Codes for Privacy and Reliability

in Information Retrieval and Distributed Computation

by

Oliver W. Gnilke

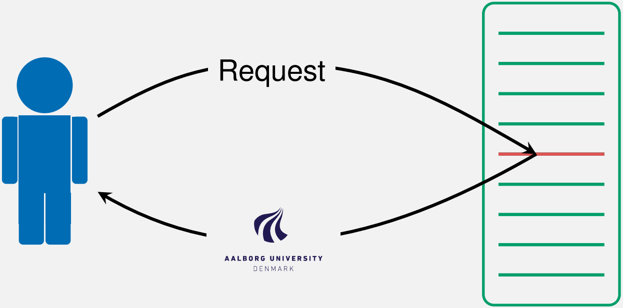
owg@math.aau.dk

AALBORG UNIVERSITY
DENMARK

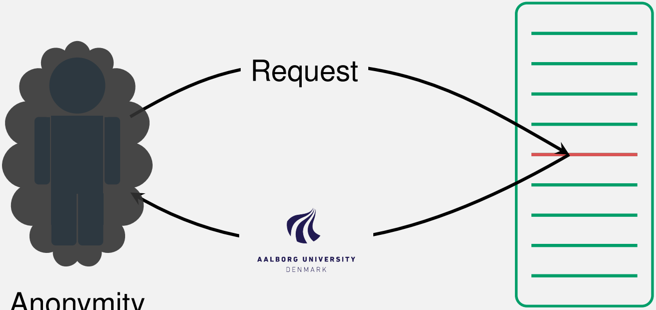
September 2, 2020

Private Information Retrieval

Securing Communications



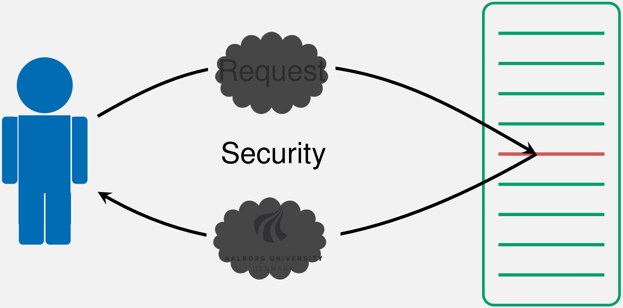
Securing Communications



Anonymity

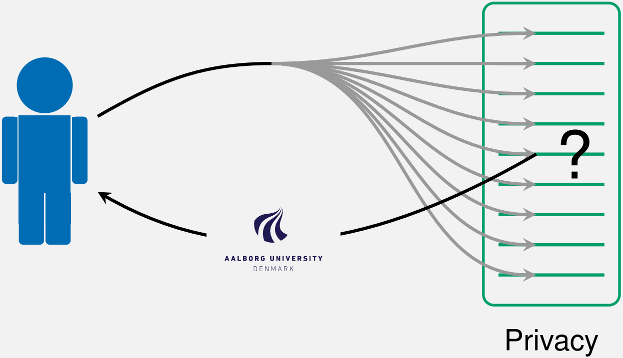
For example: TOR

Securing Communications



For example: https/TLS/SSL

Securing Communications



PIR

One-Time Pad

- Message / Information e of 'size' $|e|$.
- Key k of size $|k| \geq |e|$.

Encryption:	$c := e + k$
Decryption:	$e = c - k$

One-Time Pad

- Message / Information e of 'size' $|e|$.
- Key k of size $|k| \geq |e|$.

$$\begin{aligned} \text{Encryption:} \quad & c := e + k \\ \text{Decryption:} \quad & e = c - k \end{aligned}$$

- Information theoretically secure.
- But requires a key that is at least as big as the message.

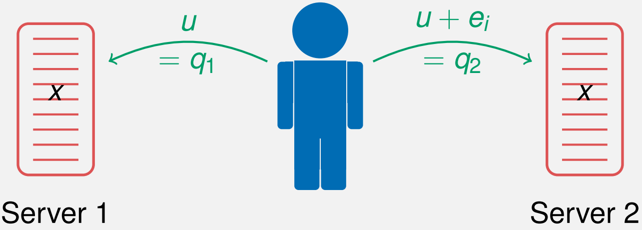
One-Time Pad

- Message / Information e of 'size' $|e|$.
- Key k of size $|k| \geq |e|$.

Encryption:	$c := e + k$
Decryption:	$e = c - k$

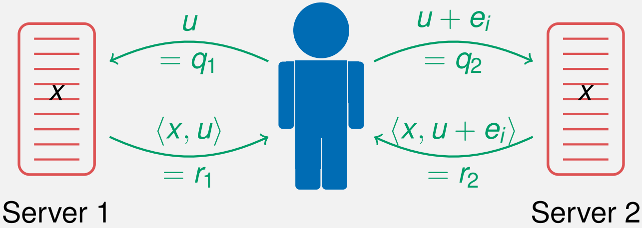
- Information theoretically secure.
- But requires a key that is at least as big as the message.
- The advantage is that an eavesdropper needs to have the ciphertext c AND the key k.

A first example



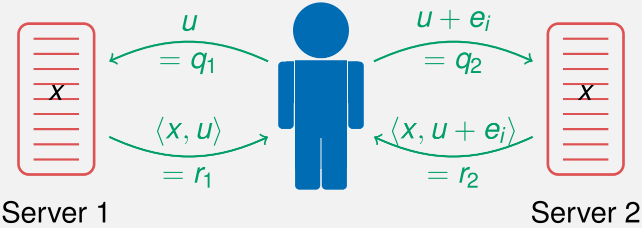
1.) The user wants file x_i. Draws $u \sim U(\mathbb{F}_q^m)$ and forms queries $q_1 = u$ and $q_2 = u + e_i$.

