On some data processing problems arising in the distributed storage systems

Vitaly Skachek

Joint works with Helger Lipmaa and with Michael Rabbat

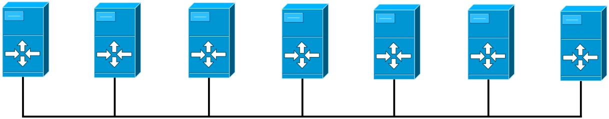
COST Action IC1104 Meeting, Palmela, Portugal
18 September 2014

Distributed Storage Systems

- Enormous amounts of data are stored in a huge number of servers.
- Occasionally servers fail.
- Failed server is replaced and the data has to be copied to the new server.

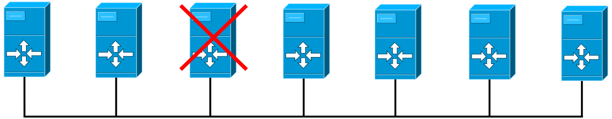
Distributed Storage Systems

- Enormous amounts of data are stored in a huge number of servers.
- Occasionally servers fail.
- Failed server is replaced and the data has to be copied to the new server.



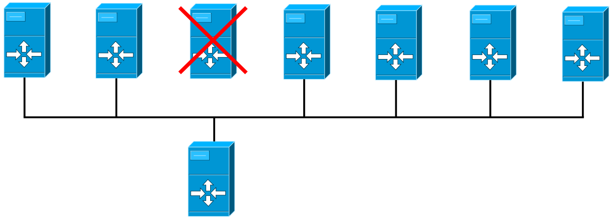
Distributed Storage Systems

- Enormous amounts of data are stored in a huge number of servers.
- Occasionally servers fail.
- Failed server is replaced and the data has to be copied to the new server.



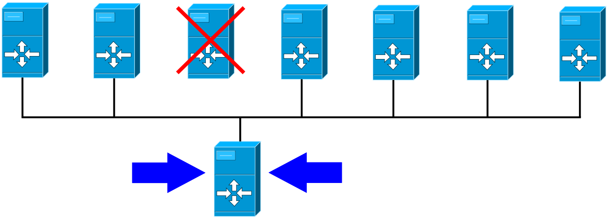
Distributed Storage Systems

- Enormous amounts of data are stored in a huge number of servers.
- Occasionally servers fail.
- Failed server is replaced and the data has to be copied to the new server.



Distributed Storage Systems

- Enormous amounts of data are stored in a huge number of servers.
- Occasionally servers fail.
- Failed server is replaced and the data has to be copied to the new server.



Example: EvenOdd Code

In the context of disk storage: [Blaum, Brady, Bruck, Menon 1995].

Example

$$\begin{array}{c||c||c||c} X_1 & Y_1 & X_1 + Y_1 & X_1 + Y_2 \\ X_2 & Y_2 & X_2 + Y_2 & X_2 + Y_1 \end{array}$$

All the information can be recovered by using any two out of four nodes.

Types of Repair

- Exact repair
- Functional repair
- Exact repair of the systematic part

Functional Repair

- The number of information blocks: M
- The number of information nodes: n
- The total number of active nodes: N
- Number of stored bits per node: α
- Maximal number of nodes used in repair: m
- Number of bits read from each node: t
- Total repair bandwidth: $\gamma = m \cdot t$.

Functional Repair

- The number of information blocks: M
- The number of information nodes: n
- The total number of active nodes: N
- Number of stored bits per node: α
- Maximal number of nodes used in repair: m
- Number of bits read from each node: t
- Total repair bandwidth: $\gamma = m \cdot t$.

Example

$$\begin{array}{c||c||c||c} X_1 & Y_1 & X_1 + Y_1 & X_1 + Y_2 \\ X_2 & Y_2 & X_2 + Y_2 & X_2 + Y_1 \end{array}$$

Here: $N = 4$, $n = 2$, $M = 4$, $m = 2$, $t = 2$ blocks, $\gamma = 4$ blocks.

