# FunBic-CCA: Function Secret Sharing for Biclusterings Applied to Cheng and Church Algorithm

Shokofeh VahidianSadegh[1][a], Alberto Ibarrondo[2][b] and Lena Wiese[1][c]
[1]*Computer Science Department, Goethe University Frankfurt, Frankfurt am Main, Germany*
[2]*Arcium, France*

Keywords: Biclustering Algorithm, Privacy-Preserving AI, Secure Multi-Party Computation, Function Secret Sharing.

Abstract: High-throughput technologies (e.g., the microarray) have fostered the rapid growth of gene expression data collection. These biomedical datasets, increasingly distributed among research institutes and hospitals, fuel various machine learning applications such as anomaly detection, prediction or clustering. In particular, unsupervised classification techniques based on biclustering like the Cheng and Church Algorithm (CCA) have proven to adapt particularly well to gene expression data. However, biomedical data is highly sensitive, hence its sharing across multiple entities introduces privacy and security concerns, with an ever-present threat of accidental disclosure or leakage of private patient information. To address such threat, this work introduces a novel, highly efficient privacy-preserving protocol based on secure multiparty computation (MPC) between two servers to compute CCA. Our protocol performs operations relying on an additive secret sharing and function secret sharing, leading us to reformulate the steps of the CCA into MPC-friendly equivalents. Leveraging lightweight cryptographic primitives, our new technique named FunBic-CCA is first to exploit the efficiency of function secret sharing to achieve fast evaluation of the CCA biclustering algorithm.

## 1 INTRODUCTION

The abundance of biomedical data, thanks to the rapid advancement and readily available high-throughput technologies, provides great opportunities for the knowledge discovery. Machine Learning (ML) has been widely used to categorise and classify large amounts of data. Among ML methods, the traditional one-way clustering methods perform analysis directly on specific characteristics. More adaptable approaches to gene expression data are biclustering algorithms that have become prevalent with a wide variety of applications, ranging from bioinformatics to recommender systems, and many more. Biclustering based on *Cheng and Church Algorithm* (CCA) groups a set of genes and conditions with a high similarity score. This was the first study to apply biclustering algorithms over gene expression data (Cheng and Church, 2000).

Despite this potential of large biomedical data analysis, a recent influx of malicious attacks on these sensitive resources has been increasingly reported. Since that, there exists an inherent, simultaneous contradiction between utility and privacy. Accordingly, there must be improved data protection and preventive information leakage measures against unauthorised users.

Considering this problem, we propose FunBic-CCA (**Fun**ction secret sharing for **Bic**lusterings Applied to **C**heng and **C**hurch **A**lgorithm), a privacy-preserving framework for gene expression data analysis by biclustering algorithms with well-established schemes, *additive secret sharing* (SS (Demmler et al., 2015)) and *Function Secret Sharing* (FSS (Boyle et al., 2015)).

After discussing preliminaries from privacy-preserving biclustering analysis methods (Section 2), we detail FunBic-CCA framework (Section 3), which provides core contributions below:

- An end-to-end privacy-preserving framework based on additive secret sharing and function secret sharing schemes;
- Support for privacy-preserving evaluation of both linear and non-linear operations with 100% correctness in less than 500 seconds;

[a] https://orcid.org/0000-0002-6464-6842
[b] https://orcid.org/0000-0003-4079-4127
[c] https://orcid.org/0000-0003-3515-9209



VahidianSadegh, S., Ibarrondo, A. and Wiese, L.
FunBic-CCA: Function Secret Sharing for Biclusterings Applied to Cheng and Church Algorithm.
DOI: 10.5220/0013455400003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 37-48
ISBN: 978-989-758-760-3; ISSN: 2184-7711

- An extensive discussion on the security aspects of FunBic-CCA as well as analyses on its accuracy and efficiency;
- An open-source implementation [1] of our framework, available under the permissive MIT license.

We evaluate the efficacy of FunBic-CCA through experiments and evaluation measures using our framework (Section 4). Finally, we present an overview of related works and elaborate on differences of existing solutions (Section 5) then conclude (Section 6).

## 2 BACKGROUND

We provide an overview of the applied techniques and methods in FunBic-CCA. We first define our notation and proceed with a description of Cheng and Church Algorithm (Cheng and Church, 2000). Afterwards, an overview of the cryptography methods used in our framework is discussed.

**Notation.** We use regular letters for scalars (e.g., s) and boldface letters for matrices in capitals (**A**) and vectors of integers and polynomials (e.g., **a**). $a_{ij}$ denotes the element of the matrix **A** in the $i$-th row and $j$-th column of a polynomial/ vector **a** with $N$ coefficients/ elements. We write $\mathbf{a} \cdot \mathbf{b} = \mathbf{c}$ to denote the element-wise two vector multiplication, while showing matrix multiplication like $\mathbf{A} \times \mathbf{B} = \mathbf{C}$.

In a 2PC paradigm, $P_0, P_1$ denote two computing parties or in general, we consider $P_d$, where $d \in \{0,1\}$. $a_{ij_d}$ is an element of expression matrix **A** for a party $d$. We use $P_a$: $q \Rightarrow P_b$ to show that party $a$ sends value $q$ to party $b$. We employ $\langle x \rangle$ to indicate that value $x$ is arithmetically secret shared into random shares $(x_0, x_1)$, that are held by two computing parties and verify $x = x_0 + x_1 \mod 2^n$. All values are encoded on $n$-bits, which lives in $\mathbb{Z}_{2^n}$. $\mathbb{Z}^*_{n^+}$ represents the range $0 \leq x \leq 2^{n-1} - 1$. We note $\mathcal{U}_{2^n}^{\lceil l_r \cdot l_c \rceil}$ as the uniform distribution in the set $2^n$ with the size of maximum number of elements in a matrix with $l_r$ rows and $l_c$ columns. We use $1_{r2del}, 1_{c2del}, 1_{r2sdel}, 1_{c2sdel}, 1_{r2add}, 1_{c2add}$ to elaborate on the comparison functions (e.g., $1_{x \geq 0} \Leftrightarrow x \geq 0$).

**Cheng and Church Algorithm (CCA).** Cheng and Church (Cheng and Church, 2000) developed a greedy algorithm and used a measure to evaluate the consistency of matrix's elements and to assess the quality of a bicluster. Mean square residue score (MSR) measures the coherence of genes and conditions of a bicluster by taking means of genes and conditions. CCA focuses mainly on particularly large, maximal biclusters with score below a predefined threshold ($\delta$). In other words, given that a matrix **A** and a threshold $\delta \geq 0$, CCA finds $\delta$-biclusters. After finding values of the mean for rows ($\mu_r$), columns ($\mu_c$) and all of elements ($\mu_{ij}$), the residue of the entry $a_{ij}$ for a bicluster $\mathbf{B}_k = (r_k, c_k)$ is

$$r_{ij} = a_{ij} - \mu_r - \mu_c + \mu_{ij}. \tag{1}$$

which is needed to determine MSR (or $H_{ij}$):

$$H_{ij} = \frac{\sum_{i \in r_k} \sum_{j \in c_k} r_{ij}^2}{|r_k||c_k|}. \tag{2}$$

Score for rows, and columns separately can be summarised as follow in which:

$$H_r = \frac{1}{|c_k|} \sum_{j \in c_k} r_{ij}^2, H_c = \frac{1}{|r_k|} \sum_{i \in r_k} r_{ij}^2. \tag{3}$$

The three different phases in CCA include multiple node deletion, single node deletion and node addition. First step is the removal of multiple rows/columns over the input data matrix, which is followed by the single row/column deletion step. At this point, the result may not be maximal and, therefore, a node addition step is required to insert some rows/columns without increasing MSR. CCA takes as input, the expression matrix (**A**), maximum acceptable mean squared residue score threshold ($\delta$), a parameter for multiple node deletion step ($\alpha$), and number of $\delta$-biclusters ($k$) (Cheng and Church, 2000).