A first example



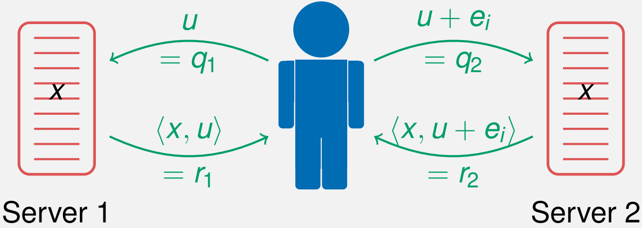
1.) The user wants file x_i. Draws $u \sim U(\mathbb{F}_q^m)$ and forms queries $q_1 = u$ and $q_2 = u + e_i$.
2.) The servers respond with $r_j := \langle x, q_j \rangle$.

A first example



1.) The user wants file x_i. Draws $u \sim U(\mathbb{F}_q^m)$ and forms queries $q_1 = u$ and $q_2 = u + e_i$.
2.) The servers respond with $r_j := \langle x, q_j \rangle$.
3.) The user calculates $r_2 - r_1 = \langle x, u + e_i \rangle - \langle x, u \rangle = x_i$.

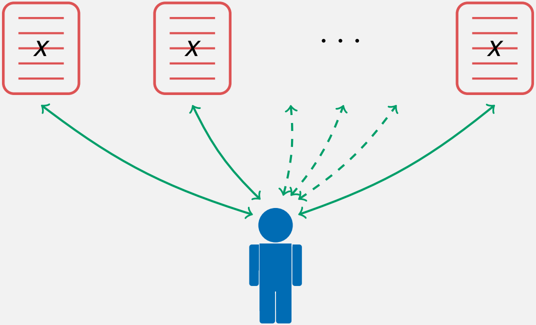
A first example



Rate

We measure *rate* as the size of the requested file over the size of all downloads, $R = \frac{|x_i|}{\sum |r_j|}$. E.g. the rate in the example is $R = \frac{1}{2}$

More Servers



- If all n servers contain the full database X we can download $n-1$ files simultaneously.
- This gives us a rate of $R = 1 - \frac{1}{n}$

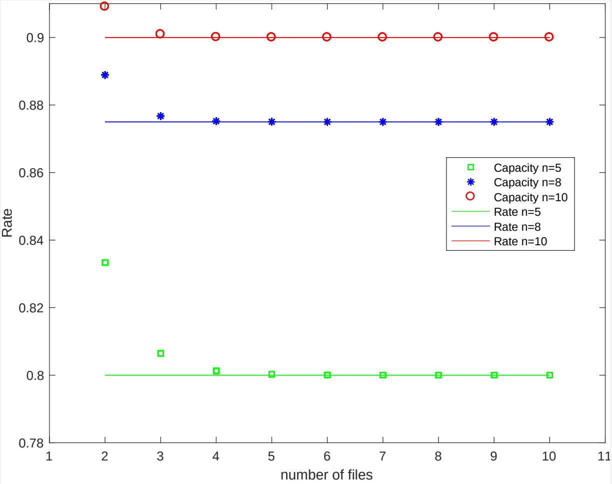
Capacity

Capacity for m files replicated on n servers [2]

$$C = \frac{1 - \frac{1}{n}}{1 - \left(\frac{1}{n}\right)^m}$$

[2] Hua Sun, Syed A. Jafar, *The Capacity of Private Information Retrieval*, IEEE Transactions on Information Theory, Volume: 63, Issue: 7, July 2017)

Capacity



[2] Hua Sun, Syed A. Jafar, *The Capacity of Private Information Retrieval*, IEEE Transactions on Information Theory, Volume: 63, Issue: 7, July 2017)

Distributed Storage Systems

- Replicating all files across all servers is rather wasteful.
- To save space a storage code C is employed.

$$\text{files}\ \begin{array}{c} x^1 \\ \vdots \\ x^m \end{array} \begin{pmatrix} x_1^1 & \dots & x_k^1 \\ \vdots & \ddots & \vdots \\ x_1^m & \dots & x_k^m \end{pmatrix} \cdot G_C = \begin{pmatrix} y_1^1 & \dots & y_n^1 \\ \vdots & \ddots & \vdots \\ y_1^m & \dots & y_n^m \end{pmatrix}$$

Distributed Storage Systems

- Replicating all files across all servers is rather wasteful.
- To save space a storage code C is employed.

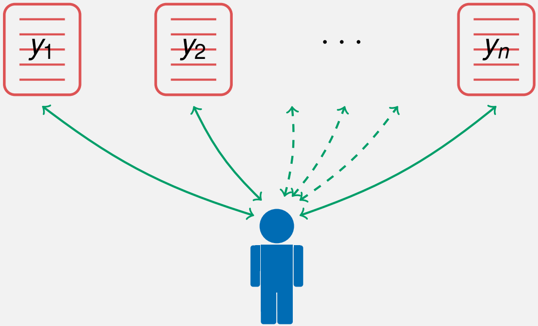
$$\text{files}\;\begin{array}{c} x^1 \\ \vdots \\ x^m \end{array} \begin{pmatrix} x_1^1 & \cdots & x_k^1 \\ \vdots & \ddots & \vdots \\ x_1^m & \cdots & x_k^m \end{pmatrix} \cdot G_C = \begin{array}{c} \text{Server 1} \cdots \text{Server n} \\ \begin{pmatrix} y_1^1 & \cdots & y_n^1 \\ \vdots & \ddots & \vdots \\ y_1^m & \cdots & y_n^m \end{pmatrix} \end{array}$$

Distributed Storage Systems

- Replicating all files across all servers is rather wasteful.
- To save space a storage code C is employed.
- Not coding across files enables easy addition or removal of files.

$$
\begin{array}{c}
\text{files} \\
\begin{array}{c} x^1 \\ \vdots \\ x^m \\ x^{m+1} \end{array}
\end{array}
\begin{pmatrix}
x_1^1 & \dots & x_k^1 \\
\vdots & \ddots & \vdots \\
x_1^m & \dots & x_k^m \\
x_1^{m+1} & \dots & x_k^{m+1}
\end{pmatrix} \cdot G_C =
\begin{array}{c}
\begin{array}{ccc} \text{Server 1} & \cdots & \text{Server n} \end{array} \\
\begin{pmatrix}
y_1^1 & \dots & y_n^1 \\
\vdots & \ddots & \vdots \\
y_1^m & \dots & y_n^m \\
y_1^{m+1} & \dots & y_n^{m+1}
\end{pmatrix}
\end{array}
$$

More Servers - Coded



- If the storage is coded, contents from two different servers no longer cancel out.

