# CIS 6930– Emerging Topics in Network Security

Topic 2.3 Secret Sharing

# Secret Sharing

# Secret Sharing

- Objective
  - Divide data $D$ into $n$ pieces $D_1$, …, $D_n$ in such a way that
    - Knowledge of any $k$ or more $D_i$ pieces makes $D$ easy to compute,
    - Knowledge of any $k - 1$ or fewer $D_i$ pieces leaves $D$ completely undetermined.
  - Such a scheme is called a ($k$, $n$) threshold scheme.
- Useful when no single entity can be trusted with the secret
  - Management of cryptographic keys

# Shamir's Secret Sharing

- Underlying fact
  - Based on polynomial equations.
  - Given $k$ points in the 2-d plane $(x_1, y_1), \ldots, (x_k, y_k)$ with distinct $x_i$'s,
  - there is one and only one polynomial $q(x)$ of degree $k-1$ such that

$$q(x_i)=y_i \text{ for all } i.$$

# Shamir's Secret Sharing (Cont'd)

- Split the secret D
  - To divide $D$ into pieces $D_i$ …
  - Pick a random $k - 1$ degree polynomial

$$q(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1}$$

    in which $a_0 = D$.
  - Compute $D_1 = q(1)$, $D_2 = q(2)$, …, $D_n = q(n)$.

  - The secret shares represent distinct points on the polynomial.

# Shamir's Secret Sharing (Cont'd)

- Reconstruction
  - Given any subset of $k$ of these $D_i$ values (with their identifying indices)
    - Find the coefficients of $q(x)$ by solving the linear equations,
    - Evaluate $D = q(0)$.
  - Given just $k - 1$ of these values,
    - $D$ could be any value
    - In other words, $D$ being any value will give one and only one possible polynomial

# Exercise

Assume we have a secret value $x = 10$. Split it into 5 shares using Shamir's ($k$, $n$) threshold scheme so that any 3 out of the 5 shares can recover $x$.

# CIS 6930– Emerging Topics in Network Security

## Topic 2.4 Rabin's Information Dispersal Algorithm

# Motivation

- IDA was developed to provide safe and reliable transmission of information in distributed systems.

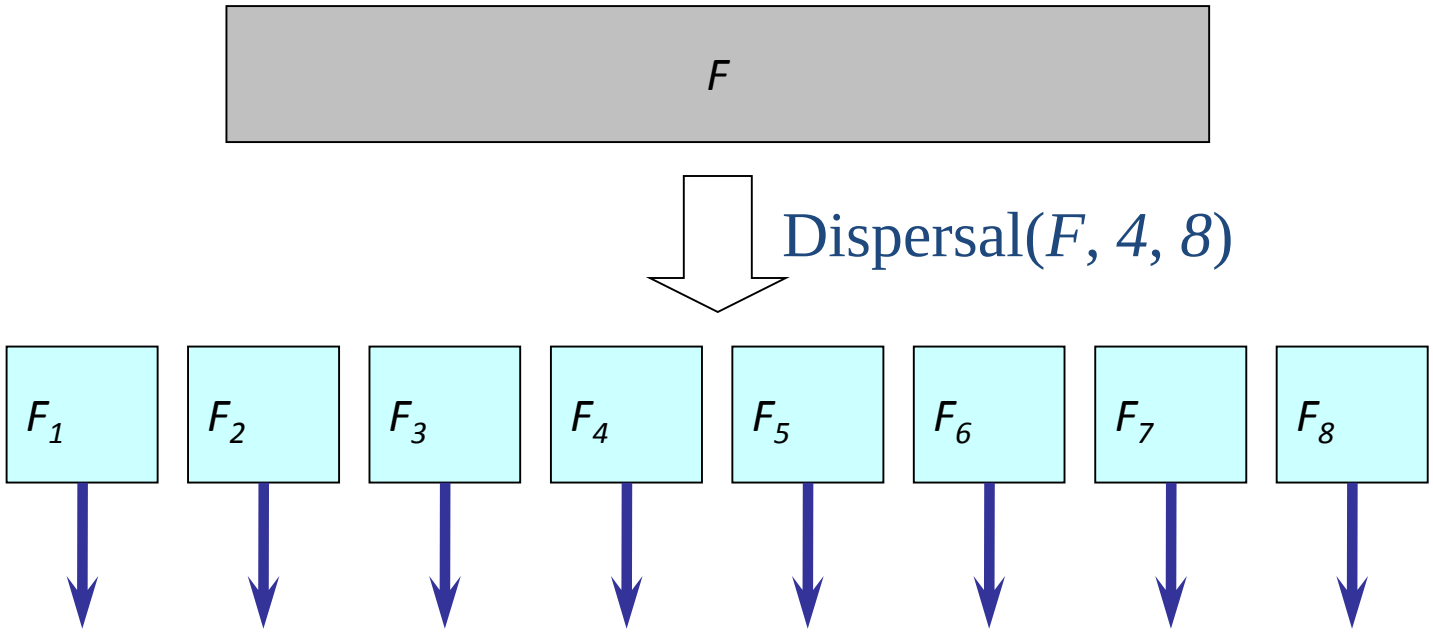
- Inefficiency of retransmission of lost packets
  - In multicast transmission, different receivers lose different sets of packets.
  - Re-request and retransmission increases delays.

