# A Secure PolyDot Matrix Multiplication Approach For Distributed Computing

***Abstract*—**

***Index Terms*—**

## I. System Models

### A. Symbolism

Before delving into the discussion of the system model, we would like to present some commonly used symbols that will be employed consistently throughout.We use uppercase letters (A, B, X, Y, ...) to denote matrices. A sufficiently large finite Galois field of size $|\mathbb{F}|$ is denoted as $\mathbb{F}$. With a real number a, let $\lceil a \rceil$ denote the ceiling function of a, which is the smallest integer greater than or equal to a. I(X) can be considered as a function used to quantify the amount of information gained with respect to the event X.

$$I(X) = -log_2(P(X)) \tag{1}$$

. We also represent $I(X, Y)$ as the amount of mutual information between two variables X and Y.
The function $H(X)$ is the information entropy function for the parameter X, understood as the measure of uncertainty that information carries that can be represented as:

$$H(X) = -\sum_i P(x_i) \log_2(P(x_i)) \tag{2}$$

with $P(x_i)$ is the probability of the event $x_i \in X$. Additionally, we have the following definitions: $H(X, Y)$ represents information contained in the pair $(X, Y)$, and $H(X|Y$ represents the information in X that is not present in Y. From here, we have related expressions for I and H as follows:

$$I(X, Y) = I(Y, X) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{3}$$

For distributed computing systems, we denote the number of workers as N, $R_c$ as the minimum recovery threshold or the minimum number of workers necessary to recover the result, $P_C$ as the number of colluding workers, B as the set of Byzantine workers (The definitions of colluding worker and Byzantine worker will be introduced in the following section) and $C_l$ as the communication load defined as as the quantity of information retrieved from the workers.

### B. PolyDot Code without security

- m = t , n = s , p = d

In this section, we will introduce a coded distributed matrix-matrix multiplication problem, named PolyDot. This method represents a trade-off between recovery threshold and communication load compared to Polynomial Code and MatDot Code. Master node needs compute Z = AB, where X,Y are data matrices, we also assume that the number of colluding workers and the number of Byzantine workers are 0. First, master node partition the matrix $X \in \mathbb{F}^{M \times N}$ and $Y \in \mathbb{F}^{N \times P}$ into submatrices as shown below:

$$X = \begin{bmatrix} X_{1,1} & \dots & X_{1,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \dots & X_{m,n} \end{bmatrix} \quad Y = \begin{bmatrix} Y_{1,1} & \dots & Y_{1,p} \\ \vdots & \ddots & \vdots \\ Y_{n,1} & \dots & Y_{s,p} \end{bmatrix}$$

where $m \mid M, n \mid N, p \mid P$, $X_{i,j} \in \mathbb{F}^{\frac{M}{m} \times \frac{N}{n}}$, $Y_{k,l} \in \mathbb{F}^{\frac{N}{n} \times \frac{P}{p}}$
The resulting matrix Z = XY is represented as follows:

$$Z = \begin{bmatrix} Z_{1,1} & \dots & Z_{1,p} \\ \vdots & \ddots & \vdots \\ Z_{m,1} & \dots & Z_{m,p} \end{bmatrix}$$

with $Z_{u,v} = \sum_{k=1}^{n} X_{u,k} Y_{k,v}, \ u \in \{1, 2, \dots, m\}, \ v \in \{1, 2, \dots, p\}$

*1) Encode:* The Master node encodes the submatrices of X and Y using the functions $F$ and $G$ below:

$$F(z) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{i,j} z^{n(i-1)+j-1} \tag{4}$$

$$G(z) = \sum_{k=1}^{n} \sum_{l=1}^{p} Y_{k,l} z^{n-k+mn(l-1)} \tag{5}$$

where $F$ and $G$are encode function satisfied $F : \mathbb{F}^{\frac{M}{m} \times \frac{N}{n}} \rightarrow \mathbb{F}^{\frac{M}{m} \times \frac{N}{n}}$ and $G : \mathbb{F}^{\frac{N}{n} \times \frac{P}{p}} \rightarrow \mathbb{F}^{\frac{N}{n} \times \frac{P}{p}}$. Subsequently, the master node sends to the i-th worker the expressions $F(z_i)$ and $G(z_i)$ , where it is noted that $z_i$ is distinct and non-zero for i $\in$ 1,2,...,p