PIR from coded storage

- Let $u \sim U(\mathbb{F}_q^m)$.
- Define the n queries as $q_j = u$ for all servers j. Then

$$r_j = \langle q_j, y_j \rangle = \langle u, y_j \rangle.$$

- The vector

$$(r_1, \ldots, r_n) = (u.y_1, \ldots, u.y_n) = u^T[y_1 \cdots y_n] = \sum_{i=1}^{m} u^i y^i$$

is a linear combination of the codewords in the storage system, and therefore itself a codeword in C.

PIR from coded storage

- Let $u \sim U(\mathbb{F}_q^m)$.
- Define the queries as $q_j = u$ for all servers but one, here the last, and let $q_n = u + e_i$

PIR from coded storage

- Let $u \sim U(\mathbb{F}_q^m)$.
- Define the queries as $q_j = u$ for all servers but one, here the last, and let $q_n = u + e_i$
- The vector $(r_1, \ldots, r_n)$ is a linear combination of the codewords in the storage system, plus one 'error' in the last coordinate

$$(r_1, \ldots, r_n) = (u.y_1, \ldots, u.y_n + y_n^i) = \sum_{i=1}^{m} u^i y^i + (0, \ldots, 0, y_n^i)$$

PIR from coded storage

- Let $u \sim U(\mathbb{F}_q^m)$.
- Define the queries as $q_j = u$ for all servers but one, here the last, and let $q_n = u + e_i$
- The vector $(r_1, \ldots, r_n)$ is a linear combination of the codewords in the storage system, plus one 'error' in the last coordinate

$$(r_1, \ldots, r_n) = (u.y_1, \ldots, u.y_n + y_n^i) = \sum_{i=1}^{m} u^i y^i + (0, \ldots, 0, y_n^i)$$

- We only need k coordinates of a codeword to uniquely determine it. Hence we can add up to $n - k$ such 'errors'.

PIR Rate for coded storage

- We receive $n-k$ blocks of information, when downloading n blocks total.

Rate for PIR from MDS coded storage

$$R = \frac{n-k}{n}$$

PIR Rate for coded storage

- We receive $n - k$ blocks of information, when downloading n blocks total.

Rate for PIR from MDS coded storage

$$R = \frac{n-k}{n}$$

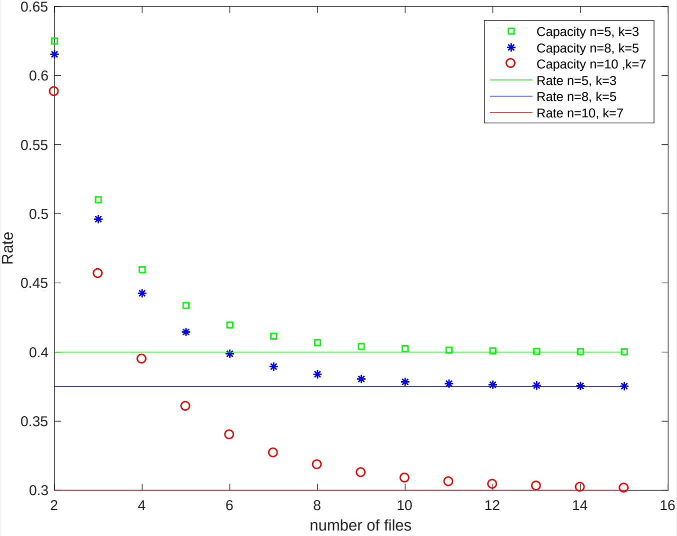
- We compare to the capacity

Capacity for PIR from coded storage [3]

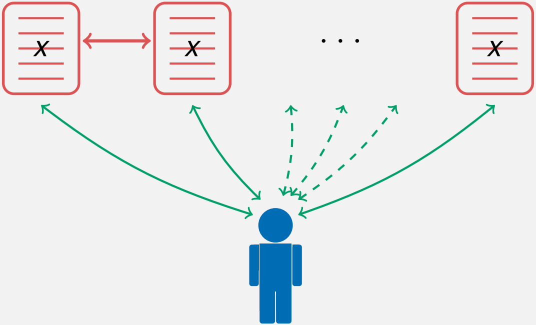
$$C = \frac{1-\frac{k}{n}}{1-\left(\frac{k}{n}\right)^m}$$

[3] Karim Banawan, Sennur Ulukus, *The Capacity of Private Information Retrieval From Coded Databases*, IEEE Transactions on Information Theory, Volume: 64, Issue: 3, March 2018

Asymptotic vs. Capacity



More Servers - Collusion



- In the replicated storage scenario, each query is masked with the same random vector.
- If two of them exchange their queries, they can unmask the request.

t-Collusion

- We want to design a scheme that remains secure, even if $t < n$ of the servers combine their queries.
- We use secret sharing to design these queries.

t-Collusion

- We want to design a scheme that remains secure, even if $t < n$ of the servers combine their queries.
- We use secret sharing to design these queries.