# High-level Operations

- Dispersal($F$, $m$, $n$):
  - Split input $F$ with redundancy into $n$ pieces $F_i$ ($1 \le i \le n$).
  - $|F_i| = |F|/m$, and $m \le n$
- Recovery($\{F_{i_j} | (1 \le j \le m), (1 \le i_j \le n)\}$, $m$, $n$):
  - Reconstruct $F$ from any $m$ out of the $n$ pieces ($F_i$ ($1 \le i \le n$))

# Dispersal(*F*, *m*, *n*) – Example 1

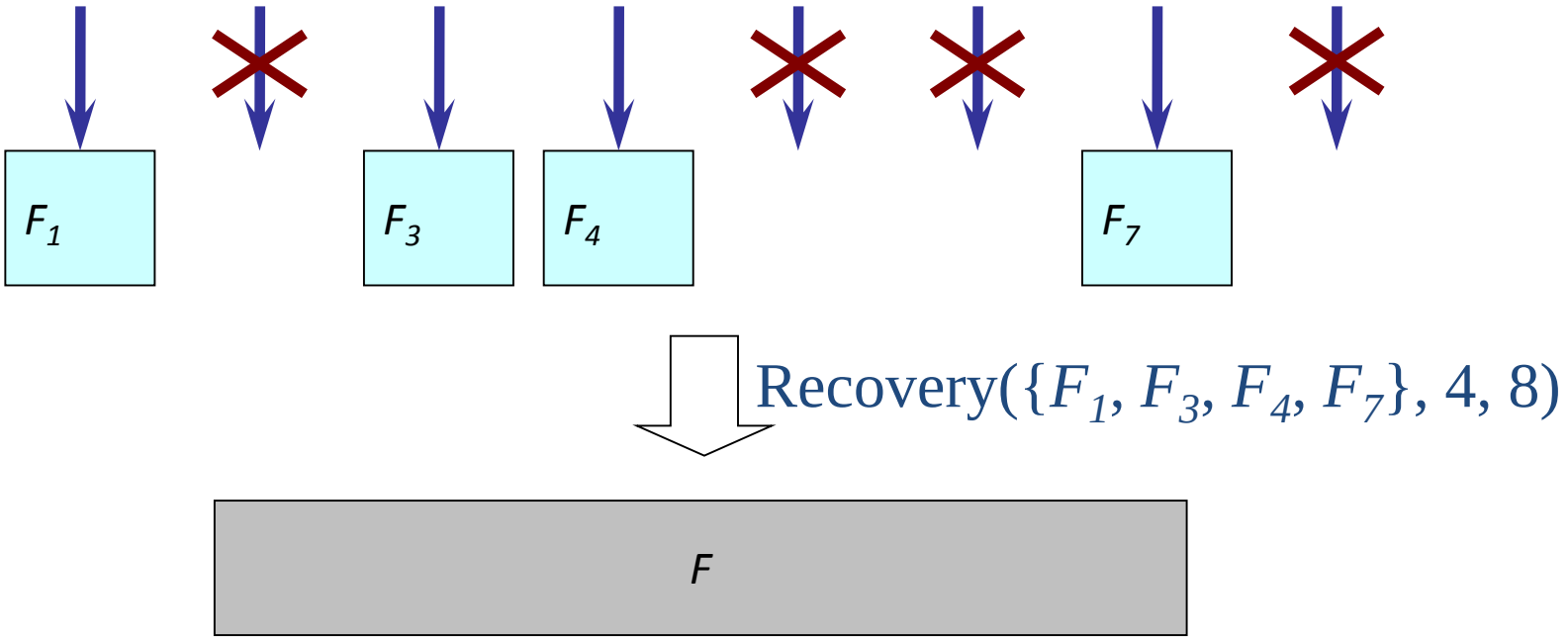
- |*F*|=32 bytes, *m*=4, *n*=8




– $|F_i| = 32/4 = 8$ bytes $(1 \leq i \leq n)$

# Recovery($\{F_{i_j} | (1 \le j \le m), (1 \le i_j \le n)\}$, $m$, $n$) – Example 2

- $|F|$=32 bytes, $m$=4, $n$=8, $|F_i|$=8 bytes ($1 \le i \le 8$)
- Assume the following 4(=$m$) pieces are received.



# Dispersal($F$, $m$, $n$)

- $F = b_1, b_2, \ldots, b_N$
  - $N = |F|$, and $b_i$ represents each byte in $F$ ($0 \le b_i \le 255$).

- $F = (b_1, \ldots, b_m), (b_{m+1}, \ldots, b_{2m}), \ldots, (b_{N-m+1}, \ldots, b_N)$
  - $\mathbf{S_i} = (b_{(i-1)m+1}, \ldots, b_{im})^T (1 \le i \le N/m)$

- The matrix $\mathbf{M}_{m \times N/m}$ is constructed as follows:
  - $\mathbf{M} = [\, \mathbf{S_1}\ \mathbf{S_2}\ \ldots\ \mathbf{S}_{N/m}\, ]$

# Dispersal(*F*, *m*, *n*)

- The matrix $A_{n \times m}$ is constructed as follows :

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix}$$

  - $\mathbf{a_i} = (a_{i1}, \dots, a_{im})$ $(1 \le i \le n)$
    - Every subset of *m* different vectors should be linearly independent.

