Exploiting Trusted Execution Environments and Distributed Computation for Genomic Association Tests

Cláudia V. Brito, Pedro G. Ferreira, and João T. Paulo

***Abstract*— Breakthroughs in sequencing technologies led to an exponential growth of genomic data, providing novel biological insights and therapeutic applications. However, analyzing large amounts of sensitive data raises key data privacy concerns, specifically when the information is outsourced to untrusted third-party infrastructures for data storage and processing (*e.g.*, cloud computing).**

We introduce *Gyosa*, a secure and privacy-preserving distributed genomic analysis solution. By leveraging trusted execution environments (TEEs), *Gyosa* allows users to confidentially delegate their GWAS analysis to untrusted infrastructures. *Gyosa* implements a computation partitioning scheme that reduces the computation done inside the TEEs while safeguarding the users' genomic data privacy. By integrating this security scheme in *Glow*, *Gyosa* provides a secure and distributed environment that facilitates diverse GWAS studies. The experimental evaluation validates the applicability and scalability of *Gyosa*, reinforcing its ability to provide enhanced security guarantees.

***Index Terms*— Privacy-preserving GWAS Distributed Systems Trusted Execution Environments.**

I. Introduction

With the emergence of next-generation sequencing (NGS) technologies, genome sequencing costs have decreased, enabling the generation of large amounts of genomic data [34]. This presents new research opportunities on genetic factors but requires efficient algorithms to handle such a volume of data.

Genomic Wide-Association Studies (GWAS) test the association of hundreds of thousands to millions of genetic variants in a cohort of individuals and find the variants statistically associated with a specific trait or disease [37]. By 2021, more than 5700 GWASes have been conducted using data from more than one million individuals for more than 3300 traits [37]. The computational requirements of a GWAS depend on the number of genetic variants, the number of individuals, and the tested traits and phenotypes. As the datasets keep increasing in size along these different variables, the computational capacity offered by a single workstation is often insufficient.

Submitted for review on 31st of May, 2024. This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020. DOI 10.54499/LA/P/0063/2020

Cláudia V. Brito and João T. Paulo are with HASLab, INESC TEC & University of Minho, Portugal (e-mail: {claudia.v.brito, joao.t.paulo}@inesctec.pt). Pedro G. Ferreira. is with the Department of Computer Science, Faculty of Sciences, University of Porto and LIAAD, INESC TEC, Portugal (e-mail: pgferreira@fc.up.pt). João T. Paulo and Pedro G. Ferreira are the corresponding authors.

The conventional approach of enhancing server capacity by augmenting core processors, memory, and storage resources may encounter significant challenges, including exponentially escalating costs and inherent hardware limitations [32]. Alternatively, distributed computing allows using several servers (clusters of servers) in parallel and provides a more scalable and efficient solution to reduce the runtime execution of parallelizable and data-intensive algorithms, such as GWAS [7]. However, acquiring, maintaining, and managing a distributed server infrastructure is a costly and complex task requiring high-end hardware and specialized human resources [32].

A more accessible option involves running GWAS analysis remotely on distributed infrastructures managed by third-party Cloud Computing (*e.g.*, GCP [15]), and HPC services (*e.g.*, TACC [36]). This approach offers significant benefits, especially when: *i*) a single entity, such as a Hospital, possesses a large genomic dataset but lacks the processing and storage power for analysis; and *ii)* a consortium of entities, including Hospitals and research laboratories (see Figure 1 from Supplemental Material), collectively aims to analyze their datasets in a unified remote infrastructure, enabling large-scale analysis using shared data.

Challenges. Users trust third-party entities to keep their information secure when offloading data storage and processing to their infrastructures. However, there have been several reports detailing both external (*e.g.*, done remotely by a hacker) and internal attacks (*e.g.*, led by a malicious system administrator with physical access to the cluster) that have successfully compromised the privacy of sensitive information kept at remote infrastructures [15]. These attacks are one of the major barriers limiting the broader adoption of cloud computing services for storing and processing genomic data since its disclosure can lead to the complete identification of individuals [14]. For instance, early studies indicated that only 75 SNPs could help identify an individual [14]. By carrying information on disease predisposition, genomic data leakage may imply privacy risks for the individual and their future and previous generations [14].

When outsourcing a genomic data processing pipeline (*i.e.,* data collection, processing, and sharing) to third-party infrastructures, users increase the attack surface and become susceptible to three major groups of attacks. Membership inference attacks allow the adversary (attacker) to leverage the knowledge it has on a specific individual (*e.g.,* dis-

ease predisposition) and query the analysis results explicitly. Specifically, Homer's attacks use the background information on the human genome available in the public domain to infer whether an individual's genetic variants information was used for a specific study [38]. Re-identification attacks aim to reveal the identities of individuals whose data have been anonymized and used in genomic analysis. Recent studies have shown that demographic information can be linked to public genomic data and databases by linking genomic data with genealogical databases [23]. The goal of data poisoning attacks is to manipulate the analysis results. These attacks can generate false assumptions and associations when applied to GWASes, introducing bias and yielding erroneous discoveries [26]. For a detailed overview of these attacks and the genomic pipeline, see Section 2 of Supplementary Material.

Privacy-preserving solutions based on Homomorphic encryption (HE), Differential Privacy (DP), and Secure Multiparty computation (SMPC) typically incur a high-performance penalty (*e.g.*, GWAS studies [16], [21]). Such drawback motivated a new line of research exploring TEEs to ensure the privacy of GWAS at untrusted infrastructures [6], [33]. However, current solutions: *i)* are designed to run on a single server (*i.e.*, are not practical, performance-wise, for studies involving large datasets) and, *ii)* require all computation to be offloaded to a secure TEE (*i.e.*, further decreasing performance). Therefore, designing a privacy-preserving solution for GWAS that takes advantage of the computational power provided by distributed infrastructures and selectively uses TEEs to protect sensitive user information is still an open research challenge.