[Dimakis, Godfrey, Wu, Wainwright, Ramchandran 2008]

Theorem

The following point is feasible:

$$\alpha \geq \begin{cases} \frac{M}{n} & \gamma \in [f(0), +\infty) \\ \frac{M-g(i)\gamma}{n-i} & \gamma \in [f(i), f(i-1)) \end{cases},$$

where

$$f(i) \triangleq \frac{2Mm}{(2n-i-1)i + 2n(m-n+1)}$$

$$g(i) \triangleq \frac{(2m-2n+i+1)i}{2m},$$

and $m < N - 1$.

Special Cases

- **MSR:** Minimum storage regenerating codes.
- **MBR:** Minimum bandwidth regenerating codes.

Special Cases

- **MSR:** Minimum storage regenerating codes.
- **MBR:** Minimum bandwidth regenerating codes.

MSR codes

$$(\alpha, \gamma) = \left(\frac{M}{n}, \frac{M}{n} \cdot \frac{N-1}{N-n} \right) .$$

- **MSR:** Minimum storage regenerating codes.
- **MBR:** Minimum bandwidth regenerating codes.

MSR codes

$$(\alpha, \gamma) = \left(\frac{M}{n}, \frac{M}{n} \cdot \frac{N-1}{N-n} \right) .$$

MBR codes

$$(\alpha, \gamma) = \left(\frac{M}{n} \cdot \frac{2N-2}{2N-n-1}, \frac{M}{n} \cdot \frac{2N-2}{2N-n-1} \right) .$$

Code Locality

[Gopalan, Huang, Simitci, Yekhanin 2012]

Code Locality

[Gopalan, Huang, Simitci, Yekhanin 2012]

Definition

Let $[n, k, d]_q$ be a linear code $\mathcal{C}$ over $\mathbb{F}_q$. We say that the $\mathcal{C}$ has locality r, if the value of any symbol in $\mathcal{C}$ can be recovered by accessing some r other coordinates of $\mathcal{C}$.

[Gopalan, Huang, Simitci, Yekhanin 2012]

Definition

Let $[n, k, d]_q$ be a linear code $\mathcal{C}$ over $\mathbb{F}_q$. We say that the $\mathcal{C}$ has locality r, if the value of any symbol in $\mathcal{C}$ can be recovered by accessing some r other coordinates of $\mathcal{C}$.

Bound

The following connection holds:

$$n - k \geq \left\lceil \frac{k}{r} \right\rceil + d - 2 \, .$$

The Pyramid codes are shown to achieve this bound.

Code Availability

[Rawat, Papailiopoulos, Dimakis, Vishwanath 2014]

Code Availability

[Rawat, Papailiopoulos, Dimakis, Vishwanath 2014]

Definition

Let $[n, k, d]_q$ be a linear code $\mathcal{C}$ over $\mathbb{F}_q$. We say that the $\mathcal{C}$ has (r, s)-availability, if the value of any symbol in $\mathcal{C}$ can be recovered by accessing s disjoint groups of other symbols, each of size at most r.

Code Availability

[Rawat, Papailiopoulos, Dimakis, Vishwanath 2014]

Definition

Let $[n, k, d]_q$ be a linear code $\mathcal{C}$ over $\mathbb{F}_q$. We say that the $\mathcal{C}$ has (r, s)-availability, if the value of any symbol in $\mathcal{C}$ can be recovered by accessing s disjoint groups of other symbols, each of size at most r.

Bound

The following connection holds:

$$n - k \geq \left\lceil \frac{ks}{r} \right\rceil + d - s - 2 \,.$$

There are explicit constructions of codes that achieve this bound for a variety of parameters.

Batch Codes

- Proposed in [Ishai, Kushilevitz, Ostrovsky, Sahai 2004].
- Can be used in:
  - Load balancing.
  - Private information retrieval.
  - Distributed storage systems.

Batch Codes

- Proposed in [Ishai, Kushilevitz, Ostrovsky, Sahai 2004].
- Can be used in:
  - Load balancing.
  - Private information retrieval.
  - Distributed storage systems.