**Additive Secret Sharing.** Additive secret sharing (SS) focuses on rings with distributed secret $x$ into two random shares $x_0$ and $x_1$, given that $x = x_0 + x_1 \mod N$ (where $N$ is the ring size) among two computing parties $P_d$. Since then, two parties are able to perform local addition/subtraction of two secret shared values. However, parties require one round of communication to do multiplication by having Beaver's multiplication triples (Beaver, 1992). Lastly, the result is reconstructed by one chosen party, which adds the two secret shares together. We work with $N = 2^n$, where $n \in \{8, 16, 32, 64\}$, by considering native integer types in modern computers that increase the speed in working with the $n$-bit modular arithmetic.

**Function Secret Sharing.** A 2PC Function Secret Sharing (FSS) scheme unlike standard secret sharing of individual elements, shares description of a function $f$ among parties (Boyle et al., 2015). In this scheme, $f$ is split into two shares $(f_0, f_1)$, where $f_d$

[1] https://github.com/ShokofehVS/FunBic-CCA

hides $f$ and $f_0(x)+f_1(x)=f(x)$ for the input value $x$ that is public for both parties. Due to the publicity of the secret input $x$, a random mask $r$ is added $\hat{x}=x+r$ before using in FSS evaluation to keep its privacy. The used mask is generally known by dealer and applied for the key generation to blind the input.

We use Interval Containment (IC) gate for comparison ($\geq \gamma, \gamma=0$) in our work [2]. Our equality check between two secret shares is relied on the Distributed Point Function (DPF) (Definition 2.5 of (Boyle et al., 2019)) as an FSS scheme for the family of all point functions [3].

## 3 FunBic-CCA FRAMEWORK

In this Section, we provide a detailed description of our FunBic-CCA framework.

**Sketch of the Solution.** We consider a scenario by having a data owner and two cloud servers as our computing parties $(P_0, P_1)$:

- *Data owner*: or patient with his genes sequenced as matrix **A** of expression level values $a_{gp}$, consisting of genes **g** under different conditions **p**.
- *Public cloud servers*: by which we intend to execute biclustering on the secret shared gene expression data with our algorithmic design.

Different types of computations are needed in the procedure that can potentially be transferred to the public cloud services. Table 1 contains these steps (further explanations on these steps in Section 2) and Figure 1 represents the general overview of our proposed solution. Prior to the introduction of our main building blocks, we summarise the main generic functions inside our algorithm (with the steps in Table 1) that we intend to protect by MPC based operations (see Section 3.1):

$$\begin{gathered} H_{ij}, H_r, H_c \\ H_r > \alpha H_{ij}, H_c > \alpha H_{ij} \\ H_r[d(i)] \geq H_c[d(j)], H_r[d(i)] < H_c[d(j)] \\ H_r \leq H_{ij}, H_c \leq H_{ij} \end{gathered} \tag{4}$$

[2]IC gate in FSS scheme computes $f_{p,q}(x) = 1_{x\in[p,q]}$ where $p=0, q=2^{n-1}-1$ to finally obtain $1_{0\leq x\leq 2^{n-1}-1}$ (Section 4.1 of (Boyle et al., 2021)).

[3]This states that for a point function $f_{\alpha,\beta}$, $f(\alpha)=\beta$ and $f(x)=0$ for $x\neq\alpha$, where $\alpha\in\mathbb{G}^{in}, \beta\in\mathbb{G}^{out}$, $f:\mathbb{G}^{in}\rightarrow\mathbb{G}^{out}$.

Table 1: Steps within Cheng and Church Algorithm.

| Step | Description |
|---|---|
| 1 | Finding mean squared residue scores $H_r, H_c$ and $H_{ij}$ |
| 2 | Evaluation in multiple node deletion $H_r > \alpha H_{ij}, H_c > \alpha H_{ij}$ |
| 3 | Evaluation in single node deletion $H_r[d(i)] \geq H_c[d(j)], H_r[d(i)] < H_c[d(j)]$ |
| 4 | Evaluation in node addition $H_r \leq H_{ij}, H_c \leq H_{ij}$ |

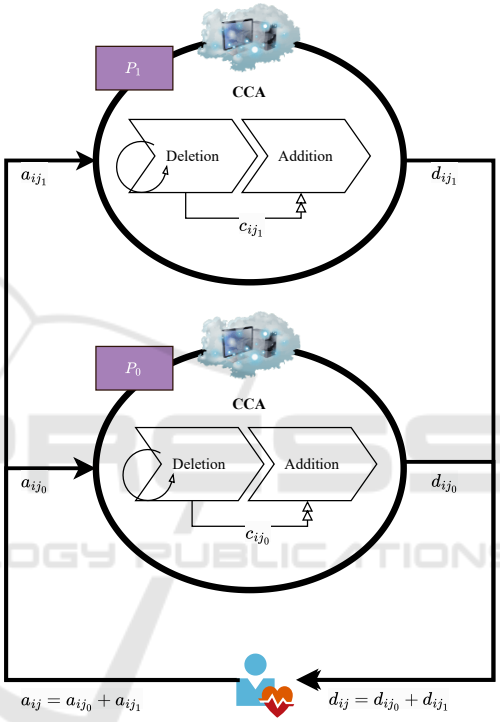



Figure 1: General overview of FunBic-CCA system.

**Two-Party Computation Scenario.** Our solution is based on the two-party computation (2PC) context, having two computing parties $(P_0, P_1)$ to perform secure computations as well as the data owner. In this two-server protocol, the data owner interacts with these two servers that are assumed not to collude (Boyle et al., 2021).

Our framework is secure against Honest-but-Curious ("semi-honest") adversaries. Therefore, FunBic-CCA provides privacy of all the input data in the case of corruption of either the two parties by a semi-honest adversary. This work is designed in MPC with a preprocessing model, having a setup/offline phase to benefit from an optimised online phase.

### 3.1 Cryptographic Primitives

Here, we provide details of our building blocks that form the core of our framework and for which we decide to apply secret sharing schemes. In all operations, $d$ refers to the number of parties, precisely $d \in \{0,1\}$. Note that, each individual element $(a_{ij})$ of the input matrix $(\mathbf{A})$ is secret shared among parties with the following condition: $a_{ij} = a_{ij_0} + a_{ij_1}$.

**Construction of Secure Linear Operations.** We require to obtain the below-mentioned operations, such as addition/subtraction, and multiplication:

- Addition/ subtraction: These operations can be computed locally. Accordingly, both parties are able to perform the following operations on their own secret shared input matrix.
  - Finding the mean values, residue, scores of the whole matrix, and nodes are based on addition/ subtraction.

$$\begin{gathered}\sum^{l_r \cdot l_c} a_{ij_d}, \sum^{l_r} a_{ij_d}, \sum^{l_c} a_{ij_d} \\ \sum^{l_r \cdot l_c} (a_{ij_d} - \mu_{r_d} - \mu_{c_d} + \mu_{ij_d}) \\ \sum^{l_r \cdot l_c} r^2_{ij_d}, \sum^{l_r} r^2_{ij_d}, \sum^{l_c} r^2_{ij_d}\end{gathered} \tag{5}$$

- Multiplication: Both parties perform SS-based multiplications in one round of communication and consumption of the beaver multiplication triples.
  - Squaring the residue $(r_{ij_0} \times r_{ij_1})$ as part of the score formula requires parties to perform element-wise multiplication for the matrices jointly using beaver triples. We follow the steps mentioned in (Knott et al., 2021). To square the residue $r^2_{ij} = r^2_{ij_0} + r^2_{ij_1} + 2 \times r_{ij_0} \times r_{ij_1}$, the parties use a beaver pair $\hat{a}, \hat{b}$, such that $\hat{b} = \hat{a}^2$. Then, parties compute $\hat{e}_d = r_{ij_d} - \hat{a}_d$, decrypt $\hat{e}_d$ to obtain result with $r^2_{ij} = \hat{b}_d + e^2 + 2 \times e \times \hat{a}_d$.