Contributions. We propose *Gyosa* [1], a novel distributed and privacy-preserving framework for securely executing GWAS in untrusted and distributed infrastructures. *Gyosa* is built on top of Apache Spark [40], a distributed computation framework, and uses Glow [11], a library for genomic processing including regression-based algorithms, statistical tests, and population stratification methods to perform GWAS. These are combined with TEEs, namely Intel SGX, to provide a secure environment where sensitive genomic information can be efficiently processed in plaintext without disclosing it to attackers.

Gyosa's novelty stems from the necessity and lack of solutions to outsource and distribute GWAS computation to untrusted third parties securely. *Gyosa* enables fine-grained differentiation between sensitive and nonsensitive information in an efficient, secure, and practical fashion. Furthermore, *Gyosa* leverages the genomic analysis pipeline offered by Glow and defines the specific set of operations from genomic association tests that can run outside of secure TEE enclaves without inducing data leakage. Thus, this distributed approach and careful differentiation allow large-scale GWAS algorithms to run while maintaining the confidentiality of critical data. Indeed, this is a key takeaway from this work, showing that by distributing GWAS computations, one can achieve practical and usable privacy-preserving designs. In brief, this work provides three major contributions:

- We introduce *Gyosa*, a distributed and privacy-preserving framework for securely executing GWAS on untrusted infrastructures, resorting to Apache Spark and Glow for distributed genomic processing.
- *Gyosa* integrates with Intel SGX, to ensure sensitive genomic data is kept private from potential attackers.
- Through fine-grained differentiation, *Gyosa* selectively places data inside secure TEEs, achieving a better trade-off between security and performance.
- We evaluate *Gyosa* with three genetic association algorithms—Logistic Regression, Linear Regression, and X^2—demonstrating that *Gyosa* maintains the accuracy of the output of these algorithms while enabling scalable computation across multiple servers.

[1]The system's name relies on the assumption that we are putting all sensitive data inside a trusted execution environment, similar to a dumpling.

II. Related Work

Next, we discuss the related work on privacy-preserving genomic analysis that resorts to: *i)* software-based or, *ii)* hardware-based cryptographic primitives. Then, we present a summary and taxonomy of the related work.

Software Approaches. To enable the execution of GWAS at untrusted infrastructures while ensuring data privacy, several approaches have been proposed based on standard cryptographic primitives such as HE, SMPC, and DP [18], [22]. HE-based solutions for GWAS have been shown to impose a high-performance execution time penalty and can only support a limited number of algorithmic operations [16]. Conversely, SMPC enables secure data storage and computation even across multiple entities that do not trust each other. However, it resorts to distributed protocols that require several rounds of network communication, adding a significant delay to the execution of GWAS [41]. In contrast, DP provides a less penalizing solution regarding performance overhead. Nevertheless, adding noise to the data and computation can compromise the accuracy and undermine the results. Also, DP cannot effectively manage high-dimensional data or cope with growing data volumes [10].

Hardware Approaches. TEEs have shown great potential as alternative solutions to ensure privacy-preserving computation and storage in untrusted infrastructures for genomic data [8], [19]. While this technology is promising for running GWAS, its application has been limited to a single-server [6], [33]. By taking advantage of distributed infrastructures (*i.e.*, multiple servers), it is possible to enhance the speed and scalability (*i.e.*, the amount of data being analyzed) of GWAS. However, developing a distributed solution for privacy-preserving GWAS requires a fundamentally new design. *Gyosa* belongs to this group of approaches while resorting to data encryption to protect the information handled outside TEEs.

Summary. In Table I, we present a taxonomy of related work systems divided into five categories: *i)* cryptographic primitives; *ii)* architecture; *iii)* stages of the genomic pipeline; *iv)* algorithms; and, *v)* availability.

For the first category, we consider the most relevant primitives for genomic data analysis: HE, SMPC, DP, and TEEs. The architecture topology of each system is decomposed into distributed, centralized, and collaborative, representing each

system's architectural design. The *centralized* sub-category refers to solutions that perform the computation in a single server. The *distributed* sub-category refers to solutions that distribute the computation across multiple servers to parallelize the computation. This model involves partitioning tasks and data to improve efficiency and scalability, with each node operating independently and concurrently handling its assigned tasks. Conversely, the *collaborative* sub-category refers to solutions that require the collaboration of multiple entities to perform the computation, a setting commonly used when resorting to SMPC or federated environments. This model prioritizes data privacy and security when nodes collaborate by sharing intermediate results rather than raw data. For the third category, we consider the stages of the genomic pipeline where the solution is applied, namely, data analysis and public release. Within *public release*, we consider the aggregation of the final results from an analysis in a collaborative setup. For the fourth category, we consider the algorithms supported by each solution, such as GWAS and read mappings.

TABLE I: Taxonomy of Related Work systems.

Systems	Crypt. Primitives	Architecture	Stage	Algorithms
[16]	○	C, H	A, PR	◑
[18]	○	C, H	A	●, ◑
[17]	○	C	A	◑
[41]	◐	H	A	●, ◑
[10]	◑, ○	H	A	◑
[6]	●, ○	C	A	◑
[2]	●,◐	C, H	A	●,○, ◑
[21]	○, ◑, ◐	H	A	●,◑
[33]	●,○	C, H	A	◑
[19]	●	C	A	●,○
[29], [30]	●	C,H	A,PR	●,◑
[20]	●	C	A	◐
[39]	●	D	A	◐
[8]	●	C, H	A	◑
This work	●	D	A	●,○

Crypt. Primitives
● - TEE
◐ - SMPC
◑ - DP
○ - HE

Architecture
C - Centralized
D - Distributed
H - Collaborative

Stage
A - Analysis
PR - Public Release

Algorithms
● - GWAS
○ - x^2 tests
◐ - Read Maps
◑ - Other Tests

Most of these solutions are tailored for the collaboration between entities in a federated way, meaning that the data is distributed across multiple entities, and computation is performed collaboratively [8], [10], [16], [41]. However, these approaches are unsuitable when outsourcing GWAS computation to a single cloud environment, and *Gyosa* differs from them as it parallelizes the computation among different servers to decrease the execution time. Another relevant aspect lies on the different types of computation considered. Privacy-preserving solutions have been applied to a wide range of tasks, typically on the data analysis stage, such as read mappings [20], [39], statistical tests [2], [8], [17], [18], [21], [29], [33], [41], queries (other tests) [6], [10], [16], [17], [33] and even GWAS [18], [19], [30]. The only distributed solution among these, *HySecFlow* [39], focuses on the problem of read mappings. While using a specific library to perform this task, the solution does not allow the seamless integration of other tasks (*e.g.*, GWAS, statistical tests, PCA).

