NSC-JST workshop

Secure Decentralized Erasure Code based Networked Storage Systems with Multiple Functionalities

Wen-Guey Tzeng 曾文貴
(joint work with Hsiao-Ying Lin)
National Chiao Tung University
Distributed Networked Storage

Servers are decentralized:
no single central authority
Objectives

- **Data robustness**
  - Storage servers may fail over time (erasure error)

- **Data confidentiality**
  - The managers of storage servers may not be honest

- **Functionalities**
  - Data forwarding
  - System repairing
  - Integrity check
  - Keyword search
  - ...
Data Robustness

- Simple replication
  - Expensive in storage space
  - Solution: decentralized erasure code
Erasure Code

Message

\[ M_1 \quad M_2 \quad M_3 \quad \ldots \quad M_k \]

Codeword

\[ C_1 \quad C_2 \quad C_3 \quad \ldots \quad C_n \]

Encode

Any \( k \) out of \( n \) symbols

Decode

\[ M_1 \quad M_2 \quad M_3 \quad \ldots \quad M_k \]
Decentralized Erasure Code

Decentralized encoding
- each codeword symbol is independently computed
- linear combination with random coefficients

\[ C_1 = a_{1,1} M_1 + a_{1,2} M_2 + \ldots + a_{1,k} M_k \]

\[
G = \begin{bmatrix}
  a_{1,1} & a_{1,2} & \ldots & a_{1,k} \\
  a_{2,1} & a_{2,2} & \ldots & a_{2,k} \\
  a_{3,1} & a_{3,2} & \ldots & a_{3,k} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n,1} & a_{n,2} & \ldots & a_{n,k}
\end{bmatrix}
\]

\[
\begin{bmatrix} M_1 & M_2 & \ldots & M_k \end{bmatrix} \cdot G^T = \begin{bmatrix} C_1 & C_2 & \ldots & C_n \end{bmatrix}
\]
Decentralized Erasure Code

Decode
• solve a linear system with $k$ equations and $k$ unknowns

$$G = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} \\ a_{3,1} & a_{3,2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,k} \end{bmatrix}$$

$k$ columns

$$K = \begin{bmatrix} M_1^T \\ M_2^T \\ \vdots \\ M_k^T \end{bmatrix} = \begin{bmatrix} C_1^T \\ C_2^T \\ \vdots \\ C_k^T \end{bmatrix} (K^T)^{-1}$$

When $K$ is invertible

When $K$ is invertible

$M_1$ $M_2$ $M_3$ $\cdots$ $M_k$
Example

\[
[M_1 \ M_2] \circ \begin{bmatrix} a_{1,1} & a_{2,1} & 0 \\ 0 & a_{2,2} & a_{3,2} \end{bmatrix} =
\begin{bmatrix} M_1^{a_{1,1}} & M_1^{a_{2,1}} M_2^{a_{2,2}} & M_2^{a_{3,2}} \end{bmatrix} =
\begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}
\]
Decentralized Erasure Code for Storage

- Decentralized encoding (in servers)
- Robust against erasure errors
- Efficient in storage
- Light confidentiality
Previous Result

Assumption: random & independent distribution,
[2006] Result: $v = c \ln(k)$, where $c > 5 \frac{n}{k}$

$\Pr[\text{success retrieval}] > 1 - \frac{k}{p} - o(1)$

$n =$ number of storage servers

$v =$ number of copies

$k =$ number of message blocks

$k =$ number of queries

User
Decentralized Erasure Code for Storage

- Robust against erasure errors
- Decentralized encoding
- Efficient in storage
- Light confidentiality
- Stronger confidentiality is wanted
- More functionalities are desired
  - Data forwarding
  - System repairing
  - ...
Security Concerns

- **Storage in public**
  - Confidentiality of stored data
  - Solution: cryptographic encryption scheme

- **Key management (key server)**
  - Store secret key at single point is risky
  - Solution: key servers in private cloud
    - secret sharing
    - key share holder performs partial decryption
    - user recovers messages from partial decrypted data
# Our work