**Construction of Secure Non-Linear Operations.** Our algorithm relies on non-linear operations, including comparison, argmax and division:

- Comparison: Main comparative functions are denoted in Table 2. The threshold in comparison i.e., zero (is referred to $\gamma$), must be kept private and hidden by parties by using a random mask. In the online phase,the parties are required to compute

$$\begin{gathered}1_{r2del(H_{r_d}, \alpha \cdot H_{ij_d}) > 0}, 1_{c2del(H_{c_d}, \alpha \cdot H_{ij_d}) > 0} \\ 1_{r2sdel(H_{r_d}[d(i)], H_{c_d}[d(j)]) \geq 0}, 1_{c2sdel(H_{c_d}[d(j)], H_{r_d}[d(i)]) > 0} \\ 1_{r2add(H_{ij_d}, H_{r_d}) \geq 0}, 1_{c2add(H_{ij_d}, H_{c_d}) \geq 0} \\ 1_{stop(H_{ij_d}, \delta) \geq 0} \\ 1_{req(I_d, I'_d) == 0}, 1_{ceq(J_d, J'_d) == 0}\end{gathered} \tag{6}$$

on the secret shares $a_{ij_d}$.

Parties evaluate IC gate to determine whether the result is greater or equal to the threshold for $1_{r2del}, 1_{c2del}, 1_{r2sdel}, 1_{c2sdel}, 1_{r2add}, 1_{c2add}$. The DPF gate handles our equality checks, $1_{req}, 1_{ceq}$.

For stopping condition $1_{stop}$, we reconstruct the intermediate results $o_{stop_0}, o_{stop_1}$ after FSS IC gate to check whether our score $H_{ij_d}$, is below or equal to the threshold $\delta$.

To this end, our condition to remove columns, $1_{con}$, is done in the cleartext, since the size of the input is public and known by both parties. Note that, each of the comparison function requires one round of communication and to send two ring elements in the online phase.

- Argmax: Single node deletion step works with removing a single row or column, whichever having the largest $d(i)$ or $d(j)$. This is achieved by finding the argmax of rows and columns with the largest score:

$$d(i) = \text{argmax}(H_r), d(j) = \text{argmax}(H_c) \tag{7}$$

Our argmax function is similar to the algorithm 6 of (Ryffel et al., 2020), using pairwise comparisons. It gets as input; secret shared scores of rows or columns $(H_r, H_c)$ and outputs the index of the row or column with the largest score value.

Algorithm 3 requires a constant number of rounds. For instance, argmax of rows with length $l_r$ requires $l_r(l_r - 1)$ parallel comparisons with IC gate and $l_r$ equality checks with DPF gate. Therefore, we need two rounds of communication and sending $O(l_r^2)$ values in an online phase.

- Division: Finding mean which is a base function for calculation of the scores relies on a division.

$$\begin{gathered}\frac{\sum^{l_r \cdot l_c} a_{ij_d}}{|I||J|}, \frac{\sum^{l_r} a_{ij_d}}{|J|}, \frac{\sum^{l_c} a_{ij_d}}{|I|} \\ \frac{\sum^{l_r \cdot l_c} r^2_{ij_d}}{|I||J|}, \frac{\sum^{l_r} r^2_{ij_d}}{|J|}, \frac{\sum^{l_c} r^2_{ij_d}}{|I|}\end{gathered} \tag{8}$$

In secured machine learning applications (Ryffel et al., 2020), it is a standard assumption to keep the data and model parameters private; however,

Table 2: FunBic-CCA comparative functions.

| Formula | Symbol |
|---|---|
| $H_r > \alpha H_{ij}, H_c > \alpha H_{ij}$ | $1_{r2del}, 1_{c2del}$ |
| $H_r[d(i)] \geq H_c[d(j)]$ | $1_{r2sdel}, 1_{c2sdel}$ |
| $H_r \leq H_{ij}, H_c \leq H_{ij}$ | $1_{r2add}, 1_{c2add}$ |
| $H_{ij} \leq \delta$ | $1_{stop}$ |
| $I == I' \wedge J == J'$ | $1_{req}, 1_{ceq}$ |
| $\lvert J\rvert <= 100$ | $1_{con}$ |

the shape (number of rows and columns) of the input and the architecture of the model are public. In addition, multiplying or dividing by a public value (Escudero, 2022) such as $|I|, |J|$ can be executed locally.

Prior to any changes on nodes (removing, adding), the dimension of matrix is clear; thus the mean values are being done as mentioned above. Additionally, parties do local operations, including addition/ subtraction simultaneously by one, once any node is added/ removed.

## 3.2 System Protocol

We describe our protocols needed to deliver the full solution based on the crytographic primitives in Section 3.1. Figure 2 shows these protocol steps.

We follow the below settings in the offline and online phases, having the two parties $(P_0, P_1)$:

- Offline phase: Correlated randomness is required to perform the secured multiplication according to the discussion in Section 3.1. The generation of the FSS keys for the comparison functions also occurs in "FunBic-CCA.setup". These preprocessing materials are distributed among the parties to be used in the online phase.

  We assume a trusted dealer, an individual entity, to only participate in the offline phase and provide the computing parties with the preprocessing materials. We can also realise a trusted dealer by having our two computing parties in a pure 2PC scenario jointly generate required preprocessing materials (Boyle et al., 2021).

- Online phase:

  1. Identification of the scores: Calculation of the mean squared residue for the whole input matrix and its rows and columns (i.e., $H_{ij_d}, H_{r_d}, H_{c_d}$) takes place in "FunBic-CCA.MSR".

     The parties perform linear operations, addition/ subtraction, along with a non-linear division locally. Further, the parties square the residue $(r_{ij_d})$ by having the correlated randomness from the setup phase in one round of communication.

Algorithm 1: FunBic-CCA.setup $(l_r, l_c, n, \lambda, \gamma) \rightarrow K_0, K_1$.

**Input:** $l_r \cdot l_c$: length of matrix $\mathbf{A}$.
$n$: number of bits for secret sharing ring $\mathbb{Z}_{2^n}$.
$\lambda$: security parameter.
$\gamma$: threshold for comparison $(\in \mathbb{Z}_{2^n}, \gamma = 0)$.
**Output:** $K_0, K_1$: keys in preprocessing phase.

**Preprocessing Steps:**

Beaver triples for multiplication:
① $\langle \hat{a} \rangle \equiv (\hat{a}_0, \hat{a}_1) \sim \mathcal{U}_{2^n}^{l_r \cdot l_c}$
② $\hat{b}_0 \leftarrow \hat{a}_0^2 \sim \mathcal{U}_{2^n}^{l_r \cdot l_c}$
③ $\hat{b}_1 \leftarrow (\hat{a}_0 + \hat{a}_1)^2 - \hat{b}_0$
④ $\langle \hat{b} \rangle \equiv (\hat{b}_0, \hat{b}_1)$

Random masks for FSS gates:
① $\langle r \rangle \equiv (r_0, r_1) \leftarrow (r_0 + r_1) \sim \mathcal{U}_{2^n}^2$
② $\langle r_\gamma \rangle \equiv (r_{\gamma_0}, r_{\gamma_1}) \leftarrow (r_0, r_1 - \gamma)$

Distribution of preprocessing materials:
① $K_d \equiv (\hat{a}_d, \hat{b}_d, r_{\gamma_d}, K_d^{IC}, K_d^{EQ}), d \in \{0,1\}$
② $K_0 \Rightarrow P_0, \quad K_1 \Rightarrow P_1$

Algorithm 2: FunBic-CCA.MSR $(a_{ij_d}, K_d, d) \rightarrow H_{ij_d}, H_{r_d}, H_{c_d}$.

**Input:** $a_{ij_d}$: input matrix $(\in \{a_{ij_0}, a_{ij_1}\}, \mathbb{Z}_{2^n}^{l_r \cdot l_c})$.
$K_d$: preprocessing keys containing $\hat{a}_d, \hat{b}_d$.
$d$: computing parties $(\in \{0,1\})$.
**Output:** $H_{ij_d}, H_{r_d}, H_{c_d}$: similarity scores $(\in l_r \cdot l_c)$.