Based on this and to the best of our knowledge, *Gyosa* is the first solution to support the execution of GWAS in a distributed environment while ensuring data privacy. Finally, by relying on *Glow*, *Gyosa* stands out from other solutions by enabling the addition of new tasks (*e.g.,* statistical tests, genomic imputation, and querying) in the genomic pipeline, making it easier to extend the secure analysis pipeline.

III. Methods

A. GWAS

Genomic pipelines comprise tasks like Short-read Sequence Alignment, Genome Imputation, Variant Call, and GWAS. This work focuses on GWAS, which tests the correlation between an SNP's allele frequency and specific phenotypes. Such studies have been performed for many phenotypes, including several diseases. Tested individuals are separated into case and control groups and require large sample sizes to achieve significant and robust results [37].

1) Statistical Methods.: To evaluate the performance of *Gyosa*, we illustrate its practical application by employing Logistic and Linear Regression algorithms and test the association with continuous and binary phenotypes. We employed the X^2 test, as in [19], to assess *Gyosa*'s scalability and performance with varying workloads and increasing workers. These algorithms are chosen based on the current state-of-the-art, corroborating their efficiency in conducting GWAS [37][2].

Linear Regression checks the relation between one dependent continuous variable (*e.g.*, weight or blood pressure) and multiple independent ones, which can include various genetic markers or environmental factors. As such, this algorithm is used for finding specific genetic correlations with traits and diseases [13]. The algorithm is efficiently implemented based on matrix multiplications and inversions. For a matrix of m x n dimensions, where m is the number of samples and n is the number of features, the time complexity approximates $O(m * n^2 + n^3)$ [35]. This reflects the costs associated with processing the data: the first term, $O(m * n^2)$, arises from calculating the product of the sample size m and the square of the number of features n^2, while the second term, $O(n^3)$, corresponds to the operations required for matrix inversion.

Logistic Regression performs binary classification based on dependent variables to distinguish between case and control cases. For instance, logistic regression can assess the likelihood of a phenotypic observation belonging to a class based on genetic markers, revealing connections between phenotypic features and genetic data [3], [37]. The time complexity of Logistic Regression can be decomposed into d as the size of the phenotype vector and n as the covariates or features of the phenotype, $O(nd)$. As the number of features or the complexity of the phenotype increases, the time required for logistic regression analysis increases linearly, making it practical for handling large genomic datasets [35].

X^2 ***Test*** tests if the observed and the expected frequencies (allele counts) in the case-control groups differ significantly. The test can be defined as follows:

[2]Principal Component Analysis (PCA) is not currently implemented in Glow, and its privacy requirements may differ from *Gyosa* current implementation due to its increasing data exchange and convergence rate.

$$X^2 = \sum \frac{(observed_i - expected_i)^2}{expected_i} \quad (1)$$

, in which we find the observed and expected frequency of the ith SNP. Differently from previous statistical methods, X^2 focuses solely on this frequency. The time complexity of X^2 is $O(n)$, where n is the number of samples [31].

B. Design

Gyosa is a privacy-preserving distributed GWAS-focused solution. It builds on top of SOTERIA [4] and resorts to Apache Spark and Glow to offer a distributed genomic analysis framework. *Gyosa* uses Intel SGX, which establishes secure memory regions called enclaves, providing a robust privacy-preserving solution and focusing on privacy and data confidentiality. These enclaves protect sensitive data during processing and prevent unauthorized access. The Supplementary Material further details Apache Spark, Glow, and Intel SGX.

SOTERIA is a privacy-preserving distributed machine learning solution that leverages Intel SGX to protect sensitive data and computations. It is designed to handle general-purpose ML workloads, which differ from GWASes. SOTERIA leverages the distributed processing nature of Apache Spark and its MLlib library. The main novelty of SOTERIA stems from its partitioning scheme that allows the distribution of the computation across secure and non-secure workers.

We emphasize that *Gyosa* differs from SOTERIA by considering a different processing pipeline for genomic association tests. This is highlighted in three main aspects, further detailed: *i)* supporting a new framework (*i.e.*, Glow), *ii)* including transparent encryption of a new type of dataset file format (*i.e.*, VCF files), and *iii)* redefining the sensitive analysis steps that must be performed inside secure enclaves.

Furthermore, *Gyosa* differs from state-of-the-art solutions which *i)* perform all computations inside TEEs, and *ii)* are designed for single-node setups. To overcome TEEs' memory constraints and computational bottlenecks due to the limited enclave size, *Gyosa* adopts a hybrid partitioning scheme. This approach allows efficiently distributing the computational workload between secure enclaves (for sensitive genomic data) and non-secure environments (for non-sensitive tasks), providing better performance than previous solutions. Additionally, *Gyosa* leverages a distributed architecture that scales across multiple servers, addressing the scalability limitations inherent in single-node SGX solutions. By integrating Apache Spark and Glow with Intel SGX, *Gyosa* ensures that computations are both privacy-preserving and performant, making it a more practical choice for large-scale genomic studies.

1) Threat Model: *Gyosa* adopts the standard SGX threat model supported by existing research [4], [8], [16], [19]. We consider a scenario where a client seeks to use sensitive genomic data and perform computation on top of it at third-party infrastructures. In this model, the client and the hardware are deemed trustworthy, whereas the third-party infrastructure's other components (*i.e.,* host OS, libraries) are regarded as untrusted. This leads to an honest-but-curious adversary model, where the adversary is honest and adheres to the protocol but is curious and seeks to obtain information.