Constructions:

- [Ishai *et al.* 2004]: algebraic, expander graphs, subsets, RM codes, locally-decodable codes

Prior Art

Design-based constructions and bounds:

- [Stinson, Wei, Paterson 2009]
- [Brualdi, Kiernan, Meyer, Schroeder 2010]
- [Bujtas, Tuza 2011]
- [Bhattacharya, Ruj, Roy 2012]
- [Silberstein, Gal 2013]

Prior Art

Design-based constructions and bounds:

- [Stinson, Wei, Paterson 2009]
- [Brualdi, Kiernan, Meyer, Schroeder 2010]
- [Bujtas, Tuza 2011]
- [Bhattacharya, Ruj, Roy 2012]
- [Silberstein, Gal 2013]

Application to distributed storage:

- [Rawat, Papailiopoulos, Dimakis, Vishwanath 2014]
- [Silberstein 2014]

Batch Codes

Definition [Ishai *et al.* 2004]

$\mathcal{C}$ is an $(n, N, m, M, t)_\Sigma$ batch code over Σ if it encodes any string $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \Sigma^n$ into M strings (buckets) of total length N over Σ, namely $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_M$, such that for each m-tuple (batch) of (not neccessarily distinct) indices $i_1, i_2, \cdots, i_m \in [n]$, the symbols $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ can be retrieved by m users, respectively, by reading $\leq t$ symbols from each bucket, such that x_{i_ℓ} is recovered from the symbols read by the ℓ-th user alone.

Batch Codes

Definition [Ishai *et al.* 2004]

$\mathcal{C}$ is an $(n, N, m, M, t)_\Sigma$ batch code over Σ if it encodes any string $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \Sigma^n$ into M strings (buckets) of total length N over Σ, namely $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_M$, such that for each m-tuple (batch) of (not neccessarily distinct) indices $i_1, i_2, \cdots, i_m \in [n]$, the symbols $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ can be retrieved by m users, respectively, by reading $\leq t$ symbols from each bucket, such that x_{i_ℓ} is recovered from the symbols read by the ℓ-th user alone.

Definition

If $t = 1$, then we use notation $(n, N, m, M)_\Sigma$ for it. Only one symbol is read from each bucket.

Batch Codes (cont.)

Definition

An $(n, N, m, M, t)_q$ batch code is *linear*, if every symbol in every bucket is a linear combination of original symbols.

Batch Codes (cont.)

Definition

An $(n, N, m, M, t)_q$ batch code is *linear*, if every symbol in every bucket is a linear combination of original symbols.

In what follows, consider *linear codes* with $t = 1$ and $N = M$: each encoded bucket contains just one symbol in $\mathbb{F}_q$.

Linear Batch Codes: Our Settings

For simplicity we refer to a linear $(n, N = M, m, M)_q$ batch code as $[M, n, m]_q$ batch code.

Linear Batch Codes: Our Settings

For simplicity we refer to a linear $(n, N = M, m, M)_q$ batch code as $[M, n, m]_q$ batch code.

- Let $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ be an information string.
- Let $\mathbf{y} = (y_1, y_2, \cdots, y_M)$ be an encoding of $\mathbf{x}$.
- Each encoded symbol y_i, $i \in [M]$, is written as $y_i = \sum_{j=1}^{n} g_{j,i} x_j$.
- Form the matrix $\mathbf{G}$:

$$\mathbf{G} = \left(g_{j,i} \right)_{j \in [n], i \in [M]} ;$$

the encoding is $\mathbf{y} = \mathbf{x}\mathbf{G}$.

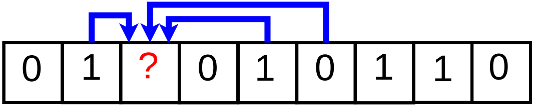
Code Comparison

Locally repairable codes, codes with locality.

0	1	?	0	1	0	1	1	0

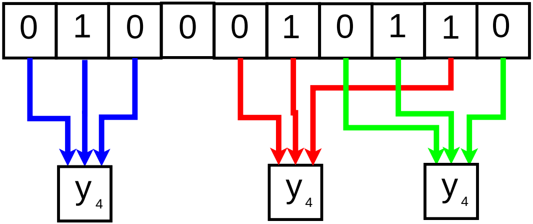
Code Comparison

Locally repairable codes, codes with locality.

