

A DISTRIBUTED DATA STORAGE SCHEME FOR SENSOR NETWORKS

Abhishek Parakh and Subhash Kak
Computer Science Department
Oklahoma State University

MobiSec 2009, Turin, Italy

Introduction

- Secure data storage in distributed networks, as sensor networks, is a challenge.
- Data encryption using a key moves the problem of storing the data to storing the key securely.
- Loss of data is no good.
- Fault tolerance of encrypted data needed.
- Fault tolerance in encryption key needed.
 - ▣ To solve above problems
- We propose an implicitly secure scheme for data storage in sensor networks.

Our Proposal

- Split the data into pieces, such that
 - ▣ Each piece in itself does not reveal any information
 - ▣ A certain number of pieces are redundant – fault tolerance
 - ▣ Each piece of data is stored on different sensors
 - ▣ No explicit encryption key is required
 - ▣ Data is implicitly secure
 - To achieve the above,
- We propose a new data partitioning scheme

The Proposed Data Partitioning Scheme

- An equation of degree k has k roots

$$x^k + a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x + a_0 = 0$$

- If we replace a_0 with data d , then the roots of the equation represent the data partitions/pieces

$$x^k + \sum_{i=1}^{k-1} a_{k-i}x^{k-i} + d \equiv 0 \pmod{p}$$

$$\prod_{i=1}^k (x - r_i) \equiv 0 \pmod{p}$$

- Since d is the coefficient free term, reconstruction is given by

$$\prod_{i=1}^k r_i \equiv d \pmod{p}$$

- However, all roots are required for reconstruction

Restriction on Coefficients

- Not all coefficients must be simultaneously 0
 - Equation will have k roots only if d is k th power of x
 - Further, any one partition would suffice to reconstruct the data by raising that partition to power k
- Coefficients must be so chosen that k roots exist in finite field – an algorithm given (later)

Introducing Redundancy

- Extend k partitions to n partitions, $n \geq k$
 - $n - k =$ redundant partitions
- If $\{r_1, r_2, \dots, r_k\}$ is the set of partitions
- Map it into a set of n partitions $\{p_1, p_2, \dots, p_n\}$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nk} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

here a_{ij} are randomly chosen numbers from the field such that any k rows of the matrix A on LHS are linearly independent

Partitions are given by $p_i = \{a_{i1}, a_{i2}, \dots, a_{ik}, c_i\}$

- Reconstruction requires any k out of n partitions p_i

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{bmatrix} = \begin{bmatrix} a_{m1} & a_{m2} & \dots & a_{mk} \\ a_{n1} & a_{n2} & \dots & a_{nk} \\ \vdots & & & \\ a_{i1} & a_{i2} & \dots & a_{ik} \end{bmatrix}_{k \times k}^{-1} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}_{k \times 1}$$

- And a matrix inversion operation and $O(k^2)$ multiplications

Complexity of the Proposed Protocol

□ Complexity of (k,k) partitioning scheme – $O(k)$

Algorithm 1a (k,k) Data Partitioning Scheme

Input \leftarrow data d

1. Choose randomly and uniformly from a finite field \mathbb{Z}_p , $k - 1$ numbers

$$r_1, r_2, \dots, r_{k-1}.$$

2. Compute $r_k = d \cdot (r_1 \cdot r_2 \cdot \dots \cdot r_{k-1})^{-1} \bmod p$.

3. Construct the k^{th} degree polynomial: $p(k) = (x - r_1)(x - r_2) \dots (x - r_k)$

$$\bmod p = x^k + a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x + a_0 \bmod p.$$

