GASP Codes for Secure Distributed Matrix Multiplication

Introduction to Blockchain course

Team project

Designed by:
Alexander Blagodarnyi
Stanislav Krikunov
Mariia Kopylova
Mikhail Konovalov

Skoltech
Skolkovo Institute of Science and Technology

Our team

Alexander Blagodarnyi

Stanislav Krikunov

Mikhail Konovalov

Mariia Kopylova

What is the GASP and SDMM?

SDMM

$$\begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \times \begin{bmatrix} 6 & 4 & 3 \end{bmatrix} = \begin{bmatrix} 12 & 8 & 6 \\ 18 & 12 & 9 \\ 24 & 16 & 12 \end{bmatrix}$$

GASP

$$\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_2(x) &= \tfrac{1}{2}(3x^2 - 1) \\
P_3(x) &= \tfrac{1}{2}(5x^3 - 3x) \\
P_4(x) &= \tfrac{1}{8}(35x^4 - 30x^2 + 3) \\
P_5(x) &= \tfrac{1}{8}(63x^5 - 70x^3 + 15x) \\
P_6(x) &= \tfrac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5) \\
P_7(x) &= \tfrac{1}{16}(429x^7 - 693x^5 + 315x^3 - 35x) \\
P_8(x) &= \tfrac{1}{128}(6435x^8 - 12012x^6 + 6930x^4 - 1260x^2 + 35) \\
P_9(x) &= \tfrac{1}{128}(12155x^9 - 25740x^7 + 18018x^5 - 4620x^3 + 315x) \\
P_{10}(x) &= \tfrac{1}{256}(46189x^{10} - 109395x^8 + 90090x^6 - 30030x^4 + 3465x^2 - 63) \\
P_{11}(x) &= \tfrac{1}{256}(88179x^{11} - 230945x^9 + 218790x^7 - 90090x^5 + 15015x^3 - 693x)
\end{aligned}$$