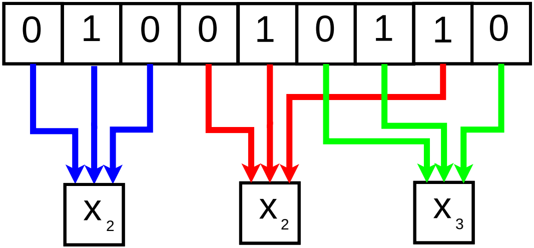


Code Comparison

Codes with locality and availability.



Batch codes.



Theorem

Let $\mathcal{C}$ be an $[M, n, m]_q$ batch code. It is possible to retrieve $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ simultaneously if and only if there exist m non-intersecting sets $T_1, T_2, \cdots, T_m$ of indices of columns in $\mathbf{G}$, and for T_r there exists a linear combination of columns of $\mathbf{G}$ indexed by that set, which equals to the column vector $\mathbf{e}_{i_r}^T$, for all $r \in [m]$.

Retrieval

Theorem

Let $\mathcal{C}$ be an $[M, n, m]_q$ batch code. It is possible to retrieve $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ simultaneously if and only if there exist m non-intersecting sets $T_1, T_2, \cdots, T_m$ of indices of columns in $\mathbf{G}$, and for T_r there exists a linear combination of columns of $\mathbf{G}$ indexed by that set, which equals to the column vector $\mathbf{e}_{i_r}^T$, for all $r \in [m]$.

Example

[Ishai *et al.* 2004] Consider the following linear binary batch code $\mathcal{C}$ whose 4×9 generator matrix is given by

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} .$$

Retrieval (cont.)

Example

Let $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = \mathbf{x}\mathbf{G}$.
Assume that we want to retrieve the values of (x_1, x_1, x_2, x_2). We can retrieve (x_1, x_1, x_2, x_2) from the following set of equations:

$$\left\{ \begin{array}{lcl} x_1 & = & y_1 \\ x_1 & = & y_2 + y_3 \\ x_2 & = & y_5 + y_8 \\ x_2 & = & y_4 + y_6 + y_7 + y_9 \end{array} \right. .$$

It is straightforward to verify that any 4-tuple $(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4})$, where $i_1, i_2, i_3, i_4 \in [4]$, can be retrieved by using columns indexed by some four non-intersecting sets of indices in [9]. Therefore, the code $\mathcal{C}$ is a $[9, 4, 4]_2$ batch code.

Theorem

Let $\mathcal{C}$ be an $[M, n, m]_2$ batch code $\mathcal{C}$ over $\mathbb{F}_2$. Then, $\mathbf{G}$ *is a generator matrix of the classical error-correcting $[M, n, \geq m]_2$ code.*

Theorem

Let $\mathcal{C}$ be an $[M, n, m]_2$ batch code $\mathcal{C}$ over $\mathbb{F}_2$. Then, $\mathbf{G}$ is a generator matrix of the classical error-correcting $[M, n, \geq m]_2$ code.

Example

The converse is not true. For example, take $\mathbf{G}$ to be a generator matrix of the classical $[4, 3, 2]_2$ ECC as follows:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} .$$

Let $\mathbf{x} = (x_1, x_2, x_3)$. Then, it is impossible to retrieve (x_2, x_3).

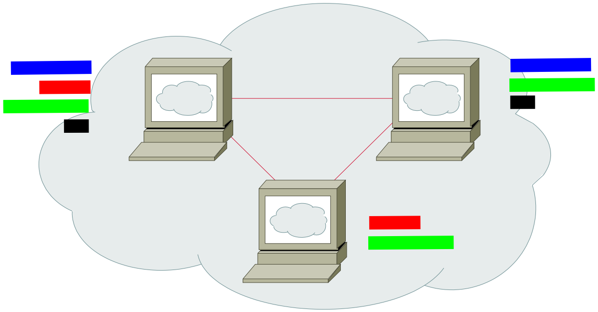
Bounds on the Parameters

- Various well-studied properties of linear ECCs, such as MacWilliams identities, apply also to linear batch codes (for $t = 1$, $M = N$ and $q = 2$).

Bounds on the Parameters

- Various well-studied properties of linear ECCs, such as MacWilliams identities, apply also to linear batch codes (for $t = 1$, $M = N$ and $q = 2$).
- A variety of bounds on the parameters of ECCs, such as sphere-packing bound, Plotkin bound, Griesmer bound, Elias-Bassalygo bound, McEliece-Rodemich-Rumsey-Welch bound apply to the parameters of $[M, n, m]_2$ batch codes.