**Score Steps:**

Local linear functions (addition and subtraction):
① $\sum^{l_r \cdot l_c} a_{ij_d}, \sum^{l_r} a_{ij_d}, \sum^{l_c} a_{ij_d}$
② $\sum^{l_r \cdot l_c} (a_{ij_d} - \mu_{r_d} - \mu_{c_d} + \mu_{ij_d})$
③ $\sum^{l_r \cdot l_c} r_{ij_d}^2, \sum^{l_r} r_{ij_d}^2, \sum^{l_c} r_{ij_d}^2$

Local non-linear function (division):
① $\frac{\sum^{l_r \cdot l_c} a_{ij_d}}{|I||J|}, \frac{\sum^{l_r} a_{ij_d}}{|J|}, \frac{\sum^{l_c} a_{ij_d}}{|I|}$
② $\frac{\sum^{l_r \cdot l_c} r_{ij_d}^2}{|I||J|}, \frac{\sum^{l_r} r_{ij_d}^2}{|J|}, \frac{\sum^{l_c} r_{ij_d}^2}{|I|}$

Reconstruction of $\hat{e}$ for multiplication:
① $\hat{e}_d \leftarrow (r_{ij_d} - \hat{a}_d); \langle \hat{e} \rangle \equiv (\hat{e}_0, \hat{e}_1) \sim \mathcal{U}_{2^n}^{l_r \cdot l_c}$
② $\hat{e}_d \Rightarrow P_{1-d}; \quad \hat{e} \leftarrow \hat{e}_0 + \hat{e}_1$

Squared residue:
① $r_{ij_d}^2 = \hat{b}_d + e + 2 \times e \times \hat{a}_d$

  2. Deletion/ addition of the nodes: Parties do the comparisons, composing of $1_{r2del}, 1_{c2del}$ inside the multiple node deletion, $1_{r2sdel}, 1_{c2sdel}$ of the

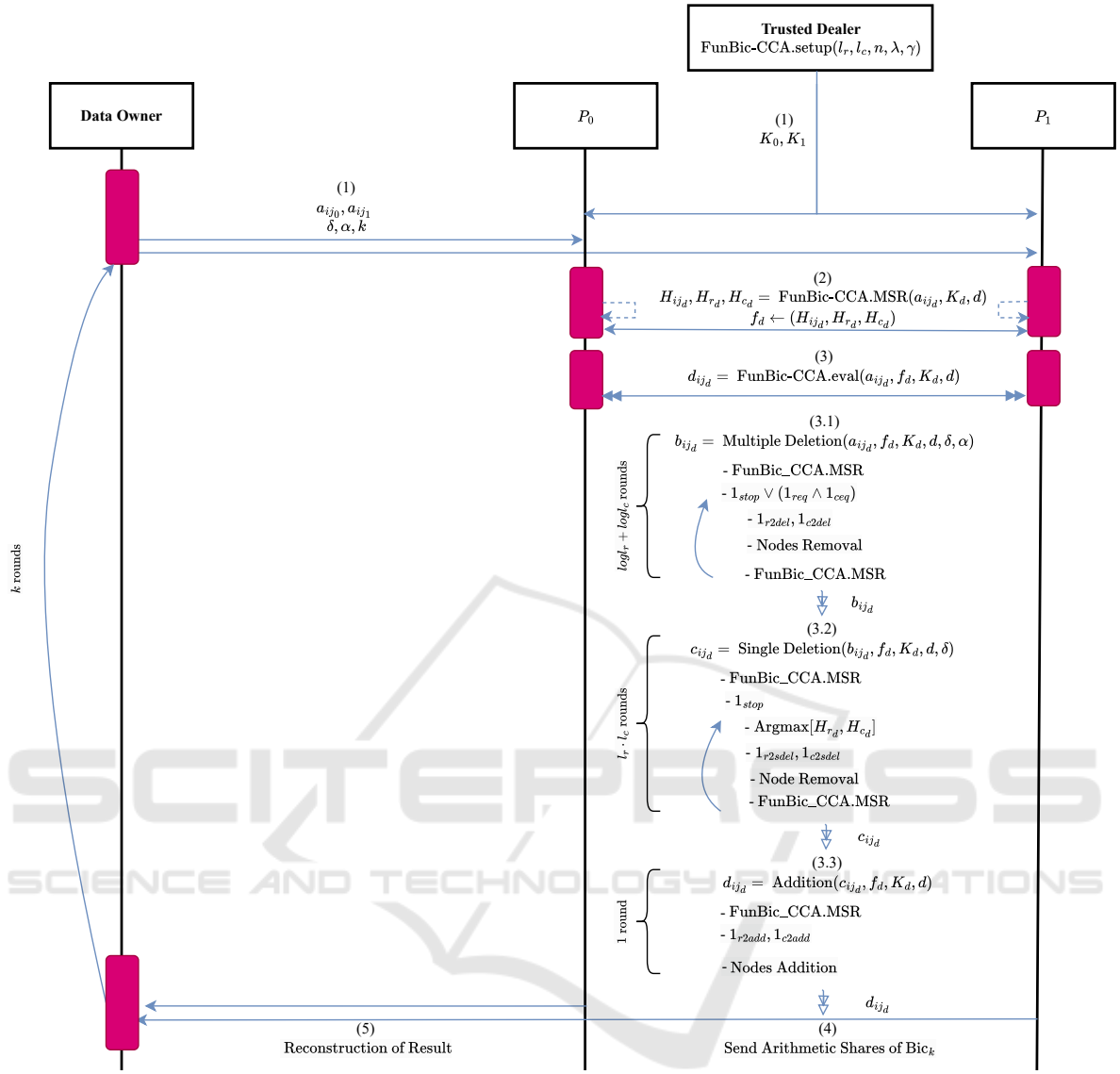



Figure 2: System diagram of the end-to-end secured computation of CCA using FunBic-CCA's algorithm.

single node deletion, $1_{r2add}, 1_{c2add}$ in the node addition, and the stopping function $1_{stop}$ as explained in Table 2, by performing the "FunBic-CCA.eval" with the IC gate [4].

Further, the parties benefit from the DPF gate [5] to obtain $1_{req}, 1_{ceq}$ and to also decide which nodes have to be removed/ added in the deletion and addition steps. Deletion of a single node depends on the result of the argmax, which invokes both FSS gates for the comparison and equality check. To this end, parties can locally increase/ decrease with one simultaneously to track the actual size of the secret shared matrix (i.e., some rows/ columns are masked with zeros in the deletion steps). The resulting matrices are sent back to the next step.

3. Final result: Parties return their final output to the data owner for its reconstruction in the "FunBic-CCA.result".

## 3.3 Security Analysis

**Overview.** As explained earlier in Section 3, our framework is secure against Honest-but-Curious

[4]We keep FSS.Gen$^{IC}$, FSS.Eval$^{IC}$ calls to the original protocols 1, and 2 in (Ibarrondo et al., 2023).

[5]Construction of FSS.Gen$^{EQ}$, FSS.Eval$^{EQ}$ for equality check maintains a call to DPF gate (Boyle et al., 2019).

Algorithm 3: Argmax ($H_{r_d} \rightarrow \text{argmax}_{w\in[1,l_r]} H_{r_d}$).

**Input:** $H_{r_d}$: secret shared score for rows.
**Output:** $\text{argmax}_{w\in[1,l_r]} H_{r_d}$.

**for** $w \leftarrow 1$ **to** $l_r$ **do**
1 **for** $v \leftarrow 1$ **to** $l_r$ **do**
2 **if** $w \neq v$ **then**
3 $\hat{z}_{\gamma_d} \leftarrow H_{r_d}[w] - H_{r_d}[v] + r_{\gamma_d}$
4 Comparison of $H_{r_d}[w] \geq H_{r_d}[v]$:
5 ① $\hat{z}_{\gamma_d} \Rightarrow P_{1-d}$ $\hat{z}_\gamma \leftarrow \hat{z}_{\gamma_0} + \hat{z}_{\gamma_1}$
6 ② $geq_d \leftarrow \text{FSS.Eval}^{IC}(d, K_d^{IC}, \hat{z}_\gamma)$
7 ③ $s_d \leftarrow \sum^{l_r} geq_d$
8 Equality check on $s_d == l_r - 1$:
9 ① $\hat{z}_{\gamma_d} \leftarrow s_d - (l_r - 1) + r_{\gamma_d}$
10 ② $\hat{z}_{\gamma_d} \Rightarrow P_{1-d}$ $\hat{z}_\gamma \leftarrow \hat{z}_{\gamma_0} + \hat{z}_{\gamma_1}$
11 ③ $eq_w \leftarrow \text{FSS.Eval}^{EQ}(d, K_d^{EQ}, \hat{z}_\gamma)$
12 **if** $eq_w == 1$ **then**
13 **return** $w$

Algorithm 4: FunBic-CCA.eval ($a_{ij_d}, f_d, K_d, d$) $\rightarrow d_{ij_d}$.