*2) Task computing:* i-th worker calculate $F(z_i)G(z_i)$. As soon as worker $i$ completes the computation, it will send the result back to the master node, noting that the returned results from the workers may vary in speed depending on each worker's computational capabilities, and the returned results may not be accurate.

$$\begin{aligned} F(z)G(z) &= \left( \sum_{i=1}^{m} \sum_{j=1}^{n} X_{i,j} z^{n(i-1)+j-1} \right) \\ &\quad \times \left( \sum_{k=1}^{n} \sum_{l=1}^{p} Y_{k,l} z^{n-k+mn(l-1)} \right) \\ &= \sum_{i,j,k,l} X_{i,j} Y_{k,l} z^{n(i-1)+j-1+n-k+mn(l-1)} \end{aligned}$$

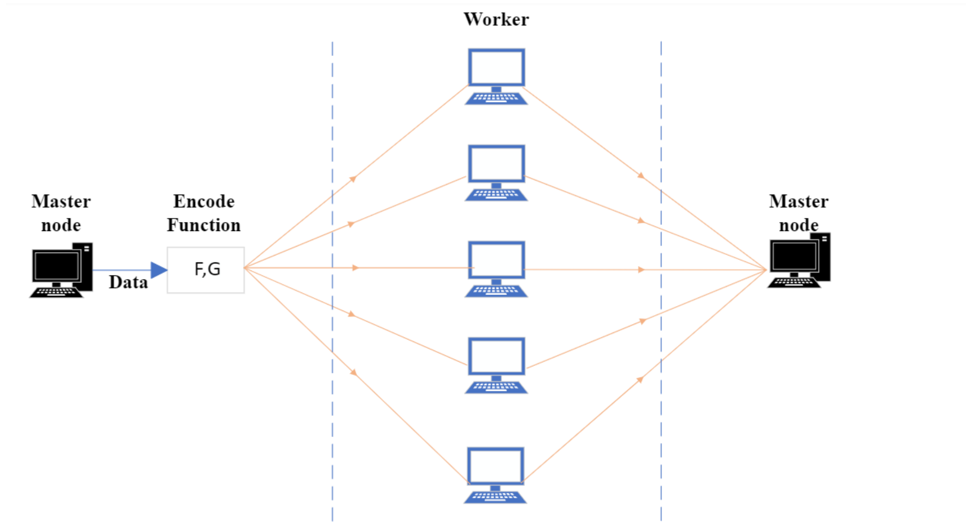

Fig. 1. General overview of distributed computing systems.

The coefficient of $Z_{u,v}$ in the polynomial corresponding to the exponent z is $n(u-1)+n-1+mn(v-1)$ in the case where the index j = k in $X_{i,j}$ and $Y_{k,l}$

*3) Result recovering (Decoding):* The master node will collect the fastest R results returned by the workers. Afterward, the master node will recover the result $Z_{i,j}$ with $i \in \{1,2,\ldots,m\}$, $j \in \{1,2,\ldots,p\}$ based on polynomial interpolation.

Since the degree of the polynomial is mnp + n - 2, when applying the polynomial interpolation method, a minimum of mnp + n - 1 values is required to achieve the fastest results from the workers for the recovery of the matrix Z. Therefore, the recovery threshold is:

$$P_R = mnp + n - 1 \tag{7}$$

and communication load is:

$$C_L = P_R \frac{MP}{mp} \tag{8}$$

Example 1: m = 1, n = 2, p = 3

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} \end{bmatrix} \quad Y = \begin{bmatrix} Y_{1,1} & Y_{1,2} & Y_{1,3} \\ Y_{2,1} & Y_{2,2} & Y_{2,3} \end{bmatrix}$$

$$Z = XY = \begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} \end{bmatrix} \tag{9}$$