- **No central authority (decentralized)**
- **Data robustness**
- **Strong confidentiality**
- **Secure data forwarding**
- **Key management**
- **Repair mechanism**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>No central authority (decentralized)</td>
<td>Decentralized random linear code</td>
</tr>
<tr>
<td>Data robustness</td>
<td>Homomorphic public-key encryption</td>
</tr>
<tr>
<td>Strong confidentiality</td>
<td>Proxy re-encryption</td>
</tr>
<tr>
<td>Secure data forwarding</td>
<td>Partial decryption</td>
</tr>
<tr>
<td>Key management</td>
<td>Linear code repair</td>
</tr>
<tr>
<td>Repair mechanism</td>
<td></td>
</tr>
</tbody>
</table>
**Erasure coding over ciphertext**

- Homomorphic encryption (multiplicative)

\[
E_A(M_1) \otimes E_A(M_2) = E_A(M_1 M_2)
\]

\[
E_A(M_1)^a \otimes E_A(M_2)^b = E_A(M_1^a M_2^b)
\]
Proxy Re-encryption

- Required properties
  - Support decentralized/secure erasure coding
  - Support decentralized partial decryption
- We construct one satisfying all. It is
  - Based on bilinear map
  - Multiplicatively homomorphic
Decentralized Partial Decryption

storage servers

\[ E_A(M_1^{a_{1,1}}), a_{1,1}, 0 \]
\[ E_A(M_1^{a_{2,1}} \cdot M_2^{a_{2,2}}), a_{2,1}, a_{2,2} \]
\[ E_A(M_2^{a_{3,2}}), 0, a_{3,2} \]

key servers (with key shares)

\[ M_1, M_2 \]
System Overview

$t$-out-of-$m$ secret sharing

$n$ storage servers

$m$ key servers

Encryption key: $PK_A$

Decryption key: $SK_A$
Storing Process

Coding over encrypted messages

$n$ storage servers $m$ key servers

$E_A(M_1)$
$E_A(M_2)$
...
$E_A(M_k)$

$E_A(c_1),a_{1,1},...,a_{1,k}$
$E_A(c_2),a_{2,1},...,a_{2,k}$
$E_A(c_n),a_{n,1},...,a_{n,k}$
Forwarding Process

Compute and send re-encryption key

\[ RK_{A \rightarrow B} \]

ReEnc

\[ \widetilde{E}_A(c_1), a_{1,1}, \ldots, a_{1,k} \]
\[ \widetilde{E}_B(c_1), a_{1,1}, \ldots, a_{1,k} \]
\[ \widetilde{E}_A(c_2), a_{2,1}, \ldots, a_{2,k} \]
\[ \widetilde{E}_B(c_2), a_{2,1}, \ldots, a_{2,k} \]
\[ \widetilde{E}_A(c_n), a_{n,1}, \ldots, a_{n,k} \]
\[ \widetilde{E}_B(c_n), a_{n,1}, \ldots, a_{n,k} \]

\( m \) key servers
Retrieval Process - Owner

Decentralized partial decryption

$n$ storage servers

$E_{A(c_1)}, a_{1,1}, ..., a_{1,k}$

$E_{A(c_2)}, a_{2,1}, ..., a_{2,k}$

$E_{A(c_n)}, a_{n,1}, ..., a_{n,k}$

$m$ key servers

$sk_{A,1}$

$sk_{A,2}$

$sk_{A,m}$

$U$
Retrieval Process - Owner

Combine partial decryption and decoding

$n$ storage servers

$E_A(c_1), a_{1,1}, \ldots, a_{1,k}$

$E_A(c_2), a_{2,1}, \ldots, a_{2,k}$

$E_A(c_n), a_{n,1}, \ldots, a_{n,k}$

$m$ key servers

$sk_{A,1}$

$sk_{A,2}$

$sk_{A,m}$

At least $t$ key servers response

$M_1, M_2, \ldots, M_k$
Success Retrieval

Conditions
1. #SSs chosen by KSs is at least $k$
2. $\det(\text{submatrix}) \not\equiv 0 \mod p$ $\rightarrow$ $\det(\text{submatrix}) \not\equiv 0$
   $\rightarrow$ $\det(\text{submatrix}) \not\equiv rp$