4. The roots r_1, r_2, \dots, r_k of the polynomial $p(k)$ represent the data partitions.

Output \rightarrow vector $\vec{R} = [r_1, r_2, \dots, r_k]$

□ Data reconstruction (k,k) partitioning – $O(k)$

Algorithm 1b (k,k) Data Reconstruction

Input \leftarrow vector $\vec{R} = [r_1, r_2, \dots, r_k]$

1. Data $d = r_1 \cdot r_2 \cdot \dots \cdot r_k \pmod{p}$.

Output $\rightarrow d$

- Complexity of partition creation - (k,n) scheme – $O(n \log^2 n)$
 - Use (k,k) scheme as a sub-algorithm
 - Generate a $n \times k$ matrix A with linearly independent rows
 - Compute $[A]_{n \times k} \cdot [\vec{R}^T]_{k \times 1} = [C]_{n \times 1}$
 - If A is chosen to be a Vandermonde matrix then, partitions are given by $p_i = \{x_i, c_i\}, 1 \leq i \leq n.$

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^{k-1} \\ 1 & x_2 & x_2^2 & x_2^3 & \dots & x_2^{k-1} \\ \vdots & & & & & \\ 1 & x_n & x_n^2 & x_n^3 & \dots & x_n^{k-1} \end{bmatrix}$$

□ Data reconstruction (k,n) scheme – $O(k \log^2 k)$

Input \leftarrow any k partitions $p_i = \{x_i, c_i\}$

1. Construct the $k \times k$ matrix B by choosing the rows of matrix A corresponding to the given pairs p_i and compute B^{-1} .
2. Reconstruct the k shares by evaluating $[\vec{R}^T]_{k \times 1} = [B^{-1}]_{k \times k} \cdot [C]_{k \times 1}$.
3. Use Algorithm 1b to reconstruct data d , where the input is the algorithm is \vec{R} .

Reducing computations during partition creation

- Often data collection and storage is done on sensors but reconstruction may be done only at the sink
- Sinks usually have greater computational power compared to sensors
- Partition creation in (k,n) scheme can be done in linear time with $O(k)$ multiplications and k additions if elements of matrix A are chosen from $GF(2)$

- Example:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

- Here we are ignoring the cost multiplications with 1 and 0.

- However, now the reconstruction would require $O(k^3)$ operations to invert the matrix and $O(k^2)$ multiplications for data reconstruction.

Note on security

- Data is mapped into multiplication of roots

$$\prod_{i=1}^k r_i \equiv d \pmod{p}$$

- Therefore if the coefficients of the equation are chosen randomly and uniformly from the field then the knowledge of $k-1$ roots of equation does not reveal the k th root
- Alternatively, using algorithm 1a we can randomly choose $k-1$ roots and compute the k th root using

$$r_k = d \cdot (r_1 \cdot r_2 \cdot \dots \cdot r_{k-1})^{-1} \pmod{p}$$

- As seen, if d is not known and if r_i are randomly and uniformly chosen then all values of r_k remain equally probable

An alternate approach to partitioning

- Instead of storing all the k roots of the equation one may choose to store 1 root and $k-1$ coefficients of the equation on the network.
- Additional security may be added if computations are performed modulo a composite number, $n=pq$
- And using an encryption exponent to encrypt the data before computing the roots of the equation.

$$x^k + \sum_{i=1}^{k-1} a_{k-i} x^{k-i} + d^y \equiv 0 \pmod{n}$$

$$\prod_{i=1}^k r_i \equiv d^y \pmod{n}$$

Conclusions

- We have provided an efficient scheme for data partitioning
- (k,k) scheme
 - $O(k)$ – partitioning and reconstruction
- (k,n) scheme (in general)
 - $O(n \log^2 n)$ – partitioning
 - $O(k \log^2 k)$ – reconstruction
- (k,n) scheme where complexity moved from sensors to sink
 - $O(k)$ – partitioning
 - $O(k^3)$ – reconstruction
- Elimination of encryption keys

Future Work and Applications

- Addressing of data partitions
- Determine the optimal way to distribute the data pieces in a network
- Partition locations – static / adaptive relocation in case sensors move – so that the original sensor has access to its own data's partitions
- Algorithm can be used for key distribution as well
- Parallel transmission of data from sensor to sink using data partitioning



Thank you