2) Gyosa's Key Components: *Gyosa* is split into the client module, deployed on a trusted infrastructure, and the cluster module, deployed on an untrusted site (Figure 1). The client module encompasses the encryption of the genomic data and the submission of the GWASes to the untrusted infrastructure. The cluster module includes a distributed Apache Spark and Glow cluster to which the client will submit the analysis.

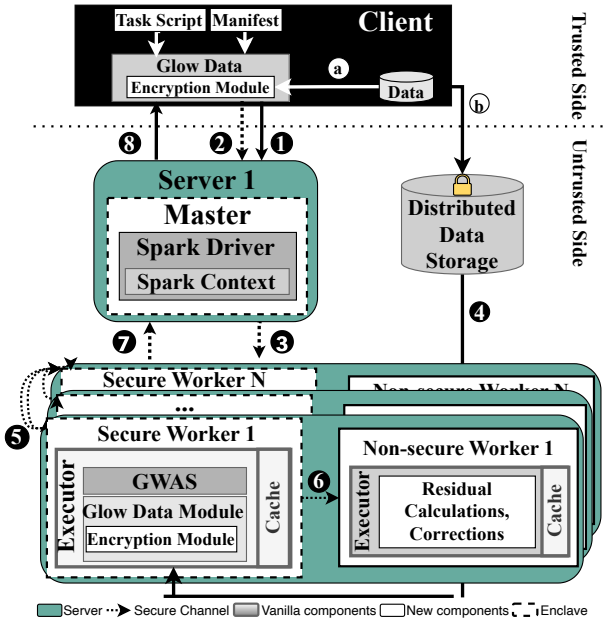


Fig. 1: *Gyosa* architecture schematic. A Biocenter encrypts and sends its data to distributed data storage. The genomic association studies run securely and distributed with *Gyosa*.

Client Module. The client module includes three main operations. First, it allows the encryption of VCF files based on authenticated encryption. To provide this, *Gyosa* employs an AES-GCM-128 authenticated encryption cipher mode, which is added to Glow by providing a new class to encrypt VCFs. Sensitive data is transparently encrypted before leaving the trusted premises, thus avoiding changes in how users implement or specify their GWAS. Second, the client module handles the secure outsourcing of the GWAS requests issued by users to the Spark Cluster. Finally, this module allows the decryption of the analysis results when returned to users.

To perform a GWAS, users must specify a task script file where the analysis steps and all the required parameters are defined. This file contains sensitive information about the analysis task that cannot be leaked or tampered with. Thus, the encryption module encrypts the task script, which can then be sent via unprotected network channels.

The manifest is a predefined file that contains the libraries to run the Glow pipeline and the path for the dataset. To ensure data security and integrity when exchanging this file, a secure channel between the client and the Spark Cluster is created only once at the bootstrap phase. This secure channel is also used to transmit the user's encryption key, which is then used and handled only inside the SGX enclaves. Implementation-

wise, the channel can be created by resorting to a protocol such as TLS with Pre-Shared Keys (TLS-PSK), which provides robust authentication and encryption while being less resource-intensive than traditional SSL/TLS handshakes.

Cluster Module. The cluster follows Spark's workflow on the untrusted site, with a master and N workers running on distinct servers. The master is deployed inside an SGX enclave at the untrusted server since the Spark Driver and Spark Context modules require reading plaintext information (*i.e.,* task script) to distribute the processing tasks to the workers.

We deploy a secure and non-secure worker at each cluster server. The secure worker runs inside an SGX enclave and handles all the computation over sensitive data, while the non-secure worker handles non-sensitive data and runs outside of SGX. The exchange of sensitive information between secure workers and with the master is done via secure network channels using the TLS-PSK protocol (Figure 1).

Partitioned design. For its partitioned scheme, *Gyosa* redefines the computation partitioning across secure and non-secure workers. In *Gyosa*, non-sensitive computations involve residual values (*e.g.*, matrix calculation of metadata or calculations over single genotype information) and the correction of statistical tests, which in Glow are based on the Firth's approximation algorithm [24]. This correction is performed as a score test, comparing the predicted and observed values to validate the resulting P-values. All remaining operations (*e.g.*, read operations on top of the VCFs, dataframe transformations, and regression operations) are conducted on sensitive data within SGX enclaves at the secure workers.

For example, in Logistic Regression, p-values are determined through the Firth approximation algorithm (detailed in Supplementary Material, Section 4). Since this algorithm processes only p-values, and not sensitive data, it is performed outside the secure worker's enclave. This design ensures that all sensitive data is consistently encrypted, even when transmitted and stored outside the secure environment.

3) Gyosa's Workflow: The client ⓐ resorts to *Gyosa*'s encryption module to encrypt VCF files at the trusted premises. Then, ⓑ encrypted data is sent to a distributed data storage shared by various servers on the untrusted infrastructure. Similarly, the client specifies the studies it wants to run as task scripts and encrypts these before sending them to the untrusted infrastructure ❶. *Gyosa* assumes a single bootstrapping phase between client and master in which a secure channel is established and used to share the manifest file and the client's key, required to encrypt the VCF files and tasks' scripts. This key is also used to decrypt the final results ❷.

Following a master–worker architecture, the master, running inside an SGX-enabled server, receives the task the client wants to perform and the path (within the manifest file) for the encrypted dataset. The former is sent encrypted through an insecure channel, while the latter is forwarded inside the previously established secure channel. After decrypting the task script inside the secure enclave, the master forwards specific sub-tasks to each secure worker ❸ through secure channels established between their enclaves. With these sub-tasks, secure workers can fetch the required data from the distributed storage backend and perform the computation. Since data is encrypted at the storage backend, it must be fetched and decrypted at each secure worker enclave to be processed in plaintext (*i.e.*, inside an SGX enclave) ❹.

