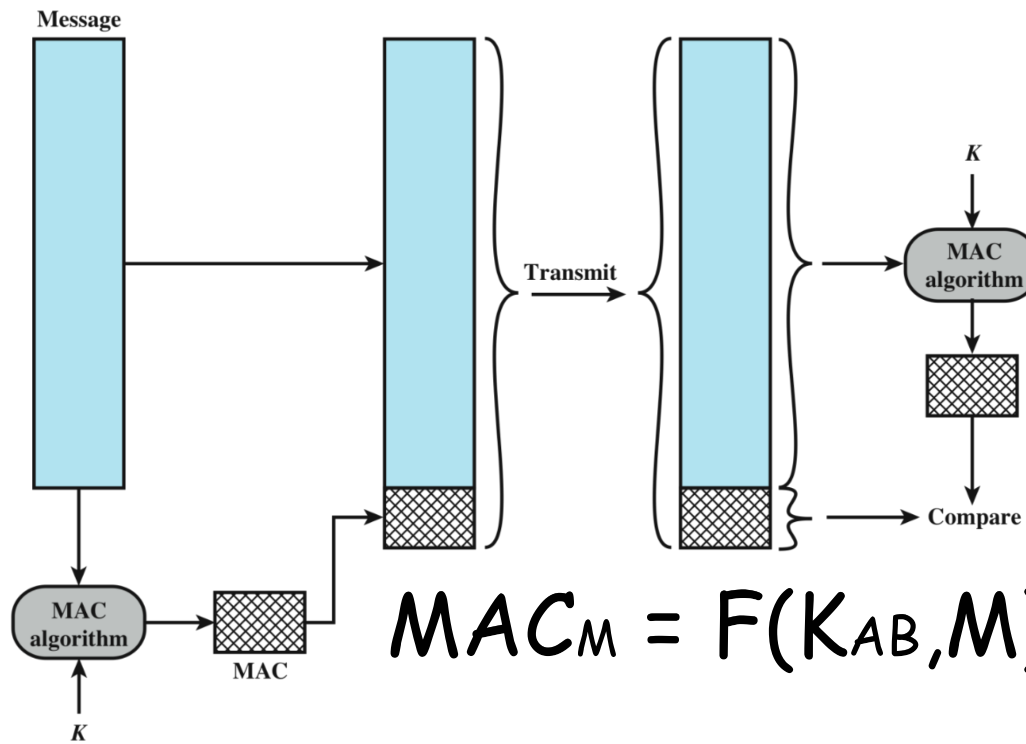
Message Authentication Code

- Message authentication is a property, which promise the following:
  - **Data integrity:** the message has not been altered while in transit
  - **Sender authenticity :** the receiving party can verify the source of the message (authenticate the source).
  - **Freshness:** The message is timely (fresh) and in correct sequence
- Message authentication does not necessarily include the property of **non-repudiation** (i.e., the sender of a message will not be able to successfully challenge the authorship of the message)

Message Authentication Codes (MAC)

- A message authentication code (MAC), is a short piece of information used to authenticate message
- The MAC value protects both a message's <u>data integrity</u> as well as its <u>authenticity.</u>

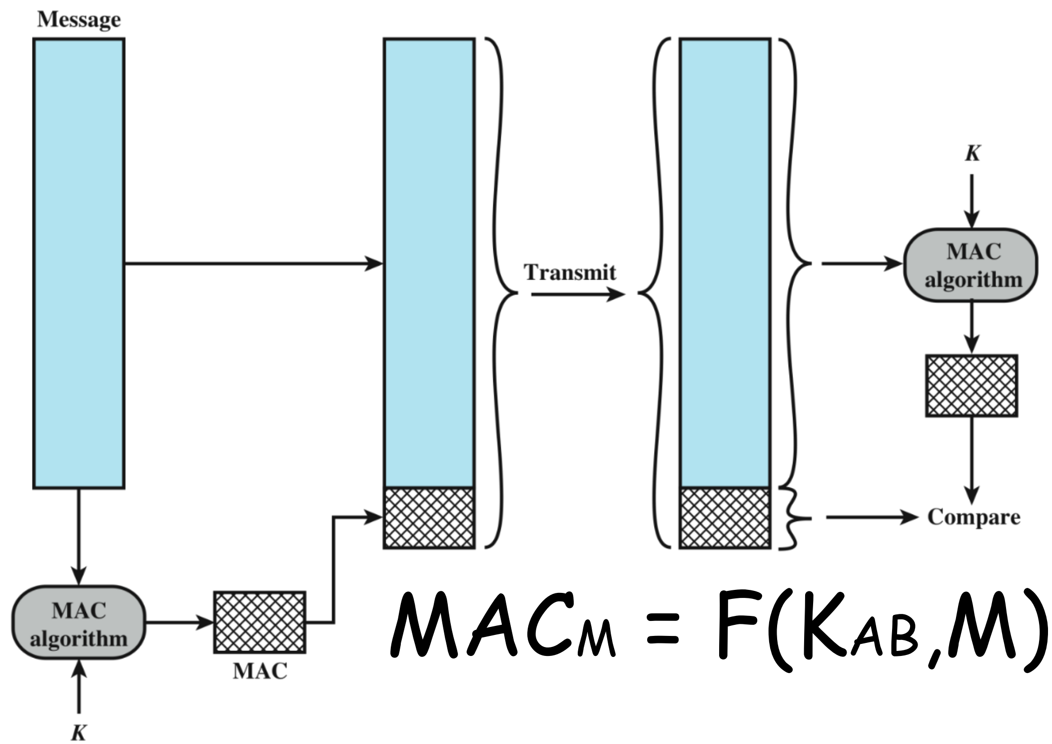
- The sender and receiver must hold a symmetric secret key (denoted as K_{ab})