Shamir Secret Sharing

Let (α_i) be a list of n pairwise different elements of $\mathbb{F}_q$. Let $u_0, \ldots, u_{t-1} \sim U(\mathbb{F}_q)$, and e_t our information.

$$
\begin{gathered}
(u_0, \ldots, u_{t-1}, e_t) \in \mathbb{F}_q^k \\
\downarrow \\
f(z) = u_0 + \cdots + u_{t-1} z^{t-1} + e_t z^t \quad \mapsto \quad (f(\alpha_1), \ldots, f(\alpha_n))
\end{gathered}
$$

t-Collusion

- We want to design a scheme that remains secure, even if $t < n$ of the servers combine their queries.
- We use secret sharing to design these queries.

Shamir Secret Sharing

Let (α_i) be a list of n pairwise different elements of $\mathbb{F}_q$. Let $u_0, \ldots, u_{t-1} \sim U(\mathbb{F}_q)$, and e_t our information.

$$
\begin{array}{ccc}
(u_0, \ldots, u_{t-1}, e_t) \in \mathbb{F}_q^k & & (s_1, \ldots, s_n) \\
\downarrow & & \parallel \\
f(z) = u_0 + \cdots + u_{t-1} z^{t-1} + e_t z^t & \mapsto & (f(\alpha_1), \ldots, f(\alpha_n))
\end{array}
$$

Any subset of t shares s_i, reveals no information about e_t.

t-Collusion

- We want to design a scheme that remains secure, even if $t < n$ of the servers combine their queries.
- We use secret sharing to design these queries.

Shamir Secret Sharing

Let (α_i) be a list of n pairwise different elements of $\mathbb{F}_q$. Let $u_0, \ldots, u_{t-1} \sim U(\mathbb{F}_q)$, and e_t our information.

$$\begin{array}{ccc} (u_0, \ldots, u_{t-1}, e_t) \in \mathbb{F}_q^k & & (s_1, \ldots, s_n) \\ \downarrow & & \shortparallel \\ f(z) = u_0 + \cdots + u_{t-1} z^{t-1} + e_t z^t & \mapsto & (f(\alpha_1), \ldots, f(\alpha_n)) \end{array}$$

Any subset of t shares s_i, reveals no information about e_t.

- We use a slightly altered version for our scheme.

PIR with Collusion

- The scheme is dual to the coded storage scheme. Instead of the files, our queries are encoded with an $[n, t]$ MDS code D.

$$\text{randomness}\quad \begin{pmatrix} u_1^1 & \dots & u_t^1 \\ \vdots & \ddots & \vdots \\ u_1^m & \dots & u_t^m \end{pmatrix} \cdot G_D + E = \begin{pmatrix} q_1^1 & \dots & q_n^1 \\ \vdots & \ddots & \vdots \\ q_1^m & \dots & q_n^m \end{pmatrix} + E$$

PIR with Collusion

- The scheme is dual to the coded storage scheme. Instead of the files, our queries are encoded with an $[n, t]$ MDS code D.

$$
\begin{array}{cc}
 & \begin{array}{ccc} & & \text{Query } 1 \;\ldots\; \text{Query } n \end{array} \\
\text{randomness} & \begin{pmatrix} \boxed{u_1^1 \quad \cdots \quad u_t^1} \\ \vdots \quad \ddots \quad \vdots \\ \boxed{u_1^m \quad \cdots \quad u_t^m} \end{pmatrix} \cdot G_D + {\color{green}E} = \begin{pmatrix} \boxed{\begin{matrix} d_1^1 \\ \vdots \\ d_1^m \end{matrix}} & \begin{matrix} \cdots \\ \ddots \\ \cdots \end{matrix} & \boxed{\begin{matrix} d_n^1 \\ \vdots \\ d_n^m \end{matrix}} \end{pmatrix}
\end{array}
$$

PIR with Collusion

- The scheme is dual to the coded storage scheme. Instead of the files, our queries are encoded with an $[n, t]$ MDS code D.

$$
\text{randomness}\quad
\begin{pmatrix}
\boxed{u_1^1 \quad \dots \quad u_t^1} \\
\vdots \quad \ddots \quad \vdots \\
\boxed{u_1^m \quad \dots \quad u_t^m}
\end{pmatrix} \cdot G_D + E =
\begin{array}{ccc} \text{Query 1} & \dots & \text{Query n} \end{array}
\begin{pmatrix}
d_1^1 & \dots & d_n^1 \\
\vdots & \ddots & \vdots \\
d_1^m & \dots & d_n^m
\end{pmatrix}
$$

- The responses now again take the form,

$$
(r_1, \dots, r_n) = (d_1.x, \dots, d_n.x) + e = \sum_{i=1}^{m} d^i x^i + e.
$$

- $\sum_{i=1}^{m} d^i x^i$ is a codeword in D

Rate and Capacity for t-collusion

- We receive $n - t$ blocks of information, when downloading n blocks total.

Rate for PIR with t-collusion

$$R = \frac{n-t}{n}$$

Rate and Capacity for t-collusion

- We receive $n - t$ blocks of information, when downloading n blocks total.