Then, workers broadcast intermediate results between them ❺. In addition, following the partitioning of computation, secure workers broadcast metadata to non-secure workers. Again, after performing their computation, the non-secure workers broadcast the information back to the secure workers ❻. Sensitive information shared across secure workers is done through secure channels established between their enclaves. In the final worker-related stage, a consensus regarding the result is reached and sent to the master's enclave through a secure channel ❼. Final results are aggregated and encrypted by the master, with the client's key, and sent to the client for transparent decryption at the trusted premises ❽.

IV. Results

Gyosa was evaluated to understand the impact of adding privacy protection on top of a baseline stack composed by Apache Spark and Glow, which does not provide such guarantees. Two main questions are answered with this evaluation: *i)* How does the execution time of *Gyosa* compare with a non-secure baseline setup? *ii)* How does *Gyosa* behave when increasing the workload size and the number of servers?

A. Testbed

Dataset. For the benchmark, we used a real-world dataset by the *Genome in a Bottle* Consortium [42], with genomes sequenced as part of the Human Genome Project, namely, data from the Ashkenazim Trio family (father, mother, and son). Phenotype information was simulated with the PhenotypeSimulator [25]. The algorithms were tested for different workloads by scaling the original dataset by several factors to reach sizes 1, 4, 16, and 32 GB. For the scalability tests, we generated a synthetic dataset of 80,000 VCF files with $1 * 10^6$ unique SNPs, each VCF file representing one individual. We define the workload size as 20k, 40k, 60k, and 80k individuals.

Environment. Tests were performed in a cluster of 4 servers with OS Ubuntu 18.04.4 LTS and Linux kernel 4.15.0. Each machine has a 10Gbps Ethernet card connected to a dedicated local network and 16 GB of memory. *Gyosa* uses Apache Spark 3.2.1 and was deployed with version 2.6 of the Intel SGX Linux SDK with driver 1.8, with 4 GB of memory. The client and Spark master ran on one server, while Spark workers were deployed on the remaining servers.

Encryption Mode. We used AES-GCM-128 for its strong security and efficiency, ensuring confidentiality and data integrity. AES-GCM-128 adds a fixed overhead of 28 bytes per encrypted message (16 bytes for the authentication tag and 12 bytes for the IV). This overhead is negligible for large messages, such as those in HDFS (Apache Spark's storage backend), which uses a 128 MB block size — making the 28 bytes insignificant compared to the block size.

Setups. We compare *Gyosa* against a baseline setup without security measures (*i.e.,* Glow) to quantify the performance impact of our privacy-preserving approach. Further, we compare *Gyosa*' distributed design and partitioning scheme against

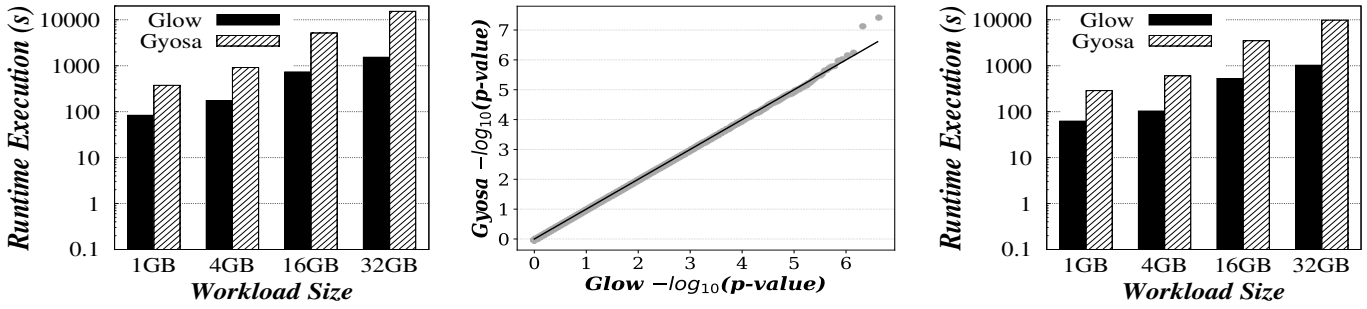


Fig. 2: The impact of *Gyosa* on variable workloads (1GB, 4GB, 16GB, and 32GB) with the Linear Regression and Logistic Regression algorithm compared to baseline. *a)* Execution time of Linear Regression in logarithmic scale; *b)* Comparison of p-values between approaches for Linear Regression; *c)* Execution time of Logistic Regression in logarithmic scale.

state-of-the-art approaches, *i.e.*, that are designed for a single-node setup and perform all computations inside SGX enclaves (see also Section 5 of the Supplementary Material).

Performance Metrics. We focused on three key performance metrics: runtime execution, p-value comparison, and memory usage. We measured the time taken for each algorithm across various dataset sizes to assess efficiency and scalability and compared p-values from our solution with a baseline lacking privacy guarantees to evaluate accuracy. Lastly, we monitored the memory consumption of our system while increasing the number of servers to assess resource efficiency.

Experimentation. All experiments were repeated at least 3 times. This approach allowed us to average results, ensuring statistical stability and minimizing the impact of potential variability in the environment. The scalability tests include over 72 experiments conducted across nine days, while runtime results are based on over 80 experiments carried out over 56 hours.

B. Secure GWAS

To assess the impact of *Gyosa*'s security mechanisms on the execution time of the algorithms, we compare the results with the ones for the baseline setup. Figure 2 shows the Linear Regression and Logistic Regression algorithm results. In Figure 2a, for a workload of 1 GB, the runtime overhead of the linear regression algorithm is around 4.5x. For 32 GB, the overhead of *Gyosa* reaches the maximum for the performed tests, with 10x compared to the baseline setup. Furthermore, this result underlines the trade-off between enhanced privacy and processing efficiency, demonstrating *Gyosa*'s ability to handle large datasets with acceptable performance cost despite the added layer of security. The comparison of *p-values* between the two approaches shows negligible differences (see Figure 2b). The slightly different values observed result from the final approximation since it deals with small values and reverts them to $-log_{10}(p-value)$. Similarly, for the Logistic Regression algorithm and workload size of 1 GB and 32 GB, *Gyosa* has a runtime overhead of 4x and 9.5x (see Figure 2c).