We define the encode funtion:

$$F(z) = \sum_{i=1}^{1}\sum_{j=1}^{2} X_{i,j} z^{2(i-1)+j-1}$$
$$= X_{1,1} + X_{1,2}z$$
$$G(z) = \sum_{k=1}^{2}\sum_{l=1}^{3} Y_{k,l} z^{2-k+2*(l-1)}$$
$$= Y_{1,1}z + Y_{1,2}z^3 + Y_{1,3}z^5 + Y_{2,1} + Y_{2,2}z^2 + Y_{2,3}z^6$$

Consider expression $F_z = F(z)G(z)$:

| Coefficient | Exponent of $z$ |
|---|---|
| $Z_{1,1}$ | 1 |
| $Z_{1,2}$ | 3 |
| $Z_{1,3}$ | 5 |

Retrieve distinct points $z_1, z_2, ...z_p$ with $z_i \in \mathbb{F}$, $i \in \{1,\ldots,P\}$. Then, master node send $F(z_i)$and $G(z_i)$ to ith-worker i. Then, i-th worker will compute $F(z_i)G(z_i)$ and send the result back to the master node. Because the highest degree of the polynomial$F_z$ is 11, it is necessary to have a minimum of 12 equations to determine the polynomial $F_z$, or in other words, to calculate its coefficients. When the fastest 12 nodes complete the computation and return the results, the master node will restore the coefficients $Z_{u,v}$ of matrices Z using polynomial interpolation.

**Remark:** The PolyDot code method presented above represents a trade-off between recovery threshold and communication load. However, in practical distributed computing systems, issues related to malicious workers and colluding workers are common challenges. Malicious (Byzantine) workers may provide deliberately incorrect computations to sabotage the overall computation results, while colluding workers, typically located at edge nodes, collaborate to share information gathered from the input data. The Polydot Code mentioned above is not designed to withstand colluding workers as well as Byzantine workers. Therefore, in the following section, we will introduce the Secure PolyDot Code (SPC), which addresses both colluding worker and Byzantine worker scenarios.

## II. PolyDot with Security.

### A. *Generalized Polydot with Security*

In the preceding section, we analyzed the PolyDot code under the assumption of an absence of colluding and malicious workers. Nevertheless, in practical scenarios, colluding workers may conspire to exchange information, and malicious workers might provide inaccurate results, thereby influencing

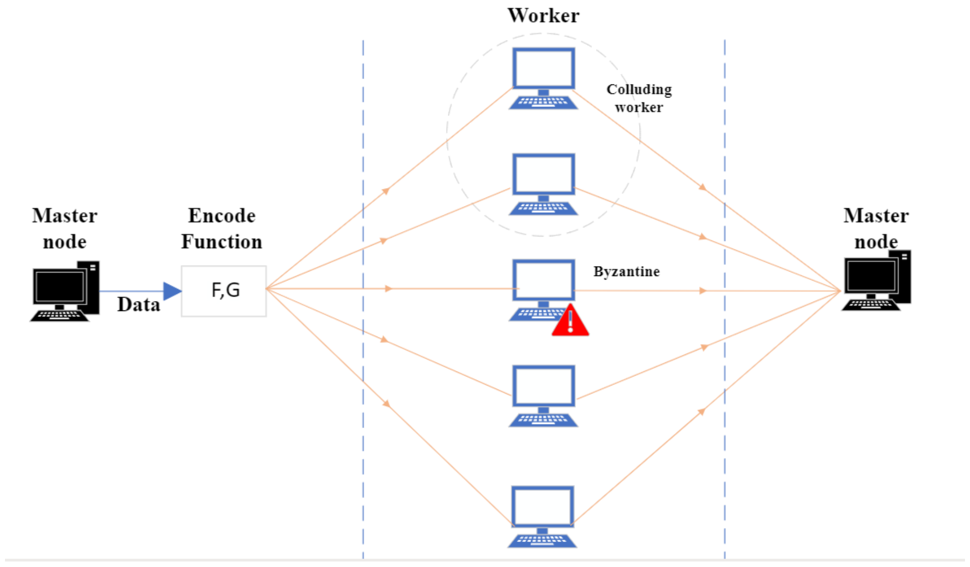

Fig. 2. A description of a distributed computing system with security issues. Due to varying computational capabilities and different return times for results from individual workers, the overall computation time of the master node is affected. Colluding workers may share information to obtain data details, while some workers might intentionally send inaccurate results with the goal of distorting the final outcome.