Rate for PIR with t-collusion

$$R = \frac{n-t}{n}$$

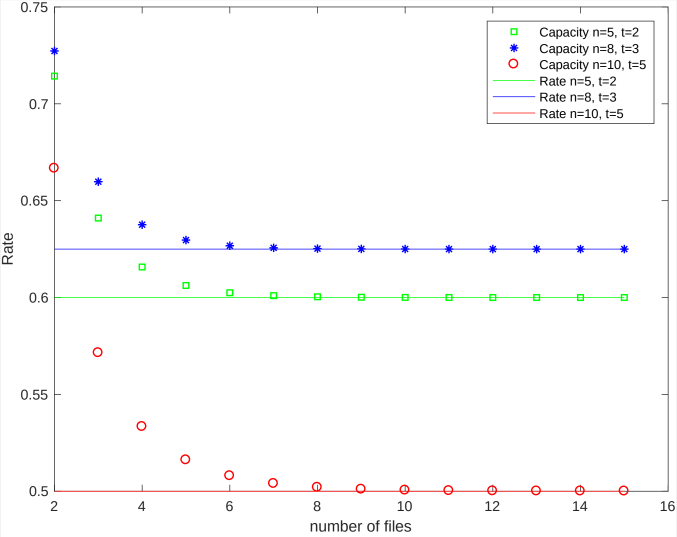
- We compare to the capacity expression for this scenario

Capacity for PIR with t-collusion [4]

$$C = \frac{1-\frac{t}{n}}{1-\left(\frac{t}{n}\right)^m}$$

[4] Hua Sun, Syed A. Jafar, *The capacity of private information retrieval with colluding databases*, 2016 IEEE Global Conference on Signal and Information Processing

Asymptotic vs. Capacity



Combining both schemes

- We combine both schemes, *i.e.*, encode both the data and the queries.
- The responses then take a different form

$$(r_1, \ldots, r_n) = (d_1.y_1, \ldots, d_n.y_n) + e$$

Combining both schemes

- We combine both schemes, *i.e.*, encode both the data and the queries.
- The responses then take a different form

$$(r_1, \dots, r_n) = (d_1.y_1, \dots, d_n.y_n) + e = \sum_{i=1}^{m} d^i \star y^i + e$$

where $(c_1, \dots, c_n) \star (d_1, \dots, d_n) := (c_1 d_1, \dots, c_n d_n)$ is the Schur product of vectors.

Combining both schemes

- We combine both schemes, *i.e.*, encode both the data and the queries.
- The responses then take a different form

$$(r_1, \ldots, r_n) = (d_1.y_1, \ldots, d_n.y_n) + e = \sum_{i=1}^{m} d^i \star y^i + e$$

  where $(c_1, \ldots, c_n) \star (d_1, \ldots, d_n) := (c_1 d_1, \ldots, c_n d_n)$ is the Schur product of vectors.
- Do these responses lie inside some code (plus some errors) that we can easily describe?

Products of Codes

Schur Product

Let C and D be two linear codes of length n. Then we define their product code as the span of all Schur products of codewords in C with codewords in D.

$$C \star D := \langle \{ c \star d \ : \ c \in C, d \in D \} \rangle$$

Products of Codes

Schur Product

Let C and D be two linear codes of length n. Then we define their product code as the span of all Schur products of codewords in C with codewords in D.

$$C \star D := \langle \{ c \star d \; : \; c \in C, d \in D \} \rangle$$

- The rate of our scheme will again depend on the minimum distance of the response code.

Products of Codes

Schur Product

Let C and D be two linear codes of length n. Then we define their product code as the span of all Schur products of codewords in C with codewords in D.

$$C \star D := \langle \{ c \star d \ : \ c \in C, d \in D \} \rangle$$

Product Singleton Bound[5]

$$d_{C \star D} - 1 \leq \max\{0, n - k_C - k_D + 1\}$$

[5] H. Randriambololona, *An upper bound of Singleton type for componentwise products of linear codes*, IEEE Transactions on Information Theory vol 59., 2013

Optimal Products

Product Singleton Bound[5]

$$d_{C\star D} - 1 \leq \max\{0, n - k_C - k_D + 1\}$$

- Equality is achieved in a handful of cases only

[5] H. Randriambololona, *An upper bound of Singleton type for componentwise products of linear codes*, IEEE Transactions on Information Theory vol 59., 2013

Optimal Products

Product Singleton Bound[5]

$$d_{C\star D} - 1 \leq \max\{0, n - k_C - k_D + 1\}$$

- Equality is achieved in a handful of cases only
  - C or D are the repetition code.
  - $C = D^{\perp}$.
  - C and D are generalized Reed-Solomon (GRS) codes on the same evaluation set.

[5] H. Randriambololona, *An upper bound of Singleton type for componentwise products of linear codes*, IEEE Transactions on Information Theory vol 59., 2013

Optimal Products

Product Singleton Bound[5]

$$d_{C\star D} - 1 \leq \max\{0, n - k_C - k_D + 1\}$$

- Equality is achieved in a handful of cases only
  - C or D are the repetition code.
    Then we have no collusion or no coding.
  - $C = D^{\perp}$.
  - C and D are generalized Reed-Solomon (GRS) codes on the same evaluation set.

[5] H. Randriambololona, *An upper bound of Singleton type for componentwise products of linear codes*, IEEE Transactions on Information Theory vol 59., 2013

Optimal Products

Product Singleton Bound[5]

$$d_{C\star D} - 1 \leq \max\{0, n - k_C - k_D + 1\}$$

- Equality is achieved in a handful of cases only
  - C or D are the repetition code.
    Then we have no collusion or no coding.
  - $C = D^{\perp}$.
    Then we get a rate $\frac{1}{n}$ scheme.
  - C and D are generalized Reed-Solomon (GRS) codes on the same evaluation set.

[5] H. Randriambololona, *An upper bound of Singleton type for componentwise products of linear codes*, IEEE Transactions on Information Theory vol 59., 2013

Optimal Products

Product Singleton Bound[5]

$$d_{C\star D} - 1 \leq \max\{0, n - k_C - k_D + 1\}$$

- Equality is achieved in a handful of cases only
  - C or D are the repetition code.
    Then we have no collusion or no coding.
  - $C = D^\perp$.
    Then we get a rate $\frac{1}{n}$ scheme.
  - C and D are generalized Reed-Solomon (GRS) codes on the same evaluation set.
    This allows for a flexible schemes with varied parameters.