**Input:** $a_{ij_d}$: input matrix
$(\in \{a_{ij_0}, a_{ij_1}\}, \mathbb{Z}_{2^n}^{l_r \cdot l_c})$.
$f_d$: similarity scores ($\in \{H_{ij_d}, H_{r_d}, H_{c_d}\}$).
$K_d$: preprocessing keys from setup phase.
$d$: computing parties ($\in \{0,1\}$).
**Output:** $d_{ij_d}$: arithmetic shares of output matrix after deletion/ addition steps.

**Evaluation Steps:**
Preparation of input to FSS gate ($\hat{z}_{\gamma_d}$):
① $1_{r2del} : \alpha H_{ij_d} - H_{r_d} + r_{\gamma_d}$
② $1_{r2sdel} : H_{r_d}[d(i)] - H_{c_d}[d(j)] + r_{\gamma_d}$
③ $1_{r2add} : H_{ij_d} - H_{r_d} + r_{\gamma_d}$
④ $1_{stop} : H_{ij_d} - \delta + r_{\gamma_d}$
⑤ $1_{req} : I'_d - I_d + r_{\gamma_d}$
Reconstruction of masked input:
① $\hat{z}_{\gamma_d} \Rightarrow P_{1-d}$; $\hat{z}_\gamma \leftarrow \hat{z}_{\gamma_0} + \hat{z}_{\gamma_1}$
Comparison (IC) and equality check (DPF):
① $o_{geq_d} \leftarrow \text{FSS.Eval}^{IC}(d, K_d^{IC}, \hat{z}_\gamma)$
② $o_{eq_d} \leftarrow \text{FSS.Eval}^{EQ}(d, K_d^{EQ}, \hat{z}_\gamma)$

("semi-honest") adversary ($\mathcal{T}$) that corrupts up to one of computing parties ($P_0, P_1$). We consider standard security model in line with the related works (Ibarrondo et al., 2023; Boyle et al., 2021), having a static corruption model. Therefore, the adversary must choose a participant to corrupt prior to computations.

We simulate the corruption of a party $P_d$ by resorting to the standard real world - ideal world paradigm (Canetti, 2001). The ideal world requires an additional trusted party that obtains all the inputs from the involved parties, which will then receive correct results by the correct computation of the ideal functionality. On the other hand, we execute the aforementioned protocols of the FunBic-CCA algorithm in the real world with $\mathcal{T}$. Accordingly, we begin with representing our ideal functionality of FunBic-CCA in FUNCTIONALITY 5 to prove that our security works in the $I_{\text{FunBic-CCA.setup}}$-hybrid model. Thus, the model is based on the faithful execution of the FunBic-CCA.setup by our defined trusted party to provide designated parties with each piece of setup material.

Algorithm 5: FunBic-CCA.result $d_{ij_d} \rightarrow \text{Bic}_k(I,J)$.

**Input:** $d_{ij_d}$: output arithmetic shares.
**Output:** $\text{Bic}_k(I,J) : k$ $\delta$-biclusters.

**Result Steps:**
① $d_{ij} = d_{ij_0} + d_{ij_1}$
② return $\text{Bic}_k(I,J) = d_{ij}$

**Security Proof.** We assert that there is a Probabilistic Polynomial Time (PPT) key generation algorithm. Simulator ($\mathcal{S}$) realises the mentioned ideal functionality $I_{\text{bic-identif}}$ for each individual participant (given $\forall \mathbf{A} \in \mathbb{R}^{l_r \cdot l_c}, \forall \gamma \in \mathbb{Z}_{n^+}^*$) in which $\mathcal{S}$'s behaviour is computationally or statistically indistinguishable from a real world execution of our protocols 2,4, and 5, having a semi-honest adversary $\mathcal{T}$.

**Proof.** We prove the security of our framework by considering $\mathcal{S}$ and possible scenarios during execution of the protocols:

- FunBic-CCA.setup: This offline phase can be seen as a black-box access in an ideal world that provides parties with preprocessing materials. We realise this setup phase as an ideal functionality $I_{\text{FunBic-CCA.setup}}$.
  Its simulation is grounded on the security of the underlying primitive used to instantiate it, generic 2PC for the FSS keys (Appendix A.2 of (Boyle et al., 2021)), and OT (Oblivious Transfer), HE (Homomorphic Encryption) for the SS preprocessing materials in (Demmler et al., 2015).
- FunBic-CCA.MSR: For the messages that a particular party is holding without sharing with the other one, nothing is needed to be considered by $\mathcal{S}$, because $\mathcal{T}$ does not have access to any messages. In the other cases in which $P_{1-d}$ is the owner, $\mathcal{S}$ still executes the protocols honestly.
  Our local functions, including addition/ subtraction and division do not need to be simulated as of being done in a non-interactive way. The security proof of our secured multiplication is the same as for the Beaver-triple-based secure multiplication protocol.

- FunBic-CCA.eval: In the online phase, $\mathcal{S}$ honestly follows the protocol steps for finding the score using the data from the MSR phase. This is followed by the reconstruction of $\langle \hat{z}_\gamma \rangle$ as $\hat{z}_\gamma = \hat{z}_{\gamma 0} + \hat{z}_{\gamma 1}$ which is our input to FSS gates.

  We argue computationally indistinguishability of the ideal world - real world executions for FSS gates based on (Boyle et al., 2021) (Definition 2). In this simulation, the information in FSS keys $K_0, K_1$ as well as mask $r_\gamma$ is preserved.
- FunBic-CCA.result: $\mathcal{S}$ simulates the output result with receiving $\langle o \rangle$ from $\mathcal{T}$ and $o_{1-d}$, associated with $P_{1-d}$, then can compute $o_d = 1 - o_{1-d}$. $\mathcal{S}$ provides $\mathcal{T}$ with $o_d$ on behalf of $P_{1-d}$. After all, $\mathcal{S}$ has the output of $P_d$ from $\mathcal{T}$.

**FUNCTIONALITY 5** ($I_{\text{bic-identif}}(\mathbf{A}) \rightarrow \mathbf{D}$): Upon receiving a share of input matrix $\mathbf{A}$, known parameters $\delta, \alpha, k$, preprocessing materials $\hat{a}_d, \hat{b}_d, \hat{e}_d, K^{IC}, K^{DPF}, r_\gamma$ from $P_d, d \in \{0,1\}$, $I_{\text{bic-identif}}$ reconstructs $\hat{e}_d = r_{ij_d} - \hat{a}_d$, computes $H_{ij}, H_r, H_c$ and $1_{r2del}, 1_{c2del}, 1_{r2sdel}, 1_{c2sdel}, 1_{r2add}, 1_{c2add}$, finally returns the shares of output matrix $\mathbf{D}$.

**Correctness Proof.** The execution of FunBic-CCA protocols satisfies:

$$\begin{aligned} &\text{if FunBic-CCA.setup}(l_r, l_c, n, \lambda, \gamma) \rightarrow K_0, K_1, \\ &\text{and FunBic-CCA.MSR}(a_{ij_d}, K_d, d) \rightarrow f_0, f_1 \\ &\text{then Pr}[\text{FunBic-CCA.eval}(a_{ij_0}, f_0, K_0, 0) \\ &\quad + \text{FunBic-CCA.eval}(a_{ij_1}, f_1, K_1, 1) \\ &= \{1_{r2del(H_{r_d}, \alpha \cdot H_{ij_d})>0}, 1_{c2del(H_{c_d}, \alpha \cdot H_{ij_d})>0}, \\ &1_{r2sdel(H_{r_d}[d(i)], H_{c_d}[d(j)])\geq 0}, 1_{c2sdel(H_{c_d}[d(j)], H_{r_d}[d(i)])>0} \\ &1_{r2add(H_{r_d}, H_{ij_d})\geq 0}, 1_{c2add(H_{c_d}, H_{ij_d})\geq 0}\} \rightarrow d_{ij_0}, d_{ij_1}] = 1 \end{aligned} \tag{9}$$

for threshold $\gamma (\in \mathbb{Z}^*_{n^+})$, input matrix $\mathbf{A} (\in l_r \cdot l_c)$, and suitable choice of FSS as well as computing functions including deletion and addition.