Gyosa can be used in a cluster setup with multiple servers, each including one secure worker and one non-secure worker. Figure 3a shows the results of the overhead imposed for the X^2 frequency test. *Gyosa* leverages the scalability offered by Apache Spark and Glow, as shown by the experiments with up to 3 servers. We verify a linear decrease in the runtime execution when increasing the number of servers for both *Gyosa* and baseline setups. Namely, when comparing the execution time obtained by running this experiment over 80,000 VCF files with one and three servers, the runtime execution decreases up to 2.7X (*i.e.,* 2.4 hours). Regarding the security guarantees, the runtime overhead ranges from 1.3x to 3x for a workload of 40k individuals with three servers. These results confirm *Gyosa*'s ability to effectively use a distributed architecture to balance performance with privacy requirements.

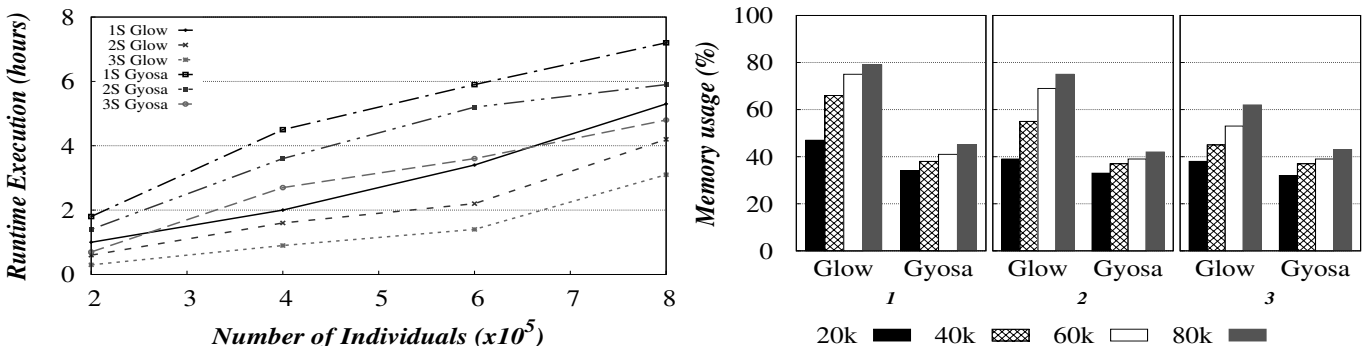
V. Discussion

Next, we analyze the security guarantees of *Gyosa* and pinpoint how our design ensures data confidentiality and protection against known attacks. These guarantees follow Soteria's theoretical proofs and are complemented by an empirical performance-based security analysis.

Security Analysis. *Gyosa* combines different mechanisms to safeguard users from attacks (see Section 2 of Supplementary Material for more details on these attacks). It provides transparent authenticated encryption, which protects sensitive data from being disclosed to unwanted parties and ensures anti-tampering properties for clients' data stored in untrusted infrastructures. This feature protects users from poisoning attacks by limiting access to the plaintext data and not allowing the addition of poisoned data. Membership inference and re-identification attacks are subject to an attacker's previous knowledge of the genomic data. By ensuring that private data, while at rest and in transit, is always encrypted and that any sensitive computation is performed on SGX enclaves, *Gyosa* avoids disclosing such knowledge to attackers.

Partitioning the computation across the secure and non-secure workers improves the performance but increases the attack surface. However, previous work shows that genomic data cannot be inferred from the information leaked from sharing metadata and statistical information [27]. Given this assumption, and by not changing the main security protocol specified by SOTERIA, *Gyosa* can keep the information leakage contained to avoid the success of the aforementioned attacks. The full proofs for SOTERIA's protocol, followed by *Gyosa*, are available at [1].

Our benchmarks, with over 152 combined experiments, reveal that the performance penalty in *Gyosa* is proportional to the data size for regression and classification tasks. Notably,



(a) (b)

Fig. 3: Runtime execution (hours) of the X^2 test for 1 up to 3 servers, supporting workers, and 20k to 80k VCFs from the synthetic dataset. a) Runtime Execution for X^2 test in hours; b) Memory Usage in percentage (%) for X^2 statistical test.

adding security guarantees in *Gyosa* does not alter the results from the statistical tests, making *Gyosa* a novel and practical option for privacy-preserving genomic analysis.

Performance and Scalability Analysis. The performance overhead of *Gyosa* 's security features, including encryption and secure enclave processing, was assessed by comparing execution times with and without these mechanisms. The results highlight the trade-offs between data privacy and computational efficiency across various workloads, evaluating *Gyosa*'s security performance regarding computational costs.

Increasing the number of servers allows for distributed and parallelized GWAS computations, significantly reducing runtime and memory usage, as shown in Figures 3a and 3b. Figure 3b shows that despite Glow being a memory-intensive solution [11], increasing the number of servers leads to a decrease in the mean memory usage. Compared with other SGX-based solutions, namely [5], *Gyosa* shows comparable runtime overhead. Notably, *Gyosa* distinguishes from all these state-of-the-art solutions [8], [19] by allowing distributed computation across several servers in a cloud environment.

High-end servers used by state-of-the-art solutions (*e.g.*, a configuration with > 40 cores, > 2.0TB of physical memory, and 10TB of disk space [8]) are not widely available and require substantial resources for their setup and maintenance. A cost-efficient alternative is to use cloud environments that allow the distribution of computation by relying on several servers, reducing the execution time of genomic analysis. In Google Cloud, as of 2024, with $391.35€$ per month, one could opt for *i)* one server with 16 cores and 64 GiB of memory or *ii)* four servers with four cores each and 16 GB of memory. While solution *i)* provides 730 h of computation, solution *ii)* provides a total of $2,920$ monthly hours of computation [12].

