UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM

INSTITUTO DE COMPUTAÇÃO- ICOMP

PROGRAMA PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

Protecting confidential data in cloud environments

Ronny Peterson Guimarães

Manaus - AM

December - 2024

Ronny Peterson Guimarães

Protecting confidential data in cloud environments

Thesis presented to the Graduate Program in Informatics at the Federal University of Amazonas, as a partial requirement for the degree of Doctor in Informatics.

Advisors

Advisor: Altigran Soares da Silva, PhD.

Co-Advisor: André Luiz da Costa Carvalho, PhD.

Universidade Federal do Amazonas - UFAM

Instituto de Computação- IComp

Manaus - AM

December - 2024

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Guimarães, Ronny Peterson
G963p Protecting confidential data in cloud environments / Ronny Peterson Guimarães . 2024
90 f.: il. color; 31 cm.

Orientador: Altigran Soares da Silva
Coorientador: André Luiz da Costa Carvalho
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. Data privacy. 2. Cloud computing. 3. Security. 4. Sgx. I. Silva, Altigran Soares da. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"PROTECTING CONFIDENTIAL DATA IN CLOUD ENVIRONMENTS"

RONNY PETERSON GUIMARÃES

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Dr. Altigran Soares da Silva - **Presidente**

Prof. Dr. Eduardo Luzeiro Feitosa - **Membro Interno**

Profa. Dra. Tanara Lauschne - **Membro Externo**

Prof. Dr. Joao Marcos Bastos Cavalcanti - **Membro Externo**

Manaus, 18 de dezembro de 2024.

Documento assinado eletronicamente por **Altigran Soares da Silva**, **Professor do Magistério Superior**, em 22/01/2025, às 12:48, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Eduardo Luzeiro Feitosa**, **Professor do Magistério Superior**, em 22/01/2025, às 14:28, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Tanara Lauschner**, **Professor do Magistério Superior**, em 24/01/2025, às 09:18, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **João Marcos Bastos Cavalcanti**, **Professor do Magistério Superior**, em 24/01/2025, às 16:12, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2418833** e o código CRC **BE523AAE**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.053380/2024-98 SEI nº 2418833

I dedicate this work, first and foremost, to God, my source of strength and inspiration, who has been my refuge and fortress throughout this journey. I am deeply grateful to my parents and my entire family, whose unwavering support was crucial in accomplishing this challenge. In particular, I dedicate this achievement to my wife, whose understanding and affection motivated me to keep going, and to my sons and daughters, who have always been by my side, offering love and encouragement.

Acknowledgements

First and foremost, I thank God, whose light and strength guided me every step of this journey.

I am grateful to my advisor, Professor Altigran Soares da Silva, and my co-advisor, Professor André Luis Carvalho, for their guidance, unwavering support, and the trust they placed in me throughout this process.

I thank Professor Christof Fetzer and Professor André Martin for their support and assistance with the use of SCONE during my studies.

I am immensely thankful to my parents, Vera da Silva and Francisco de Assis, and to my father figure, Ocimar Bezerra, for always believing in me and encouraging me to pursue my dreams.

I would also like to thank my sisters, Carla and Deborah, and all my nephews and nieces, who have always brought joy and motivation to my life.

A special thank you goes to my family, particularly to my wife, Gabriele Guimarães, for her love and steadfast support. I am grateful to my children, Jhonatan, Joabe, Rafaelle, and Fernanda, for being the reason for my efforts and for inspiring me to be a better person every day.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work was partially supported by Amazonas State Research Support Foundation - FAPEAM - through the POSGRAD project.

"I can do all things through Him who strengthens me".

The Bible passage from Philippians 4:13 states

Protecting confidential data in cloud environments

Author: Ronny Peterson Guimarães

Advisor: Altigran Soares da Silva, PhD.

Co-Advisor: André Luiz da Costa Carvalho, PhD.

Abstract

The growing adoption of cloud computing has brought significant challenges for protecting sensitive data, particularly when such data is stored and processed in shared public infrastructures. This work addresses these challenges by proposing solutions for the protection of sensitive data in cloud environments, focusing on methods that ensure privacy without compromising efficiency in data access and manipulation.

This thesis presents the *Vallum* platform, which leverages hardware-based security (Intel *SGX*) to protect sensitive data, alongside an optimized version that adopts selective protection through vertical partitioning, aiming to improve performance. The research examines the impact of different privacy-preserving mechanisms on system performance, particularly in terms of throughput and response time, and evaluates the trade-offs between implementing robust security measures and the need to maintain processing efficiency.