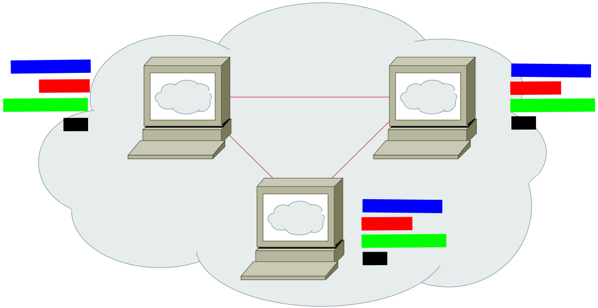
File Synchronization Problem



Before synchronization:

- **User** A**:** f_1, f_2, f_3 and f_4.
- **User** B**:** f_1, f_3, f_4.
- **User** C**:** f_2, f_3.

File Synchronization Problem



Before synchronization:

- **User** A**:** f_1, f_2, f_3 and f_4.
- **User** B**:** f_1, f_3, f_4.
- **User** C**:** f_2, f_3.

After synchronization:

- **Users** A, B, C**:** f_1, f_2, f_3 and f_4.

Prior Art

- Mitzenmacher and Varghese '2012

Prior Art

- Mitzenmacher and Varghese '2012

Parameters to Consider

- Communication cost $\textsc{Communication}(\mathcal{A})$: the worst case number of bits sent between the devices;
- Computational complexity $\textsc{Computation}(\mathcal{A})$: the worst case number of operations performed at each device;
- Time $\textsc{Time}(\mathcal{A})$: the length of the largest chain of messages in the communication protocol.

Prior Art

- Mitzenmacher and Varghese '2012

Parameters to Consider

- Communication cost $\textsc{Communication}(\mathcal{A})$: the worst case number of bits sent between the devices;
- Computational complexity $\textsc{Computation}(\mathcal{A})$: the worst case number of operations performed at each device;
- Time $\textsc{Time}(\mathcal{A})$: the length of the largest chain of messages in the communication protocol.

- k is the total number of objects in possession of A and B;
- d is the number of objects possessed by only one user;
- u is the size of the space where the objects are taken from.

Prior Art

- Minsky, Trachtenberg and Zippel '2003: characteristic polynomials.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(d^3)$,
  $\textsc{Time}(\mathcal{A}) = O(\log k)$

Prior Art

- Minsky, Trachtenberg and Zippel '2003: characteristic polynomials.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(d^3)$,
  $\textsc{Time}(\mathcal{A}) = O(\log k)$
- Goodrich and Mitzenmacher '2011: invertible Bloom filters.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(d)$,
  $\textsc{Time}(\mathcal{A}) = 3$

Prior Art

- Minsky, Trachtenberg and Zippel '2003: characteristic polynomials.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(d^3)$,
  $\textsc{Time}(\mathcal{A}) = O(\log k)$
- Goodrich and Mitzenmacher '2011: invertible Bloom filters.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(d)$,
  $\textsc{Time}(\mathcal{A}) = 3$
- Mitzenmacher and Varghese '2012: Biff codes.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(k \log u)$,
  $\textsc{Time}(\mathcal{A}) = 3$.

Prior Art

- Minsky, Trachtenberg and Zippel '2003: characteristic polynomials.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(d^3)$,
  $\textsc{Time}(\mathcal{A}) = O(\log k)$
- Goodrich and Mitzenmacher '2011: invertible Bloom filters.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(d)$,
  $\textsc{Time}(\mathcal{A}) = 3$
- Mitzenmacher and Varghese '2012: Biff codes.
  $\textsc{Communication}(\mathcal{A}) = O(d \log u)$,
  $\textsc{Computation}(\mathcal{A}) = O(k \log u)$,
  $\textsc{Time}(\mathcal{A}) = 3$.

with high probability.

Subspace Synchronization for Two Users

- Finite field $\mathbb{F}$ with q elements.
- Two users w and v.
- The users own vector spaces $U \subseteq \mathbb{F}^n$ and $V \subseteq \mathbb{F}^n$, respectively.
- Goal: w and v own vector space $U + V$.