# Dispersal(*F*, *m*, *n*)

- The following *Vandermonde matrix* satisfies the property required for A.

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{m-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{m-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{m-1} \\ 1 & x_n & x_n^2 & \dots & x_n^{m-1} \end{bmatrix}$$

  - $m \le n$, and all $x_i$'s are nonzero elements and pairwise different.
  - Any *m* different rows are linearly independent, so any submatrix composed of a set of any *m* different rows is invertible.

# Dispersal(*F*, *m*, *n*)

- The *n* pieces $F_i$ ($1 \le i \le n$) are computed as follows:

$$A \cdot M = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix} \begin{bmatrix} S_1 & S_2 & \dots & S_{N/m} \end{bmatrix}$$

$$= \begin{bmatrix} a_1 \cdot S_1 & a_1 \cdot S_2 & \dots & a_1 \cdot S_{N/m} \\ a_2 \cdot S_1 & a_2 \cdot S_2 & \dots & a_2 \cdot S_{N/m} \\ \dots & \dots & \dots & \dots \\ a_n \cdot S_1 & a_n \cdot S_2 & \dots & a_n \cdot S_{N/m} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \dots \\ F_n \end{bmatrix}$$

where $a_i \cdot S_k = a_{i1}b_{(k-1)m+1} + \dots + a_{im}b_{km}$

# Dispersal(*F*, *m*, *n*) – Example 3

- $|F|$=32 bytes, $m$=4, $n$=8
  - $F = b_1, b_2, \ldots, b_{32}$
  - Represented as $M_{4\times 8}$

$$M = \begin{bmatrix} S_1 & S_2 & \ldots & S_8 \end{bmatrix} = \begin{bmatrix} b_1 & b_5 & \ldots & b_{29} \\ b_2 & b_6 & \ldots & b_{30} \\ b_3 & b_7 & \ldots & b_{31} \\ b_4 & b_8 & \ldots & b_{32} \end{bmatrix}$$