Observations
- $\det(\text{submatrix}) \not\equiv 0$ iff a perfect matching
Parameters

For $n = ak^c, m \geq t \geq k, a > \sqrt{2}, v = bk^{c-1} \ln k, c \geq 3/2$

$u = 2, b > 5a, \Pr[\text{success retrieval}] > 1 - k/p - o(1)$

By combinatorial bound:

$\Pr[\#SS < k] \leq C^n_{k-1} \left(1/n\right)^{uk} = o(1)$

By Hall’s Theorem:

$\Pr[\det = 0 \mid \#SS \geq k] = \Pr[\text{no perfect matching}] = o(1)$

By Schwartz-Zippel Theorem (for random coefficients):

$\Pr[\det = rp \text{ for some integer } r \mid \#SS \geq k, \det \neq 0] \leq k/p$
Repair Issue

Maintain data robustness against server failure

Repair Mechanism

Any k out of n servers

Decoding

\[ \begin{align*}
M_1 & \quad M_2 & \quad M_3 & \quad \ldots & \quad M_k \\
\end{align*} \]
Repair Mechanisms

- Straightforward solution:
  - Reconstruct the original message and encode it again
  - Need to query \( k \) old servers
- Another approach
  - Generate a missing symbol by combining \( q \) available symbols from old servers
  - Objective: less repair bandwidth and storage cost
  - Question: can \( q \) be less than \( k \)?
Our Repair Mechanism

A queried server sends the symbol (with coefficients)

A new server independently queries q servers

A new server encodes received symbols as one (new symbol)

\[
\begin{align*}
  a & : C_3, (a_{3,1}, a_{3,2}, \ldots, a_{3,k}) \\
  b & : C_4, (a_{4,1}, a_{4,2}, \ldots, a_{4,k}) \\
\end{align*}
\]

\[
C_3^a \otimes C_4^b,
\]

\[
(aa_{3,1} + ba_{4,1}, aa_{3,2} + ba_{4,2}, \ldots, aa_{3,k} + ba_{4,k})
\]

#Old servers = \(\alpha n\)

#New servers = \((1-\alpha) n\)
About q

- When q is small, a new server gets less information → K may not have full rank (not invertible) → decrease Pr[ successful message retrieval ]

- Can q < k?

- Any k out of n symbols

- Need an invertible K

- \#old servers = \( \alpha n \)

- \#New servers = (1-\( \alpha \)) n

\[
G = \begin{bmatrix}
\end{bmatrix} \quad K = \begin{bmatrix}
\end{bmatrix}
\]
Main Result

There are $k^d$ old servers. $(1-\alpha)n$ new servers join the system, where

$$n = ak^c, \quad a > \sqrt{2}, \quad c \geq 1, \quad \alpha n = k^d, \quad d > 1, \quad \alpha < 1$$

Let $q$ be set s.t.

$$q \geq \min\{k, \max\left\{\frac{2k}{(d-1)\ln k}, \frac{k}{(d-1)\ln k} + \frac{d}{d-1}\right\}\}$$

After the system is repaired,

$$\Pr[\text{successful message retrieval}] \geq 1 - \frac{2k}{p} - o(1)$$

“q can be less than k”

The bound on q is related to k and d
Numerical Results

- Bring \( k \) and \( d \) to find the smallest \( q \)
Summary

- Decentralized networked storage system
  - Data robustness
  - Strong data confidentiality
  - Key management
  - Multiple functionalities
    - Secure data forwarding
    - Repair mechanism
    - Integrity check (not in this talk)
Future Work

- Data robustness against faulty errors
  - Detect/correct when stored data are altered
  - Support efficient coding operations
- Different repair model
  - Mutual communication among new servers
- More functionality
  - Support decentralized integrity check
  - Support keyword search