[5] H. Randriambololona, *An upper bound of Singleton type for componentwise products of linear codes*, IEEE Transactions on Information Theory vol 59., 2013

PIR - coded storage & collusion

- Use GRS codes for the storage and the queries.

$$(r_1, \ldots, r_n) = (d_1.y_1, \ldots, d_n.y_n) + e \qquad .$$

PIR - coded storage & collusion

- Use GRS codes for the storage and the queries.

$$(r_1, \ldots, r_n) = (d_1.y_1, \ldots, d_n.y_n) + e = \sum_{i=1}^{m} d^i \star y^i + e.$$

Then $\sum_{i=1}^{m} d^i \star y^i$ is again a codeword in an $[n, k+t-1]$ GRS code

PIR - coded storage & collusion

- Use GRS codes for the storage and the queries.

$$(r_1, \ldots, r_n) = (d_1.y_1, \ldots, d_n.y_n) + e = \sum_{i=1}^{m} d^i \star y^i + e.$$

  Then $\sum_{i=1}^{m} d^i \star y^i$ is again a codeword in an $[n, k + t - 1]$ GRS code
- Hence we can add up to $n - k - t + 1$ 'errors' via e that we can correct.
- We therefore achieve a rate of $\frac{n-k-t+1}{n}$, whenever this is positive.

Capacity

- The capacity for the coded, colluding case has been a long standing problem.

[6] Lukas Holzbaur, Ragnar Freij-Hollanti, Jie Li, Camilla Hollanti, *Towards the Capacity of Private Information Retrieval from Coded and Colluding Servers*, arXiv:1903.12552.

Capacity

- The capacity for the coded, colluding case has been a long standing problem.
- A recent paper[6] 'solves' this by only considering schemes that are 'symbol separated' or 'strongly linear'.
- This covers a lot of schemes in the literature and especially the ones presented here, and they are indeed capacity achieving under these restrictions.

[6] Lukas Holzbaur, Ragnar Freij-Hollanti, Jie Li, Camilla Hollanti, *Towards the Capacity of Private Information Retrieval from Coded and Colluding Servers*, arXiv:1903.12552.

Byzantine and Non-Responsive

- Up till now we have considered *honest but curious* servers.
- But what if some servers are slow / fail to respond or introduce errors into their responses.

Byzantine and Non-Responsive

- Up till now we have considered *honest but curious* servers.
- But what if some servers are slow / fail to respond or introduce errors into their responses.
- The previous scheme uses the codes erasure correction capability to retrieve the data.

$$(r_1, \ldots, r_n) = (d_1.y_1, \ldots, d_n.y_n) + e =$$
$$(d_1.y_1, \ldots, d_{t+k-1}.y_{t+k-1}, d_{t+k}.y_{t+k} + e_{t+k}, \ldots, d_n.y_n + e_n)$$

Byzantine and Non-Responsive

- Up till now we have considered *honest but curious* servers.
- But what if some servers are slow / fail to respond or introduce errors into their responses.
- The previous scheme uses the codes erasure correction capability to retrieve the data.

$$(r_1, \ldots, r_n) = (d_1.y_1, \ldots, d_n.y_n) + e =$$

$$(d_1.y_1, \ldots, d_{t+k-1}.y_{t+k-1}, d_{t+k}.y_{t+k} + e_{t+k}, \ldots, d_n.y_n + e_n)$$

$$(d_1.y_1, \ldots, d_{t+k-1}.y_{t+k-1}, \cancel{d_{t+k}.y_{t+k} + e_{t+k}}, \ldots, d_n.y_n + e_n + \underline{b_n})$$

- If an entry with an 'error' is lost, that information is lost. If it is altered by an additional error then we will receive a wrong symbol.

Polynomial Scheme

- Assumptions $\leq r$ non-responsive servers,
  $\leq b$ byzantine servers.
- For simplicity assume $k = n - k - t - r - 2b + 1$.
  (This is not necessary but avoids some complications)

	polynomial	code
files	$f(z)$	C=GRS $[n, k]$
query $j \neq i$	$g(z)$	D=GRS $[n, t]$
response $j \neq i$	$f(z)g(z)$	$C \star D$=GRS $[n, k+t-1]$
response $j = i$	$f(z)g(z)$	

Polynomial Scheme

- Assumptions $\leq r$ non-responsive servers,
  $\leq b$ byzantine servers.
- For simplicity assume $k = n - k - t - r - 2b + 1$.
  (This is not necessary but avoids some complications)

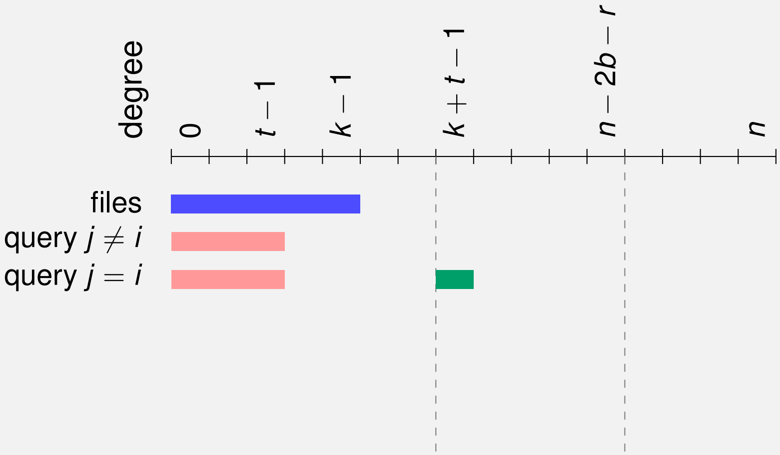
	polynomial	code
files	$f(z)$	C=GRS $[n, k]$
query $j \neq i$	$g(z)$	D=GRS $[n, t]$
response $j \neq i$	$f(z)g(z)$	$C \star D$=GRS $[n, k + t - 1]$
response $j = i$	$f(z)g(z)$	GRS $[n, n - r - 2b]$