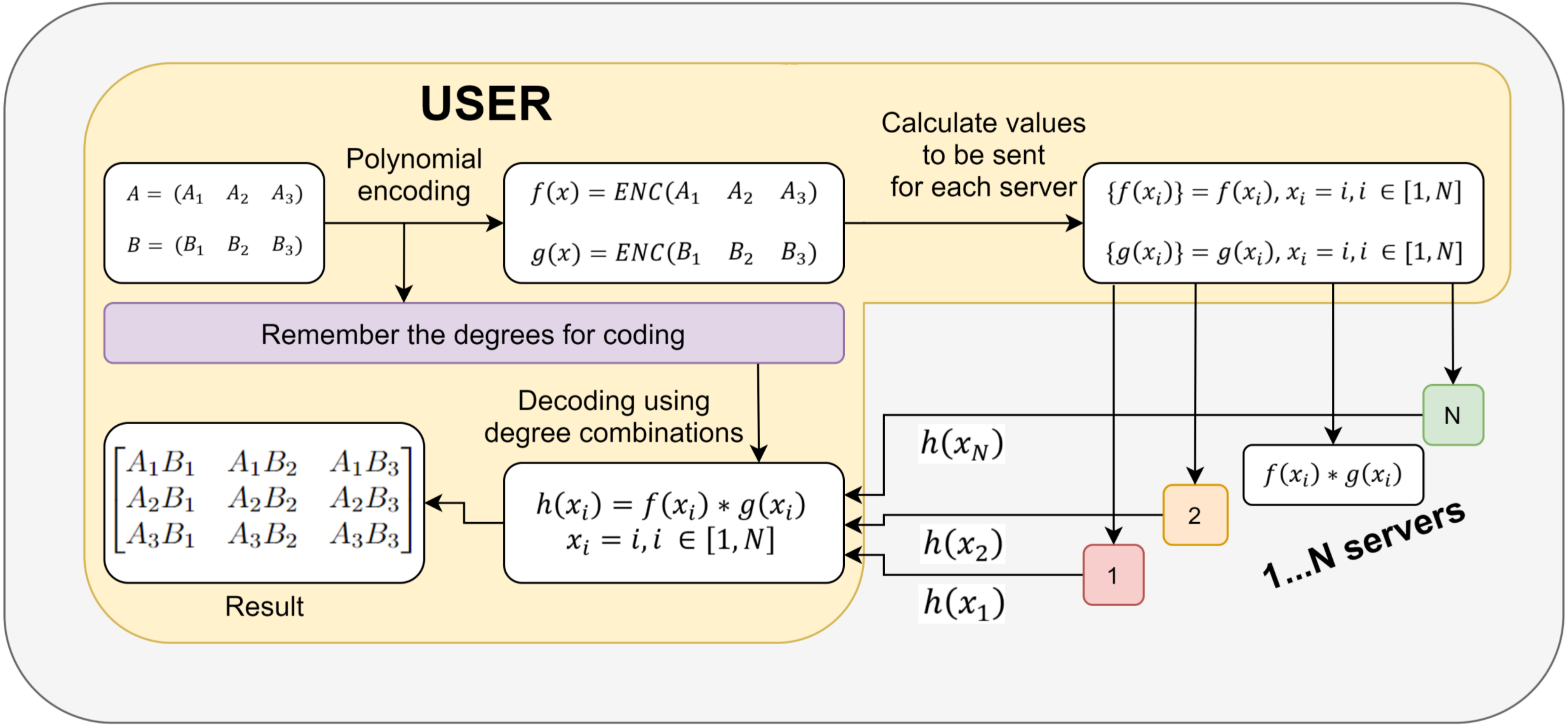
Formulation of the problem

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & B_2 & B_3 \end{bmatrix}$$ **- User has**

$$AB = \begin{bmatrix} A_1B_1 & A_1B_2 & A_1B_3 \\ A_2B_1 & A_2B_2 & A_2B_3 \\ A_3B_1 & A_3B_2 & A_3B_3 \end{bmatrix}$$ **- We need to calculate**

How to calculate AB?

GASP Block-diagram



Algorithm

Consider K = L = 3, T = 2

$$f(x) = A_1 x^{\alpha_1} + A_2 x^{\alpha_2} + A_3 x^{\alpha_3} + R_1 x^{\alpha_4} + R_2 x^{\alpha_5}$$
$$g(x) = B_1 x^{\beta_1} + B_2 x^{\beta_2} + B_3 x^{\beta_3} + S_1 x^{\beta_4} + S_2 x^{\beta_5}$$

$$h(x_i) = f(x_i) * g(x_i), x_\mathrm{i} \epsilon [1, N]$$

N points: $\begin{cases} h(x_1) = A_1 B_1 {x_1}^0 + \ldots + A_3 B_3 {x_1}^8 + C_9 {x_1}^9 + \ldots + C_{22} {x_1}^{22} \\ \circ \quad \circ \quad \circ \\ h(x_N) = A_1 B_1 {x_N}^0 + \ldots + A_3 B_3 {x_N}^8 + C_9 {x_N}^9 + \ldots + C_{22} {x_N}^{22} \end{cases}$, where

$h(x_N) - we\ know\ in\ N\ points, N\ -\ an\ amount\ of\ servers$

$h(x_N)\ -\ some\ matrix\ coding\ the\ part\ of\ \ result$

Degree table: sum of degrees combination

Left top square - **decodability**

	β_1	β_2	β_3	β_4	β_5
α_1	$\alpha_1+\beta_1$	$\alpha_1+\beta_2$	$\alpha_1+\beta_3$	$\alpha_1+\beta_4$	$\alpha_1+\beta_5$
α_2	$\alpha_2+\beta_1$	$\alpha_2+\beta_2$	$\alpha_2+\beta_3$	$\alpha_2+\beta_4$	$\alpha_2+\beta_5$
α_3	$\alpha_3+\beta_1$	$\alpha_3+\beta_2$	$\alpha_3+\beta_3$	$\alpha_3+\beta_4$	$\alpha_3+\beta_5$
α_4	$\alpha_4+\beta_1$	$\alpha_4+\beta_2$	$\alpha_4+\beta_3$	$\alpha_4+\beta_4$	$\alpha_4+\beta_5$
α_5	$\alpha_5+\beta_1$	$\alpha_5+\beta_2$	$\alpha_5+\beta_3$	$\alpha_5+\beta_4$	$\alpha_5+\beta_5$