$$MAC_M = F(K_{AB}, M)$$

Message Authentication Codes (MAC)

- If the sender find a match between the received MAC and his calculated code then:
  - **Data integrity:** An attacker can alter the message but not the code since doesn't have the secret key
  - **Sender authenticity:** no one else is able to produce the same MAC due to the secret key
  - **Freshness:** attacker cannot alter the sequence number within a message

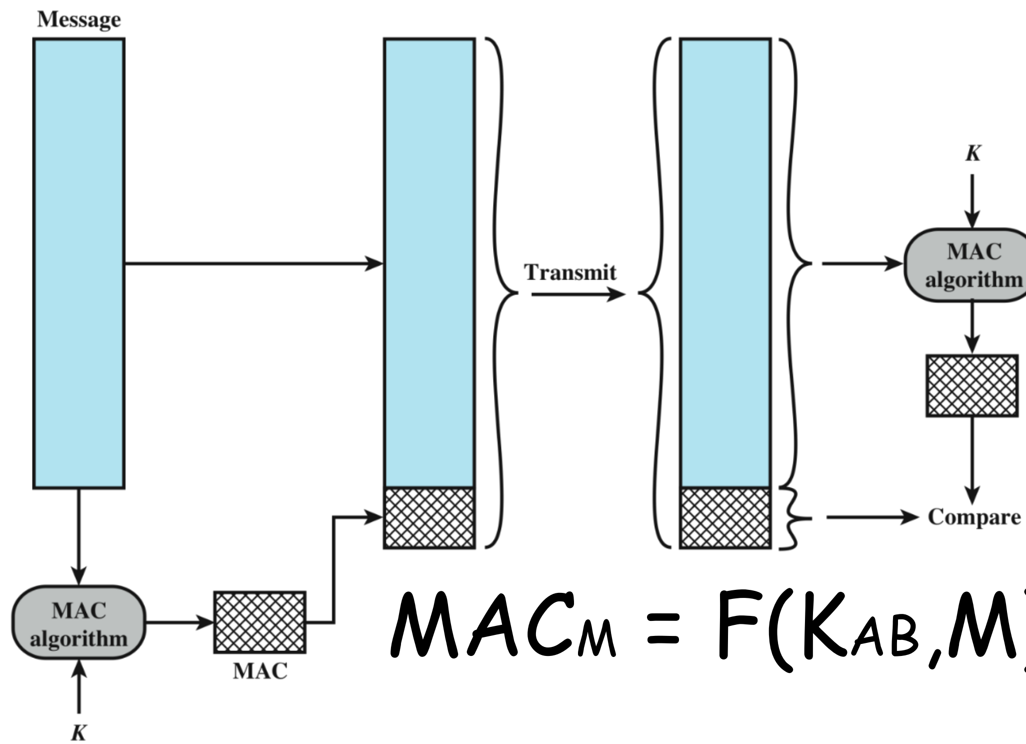


$$MAC_M = F(K_{AB}, M)$$

Does MAC provide the property of non-repudiation?