# Dispersal(*F*, *m*, *n*) – Example 3

– $A_{8\times4}$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_8 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \dots & \dots & \dots & \dots \\ 1 & x_8 & x_8^2 & x_8^3 \end{bmatrix}$$

# Dispersal(*F*, *m*, *n*) – Example 3

- $F_i$ ($1 \le i \le 8$) are computed as follows:

$$\mathrm{A} \cdot \mathrm{M} = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_8 \end{bmatrix} \begin{bmatrix} S_1 & S_2 & \dots & S_8 \end{bmatrix}$$

$$= \begin{bmatrix} a_1 \cdot S_1 & a_1 \cdot S_2 & \dots & a_1 \cdot S_8 \\ a_2 \cdot S_1 & a_2 \cdot S_2 & \dots & a_2 \cdot S_8 \\ \dots & \dots & \dots & \dots \\ a_8 \cdot S_1 & a_8 \cdot S_2 & \dots & a_8 \cdot S_8 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \dots \\ F_8 \end{bmatrix}$$

# Recovery($\{F_{i_j} | (1 \le j \le m), (1 \le i_j \le n)\}, m, n$)

- Given $m$ pieces $F_{i_j}$ ( $(1 \le j \le m), (1 \le i_j \le n)$ ),

$$\begin{bmatrix} F_{i_1} \\ F_{i_2} \\ \dots \\ F_{i_m} \end{bmatrix} = \begin{bmatrix} a_{i_1} \\ a_{i_2} \\ \dots \\ a_{i_m} \end{bmatrix} \cdot M = A' \cdot M$$

- M can be recovered from the given $m$ pieces $F_{i_j}$ ( $(1 \le j \le m), (1 \le i_j \le n)$ ) because A′ is invertible.

$$\begin{bmatrix} a_{i_1} \\ a_{i_2} \\ \dots \\ a_{i_m} \end{bmatrix}^{-1} \begin{bmatrix} F_{i_1} \\ F_{i_2} \\ \dots \\ F_{i_m} \end{bmatrix} = M$$

# Recovery($\{F_{i_j} | (1 \le j \le m), (1 \le i_j \le n)\}, m, n$) – Example 4

- $|F|$=32 bytes, $m$=4, $n$=8
- In example 3, $F_i$ ($1 \le i \le 8$) pieces of 8 bytes are resulted.
- Assume that $\{F_1, F_3, F_4, F_7\}$ are received among them.

$$\begin{bmatrix} F_1 \\ F_3 \\ F_4 \\ F_7 \end{bmatrix} = \begin{bmatrix} a_1 \cdot S_1 & a_1 \cdot S_2 & \dots & a_1 \cdot S_8 \\ a_3 \cdot S_1 & a_3 \cdot S_2 & \dots & a_3 \cdot S_8 \\ a_4 \cdot S_1 & a_4 \cdot S_2 & \dots & a_4 \cdot S_8 \\ a_7 \cdot S_1 & a_7 \cdot S_2 & \dots & a_7 \cdot S_8 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_3 \\ a_4 \\ a_7 \end{bmatrix} \cdot M$$

# Recovery($\{F_{i_j} | (1 \leq j \leq m), (1 \leq i_j \leq n)\}, m, n$) – Example 4

- The original data M can be recovered by the following computation:

$$\begin{bmatrix} a_1 \\ a_3 \\ a_4 \\ a_7 \end{bmatrix}^{-1} \begin{bmatrix} F_1 \\ F_3 \\ F_4 \\ F_7 \end{bmatrix} = \mathrm{M}$$

# Exercise

Alice wants to deliver the following data to Bob.

1A 5D 3C 24 26 71 8E 9E 74 65 29 BF CD C0 28

Use IDA to disperse the transmitted data into 5 pieces (packets), such that with any 3 of the 5 pieces Bob can reconstruct the original data