Right top, right bottom, left bottom **- sequrity**

	$\beta_1=0$	$\beta_2=3$	$\beta_3=6$	$\beta_4=9$	$\beta_5=10$
$\alpha_1=0$	0	3	6	9	10
$\alpha_2=1$	1	4	7	10	11
$\alpha_3=2$	2	5	8	11	12
$\alpha_4=9$	9	12	15	18	19
$\alpha_5=12$	12	15	18	21	22

The degree table used for decoding

Decoding

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{22} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{18} & x_{18}^2 & \cdots & x_{18}^{22} \end{pmatrix} \begin{pmatrix} A_1 B_1 \\ \vdots \\ R_5 S_5 \end{pmatrix} = \begin{pmatrix} h(x_0) \\ \vdots \\ h(x_{18}) \end{pmatrix}$$

$$V * a = h \Rightarrow$$

$$V^{-1} * V * a = V^{-1} * \mathrm{h} \Rightarrow$$

$$a = V^{-1} * \mathrm{h}$$

Vandermond matrix has powers from the degree table for 18 x values (19 servers) – 18 by 18 matrix

Performance of GASP

Download rate – main performance metric

Let N – amount of servers

$$R = \frac{SIZE\_OF\_DESIRED\_RESULT_{bits}}{TOTAL\_DOWNLOADET\_DATA_{bits}} = \mathrm{R(K, L, T), where}$$

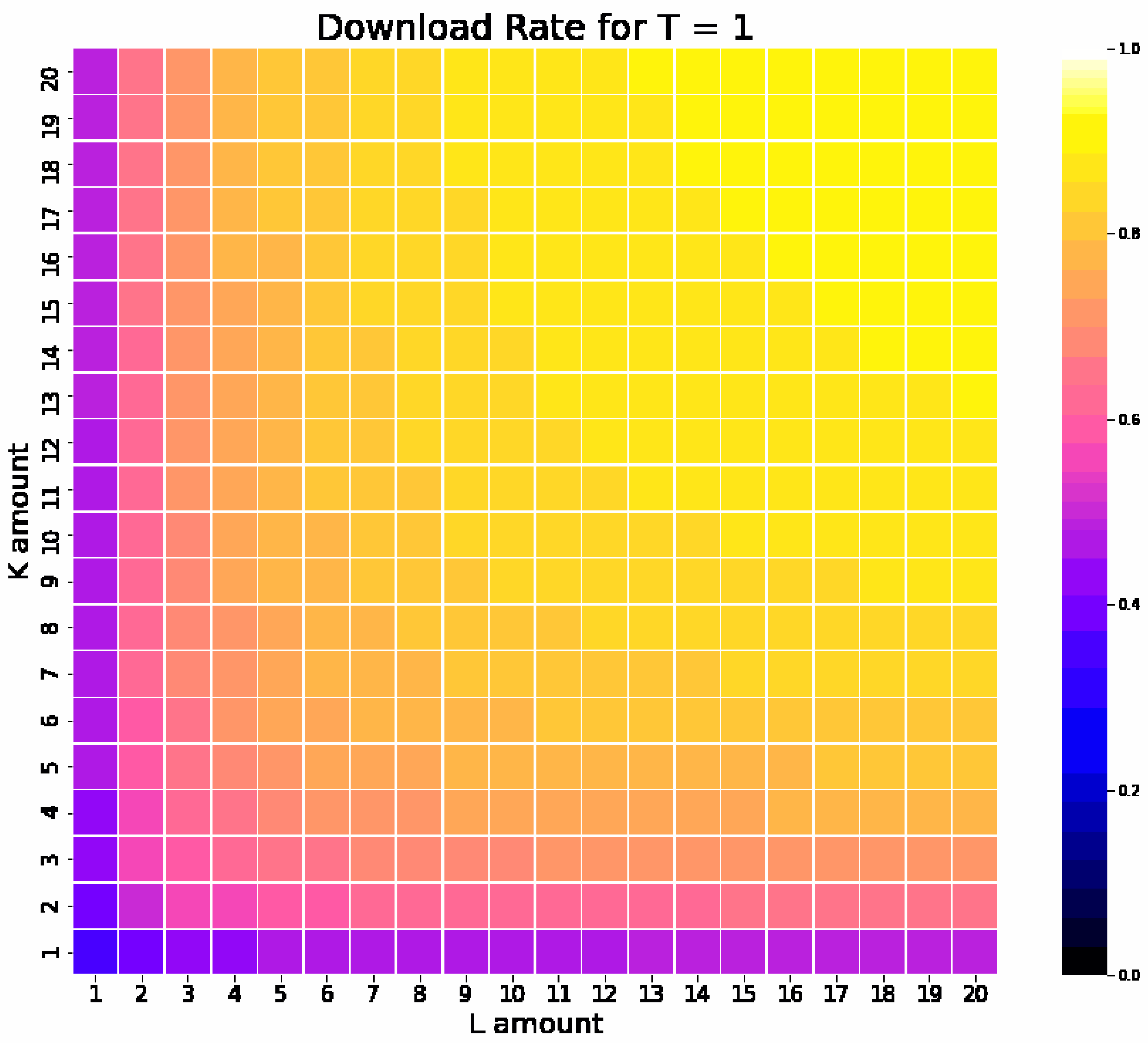
K $-$ A submatrixies
L $-$ B submatrixies
T $-$ T of N servers which may collude