Subspace Synchronization for Two Users

- Finite field $\mathbb{F}$ with q elements.
- Two users w and v.
- The users own vector spaces $U \subseteq \mathbb{F}^n$ and $V \subseteq \mathbb{F}^n$, respectively.
- Goal: w and v own vector space $U + V$.

Algorithm $\mathcal{A}$

(1) The user w draws a nonzero vector $\mathbf{x} \in U$ randomly and uniformly and communicates it to v.

(2) The node v checks if $\mathbf{x} \in V$. If not, performs

$$V \leftarrow V \oplus \langle \mathbf{x} \rangle \ .$$

(3) Repeat (1)-(2) for $\Theta(d)$ rounds.

(4) Switch the roles of w and v.

Subspace Synchronization for Two Users (cont.)

With high probability,
$\textsc{Communication}(\mathcal{A}) = O(d \cdot n \log q)$,
$\textsc{Computation}(\mathcal{A}) = O(k^2 \cdot n)$,
$\textsc{Time}(\mathcal{A}) = 2$.

With high probability,
$\textsc{Communication}(\mathcal{A}) = O(d \cdot n \log q)$,
$\textsc{Computation}(\mathcal{A}) = O(k^2 \cdot n)$,
$\textsc{Time}(\mathcal{A}) = 2$.

The scheme is easily extendable extendable to networks with many users.

Consider a classical $[n, k, d]$-linear code $\mathcal{C}$ over the finite field $\mathbb{F} = \mathbb{F}_q$, such that $n \geq 2^m$ for some integer $m > 0$. (For example, RS code with $n + 1 = k + d$). Let the $(n - k) \times n$ parity-check matrix of $\mathcal{C}$ be

$$H = [\mathbf{h}_1 \mid \mathbf{h}_2 \mid \cdots \mid \mathbf{h}_n] ,$$

$\mathbf{h}_i$'s are the columns of H.

Consider a classical $[n, k, d]$-linear code $\mathcal{C}$ over the finite field $\mathbb{F} = \mathbb{F}_q$, such that $n \geq 2^m$ for some integer $m > 0$. (For example, RS code with $n + 1 = k + d$). Let the $(n - k) \times n$ parity-check matrix of $\mathcal{C}$ be

$$H = [\mathbf{h}_1 \mid \mathbf{h}_2 \mid \cdots \mid \mathbf{h}_n] ,$$

$\mathbf{h}_i$'s are the columns of H.

With every vector $\mathbf{x} \in \{0, 1\}^m$ associate a unique integer index $\phi(\mathbf{x}) \in [n]$. If $\mathbf{x}_1 \neq \mathbf{x}_2$, we have $\phi(\mathbf{x}_1) \neq \phi(\mathbf{x}_2)$. Assume that $O = \{\mathbf{x}_i\}_{i \in S}$ is a collection of objects for some $S \subseteq [n]$.

Using Reed-Solomon Codes

Consider a classical $[\mathrm{n},\mathrm{k},\mathrm{d}]$-linear code $\mathcal{C}$ over the finite field $\mathbb{F}=\mathbb{F}_q$, such that $\mathrm{n} \geq 2^m$ for some integer $m>0$. (For example, RS code with $\mathrm{n}+1=\mathrm{k}+\mathrm{d}$). Let the $(\mathrm{n}-\mathrm{k})\times \mathrm{n}$ parity-check matrix of $\mathcal{C}$ be

$$H = [\mathbf{h}_1 \mid \mathbf{h}_2 \mid \cdots \mid \mathbf{h}_\mathrm{n}] \ ,$$

$\mathbf{h}_i$'s are the columns of H.

With every vector $\mathbf{x} \in \{0,1\}^m$ associate a unique integer index $\phi(\mathbf{x}) \in [\mathrm{n}]$. If $\mathbf{x}_1 \neq \mathbf{x}_2$, we have $\phi(\mathbf{x}_1) \neq \phi(\mathbf{x}_2)$. Assume that $O=\{\mathbf{x}_i\}_{i\in S}$ is a collection of objects for some $S \subseteq [\mathrm{n}]$.