**Proof.** Our proof on the correctness of IC gate is grounded on Theorem 3 of (Boyle et al., 2021). The two protocols $\text{FSS.Gen}^{IC}(\lambda, n, r)$, $\text{FSS.Eval}^{IC}(d, k_d^{IC}, \hat{z}_\gamma)$ establish the IC gate correctly, maintaining the condition of $f(z_\gamma, \gamma) = z_\gamma \geq \gamma$. Lastly, we argue what we mentioned earlier based on the Definition 2 of (Boyle et al., 2021) that

$$\begin{aligned} &\text{Pr}[\text{FSS.Eval}^{IC}(0, k_0^{IC}, \hat{z}_\gamma) + \\ &\text{FSS.Eval}^{IC}(1, k_1^{IC}, \hat{z}_\gamma) = (z_\gamma \geq \gamma)] = 1 \end{aligned} \tag{10}$$

We analyse the correctness of the equality test according to Theorem 4.3 of (Boyle et al., 2019). The correctness can be seen since point function $f_{\alpha,\beta}(x)$ evaluates to $\beta = 1$ once $(x_1 - x_2) = \alpha = (r_1^{in} - r_2^{in})$ or having $(x_1 - r_1^{in}) = (x_2 - r_2^{in})$.

Afterwards, we prove the correctness of SS-based multiplication (i.e., $r_{ij}^2$) by resorting to the definition of scheme about random shares ($a_{ij_d} = a_{ij_0} + a_{ij_1}$), beaver multiplication triples ($\hat{a}_d, \hat{b}_d, e : \text{Dec}(\hat{e}_d)$) and having secret shares ($r_{ij_d} = r_{ij_0} + r_{ij_1}, \hat{e}_d = r_{ij_d} - \hat{a}_d$) after applying the local functions, including addition/ subtraction and secured division:

$$\begin{aligned} &e^2 + (\hat{b}_0 + \hat{b}_1) + 2 \times e \times (\hat{a}_0 + \hat{a}_1) \\ &= e^2 + \hat{b} + 2 \times e \times \hat{a} = e^2 + \hat{a}^2 + 2 \times e \times \hat{a} = (e + \hat{a})^2 \\ &= [(r_{ij_1} - \hat{a}_1) + (r_{ij_0} - \hat{a}_0) + \hat{a}]^2 \\ &= [(r_{ij_1} - \hat{a}_1) + (r_{ij_0} - \hat{a}_0) + (\hat{a}_0 + \hat{a}_1)]^2 \\ &= [(r_{ij_1} + r_{ij_0}) - (\hat{a}_1 + \hat{a}_0) + (\hat{a}_0 + \hat{a}_1)]^2 \\ &= (r_{ij_1} + r_{ij_0})^2 = r_{ij}^2 \end{aligned} \tag{11}$$

# 4 EXPERIMENT

In this Section, we represent our implementation of the FunBic-CCA framework, evaluation of resulting biclusters and finally summarise by discussing time performance and accuracy obtained.

**Implementation and Environment.** Our privacy-preserving implementation drives from Funshade (Ibarrondo et al., 2023) to construct the protocol steps (1, 2, 4, and 5) efficiently. Additionally, we use sycret (Ryffel et al., 2020) to implement only the equality checks. We also rely on biclustlib (Padilha and Campello, 2017) for the implementation of the original CCA, yeast cell cycle dataset and accuracy measures.

In the original algorithm (Cheng and Church, 2000), rows that form mirror images are added in bicluster throughout the node addition phase. However, they are not of interest when defining a bicluster in a general framework (Galvani et al., 2021). Accordingly, we change our focus in this work only on rows in the addition phase to be applicable to a broader set of data and increase the performance.

FunBic-CCA is implemented and executed on an Intel Core i7-1185 CPU, 3.00 GHz with 8 physical cores available and 31 GB system memory over at least 10 runs.

**Datasets.** We have utilised yeast cell cycle (Tavazoie et al., 1999) and human expression data (Alizadeh et al., 2000) that were used for testing the

Table 3: Number of rounds and the communication size for one time execution of the FunBic-CCA protocol steps in the online phase. $n$ is the bit size of the matrix elements, $l_r \cdot l_c$ is the matrix size.

| Step | No. rounds | Communication size (bits) |
|---|---|---|
| MSR | 1 | $2n(l_r \cdot l_c)$ |
| Eval | 26 | $4n[1+6(l_r+l_c)]$ |
| Result | 1 | $2n(l_r \cdot l_c)$ |
| Total | 28 | $4n[1+(l_r \cdot l_c)+6(l_r+l_c)]$ |

implementation of the original algorithm (Cheng and Church, 2000).

**Parameter Selection.** In MPC, integers are naturally encoded into native data types, such as *uint*32_*t* or *uint*64_*t*, due to the implementation of protocols over finite rings or fields. We work with $n = 64$-bit modular arithmetic with $\lambda = 128$, considering the scale of input matrices and maximum bit width during secure computations. We leverage the normalisation of the input matrices and integer division, while such bounds prevent from inherently impeding any secure computation protocol.

We assess maximum communication size and round of communication for each protocol step in the online phase, and list them in Table 3, given that our biclustering algorithm runs through $logl_r + logl_c$ iterations in the multiple node deletion, $l_r \cdot l_c$ rounds in the single node deletion with only one iterate of the node addition for each $k$ bicluster (Cheng and Church, 2000). We consider here, one time execution of the online protocol steps, including FunBic-CCA.MSR with local functions (addition, subtraction, division) and one secured squaring (consists of one round of communications). Regarding FunBic-CCA.eval, our node deletion and addition steps rely on the reconstruction of the masked inputs to FSS gates with one round of communication.

**Accuracy Measures.** The quality of a bicluster needs to be assessed by evaluation functions, aligned with biclustering algorithms. These external evaluation criteria measure how close FunBic-CCA is to the ground truth. In this paper, we apply similarity measures such as Liu and Wang (Liu and Wang, 2007) and further extend it to the Prelic relevancy score (Prelić et al., 2006).

## 4.1 Results

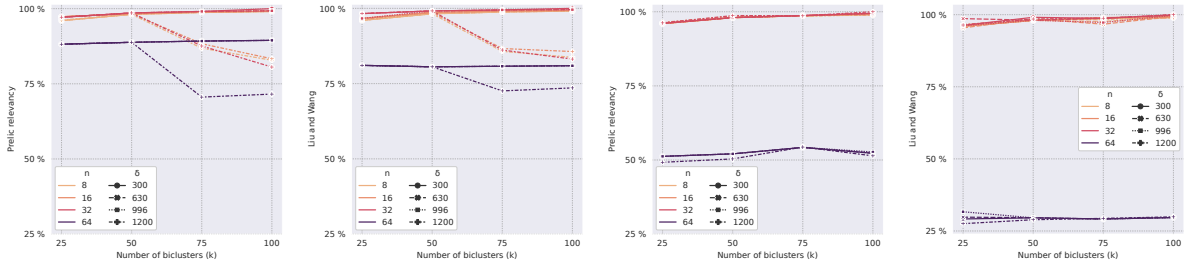
**Impact on Accuracy.** Here, we study the accuracy of our solution based on the above-mentioned key measures, which range over the interval [0, 1], with higher values indicating better solutions. Missing elements for both datasets are randomly replaced with values greater than or equal to 0. We choose $\alpha = 1.2$ and minimum number of the columns to be 100, according to the original study for both datasets (Cheng and Church, 2000). $\delta$ for experiments with yeast cell cycle and human gene expression data is selected based on the experiment by Tavazoie et al. (Tavazoie et al., 1999), whose reported clusters had scores in the range of between 261 and 996, with a median of 630.