Through detailed experimental testing, the results show that while full protection with SCONE/*SGX* (Vallum 1) leads to significant performance degradation, the selective protection approach (Vallum 2) provides a more effective balance, improving system scalability without compromising security. These results provide a foundation for understanding how cloud database systems can balance confidentiality requirements with performance demands, making them more suitable for large-scale applications.

Furthermore, this work contributes to the field with academic publications, including presentations at renowned international conferences and journal articles. The first

version of the Vallum platform was developed within the context of the international $ATMOSHPERE$ project, a collaboration between research institutions and companies in Brazil and Europe, validating its applicability in real-world scenarios.
Therefore, this research proposes solutions that ensure the privacy protection of sensitive data in cloud environments while maintaining efficiency in processing large volumes of data, enabling more secure use of cloud computing by organizations.

Keywords: Data privacy, Cloud computing, Security, Sgx.

Protecting confidential data in cloud environments

Author: Ronny Peterson Guimarães
Advisor: Altigran Soares da Silva, PhD.
Co-Advisor: André Luiz da Costa Carvalho, PhD.

Resumo

A crescente adoção da computação em nuvem trouxe desafios significativos para a proteção de dados sensíveis, particularmente quando esses dados são armazenados e processados em infraestruturas públicas compartilhadas. Este trabalho aborda esses desafios propondo soluções para a proteção de dados sensíveis em ambientes de nuvem, com foco em métodos que garantam privacidade sem comprometer a eficiência no acesso e manipulação dos dados.

Esta tese apresenta a plataforma *Vallum*, que utiliza segurança baseada em hardware (Intel *SGX*) para proteger dados sensíveis, juntamente com uma versão otimizada que adota proteção seletiva por meio de particionamento vertical, visando melhorar o desempenho. A pesquisa examina o impacto de diferentes mecanismos de preservação de privacidade no desempenho do sistema, particularmente em termos de taxa de transferência e tempo de resposta, e avalia os trade-offs entre a implementação de medidas de segurança robustas e a necessidade de manter a eficiência no processamento.

Por meio de testes experimentais detalhados, os resultados mostram que, enquanto a proteção total com SCONE/*SGX* (Vallum 1) leva a uma degradação significativa no desempenho, a abordagem de proteção seletiva (Vallum 2) proporciona um equilíbrio mais eficaz, melhorando a escalabilidade do sistema sem comprometer a segurança. Esses resultados fornecem uma base para entender como os sistemas de banco de dados

em nuvem podem equilibrar os requisitos de confidencialidade com as demandas de desempenho, tornando-os mais adequados para aplicações em larga escala.

Além disso, este trabalho contribui para a área com publicações acadêmicas, incluindo apresentações em renomadas conferências internacionais e artigos em periódicos. A primeira versão da plataforma Vallum foi desenvolvida no contexto do projeto internacional $ATMOSPHERE$, uma colaboração entre instituições de pesquisa e empresas no Brasil e na Europa, validando sua aplicabilidade em cenários do mundo real.

Portanto, esta pesquisa propõe soluções que asseguram a proteção da privacidade de dados sensíveis em ambientes de nuvem, mantendo a eficiência no processamento de grandes volumes de dados, permitindo um uso mais seguro da computação em nuvem por organizações.

Palavras-chave: Privacidade de dados, Computação em nuvem, Segurança, Sgx.

List of Figures

Figura 1 – Detailed architecture of Vallum 1, highlighting its components and their interactions for ensuring sensitive data protection in cloud environments. 43
Figura 2 – Case study showcasing queries and their decompositions: Q represents the original query, while Q_1 and Q_2 demonstrate the partitioning of sensitive and non-sensitive data to enhance privacy during processing. 51
Figura 3 – Illustration of query division and corresponding results, showcasing the separation of sensitive and non-sensitive data during processing. 52
Figura 4 – Illustration of an inner join operation, demonstrating how data from multiple tables is merged based on common attributes to create a unified result set. 53
Figura 5 – Illustration of anonymized results derived from the case study, showcasing the application of data anonymization techniques to protect sensitive information while preserving the utility of the dataset for analysis. 53
Figura 6 – Proposed architecture for the Vallum platform, highlighting its improved design to ensure enhanced protection of sensitive data while optimizing performance and scalability in cloud environments. . . . 54
Figura 7 – Illustration of an *AST* generated from the case study, providing a detailed representation of the hierarchical structure and relationships within the analyzed data. 58