Currently, the second generation of SGX includes a 128 MB page cache and data that exceeds this must be swapped to/from an encrypted memory region, resulting a performance penalty [9]. In our setup, the amount of encrypted memory attributed to SGX is limited to 4GB in each server, leading to additional disk swapping for memory-intensive GWAS algorithms [9]. This SGX's limitation cannot be solved through hardware updated but is mitigated by *Gyosa*'s distributed design. By distributing computation across multiple servers, *Gyosa* leverages from the aggregated page cache and encrypted memory sizes of multiple servers, justifying the decreased runtime execution observed for *Gyosa* in Figure 3a.

Contributions and Limitations.

Unlike previous solutions, *Gyosa* leverages a hybrid partitioning scheme and distributed execution model to improve performance while maintaining data privacy. Our comparison against a fully enclave-based SOTA solution (detailed in the Supplementary Material) further underscores the substantial runtime and scalability improvements enabled by *Gyosa*'s design. The results demonstrate that the security overhead is lower for smaller workloads since computations fit within the enclave's memory. For larger workloads, *Gyosa* mitigates performance penalties by distributing computations across multiple servers. These trade-offs make *Gyosa* an effective and scalable privacy-preserving solution for genomic data analysis, balancing computational efficiency with strong security guarantees. Our work leaves some open challenges to be further explored. First, the current implementation of *Gyosa* supports association tests, namely regression-based algorithms and statistical tests. In the future, it would be valuable to include other GWAS algorithms, such as Principal Component Analysis. Second, *Gyosa* does not focus on the mitigation of known SGX side-channel attacks, which are a known threat to the security of the system. We highlight that solutions addressing concerns such as *Denial of Service (DoS)*, *side-channel* attacks, or *memory access patterns* can be employed in *Gyosa* [28]. However, this research is orthogonal to the one proposed here. Additionally, our solution could incorporate other techniques, such as DP, to provide privacy for collaborative and federated GWAS studies involving multiple institutions. Notably, these limitations do not hinder *Gyosa* contributions and applicability. Indeed, solving such limitations is an important direction for future research.

VI. Conclusion

Gyosa offers the first end-to-end privacy-preserving genomic data analytics solution built on top of Apache Spark and Glow. Distributing GWAS computation across multiple untrusted servers allows researchers to study larger amounts of sensitive genomic data efficiently. Furthermore, by following a computation partitioning scheme tailored for GWAS, *Gyosa* decreases the amount of data transferred and processed at

secure enclaves, boosting the algorithms' performance while not compromising security or the quality of the analysis.

Finally, *Gyosa* stands out from other solutions by enabling the addition of new tasks (*e.g.,* statistical tests, genomic imputation, and querying) in the genomic pipeline, making it easier to extend the secure analysis pipeline.

REFERENCES

[1] Soteria proofs, https://dbr-haslab.github.io/repository/sac23.pdf

[2] Asvadishirehjini, A., Kantarcioglu, M., Malin, B.: A framework for privacy-preserving genomic data analysis using trusted execution environments. In: 2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA). pp. 138–147. IEEE (2020)

[3] Bagos, P.G., Nikolopoulos, G.K.: A method for meta-analysis of case-control genetic association studies using logistic regression. Statistical applications in genetics and molecular biology **6**(1) (2007)

[4] Brito, C., Ferreira, P., Portela, B., Oliveira, R., Paulo, J.: Soteria: Preserving privacy in distributed machine learning. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. SAC '23, ACM (2023)

[5] Chen, F., Wang, S., Jiang, X., Ding, S., Lu, Y., Kim, J., Sahinalp, S.C., Shimizu, C., Burns, J.C., Wright, V.J., et al.: Princess: Privacy-protecting rare disease international network collaboration via encryption through software guard extensions. Bioinformatics **33**(6), 871–878 (2017)

[6] Chenghong, W., Jiang, Y., Mohammed, N., Chen, F., Jiang, X., Al Aziz, M.M., Sadat, M.N., Wang, S.: Scotch: Secure counting of encrypted genomic data using a hybrid approach. In: AMIA Annual Symposium Proceedings. vol. 2017, p. 1744. American Medical Informatics Association (2017)

[7] Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008)

[8] Dokmai, N., Kockan, C., Zhu, K., Wang, X., Sahinalp, S.C., Cho, H.: Privacy-preserving genotype imputation in a trusted execution environment. Cell Systems **12**(10), 983–993 (2021)

[9] El-Hindi, M., Ziegler, T., Heinrich, M., Lutsch, A., et al.: Benchmarking the second generation of intel sgx hardware. In: Data Management on New Hardware (2022)

[10] Froelicher, D., Troncoso-Pastoriza, J.R., Sousa, J.S., Hubaux, J.P.: Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets. IEEE Transactions on Information Forensics and Security **15**, 3035–3050 (2020)

[11] Glow: An open-source toolkit for large-scale genomic analysis. https://projectglow.io/, (02/06/2023)

[12] Google: Google cloud pricing calculator. https://cloud.google.com/products/calculator, (02/23/2023)

[13] Hayes, B.: Overview of statistical methods for genome-wide association studies (gwas). Genome-wide association studies and genomic prediction pp. 149–169 (2013)

[14] Hwang, S., Ozturk, E., Tsudik, G.: Balancing security and privacy in genomic range queries. ACM Transactions on Privacy and Security (2022)

[15] Iqbal, S., Kiah, M.L.M., Dhaghighi, B., Hussain, M., Khan, S., Khan, M.K., Choo, K.K.R.: On cloud security attacks: A taxonomy and intrusion detection and prevention as a service. Journal of Network and Computer Applications (2016)

[16] Jafarbeiki, S., Sakzad, A., Kermanshahi, S.K., Gaire, R., Steinfeld, R., Lai, S., Abraham, G., Thapa, C.: Privgendb: Efficient and privacy-preserving query executions over encrypted snp-phenotype database. Informatics in Medicine Unlocked **31**, 100988 (2022)