Among the contributing parameters, $n$ has a direct impact on the accuracy (see Figure 3). Lower numerical precision leads to a drop in accuracy; while its improvement also occurs as long as natural overflow is avoided and computation is not exceed $[0, 2^n - 1]$ (unsigned integers). Best accurate biclusters are derived from $\delta$ being close to the lower end of the range ($\delta = 300$) (Tavazoie et al., 1999) to detect more refined patterns. Because the size and the variance in the human data are doable and quadrupling compared to yeast, the high quality biclusters are achieved with $\delta = 1200$. In accordance to the findings of (Liu and Wang, 2007; Prelić et al., 2006), when the number of the biclusters ($k$) is small, the match scores decrease; because CCA itself is not powerful enough when dealing with small set of biclusters. To conclude, we record the 100% correctness, when inputting both datasets, for 100 biclusters, and $n = 32$-bit.

**Performance Analysis.** In this Section, we analyse the performance of FunBic-CCA. According to Figure 4, we notice that by enlarging the bit width of the input matrix, both latency and bandwidth in the online phase are increased. We set the original values of $\delta$ for the yeast ($\delta = 300$) and human data ($\delta = 1200$).

Parties evaluate "FunBic-CCA.MSR" jointly $\cong 42$ seconds (s), and $\cong 110$ (s) on yeast and human data. We record on average 96.8225 (s) having yeast cell cycle, while 345.64 (s) on human data for another step in the online phase, "FunBic-CCA.eval". We extend the analyses to test how other parameters, including $k$ and $\delta$ can affect the overall performance in the online phase. In this regard, $k$ linearly increases the latency, while lowering $\delta$ links to the higher number of computations inside the algorithm, thus increases the latency (see Figure 5).

In addition, we time the total latency by each actor, including the data owner plus trusted dealer (DT), and the parties, in Table 4 and 5. Timings for parties consist of the three protocol steps "FunBic-CCA.MSR", "FunBic-CCA.eval" and "FunBic-CCA.result" to identify the scores, perform deletion/ addition on the nodes, then send back the secret shared of the outputs. We determine the to-




(a) . (b) . (c) . (d) .

Figure 3: Accuracy of the final biclusters with relevance, and Liu and Wang match scores with varying $n$, $k$ and $\delta$ over yeast cell cycle (a, b) and human expression data (c, d).

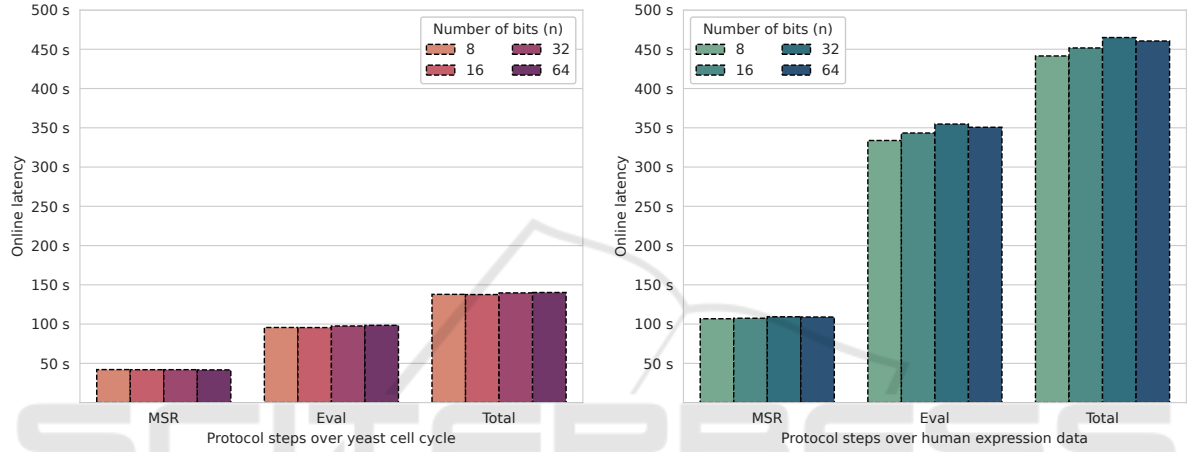



Figure 4: Computation and communication overhead of FunBic-CCA for each protocol step in the online phase over yeast cell cycle ($\delta = 300$) and human expression data ($\delta = 1200$).

Table 4: Computation and communication overhead of FunBic-CCA (online phase) for each actor over yeast cell cycle with $\delta = 300$.

| $n$ | Time (s) | | Bandwidth (KB) | |
|---|---|---|---|---|
| | DT | $P_d$ | $DT \leftrightarrow P_d$ | $P_{1-d} \leftrightarrow P_d$ |
| 8 | 14.2383 | 123.6316 | 1221.86 | 400.9778 |
| 16 | 14.2110 | 123.4150 | 1370.52 | 400.9453 |
| 32 | 14.2907 | 125.4250 | 1371.9 | 400.9676 |
| 64 | 14.1132 | 126.1317 | 1376.26 | 403.0315 |

Table 5: Computation and communication overhead of FunBic-CCA (online phase) for each actor over human data with $\delta = 1200$.

| $n$ | Time (s) | | Bandwidth (KB) | |
|---|---|---|---|---|
| | DT | $P_d$ | $DT \leftrightarrow P_d$ | $P_{1-d} \leftrightarrow P_d$ |
| 8 | 36.9292 | 404.5446 | 1224.74 | 2247.0354 |
| 16 | 37.1300 | 414.5372 | 1373.26 | 2247.9475 |
| 32 | 37.7045 | 427.2254 | 1373.17 | 2247.7641 |
| 64 | 37.6850 | 422.8491 | 1380.47 | 2251.3945 |

tal latency of the DT, which is referred to "FunBic-CCA.setup" and reconstruction of the biclusters. Almost 89.93% and 91.81% of the total execution time allocate to the parties performing the protocol steps in the online phase, remaining the rest for the DT on 64-bits yeast cell cycle and human expression data respectively.

Besides, the communication size between the DT and parties is calculated to less than 1380 kilobytes (kb) for both datasets. According to Figure 2, the communicated elements are the secret shares of the input matrix $(a_{ij_0}, a_{ij_1})$, known parameters for our algorithm $(\delta, \alpha, k)$ and the preprocessing materials (i.e., $K_0, K_1$ or the correlated randomness and FSS keys). On the other hand, the parties send back the secret shares of the output matrix to the data owner for its reconstruction. We also record the communication size between the two computing parties, exchanging the masked secret shares in performing multiplication (refer to Section 3.1) and the comparative functions (see Table 2). On average, 2248.5353 (kb) is communicated between two computing parties for human data, which is approximately $\times$ 5 higher than the size of communication in yeast data due to the difference in their range and size.

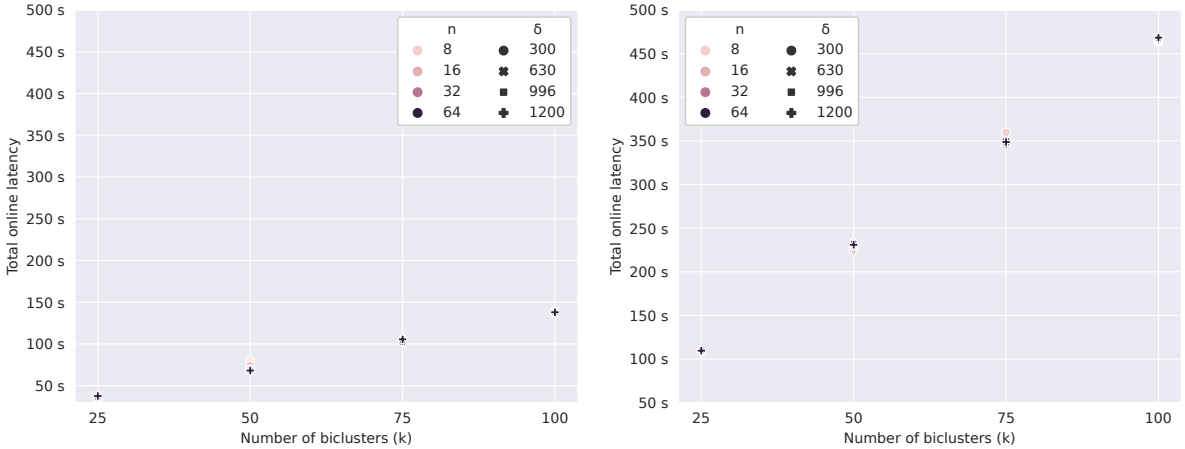



Figure 5: Total online latency with varying $n$, $k$ and $\delta$ over yeast cell cycle (left) and human expression data (right).