Polynomial Scheme

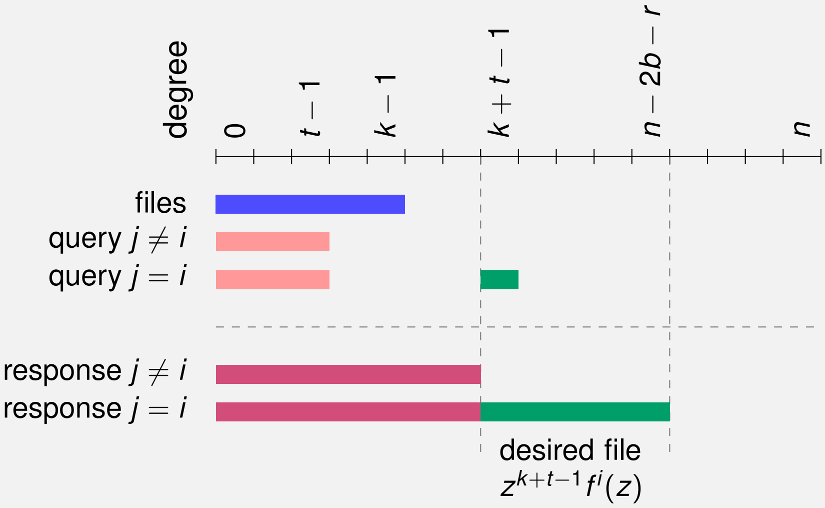
- Assumptions $\leq r$ non-responsive servers, $\leq b$ byzantine servers.
- For simplicity assume $k = n - k - t - r - 2b + 1$. (This is not necessary but avoids some complications)

	polynomial	code
files	$f(z)$	C=GRS $[n, k]$
query $j \neq i$	$g(z)$	D=GRS $[n, t]$
query $j = i$	$g(z) + z^{k+t-1}$	$\subset$GRS $[n, k+t]$
response $j \neq i$	$f(z)g(z)$	$C \star D$=GRS $[n, k+t-1]$
response $j = i$	$f(z)g(z) + z^{k+t-1} f^i(z)$	GRS $[n, n-r-2b]$

Polynomial scheme

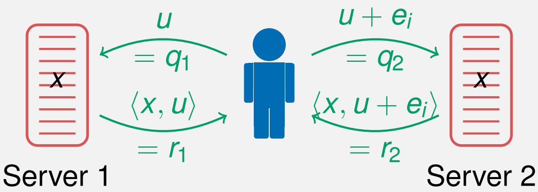


Polynomial scheme



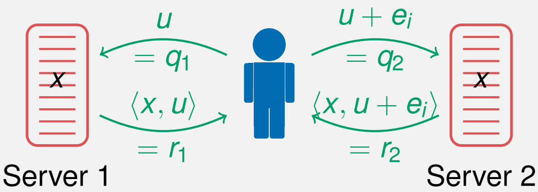
Computation

Linear Functions



- In the simple scheme, we only want to hide the index i. Hence about $\log_2(m)$ bits of information. (Assuming every request is equally likely)
- But we use $u \sim U(\mathbb{F}_q^m)$ as a key, which needs $m \log_2(q)$ bits of randomness.

Linear Functions



- This is because our scheme can do more! We could hide any request for a linear combination $\sum \ell_i x_i$ of the files.

$$r_2 - r_1 = \langle x, u + \ell \rangle - \langle x, u \rangle = \langle x, \ell \rangle = \sum \ell_i x_i$$

Matrix Multiplication

- The next step is to utilize helper nodes in order to perform computations for a user.
- The standard example is the mulitplication of two big matrices A, B.
- These matrices might contain sensitive information and we do not want the helpers to learn anything about the contents of A and B.

GASP D'Oliveira, El Rouayheb, Karpuk, Heinlein

$$\begin{pmatrix} - & A_1 & - \\ & \vdots & \\ - & A_n & - \end{pmatrix} \begin{pmatrix} | & & | \\ B_1 & \dots & B_m \\ | & & | \end{pmatrix} = \begin{pmatrix} A_1B_1 & \cdots & A_1B_m \\ \vdots & \ddots & \vdots \\ A_nB_1 & \cdots & A_nB_m \end{pmatrix}$$

- Hide the contents of A and B through secret sharing.

$$f(z) := A_1 z^{\alpha_1} + \cdots + A_n^{\alpha_n} + R(z)$$
$$g(z) := B_1 z^{\beta_1} + \cdots + B_m^{\beta_m} + S(z)$$

- Send each server the evaluations $f(z_i)$ and $g(z_i)$ and ask them to compute their product.
- If we have enough evaluations of fg we can recover its coefficients via interpolation.

$$f(z) := A_1 z^{\alpha_1} + \cdots + A_n z^{\alpha_n} + R(z)$$

$$g(z) := B_1 z^{\beta_1} + \cdots + B_m z^{\beta_m} + S(z)$$

- Assume we do not care about privacy for a minute, i.e., $R(z)$ and $S(z)$ are zero.
- The term $A_a B_b$ will appear in the coefficient of $z^{\alpha_a + \beta_b}$ in the product $f(z)g(z)$.

$$f(z) := A_1 z^{\alpha_1} + \cdots + A_n z^{\alpha_n} + R(z)$$
$$g(z) := B_1 z^{\beta_1} + \cdots + B_m z^{\beta_m} + S(z)$$

- Assume we do not care about privacy for a minute, i.e., $R(z)$ and $S(z)$ are zero.
- The term $A_a B_b$ will appear in the coefficient of $z^{\alpha_a+\beta_b}$ in the product $f(z)g(z)$.