Polynomial for Big and Small T

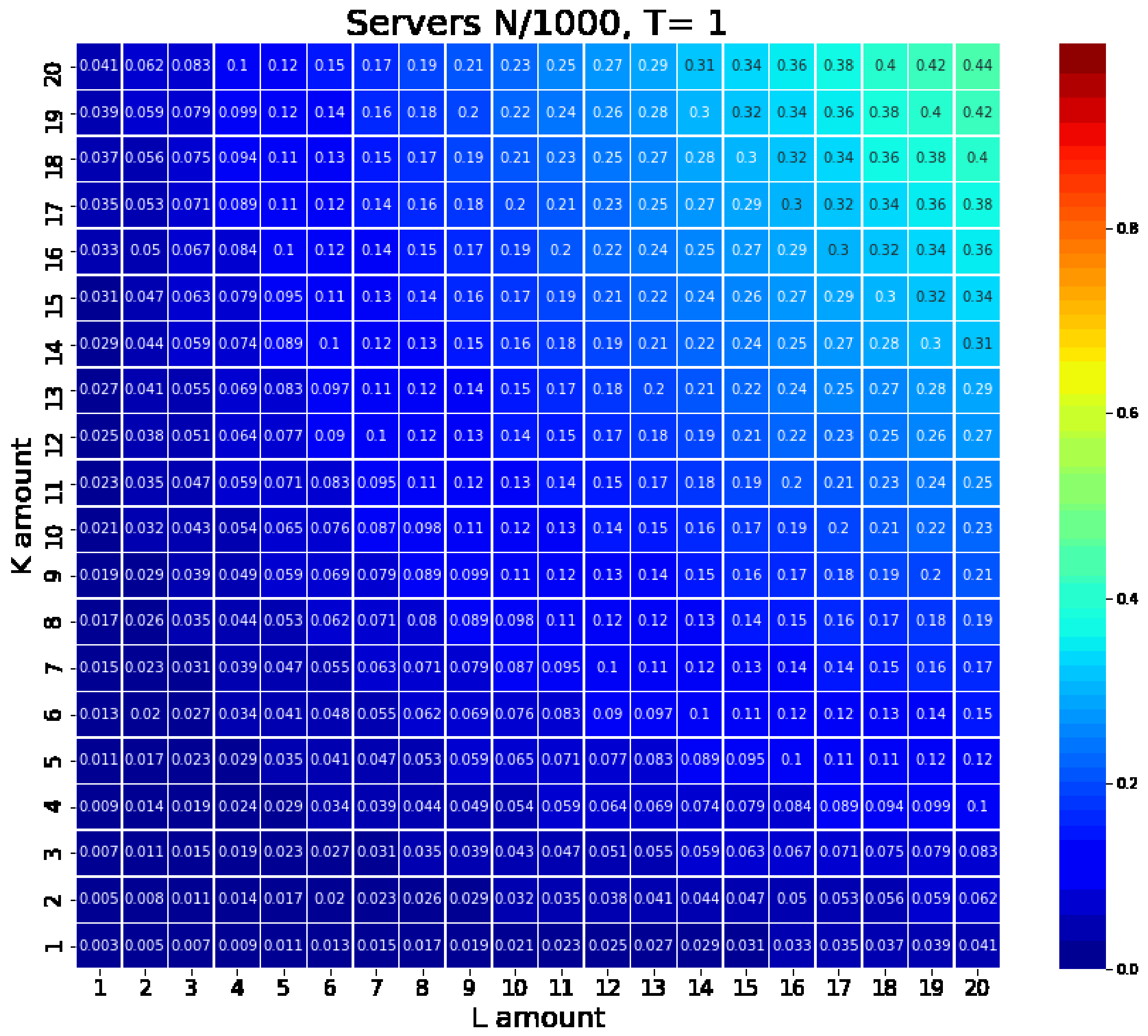
$$GASP = \begin{cases} GASP_{small} \; if \; T < \min\{K, L\} \\ GASP_{big} if \; T \geq \min\{K, L\} \end{cases} - GASP_{small} \;\; and GASP_{big} \;\; Criteria$$