Represent the collection O by the vector space

$$\Phi(O) \triangleq \left\langle \mathbf{h}_{\phi(\mathbf{x})} \right\rangle_{\mathbf{x}\in O} .$$

Using Reed-Solomon Codes and Subspaces

In order to perform reconciliation of two sets of objects, O_1 and O_2, the corresponding vector spaces V_1 and V_2 are constructed, such that $V_i = \Phi(O_i)$ for $i = 1, 2$. Then the synchronization algorithm $\mathcal{A}$ is applied to V_1 and V_2.

Using Reed-Solomon Codes and Subspaces

In order to perform reconciliation of two sets of objects, O_1 and O_2, the corresponding vector spaces V_1 and V_2 are constructed, such that $V_i = \Phi(O_i)$ for $i = 1, 2$. Then the synchronization algorithm $\mathcal{A}$ is applied to V_1 and V_2.

Performance

$\textsc{Communication}(\mathcal{A}) = O(\mathrm{d}^2 m) = O(\mathrm{d}^2 \log u)$,
$\textsc{Computation}(\mathcal{A}) = O(\mathrm{d}^2 \cdot u)$.
$\textsc{Time}(\mathcal{A}) = 2$

Network Coding with Hashing Approach

- Denote by $O_A = \{\mathbf{x}_i \in \mathbb{F}^n\}_{i \in \mathcal{X}_A}$ and $O_B = \{\mathbf{x}_i \in \mathbb{F}^n\}_{i \in \mathcal{X}_B}$ the set of objects, which are unique to A and to B, respectively.
- $O_C = \{\mathbf{x}_i \in \mathbb{F}^n\}_{i \in \mathcal{X}_O}$ the set of objects which are possessed by both A and B.
- Let $s = |\mathcal{X}_A|$ and $\tau = |\mathcal{X}_A \cup \mathcal{X}_O|$.
- As before, let $d = |\mathcal{X}_A \cup \mathcal{X}_B|$ be the number of different files for A and B.
- Assume that s, or a tight upper bound on it, is known to both A and B.

User A

- User A creates s arbitrary linear combinations of the form

$$\mathbf{y}_j = \sum_{i \in \mathcal{X}_A \cup \mathcal{X}_O} \alpha_{j,i} \mathbf{x}_i \ , \ j \in [s] \ ,$$

- The protocol uses a hash function $\mathcal{H} : \mathbb{F}^n \to \mathbb{K}$, where $\mathbb{K}$ is the finite set of possible keys.
- User A applies $\mathcal{H}$ to $\mathbf{x}_i$ for all $i \in \mathcal{X}_A \cup \mathcal{X}_O$ to produce hash values $\mathcal{H}(\mathbf{x}_i)$ for all i.
- These values are transmitted to B.

User A

- User A creates s arbitrary linear combinations of the form

$$\mathbf{y}_j = \sum_{i \in \mathcal{X}_A \cup \mathcal{X}_O} \alpha_{j,i} \mathbf{x}_i \ , \ j \in [s] \ ,$$

- The protocol uses a hash function $\mathcal{H} : \mathbb{F}^n \to \mathbb{K}$, where $\mathbb{K}$ is the finite set of possible keys.
- User A applies $\mathcal{H}$ to $\mathbf{x}_i$ for all $i \in \mathcal{X}_A \cup \mathcal{X}_O$ to produce hash values $\mathcal{H}(\mathbf{x}_i)$ for all i.
- These values are transmitted to B.

A transmits to B the following data:

- the header $\mathbf{h}$, which contains the sorted list of values $\mathcal{H}(\mathbf{x}_i)$, $i \in \mathcal{X}_A \cup \mathcal{X}_O$;
- for all $j \in [s]$, the vector pairs $(\boldsymbol{\alpha}_j, \mathbf{y}_j)$.