[17] Karimi, S., Jiang, X., Dolin, R.H., Kim, M., Boxwala, A.: A secure system for genomics clinical decision support. Journal of Biomedical Informatics **112**, 103602 (2020)

[18] Kim, A., Song, Y., Kim, M., Lee, K., Cheon, J.H.: Logistic regression model training based on the approximate homomorphic encryption. BMC medical genomics **11**(4), 23–31 (2018)

[19] Kockan, C., Zhu, K., Dokmai, N., Karpov, N., Kulekci, M.O., Woodruff, D.P., Sahinalp, S.C.: Sketching algorithms for genomic data analysis and querying in a secure enclave. Nature methods **17**(3), 295–301 (2020)

[20] Lambert, C., Fernandes, M., Decouchant, J., Esteves-Verissimo, P.: Maskal: Privacy preserving masked reads alignment using intel sgx. In: 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS). pp. 113–122. IEEE (2018)

[21] Li, W., Kim, M., Zhang, K., Chen, H., Jiang, X., Harmanci, A.: Collagene enables privacy-aware federated and collaborative genomic data analysis. Genome Biology **24**(1), 204 (2023)

[22] Lu, W.J., Yamada, Y., Sakuma, J.: Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption. In: BMC medical informatics and decision making. vol. 15, pp. 1–8. Springer (2015)

[23] Malin, B.: Re-identification of familial database records. In: AMIA annual symposium proceedings. vol. 2006, p. 524. American Medical Informatics Association (2006)

[24] Mbatchou, J., Barnard, L., Backman, J., Marcketta, A., Kosmicki, J.A., Ziyatdinov, A., Benner, C., O'Dushlaine, C., Barber, M., Boutkov, B., et al.: Computationally efficient whole-genome regression for quantitative and binary traits. Nature genetics **53**(7), 1097–1103 (2021)

[25] Meyer, H.V., Birney, E.: Phenotypesimulator: A comprehensive framework for simulating multi-trait, multi-locus genotype to phenotype relationships. Bioinformatics **34**(17), 2951–2956 (2018)

[26] Müller, N., Kowatsch, D., Böttinger, K.: Data poisoning attacks on regression learning and corresponding defenses. In: 2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC). pp. 80–89. IEEE (2020)

[27] Ney, P., Ceze, L., Kohno, T.: Genotype extraction and false relative attacks: Security risks to third-party genetic genealogy services beyond identity inference. In: NDSS (2020)

[28] Oleksenko, O., Trach, B., Krahn, R., Silberstein, M., Fetzer, C.: Varys: Protecting sgx enclaves from practical side-channel attacks. In: USENIX Annual Technical Conference (2018)

[29] Pascoal, T., Decouchant, J., Boutet, A., Esteves-Verissimo, P.: Dyps: Dynamic, private and secure gwas. Proceedings on Privacy Enhancing Technologies (2021)

[30] Pascoal, T., Decouchant, J., Boutet, A., Völp, M.: I-gwas: Privacy-preserving interdependent genome-wide association studies. Proceedings on Privacy Enhancing Technologies **1**, 437–454 (2023)

[31] Pirhaji, L., Kargar, M., Sheari, A., Poormohammadi, H., Sadeghi, M., Pezeshk, H., Eslahchi, C.: The performances of the chi-square test and complexity measures for signal recognition in biological sequences. Journal of Theoretical Biology **251**(2), 380–387 (2008)

[32] Richmond, S.: Uncovering the true costs of it infrastructure. https://www.forbes.com/sites/forbestechcouncil/2021/11/03/uncovering-the-true-costs-of-it-infrastructure/?sh=328de8b67baf, (02/16/2023)

[33] Sadat, M.N., Al Aziz, M.M., Mohammed, N., Chen, F., Jiang, X., Wang, S.: Safety: secure gwas in federated environment through a hybrid solution. IEEE/ACM transactions on computational biology and bioinformatics **16**(1), 93–102 (2018)

[34] Slatko, B.E., Gardner, A.F., Ausubel, F.M.: Overview of next-generation sequencing technologies. Current protocols in molecular biology **122**(1), e59 (2018)

[35] Surana, S.: Computational complexity of machine learning models. https://www.kaggle.com/general/263127, (02/06/2023)

[36] TACC: Texas advanced computing center. https://www.tacc.utexas.edu/, (02/06/2023)

[37] Uffelmann, E., Huang, Q.Q., Munung, N.S., De Vries, J., Okada, Y., Martin, A.R., Martin, H.C., Lappalainen, T., Posthuma, D.: Genome-wide association studies. Nature Reviews Methods Primers (2021)

[38] Wang, R., Li, Y.F., Wang, X., Tang, H., Zhou, X.: Learning your identity and disease from research papers: information leaks in genome wide association study. In: Proceedings of the 16th ACM conference on Computer and communications security. pp. 534–544 (2009)

[39] Widanage, C., Liu, W., Li, J., Chen, H., Wang, X., Tang, H., Fox, J.: Hysec-flow: Privacy-preserving genomic computing with sgx-based big-data analytics framework. In: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD). pp. 733–743. IEEE (2021)

[40] Zaharia, M., Xin, R.S., Wendell, P., Das, T., et al.: Apache spark: a unified engine for big data processing (2016)

[41] Zhu, R., Jiang, C., Wang, X., Wang, S., Zheng, H., Tang, H.: Privacy-preserving construction of generalized linear mixed model for biomedical computation. Bioinformatics **36**(Supplement_1), i128–i135 (2020)

[42] Zook, J.M., Catoe, D., McDaniel, J., Vang, L., Spies, N., Sidow, A., Weng, Z., Liu, Y., Mason, C.E., Alexander, N., et al.: Extensive sequencing of seven human genomes to characterize benchmark reference materials. Scientific data **3**(1), 1–26 (2016)