$$N, R \begin{cases} If \; GASP_{small} \Rightarrow \begin{cases} N_{small} = F_{small}(K, L, T) \\ R_{small} = F_{small}(K, L, T) \end{cases} \\ If \; GASP_{big} \quad \Rightarrow \begin{cases} N_{big} = F_{big}(K, L, T) \\ R_{big} = F_{big}(K, L, T) \end{cases} \end{cases}$$

Download rate GASP (big + small)



An amount of servers N(K, L, T)



What is already implemented?

- Generating matrix A and B with K and L sizes:

$$A = A[K,1] \quad B = B[1,L]$$

- Generating submatrices A_k and $B_l, k \in [1,K], l \in [1,L]$:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_K \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & \cdots & B_L \end{bmatrix}$$

- Generating submatrices S_t and R_t for T - security, $t \in [1,T]$,

size $A_k =$ size R_t, size $B_k =$ size S_t :

$$R = \begin{bmatrix} R_1 \\ \dots \\ R_T \end{bmatrix}$$

$$S = \begin{bmatrix} S_1 & \dots & S_T \end{bmatrix}$$

What is already implemented?

- GASP algorithms, which consists of $GASP_{big+small}$ for getting of degree sets and the optimal quantity of servers N:

$$\{\alpha\} = [\alpha_1 \quad \dots \quad \alpha_{T+K}]$$
$$\{\beta\} = [\beta_1 \quad \dots \quad \beta_{T+L}]$$
$$N = \mathrm{GASP}(K, L, T)$$

- Generating:
  - degree table
  - polynomials f(x) and g(x) for all servers:

$$f(x_i), g(x_i), x_i = i, i \in [1, N]$$

- Distributed multiplication of f(x) by g(x) and getting h(x):

$$h(x_i) = f(x_i) * g(x_i), x_i = i, i \in [1, N]$$

- Getting result of multiplication with using inverse Vandermond matrix to restore result coefficients:

$$\begin{bmatrix} A_1B_1 & A_1B_2 & A_1B_3 \\ A_2B_1 & A_2B_2 & A_2B_3 \\ A_3B_1 & A_3B_2 & A_3B_3 \end{bmatrix}$$

The main difficulties

- problems with the justification of mathematical aspects
- the need to study additional functionality of used libraries
- problems associated with selection the Vandermond matrix (regular Numpy function np.vander() experienced overflow)

Future plans

- To compare different ways of calculations:
  - Strassen algorithm
  - MPI matrix multiplication
  - Regular python functions
  - Etc…

- Try experimenting on real servers

Thank you for your attention!

Contacts:
Alexander.Blagodarnyi@skoltech.ru
Stanislav.Krikunov@skoltech.ru
Mikhail.Konovalov@skoltech.ru
Mariia.Kopylova@skoltech.ru