Let $\mathbf{X}$ be a $\tau \times n$ matrix over $\mathbb{F}$, whose rows are all vectors $\mathbf{x}_i$ indexed by $[\tau]$. Similarly, let $\mathbf{Y}$ be a $s \times n$ matrix, whose rows are vectors $\mathbf{y}_i$ for all $i \in [s]$. Denote

$$\mathbf{A} = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,\tau} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,\tau} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{s,1} & \alpha_{s,2} & \cdots & \alpha_{s,\tau} \end{pmatrix} .$$

The transmitted vector pairs can be viewed as the rows of the matrix

$$\mathbf{A} \cdot [\, \mathbf{I}_\tau \mid \mathbf{X} \,] = [\, \mathbf{A} \mid \mathbf{Y} \,] \ ,$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_\tau \end{bmatrix} \qquad \text{and} \qquad \mathbf{Y} = \mathbf{A}\mathbf{X} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_s \end{bmatrix} .$$

User B

1. Compute values of the function $\mathcal{H}$ applied to the vectors in its possession. By comparing these values to the values in the header $\mathbf{h}$, it finds the indices corresponding to elements in $\mathcal{X}_O$.
2. For each $j \in [s]$, subtract vectors $\sum_{i \in \mathcal{X}_O} \alpha_{j,i} \mathbf{x}_i$ from $\mathbf{y}_j$. Compute the resulting matrix with s rows:

$$\left[\, \tilde{\mathbf{A}} \mid \tilde{\mathbf{Y}} \, \right] \; ,$$

where rows of $\tilde{\mathbf{Y}}$ are the vectors

$$\tilde{\mathbf{y}}_j = \mathbf{y}_j - \sum_{i \in \mathcal{X}_O} \alpha_{j,i} \mathbf{x}_i \; ,$$

and $\tilde{\mathbf{A}}$ is an invertible $s \times s$ matrix obtained from $\mathbf{A}$ by removing the columns corresponding to the vectors indexed by $\mathcal{X}_O$.

Compute the matrix

$$\left[\ \mathbf{I} \mid \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{Y}}\ \right] = \left[\ \mathbf{I} \mid \tilde{\mathbf{X}}\ \right] ,$$

where, if there are no hashing collisions, $\tilde{\mathbf{X}}$ is exactly the matrix $\mathbf{X}$ having rows corresponding to the vectors indexed by $\mathcal{X}_A$.

Compute the matrix

$$\left[\mathbf{I} \mid \tilde{\mathbf{A}}^{-1} \tilde{\mathbf{Y}} \right] = \left[\mathbf{I} \mid \tilde{\mathbf{X}} \right] ,$$

where, if there are no hashing collisions, $\tilde{\mathbf{X}}$ is exactly the matrix $\mathbf{X}$ having rows corresponding to the vectors indexed by $\mathcal{X}_A$.

Peformance

$\textsc{Communication}(\mathcal{A}) = O(d \cdot n \log q)$
$\textsc{Computation}(\mathcal{A}) = O(k^2 \cdot n)$
If s is known, then $\textsc{Time}(\mathcal{A}) = 2$. If s is not known, then $\textsc{Time}(\mathcal{A}) = 3$.

Using a Pool of Hash Functions

- Large pool of different hash functions (known to both users).
- In each round, the hash function is selected randomly from the pool.
- User A sends to B the ID number of the selected hash function.

Using a Pool of Hash Functions

- Large pool of different hash functions (known to both users).
- In each round, the hash function is selected randomly from the pool.
- User A sends to B the ID number of the selected hash function.

Assume a collection $\mathbb{S}$ of k different files in $\{0,1\}^n$. Let $\mathbb{H}$ be a set of all functions $\mathcal{H} : \{0,1\}^n \to \mathbb{K}$, where $\mathbb{K}$ is the set of all possible keys. Assume that $k \ll |\mathbb{K}| \ll 2^n$.

Using a Pool of Hash Functions

- Large pool of different hash functions (known to both users).
- In each round, the hash function is selected randomly from the pool.
- User A sends to B the ID number of the selected hash function.

Assume a collection $\mathbb{S}$ of k different files in $\{0,1\}^n$. Let $\mathbb{H}$ be a set of all functions $\mathcal{H} : \{0,1\}^n \to \mathbb{K}$, where $\mathbb{K}$ is the set of all possible keys. Assume that $k \ll |\mathbb{K}| \ll 2^n$.

Theorem

If $\mathbb{K}$ is selected such that $|\mathbb{K}| > c \cdot (k-1)^2$ for some large constant $c > 0$, then the probability of success is at least $e^{-1/c}$.

Thank you!

Questions?