	β_1	β_2	β_3
α_1	$\alpha_1+\beta_1$	$\alpha_1+\beta_2$	$\alpha_1+\beta_3$
α_2	$\alpha_2+\beta_1$	$\alpha_2+\beta_2$	$\alpha_2+\beta_3$
α_3	$\alpha_3+\beta_1$	$\alpha_3+\beta_2$	$\alpha_3+\beta_3$

$$f(z) := A_1 z^{\alpha_1} + \cdots + A_n z^{\alpha_n} + R(z)$$
$$g(z) := B_1 z^{\beta_1} + \cdots + B_m z^{\beta_m} + S(z)$$

- Assume we do not care about privacy for a minute, i.e., $R(z)$ and $S(z)$ are zero.
- The term $A_a B_b$ will appear in the coefficient of $z^{\alpha_a+\beta_b}$ in the product $f(z)g(z)$.

	0	β_2	β_3
0	0		
1	1		
2	2		

GASP D'Oliveira, El Rouayheb, Karpuk, Heinlein

$$f(z) := A_1 z^{\alpha_1} + \cdots + A_n z^{\alpha_n} + R(z)$$
$$g(z) := B_1 z^{\beta_1} + \cdots + B_m z^{\beta_m} + S(z)$$

- Assume we do not care about privacy for a minute, i.e., $R(z)$ and $S(z)$ are zero.
- The term $A_a B_b$ will appear in the coefficient of $z^{\alpha_a + \beta_b}$ in the product $f(z)g(z)$.

	0	3	β_3
0	0	3	
1	1	4	
2	2	5	

$$f(z) := A_1 z^{\alpha_1} + \cdots + A_n z^{\alpha_n} + R(z)$$
$$g(z) := B_1 z^{\beta_1} + \cdots + B_m z^{\beta_m} + S(z)$$

- Assume we do not care about privacy for a minute, i.e., $R(z)$ and $S(z)$ are zero.
- The term $A_a B_b$ will appear in the coefficient of $z^{\alpha_a+\beta_b}$ in the product $f(z)g(z)$.

	0	3	6
0	0	3	6
1	1	4	7
2	2	5	8

We need $N = 9$ evaluations.

- Now we add randomness.

$$
\begin{array}{c|ccc|}
 & 0 & 3 & 6 \\
\hline
0 & 0 & 3 & 6 \\
1 & 1 & 4 & 7 \\
2 & 2 & 5 & 8 \\
\hline
9 & 9 & 12 & 15
\end{array}
$$

$N = 16$?

- Now we add randomness.

	0	3	6
0	0	3	6
1	1	4	7
2	2	5	8
9	9	12	15

Actually, there are only 12 unknowns, and we only need 12 evaluations.

- Now we add randomness.

	0	3	6	9
0	0	3	6	9
1	1	4	7	10
2	2	5	8	11
9	9	12	15	18

$N = 15$

- Now we add randomness.

	0	3	6	9	10
0	0	3	6	9	10
1	1	4	7	10	11
2	2	5	8	11	12
9	9	12	15	18	13
10	10	13	16	19	20

$N = 18$

- Now we add randomness.

	0	3	6	9	10	11
0	0	3	6	9	10	11
1	1	4	7	10	11	12
2	2	5	8	11	12	13
9	9	12	15	18	19	20
10	10	13	16	19	20	21
11	11	14	17	20	21	22

$N = 23$

- Can we improve on this?

- Now we add randomness.

	0	3	6	9	10	11
0	0	3	6	9	10	11
1	1	4	7	10	11	12
2	2	5	8	11	12	13
9	9	12	15	18	19	20
10	10	13	16	19	20	21
12	12	15	18	21	22	23

$N = 22$

- The coefficients 14 and 17 are missing.

Splitting into Blocks

- Let $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}$, and $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$. Then the blocks of their product are given by the sums

$$(AB)_{ik} = \sum_j A_{ij} B_{jk}.$$

Splitting into Blocks

- Let $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}$, and $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$. Then the blocks of their product are given by the sums

$$(AB)_{ik} = \sum_j A_{ij} B_{jk}.$$

- We can realize this as coefficients of a polynomial as well. Let

$$\begin{aligned} f(z) &:= A_{i1} + A_{i2} z \\ g(z) &:= B_{2k} + B_{1k} z, \text{ then} \\ f(z)g(z) &:= \cdots + (A_{i1} B_{1k} + A_{i2} B_{2k}) z + \cdots \end{aligned}$$

Secure Generalized PolyDot [7]

- An example with up to 2 colluding servers.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}$$

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

		B_{21}	B_{11}	B_{22}	B_{12}	S_1	S_2
		0	1	8	9	14	15
A_{11}	0	0	1	8	9	14	15
A_{12}	1	1	2	9	10	15	16
A_{21}	2	2	3	10	11	16	17
A_{22}	3	3	4	11	12	17	18
A_{31}	4	4	5	12	13	18	19
A_{32}	5	5	6	13	14	19	20
R_1	6	6	7	14	15	20	21
R_2	7	7	8	15	16	21	22

- Note that *e.g.* the coefficient of degree 5 is given by $A_{31}B_{11} + A_{32}B_{21} = (AB)_{31}$.

[7] M. Aliasgari, O. Simeone and J. Kliewer, *Distributed and Private Coded Matrix Computation with Flexible Communication Load*, 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France

Ongoing Work

- Find PIR schemes that do not fall under the restriction of being 'strongly linear' and exceed the rate of previous schemes.
- Find improved sets of exponents for the secure generalized PolyDot construction.
- Expand secure distributed computation to matrices over small fields.
- Expand secure distributed computation to other functions.

Thank You!