the computational process of the master node with deviations. Consequently, in this section, we present the Secure PolyDot Code (SPC) as a solution to mitigate the impact of colluding workers and malicious workers. SPC is carried out as follows:

*1) Encode:* To accomplish this objective, we add rows or columns containing random elements to matrices A and B, respectively. From there, we consider two distinct cases:

**a) Case:** $n < m$**:**

First, we defination:

$$\Delta_{P_C} = \left\lceil \frac{P_C}{n} \right\rceil$$

Matrix $X^*$ will be represented:

$$X^* = \begin{bmatrix} X_{1,1} & \dots & X_{1,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \dots & X_{m,n} \\ R_{1,1} & \dots & R_{1,n} \\ \vdots & \ddots & \vdots \\ R_{\Delta_{P_C},1} & \dots & R_{\Delta_{P_C},s} \end{bmatrix}$$

while the $n \times p^*$ augmented matrix $Y^* = [Y \;\; R_0]$ with $p^* = p + \Delta_{P_C}$ is obtained as

$$Y^* = \begin{bmatrix} Y_{1,1} & \dots & Y_{1,p} & R_{n,1} & \dots & R_{n,\Delta_{P_C}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Y_{n,1} & \dots & Y_{n,p} & Y_{1,1} & \dots & R_{1,\Delta_{P_C}} \end{bmatrix}$$

The encoding functions $F^*$ and $G^*$ are rewritten as follows:

$$F^*(z) = \sum_{i=1}^{m}\sum_{j=1}^{n} X^*_{i,j} z^{i-1+m(j-1)} + \sum_{i=1}^{m}\sum_{j=n+1}^{n^*} X^*_{i,j} z^{i-1+m(j-1)} \tag{10}$$

$$\begin{aligned} G^*(z) = &\sum_{k=1+\Delta_{P_C}}^{p}\sum_{l=1}^{n^*} Y^*_{k,l} z^{(n^*-k)m+mn^*(l-1)} \\ &+ \sum_{k=1}^{p}\sum_{l=1}^{n^*} Y^*_{k,l} z^{t(n^*p-\Delta_{P_C})+p(\Delta_{P_C}-k)+l-1} \end{aligned} \tag{11}$$

**b) Case:** $n \geq m$**:**

$$\Delta_{P_C} = \left\lceil \frac{P_C}{\min\{m,p\}} \right\rceil$$

Matrix $X^* \in \mathbb{F}^{t \times n^*}$ will take the form $[X \;\; R]$ with $n^* = n + \Delta_{P_C}$ is obtained as

$$X^* = \begin{bmatrix} X_{1,1} & \dots & X_{1,n} & R_{1,1} & \dots & R_{1,\Delta_{P_C}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ X_{m,1} & \dots & X_{m,n} & X_{m,1} & \dots & R_{m,\Delta_{P_C}} \end{bmatrix}$$

and the matrix $Y^* = [R \;\; Y] \in \mathbb{F}^{n^* \times d}$ is represented as:

$$Y^* = \begin{bmatrix} R_{\Delta_{P_C},1} & \dots & R_{\Delta_{P_C},p} \\ \vdots & \ddots & \vdots \\ R_{1,1} & \dots & R_{1,p} \\ Y_{1,1} & \dots & Y_{1,p} \\ \vdots & \ddots & \vdots \\ Y_{n,1} & \dots & Y_{n,p} \end{bmatrix}$$