Table 6: Overview of FSS-based secure frameworks. ◐ indicates semi-honest security model, and ●: malicious security model. Primitives include Secret Sharing (SS), Function Secret Sharing (FSS), Replicated SS (RSS), optimized SS (o-SS), Distributed Comparison Function (DCF), Interval Containment (IC). Online computation blocks are considered for both linear and non-linear functions: Matrix Multiplication (MatMult), Convolution (Conv), Comparison (Comp), Feature Selection (FeatSelect), Path Evaluation (PathEval).

| Work | Online Computation Blocks | Type | Parties | Security | Remarks |
|---|---|---|---|---|---|
| Funshade (Ibarrondo et al., 2023) | Scalar product, Comp. ($\geq \theta$) | o-SS, FSS (IC gate) | 2PC | ◐ | Optimised online phase |
| AriaNN (Ryffel et al., 2020) | MatMult., Conv., Argmax, MaxPool | SS, FSS (DCF gate) | 2PC | ◐ | Reduced FSS key size |
| FssNN (Yang et al., 2023) | MatMult., ReLU, DReLU, MaxPool | SS, FSS (DCF gate) | 2PC | ◐ | Reduced FSS key size |
| PDTE (Cheng et al., 2024) | FeatSelect, Comp., PathEval | RSS, FSS (IC, DPF gates) | 3PC | ◐ | Constant rounds of comm. |
| Waldo (Dauterman et al., 2022) | Additive and arbitrary aggregate | RSS, FSS (DCF, DPF gates) | 3PC | ● | Multi-predicate filtering |
| FunBic-CCA (OURS) | MatMult., Comp. ($\geq \theta$), Argmax | SS, FSS (IC, DPF gates) | 2PC | ◐ | 1st end-to-end secure CCA |



# 5 RELATED WORK

Personal data misuse and theft, especially when dealing with patients' genomic data, are ever-present concerns due to carrying highly private information. Accordingly, privacy-compliant patient data analyses are among typical applications of privacy-preserving computation techniques. In line with works on FSS to name but a few (Boyle et al., 2021; Ryffel et al., 2020; Ibarrondo et al., 2023), these solutions incur promising results leading to our framework. Table 6 represents recent FSS-based secure frameworks.

Regarding unsupervised machine learning algorithm based on biclustering, there exist methods in the literature that use searches reliant on traditional one-way clustering and combine additional techniques to analyse the second dimension (Fraiman and Li, 2020). For instance, clustering based on the singular value decomposition, whose privacy is protected mainly by data distortion (Lakshmi and Rani, 2013). Moreover, co-clustering algorithms with matrix factorisation are widely used in text clustering and gene expression analysis demonstrated their superiority to traditional, one-side clustering (Lin et al., 2019). Their privacy has been discussed in several recent works, particularly when using differential privacy (Guo et al., 2023). Unfortunately, the papers neither targeted the expression data by biclustering methods such as Cheng and Church (Cheng and Church, 2000) as one of the most cited and used algorithms nor building upon recent advances in FSS. Our proposed solution can be adaptable to a range of MSR-based algorithms.

# 6 CONCLUSIONS

In this paper, we introduced our framework upon an additive secret sharing scheme and function secret sharing for a full correctness for comparison with a fixed threshold in an online phase between two computing parties. Thanks to the proposed solution, FunBic-CCA is the first protocol that applies function secret sharing over Cheng and Church Algorithm and discloses similarity score, all while relying on lightweight cryptographic primitives. We implement our solution on top of the open-source libraries and showcase its 100% correctness with 32-bit precision

and latency within $\cong 500s$ against 4026 rows and 96 columns. Future research is envisioned for extending FunBic-CCA to guarantee security with abort against malicious adversaries, using MACs [6].

## REFERENCES


Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., et al. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511.

Beaver, D. (1992). Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology—CRYPTO'91: Proceedings 11*, pages 420–432. Springer.

Boyle, E., Chandran, N., Gilboa, N., Gupta, D., Ishai, Y., Kumar, N., and Rathee, M. (2021). Function secret sharing for mixed-mode and fixed-point secure computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 871–900. Springer.

Boyle, E., Gilboa, N., and Ishai, Y. (2015). Function secret sharing. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 337–367. Springer.

Boyle, E., Gilboa, N., and Ishai, Y. (2019). Secure computation with preprocessing via function secret sharing. In *Theory of Cryptography: 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part I 17*, pages 341–371. Springer.

Canetti, R. (2001). Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE.

Cheng, N., Gupta, N., Mitrokotsa, A., Morita, H., and Tozawa, K. (2024). Constant-round private decision tree evaluation for secret shared data. *Proceedings on Privacy Enhancing Technologies*.

Cheng, Y. and Church, G. M. (2000). Biclustering of expression data. In *Ismb*, volume 8, pages 93–103.

Dauterman, E., Rathee, M., Popa, R. A., and Stoica, I. (2022). Waldo: A private time-series database from function secret sharing. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2450–2468. IEEE.

Demmler, D., Schneider, T., and Zohner, M. (2015). Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*.

Escudero, D. (2022). An introduction to secret-sharing-based secure multiparty computation. *Cryptology ePrint Archive*.

Fraiman, N. and Li, Z. (2020). Biclustering with alternating k-means. *arXiv preprint arXiv:2009.04550*.

Galvani, M., Torti, A., Menafoglio, A., and Vantini, S. (2021). Funcc: A new bi-clustering algorithm for functional data with misalignment. *Computational Statistics & Data Analysis*, 160:107219.

Guo, X., Li, X., Chang, X., and Ma, S. (2023). Privacy-preserving community detection for locally distributed multiple networks. *arXiv preprint arXiv:2306.15709*.

Ibarrondo, A., Chabanne, H., and Önen, M. (2023). Funshade: Function secret sharing for two-party secure thresholded distance evaluation. *Proceedings on Privacy Enhancing Technologies*.

Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., and van der Maaten, L. (2021). Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973.

Lakshmi, M. N. and Rani, K. S. (2013). Svd based data transformation methods for privacy preserving clustering. *International Journal of Computer Applications*, 78(3).

Lin, R., Wang, S., and Guo, W. (2019). An overview of co-clustering via matrix factorization. *IEEE Access*, 7:33481–33493.

Liu, X. and Wang, L. (2007). Computing the maximum similarity bi-clusters of gene expression data. *Bioinformatics*, 23(1):50–56.

Padilha, V. A. and Campello, R. J. (2017). A systematic comparative evaluation of biclustering techniques. *BMC bioinformatics*, 18:1–25.

Prelić, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L., and Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129.

Ryffel, T., Tholoniat, P., Pointcheval, D., and Bach, F. (2020). Ariann: Low-interaction privacy-preserving deep learning via function secret sharing. *arXiv preprint arXiv:2006.04593*.

Tavazoie, S., Hughes, J. D., Campbell, M. J., Cho, R. J., and Church, G. M. (1999). Systematic determination of genetic network architecture. *Nature genetics*, 22(3):281–285.

Yang, P., Jiang, Z. L., Gao, S., Wang, H., Zhou, J., Jin, Y., Yiu, S.-M., and Fang, J. (2023). Fssnn: communication-efficient secure neural network training via function secret sharing. *Cryptology ePrint Archive*.


[6] Primitive that is used to guarantee data integrity and authenticate the parties' shares.