Message Authentication Codes (MAC)

- Any user who can verify a MAC is also capable of generating MACs for other messages.
- Hence, MACs **<u>do not</u>** provide the property of non-repudiation.



$$MAC_M = F(K_{AB}, M)$$

Sharing a Secrete

- This material is based on Prof. Eli Biham presentation on Secrete sharing

- Main reference: Adi Shamir, How to share a Secret, CACM, Vol. 22, No. 11, November 1979, pp 612-613.

The need

- Message M can be protected by encryption using key k (C=E(M,K))
- C can be stored by multiple parties but access to M can be done by only by members who have the key K.
- How can we distribute the key K to a group without "revealing" its content to any member of the group?
- Why we need such a scheme:
  - Authorize access critical action that require several members to approve.
  - Restore confidential information in a company.
  - Etc…

Secret sharing goal

- Most of the sharing a secret schemes try to achieve the following:
  - Sharing a secrete S between n parties
  - Each party receive a share Si
  - Cooperation between a predefined subgroup (for example, any k out of n) enables to reconstruct the secret.
  - Any small group (for example, smaller than k) cannot reconstruct the secret.

(k,n) - Threshold scheme

- Such schemes satisfy the following requirements:
  - Sharing a secret S between n parties
  - Each party receives a share Si
  - Cooperation of any k parties out of n enable the reconstruction of S
  - Any subgroup smaller than k cannot reconstruct the secrete S or gain any information on the secret.

Bad Example

- Let S be AES key (128 bit).
- We can share it between four parties where (k=n).
- Cooperation between all parties will allow to reconstruct the key S.
- What is wrong here?
  - Any member gain some information on the secret S.
  - Three members posses 96bit of the key and can search for 2^32 possible keys.
- This is not a valid threshold scheme.

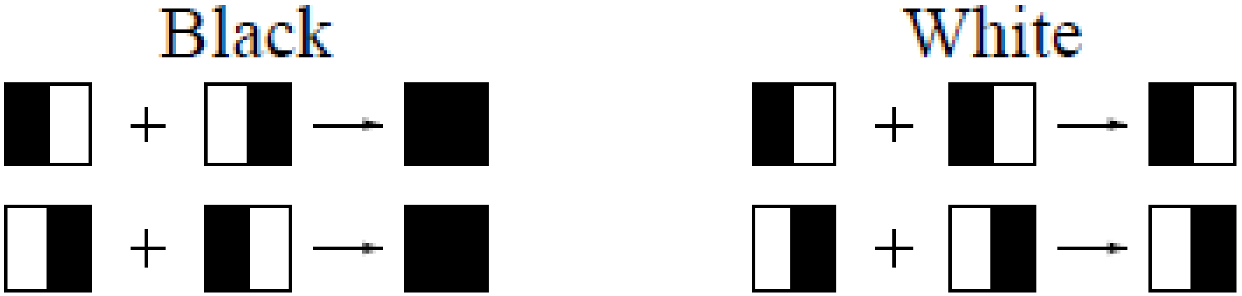
A simple (2,2) threshold scheme

- In this scheme we split the secret S into two shares, s1 and s2.
- Assume S is a secret, and it has m bits
- Let s1 be random number
- S2 can be computed: s2=S xor s1
- Each share is allocated to different entity
- Both entities can recover the secret:

  S=s1 xor s2
- No single entity can recover S

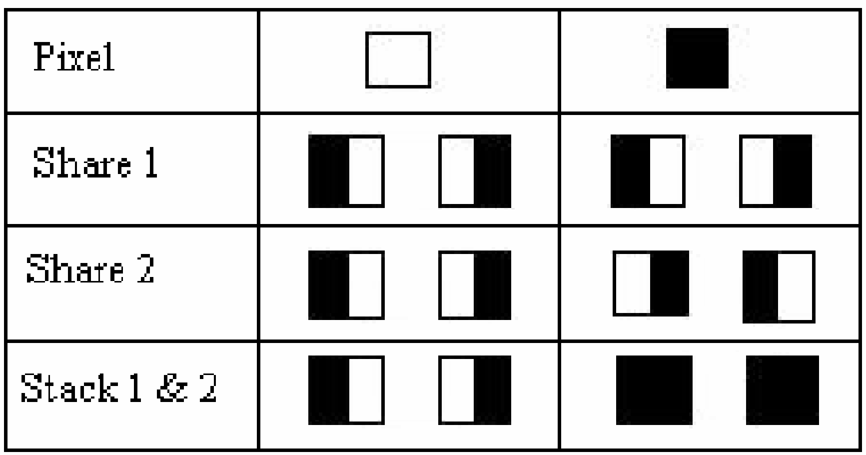
Visual secret sharing

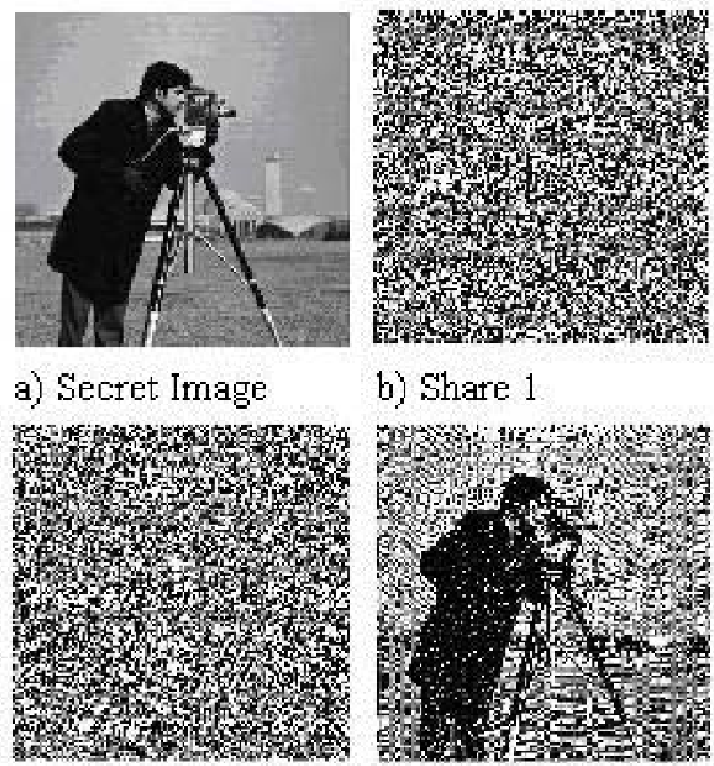
- The simple threshold scheme can be easily adopted to share a picture between two parties.
- Each pixel in the image can be "implemented" by combining two shares:



- The way each pixel is implemented and shared is chosen at random for each pixel.

Visual secret sharing





a) Secret Image
b) Share 1
c) Share 2
d) Reavealed sceret

An (n,n) threshold scheme

- In this scheme we split the secret S into n shares, $s_1..s_n$.
- Assume S is a secret and it has m bits
- Let $s_1..s_{n-1}$ be random numbers
- s_n can be computed: s_n =S xor s_1 xor s_2 xor … s_{n-1}
- Each share is allocated to different entity
- All entities together can recover the secret:

  S=s1 xor s2 xor … s_n
- No single entity can recover S

Shamir's (k,n) - Threshold schemes

- Goal: Be able to distribute shares such that every k shares can reconstruct the secret.
- These schemes are based on unique interpolation of polynomials:

Given k points on plain $(x_1,y_1),\dots,(x_k,y_k)$, where all x_i's are distinct, there exist a unique polynomial q of degree k-1 for which $q(x_i)=y_i$ for all i.

Shamir's (k,n) – Threshold schemes

- Let S be the secret that we want to share
- Select a prime modulus p where p>max(n,|S|)
- Select random polynomial q(x) such that q(0)=S, i.e. select the coefficient $a_1,a_2,\ldots,a_{k-1}$ randomly, and select $a_0=S$.
- The polynomial is:

$$q(x)=a_0+a_1x+a_2x^2+\ldots+a_{k-1}x^{k-1}(\mathrm{mod}\ p)$$

- The distributed shares are:

$$s_1=(1,q(1)),\ s_2(2,q(2)),\ldots,\ s_n(n,q(n))$$

Shamir's (k,n) - Threshold schemes

- The secret can be reconstructed from the k shares since q is a polynomial of degree k-1, thus given k points ((xi,yi), i=1,…,k) q(x) can be uniquely reconstructed by Lagrange:

$$q(x) = \sum_{i=1}^{k} y_i \prod_{j=1, j\neq i}^{k} \frac{x - xj}{x_i - x_j}$$

- The secret is the reconstructed polynom q(x) at x=0 and thus:

$$S = q(0) = \sum_{i=1}^{k} y_i \prod_{j=1, j\neq i}^{k} \frac{-xj}{x_i - xj} \ (mod\ p)$$

Limitations of cryptography

- People make other mistakes; crypto doesn't solve them
- Misuse of cryptography is fatal for security (e.g., WEP)