And the encoding function in this case will be represented:

$$\begin{aligned} F^*(z) = &\sum_{i=1}^{m}\sum_{j=1}^{n} X^*_{i,j} z^{i-1+m(j-1)} \\ &+ \sum_{i=1}^{m}\sum_{j=n+1}^{n^*} X^*_{i,j} z^{i-1+m(j-1)}, \end{aligned} \tag{12}$$

$$G^*(z) = \sum_{k=1+\Delta_{P_C}}^{p} \sum_{l=1}^{n^*} Y^*_{k,l} z^{(n^*-k)m+mn^*(l-1)} + \sum_{k=1}^{\Delta_{P_C}} \sum_{l=1}^{p} Y^*_{k,l} z^{m(n^*p-\Delta_{P_C})+p(\Delta_{P_C}-k)+l-1}. \tag{13}$$

*2) Verification key:* As mentioned above, malicious workers can return incorrect results, causing the obtained result matrix Z to be inaccurate. Therefore, there is a need for a verification method to determine whether the results returned by worker i are correct or not. In this step, the master node will generate a key $k_i$ for each encoding function and send it to the i-th worker before obtaining the encoded results.
Let $n_0 = \frac{N}{n}$, $m_0 = \frac{M}{m}$, $p_0 = \frac{P}{p}$. In case: $m_0 < p_0$ The master node will generate the key $k_i$ corresponding to $F^*(z_i)$ ,where the vectors $k_i^T \in \mathbb{F}^{m^*}$ R, as follows:

$$R_i = k_i F^*(z_i) \tag{14}$$

In case $m_0 \geq p_0$,master node will generate vector $k_i \in \mathbb{F}^{p^*}$ corresponding to $G^*(z_i)$ as follow:

$$R_i = G^*(z_i) k_i \tag{15}$$

with elements belonging to the vector $k_i$ is random number belonging to the set of real numbers

*3) Task computing:* After generating the key, the master code will send $F^*(z_i)$ and $G^*(z_i)$ to each i-th worker. The workers perform computations:

$$F_z(z_i) = F^*(z_i) G^*(z_i) \tag{16}$$

then return results to master node immediately after completing the computation.

*4) Key checking:* After receiving the result $F_z(z_i)$ returned by the i-th worker, the master code will proceed to perform a check:

$$R_i G^*(z_i) = k_i F^*(z_i) G^*(z_i) \tag{17}$$

or

$$F^*(z_i) R_i = F^*(z_i) G^*(z_i) k_i \tag{18}$$

corresponding to cases $m^* < p^*$ and $m^* \geq p^*$.
Based on this verification process, the master node can accurately identify Byzantine workers. If the result does not match, indicating that the worker may be a malicious actor, the computed result will be discarded to prevent any impact on the overall outcome. Conversely, if the results match, the master node will accept and incorporate this result.

*5) Recover result:* Similar to the recovery step mentioned above, the master node will collect the results returned from the worker nodes to determine the coefficients through polynomial interpolation. It is important to note that these returned results must satisfy the key-checking step to be considered valid. If the returned result is accurate, we add it to the $\mathbb{R}_\mathbb{Z}$, and when $|R_z| \geq R_C$, the master node proceeds with decoding to obtain the final matrix.

Example 2: Similar to the example 1, with M = 3, N = 4, P = 6, m=1, n=2, p=3 and we assume the number of and colluding workers $P_c = 2$, given that $m < n$, we redefine the encoding function as follows:

$$\begin{aligned} F^*(z) &= \sum_{i=1}^{1} \sum_{j=1}^{2} X_{i,j} z^{i-1+j-1} + \sum_{i=1}^{1} \sum_{j=3}^{4} X_{i,j} z^{i-1+j-1}, \\ &= X_{1,1} + X_{1,2} z + X_{1,3} z^2 + X_{1,4} z^3 \\ G^*(z) &= \sum_{k=3}^{4} \sum_{l=1}^{3} Y_{k,l} z^{4-k+4(l-1)} + \sum_{k=1}^{2} \sum_{l=1}^{3} Y_{k,l} z^{10+3(2-k)+l-1} \\ &= Y_{1,1} z^{13} + Y_{1,2} z^{14} + Y_{1,3} z^{15} + Y_{2,1} z^{10} \\ &+ Y_{2,2} z^{11} + Y_{2,3} z^{12} + Y_{3,1} z + Y_{3,2} z^5 \\ &+ Y_{3,3} z^9 + Y_{4,1} + Y_{4,2} z^4 + Y_{4,3} z^8 \end{aligned}$$

