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
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.
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.
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.
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.
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

\[
\begin{align*}
X_1 & \parallel Y_1 & X_1 + Y_1 & \parallel X_1 + Y_2 \\
X_2 & \parallel Y_2 & X_2 + Y_2 & \parallel X_2 + Y_1
\end{align*}
\]

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
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: \( \alpha \)

Maximal number of nodes used in repair: \( m \)

Number of bits read from each node: \( t \)

Total repair bandwidth: \( \gamma = m \cdot t \).
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: $\alpha$

Maximal number of nodes used in repair: $m$

Number of bits read from each node: $t$

Total repair bandwidth: $\gamma = m \cdot t$.

Example

\[
\begin{align*}
X_1 & \parallel Y_1 & X_1 + Y_1 & \parallel X_1 + Y_2 \\
X_2 & \parallel Y_2 & X_2 + Y_2 & \parallel X_2 + Y_1
\end{align*}
\]

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

[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**

\[ (\alpha, \gamma) = \left( \frac{M}{n}, \frac{M}{n} \cdot \frac{N - 1}{N - n} \right) \]
Special Cases

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

**MSR codes**

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

**MBR codes**

\[ (\alpha, \gamma) = \left( \frac{M}{n} \cdot \frac{2N - 2}{2N - n - 1}, \frac{M}{n} \cdot \frac{2N - 2}{2N - n - 1} \right). \]
[Gopalan, Huang, Simitci, Yekhanin 2012]
[Gopalan, Huang, Simitci, Yekhanin 2012]

**Definition**

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

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.
[Rawat, Papailiopoulos, Dimakis, Vishwanath 2014]
[Rawat, Papailiopoulos, Dimakis, Vishwanath 2014]

**Definition**

Let $[n, k, d]_q$ be a linear code $C$ over $\mathbb{F}_q$. We say that the $C$ has $(r, s)$-availability, if the value of any symbol in $C$ can be recovered by accessing $s$ disjoint groups of other symbols, each of size at most $r$. 
[Rawat, Papailiopoulos, Dimakis, Vishwanath 2014]

**Definition**

Let $[n, k, d]_q$ be a linear code $C$ over $\mathbb{F}_q$. We say that the $C$ has $(r, s)$-availability, if the value of any symbol in $C$ 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]
**Definition [Ishai et al. 2004]**

$\mathcal{C}$ is an $(n, N, m, M, t)_{\Sigma}$ batch code over $\Sigma$ if it encodes any string $x = (x_1, x_2, \ldots, x_n) \in \Sigma^n$ into $M$ strings (buckets) of total length $N$ over $\Sigma$, namely $y_1, y_2, \ldots, y_M$, such that for each $m$-tuple (batch) of (not necessarily distinct) indices $i_1, i_2, \ldots, i_m \in [n]$, the symbols $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ can be retrieved by $m$ users, respectively, by reading $\leq t$ symbols from each bucket, such that $x_{i_\ell}$ is recovered from the symbols read by the $\ell$-th user alone.
Batch Codes

**Definition [Ishai et al. 2004]**

$\mathcal{C}$ is an $(n, N, m, M, t)_{\Sigma}$ batch code over $\Sigma$ 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 $\Sigma$, namely $y_1, y_2, \cdots, y_M$, such that for each $m$-tuple (batch) of (not necessarily 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 $x_{i_{\ell}}$ is recovered from the symbols read by the $\ell$-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.
An \((n, N, m, M, t)_q\) batch code is \textit{linear}, if every symbol in every bucket is a linear combination of original symbols.
An \((n, N, m, M, t)\) batch code is \textit{linear}, if every symbol in every bucket is a linear combination of original symbols.

In what follows, consider \textit{linear codes} with \(t = 1\) and \(N = M\): each encoded bucket contains just one symbol in \(\mathbb{F}_q\).
For simplicity we refer to a linear \((n, N = M, m, M)_q\) batch code as \([M, n, m]_q\) batch code.
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 \(\mathbf{y} = \mathbf{xG}\).
Locally repairable codes, codes with locality.

\[
\begin{array}{ccccccccc}
0 & 1 & ? & 0 & 1 & 0 & 1 & 1 & 0
\end{array}
\]
Locally repairable codes, codes with locality.
Codes with locality and availability.

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
\downarrow & & & & & \downarrow & & & & \downarrow \\
y_4 & & & & & y_4 & & & & y_4 \\
\end{array}
\]
Batch codes.
Theorem

Let $C$ 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 $G$, and for $T_r$ there exists a linear combination of columns of $G$ indexed by that set, which equals to the column vector $e_{i_r}^T$, for all $r \in [m]$. 