Afterward, since $\frac{P}{p} < \frac{M}{m}$ the key is defined as:

$$k_i = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

with $a_1, a_2 \in \mathbb{F}$
Afterwards, the master node will generate a verification expression (15), and then send F and G to the corresponding i-th worker. The worker that completes the computation fastest and returns the result will be checked by the master node as illustrated in (18) From this, the accuracy of the result is ensured, and the matrix Z is successfully recovered.

## *B. Recovery Threshold and computation load*

For the SPC method, as we have considered two cases for both s and t, the recovery threshold of the algorithm will be represented as follows:
**Case : $n < m$:** For a given security level $P_c < P$, t, s, d, $n^*, p^*$are given as mentioned above, the SPC code has the recovery threshold $P_R$ as follows:

$$\begin{cases} m^* n(p+1) + n\Delta_{P_C} - 1, & \text{if } P_C \geq 1 \text{ and } \Delta_{P_C} = \frac{P_C}{n}, \\ m^* n(p+1) - n\Delta_{P_C} + 2P_C - 1, & \text{if } P_C \geq 1 \text{ and } \Delta_{P_C} > \frac{P_C}{n}, \end{cases} \tag{19}$$

**Case : $n \geq m$:** For a given security level $P_c < P$, t, s, d, $n^*, p^*$are given as mentioned above, the SPC code has the recovery threshold as follows:

$$P_R = m(n^* p - \Delta_{P_C}) + mn + 2P_C - 1 \tag{20}$$

And communication load in both case:

$$C_L = P_R \frac{M^* P^*}{m^* p^*}$$

**Prove:** Firstly, it can be observed that the exponents of z $X_{u,r} Y_{r,v}$ with $r \in \{1, \ldots, p\}$ all equal .... From there, Z... is the corresponding coefficient of $z^{\cdot}$.. in the .... We need to demonstrate that the exponents of .... do not exist within 14, 23, or 24.

[1]

**Algorithm 1** SPC Algorithm

```
1: Input: $X, Y, M, N, P, m, n, p, P_c, \Delta_{P_c}$
2: Output: $Z$
3: [I] Encode:The master node sequentially divides the matrices X and Y into mn and np submatrices, respectively.
4: if $n < m$ then
       For $i = 1$ to $N$: master node encode X and Y by Eq.(10) and Eq.(11)
5: else
       For $i = 1$ to $N$: master node encode X and Y by Eq.(12) and Eq.(13)
6: end if
7: [II] Verification key The master node will generate corresponding keys for the encode functions.
8: if $\frac{M}{m} < \frac{P}{p}$ then
       Master node generate keys by Eq.(14)
9: else
       Master node generate keys Eq.(15)
10: end if
11: Master node send $F(z_i)$ and $G(z_i)$ to i-th worker
12: [III] Task computing: After receiving $F$ and $G$ from the master node, the worker calculates $F_z = FG$
13: For $i = 1$ to $N$: i-th worker calculates $F_z(z_i) = F(z_i)G(z_i)$
14: [IV] Key checking: Master node check result $F_z(z_i)$ from i-th worker by Eq.(17) in case $\frac{M}{m} < \frac{P}{p}$ and by Eq.(18) in the other case.
15: [V] Result recovering: Master node recover matrix Z
16: if $R_Z \geq R_c$ then
       master node recovery Z by using polynomial interpolation
17: end if
```

## References

[1] Q.-U.-A. Nadeem, A. Kammoun, A. Chaaban, M. Debbah, and M.-S. Alouini, "Intelligent reflecting surface assisted wireless communication: Modeling and channel estimation," *arXiv preprint arXiv:1906.02360*, 2019.