Vitaly Skachek

Problems in DSS
Theorem

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

Example

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

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

\[
\begin{align*}
    x_1 &= y_1 \\
    x_1 &= y_2 + y_3 \\
    x_2 &= y_5 + y_8 \\
    x_2 &= y_4 + y_6 + y_7 + y_9
\end{align*}
\]

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

**Theorem**

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

**Example**

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

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

Let $x = (x_1, x_2, x_3)$. Then, it is impossible to retrieve $(x_2, x_3)$.
Various well-studied properties of linear ECCs, such as MacWilliams identities, apply also to linear batch codes (for $t = 1$, $M = N$ and $q = 2$).
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.
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
• Mitzenmacher and Varghese ’2012

Parameters to Consider

• Communication cost $\text{COMMUNICATION}(\mathcal{A})$: the worst case number of bits sent between the devices;

• Computational complexity $\text{COMPUTATION}(\mathcal{A})$: the worst case number of operations performed at each device;

• Time $\text{TIME}(\mathcal{A})$: the length of the largest chain of messages in the communication protocol.
Mitzenmacher and Varghese ’2012

**Parameters to Consider**

- **Communication cost** $\text{COMMUNICATION}(\mathcal{A})$: the worst case number of bits sent between the devices;
- **Computational complexity** $\text{COMPUTATION}(\mathcal{A})$: the worst case number of operations performed at each device;
- **Time** $\text{TIME}(\mathcal{A})$: 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.
Minsky, Trachtenberg and Zippel '2003: characteristic polynomials.

\[
\text{COMMUNICATION}(A) = O(d \log u), \\
\text{COMPUTATION}(A) = O(d^3), \\
\text{TIME}(A) = O(\log k)
\]
Prior Art

- Minsky, Trachtenberg and Zippel ’2003: characteristic polynomials.
  \[
  \text{COMMUNICATION}(A) = O(d \log u), \\
  \text{COMPUTATION}(A) = O(d^3), \\
  \text{TIME}(A) = O(\log k)
  \]

  \[
  \text{COMMUNICATION}(A) = O(d \log u), \\
  \text{COMPUTATION}(A) = O(d), \\
  \text{TIME}(A) = 3
  \]
Prior Art

- Minsky, Trachtenberg and Zippel ’2003: characteristic polynomials.
  \[ \text{COMMUNICATION}(\mathcal{A}) = O(d \log u), \]
  \[ \text{COMPUTATION}(\mathcal{A}) = O(d^3), \]
  \[ \text{TIME}(\mathcal{A}) = O(\log k) \]

  \[ \text{COMMUNICATION}(\mathcal{A}) = O(d \log u), \]
  \[ \text{COMPUTATION}(\mathcal{A}) = O(d), \]
  \[ \text{TIME}(\mathcal{A}) = 3 \]

- Mitzenmacher and Varghese ’2012: Biff codes.
  \[ \text{COMMUNICATION}(\mathcal{A}) = O(d \log u), \]
  \[ \text{COMPUTATION}(\mathcal{A}) = O(k \log u), \]
  \[ \text{TIME}(\mathcal{A}) = 3. \]
Prior Art

- Minsky, Trachtenberg and Zippel ’2003: characteristic polynomials.
  \[
  \begin{align*}
  \text{COMMUNICATION}(\mathcal{A}) &= O(d \log u), \\
  \text{COMPUTATION}(\mathcal{A}) &= O(d^3), \\
  \text{TIME}(\mathcal{A}) &= O(\log k)
  \end{align*}
  \]

  \[
  \begin{align*}
  \text{COMMUNICATION}(\mathcal{A}) &= O(d \log u), \\
  \text{COMPUTATION}(\mathcal{A}) &= O(d), \\
  \text{TIME}(\mathcal{A}) &= 3
  \end{align*}
  \]

- Mitzenmacher and Varghese ’2012: Biff codes.
  \[
  \begin{align*}
  \text{COMMUNICATION}(\mathcal{A}) &= O(d \log u), \\
  \text{COMPUTATION}(\mathcal{A}) &= O(k \log u), \\
  \text{TIME}(\mathcal{A}) &= 3.
  \end{align*}
  \]

  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 $A$**

1. The user $w$ draws a nonzero vector $x \in U$ randomly and uniformly and communicates it to $v$.
2. The node $v$ checks if $x \in V$. If not, performs
   \[ V \leftarrow V \oplus \langle x \rangle. \]
3. Repeat (1)-(2) for $\Theta(d)$ rounds.
4. Switch the roles of $w$ and $v$.  

Vitaly Skachek  
Problems in DSS
With high probability,
\[ \text{COMMUNICATION}(A) = O(d \cdot n \log q), \]
\[ \text{COMPUTATION}(A) = O(k^2 \cdot n), \]
\[ \text{TIME}(A) = 2. \]
With high probability,

\( \text{COMMUNICATION}(\mathcal{A}) = O(d \cdot n \log q), \)
\( \text{COMPUTATION}(\mathcal{A}) = O(k^2 \cdot n), \)
\( \text{TIME}(\mathcal{A}) = 2. \)

The scheme is easily extendable to networks with many users.
Consider a classical \([n, k, d]\)-linear code \(C\) 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 \(C\) be

\[
H = [h_1 | h_2 | \cdots | h_n],
\]

\(h_i\)'s are the columns of \(H\).
Consider a classical \([n, k, d]\)-linear code \(C\) 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 \(C\) be
\[
H = [h_1 \mid h_2 \mid \cdots \mid h_n],
\]
where \(h_i\)'s are the columns of \(H\).

With every vector \(x \in \{0, 1\}^m\) associate a unique integer index \(\phi(x) \in [n]\). If \(x_1 \neq x_2\), we have \(\phi(x_1) \neq \phi(x_2)\). Assume that \(O = \{x_i\}_{i \in S}\) is a collection of objects for some \(S \subseteq [n]\).
Consider a classical \([n, k, d]\)-linear code \(C\) 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 \(C\) be

\[
H = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \end{bmatrix},
\]

\(h_i\)'s are the columns of \(H\).

With every vector \(x \in \{0, 1\}^m\) associate a unique integer index \(\phi(x) \in [n]\). If \(x_1 \neq x_2\), we have \(\phi(x_1) \neq \phi(x_2)\). Assume that \(O = \{x_i\}_{i \in S}\) is a collection of objects for some \(S \subseteq [n]\).

Represent the collection \(O\) by the vector space

\[
\Phi(O) \triangleq \left\langle h_{\phi(x)} \right\rangle_{x \in O}.
\]
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 $V_2$. 

Vitaly Skachek  
Problems in DSS
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**

\[
\text{\textsc{Communication}}(\mathcal{A}) = O(d^2 m) = O(d^2 \log u), \\
\text{\textsc{Computation}}(\mathcal{A}) = O(d^2 \cdot u), \\
\text{\textsc{Time}}(\mathcal{A}) = 2
\]
Denote by $O_A = \{x_i \in \mathbb{F}^n \}_{i \in \mathcal{X}_A}$ and $O_B = \{x_i \in \mathbb{F}^n \}_{i \in \mathcal{X}_B}$ the set of objects, which are unique to $A$ and to $B$, respectively.

$O_C = \{x_i \in \mathbb{F}^n \}_{i \in \mathcal{X}_O}$ 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 $B$. 

Vitaly Skachek

Problems in DSS
User A

- User A creates $s$ arbitrary linear combinations of the form

$$y_j = \sum_{i \in \mathcal{X}_A \cup \mathcal{X}_O} \alpha_{j,i}x_i, \ j \in [s],$$

- 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 $x_i$ for all $i \in \mathcal{X}_A \cup \mathcal{X}_O$ to produce hash values $\mathcal{H}(x_i)$ for all $i$.
- These values are transmitted to $B$. 

Vitaly Skachek
Problems in DSS
User A

- User A creates $s$ arbitrary linear combinations of the form

$$y_j = \sum_{i \in X_A \cup X_O} \alpha_{j,i} x_i, \ j \in [s],$$

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

$A$ transmits to $B$ the following data:

- the header $h$, which contains the sorted list of values $\mathcal{H}(x_i)$, $i \in X_A \cup X_O$;
- for all $j \in [s]$, the vector pairs $(\alpha_j, y_j)$. 
Let $X$ be a $\tau \times n$ matrix over $\mathbb{F}$, whose rows are all vectors $x_i$ indexed by $[\tau]$. Similarly, let $Y$ be a $s \times n$ matrix, whose rows are vectors $y_i$ for all $i \in [s]$. Denote

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

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

$$A \cdot [I_\tau \mid X] = [A \mid Y],$$

where

$$X = \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_\tau
\end{bmatrix}$$

and

$$Y = AX = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_s
\end{bmatrix}. $$
Compute values of the function $H$ 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 $\mathcal{X}_O$.

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:

$$\begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{Y}} \end{bmatrix},$$

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

$$\begin{bmatrix} I & \tilde{A}^{-1}\tilde{Y} \end{bmatrix} = \begin{bmatrix} I & \tilde{X} \end{bmatrix},$$

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

\[
\begin{bmatrix}
I & \tilde{A}^{-1}\tilde{Y}
\end{bmatrix} = \begin{bmatrix}
I & \tilde{X}
\end{bmatrix},
\]

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

Performance

\[
\text{COMMUNICATION}(\mathcal{A}) = O(d \cdot n \log q)
\]

\[
\text{COMPUTATION}(\mathcal{A}) = O(k^2 \cdot n)
\]

If \(s\) is known, then \(\text{TIME}(\mathcal{A}) = 2\). If \(s\) is not known, then \(\text{TIME}(\mathcal{A}) = 3\).
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.
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 $S$ of $k$ different files in $\{0,1\}^n$. Let $H$ be a set of all functions $H : \{0,1\}^n \rightarrow K$, where $K$ is the set of all possible keys. Assume that $k \ll |K| \ll 2^n$. 
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 $\mathcal{S}$ of $k$ different files in $\{0, 1\}^n$. Let $\mathcal{H}$ be a set of all functions $\mathcal{H} : \{0, 1\}^n \rightarrow K$, where $K$ is the set of all possible keys. Assume that $k \ll |K| \ll 2^n$.

**Theorem**

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