Figura 8 – Representation of the modified *AST*, illustrating the structural changes and enhancements made to optimize data parsing and analysis efficiency. 59
Figura 9 – Comparison of response times for TPC-H benchmark queries across different systems. The graph highlights the performance variations in seconds, emphasizing the impact of privacy-preserving mechanisms on query execution times. 67
Figura 10 – Analysis of response times across privacy-preserving approaches for various values of K. This visualization emphasizes the performance differences between systems and highlights the efficiency of selective protection strategies in handling sensitive data. 74
Figura 11 – Comparison of response times for various privacy-preserving approaches across different dataset sizes. This figure highlights the scalability and efficiency of each approach, demonstrating the trade-offs between data protection and system performance. 77

List of Tables

Tabela 1 – Example of patient information. 50
Tabela 2 – Example of physical fragments. 51
Tabela 3 – Summary of the main differences between Vallum 1 and Vallum 2 . . 60
Tabela 4 – Execution times for TPC-H queries (rounded to two decimal places). 66
Tabela 5 – Response times for each scenario across different values of K (rounded to two decimal places). 73
Tabela 6 – Throughput performance results for MariaDB, Vallum 1, and Vallum 2 (rounded to two decimal places). 76

Acronyms

AES *Advanced Encryption Standard*

AMD *Advanced Micro Devices*

API *Application Programming Interface*

AST *Abstract Syntax Tree*

ATMOSPHERE *Adaptive, Trustworthy, Manageable, Orchestrated, Secure, Privacy-assuring Hybrid, Ecosystem for Resilient Cloud Computing*

CAS *Configuration and Attestation Service*

CDBMS *Common DBMS*

CPU *Central Processing Unit*

DBMS *Data Base Management System*

DRAM *Dynamic Random Access Memory*

EPC *Enclave Page Cache*

FGPA *Field Programmable Gate Array*

FHE *Fully Homomorphic Encryption*

GaV *Global as View*

GDPR *General Data Protection Regulation*

HDFS *Hadoop Distributed File System*

IBM *International Business Machines Corporation*

ODBC *Open Database Connectivity*

OLAP *On-Line Analytical Processing*

OLTP *Online Transaction Processing*

RBAC *Role-Based Access Control*

RSA *Rivest-Shamir-Adleman*

SCONE *Secure Container Orchestration*

SCPU *Secure CPU*

SDBMS *Safe DBMS*

SDK *Software Development Kit*

SGX *Software Guard eXtensions*

SQL *Structured Query Language*

TEE *Trusted Execution Environments*

TLS *Transport Layer Security*

CONTENTS

1 INTRODUCTION 17

2 BACKGROUND AND RELATED WORK 22

2.1 Background 22

2.1.1 *Trusted Execution Environments* 24

2.1.2 *Role-Based Access Control* 25

2.1.3 Privacy 26

2.1.4 Global as View 28

2.2 Related Work 30

2.2.1 Secure *DBMSs* based on Homomorphic Encryption 30

2.2.2 Secure *DBMSs* based on *TEEs* 33

3 VALLUM 38

3.1 Concepts and Assumptions 38

3.2 Vallum Overview 40

3.2.1 RBAC and Sensitive Data Protection 41

3.3 Vallum 1 42

3.3.1 Security 42

3.3.2 Authentication and Authorization 43

3.3.3 Data Engine Support and Interaction 44

3.3.4 Data Privacy 45

3.3.5 Extensibility 46

3.3.6 Monolithic vs. Microservices 46

3.4 Limitations of Vallum 1 48

4 VALLUM 2 50

4.1 Case Study . . . 50
4.2 Vallum 2 . . . 53
4.2.1 Database Management . . . 54
4.2.2 Authentication and Validation . . . 57
4.2.3 Execution Planning . . . 58
4.2.4 Query Execution and Privacy . . . 59
4.3 Vallum 1 vs. Vallum 2 . . . 60

5 EVALUATION . . . 62
5.1 Setup . . . 62
5.2 Experimental Scenarios . . . 64
5.2.1 MariaDB (Baseline) . . . 64
5.2.2 Vallum 1 . . . 64
5.2.3 Vallum 2 . . . 65
5.3 Benchmark for Performance Evaluation . . . 65
5.3.1 Results for TPC-H Queries . . . 65
5.3.2 Average Response Times . . . 67
5.3.3 Compatibility with TPC-H Queries . . . 68
5.3.4 TPC-H Analysis . . . 68
5.4 Privacy and Throughput Performance Evaluation . . . 71
5.4.1 Privacy: Vallum 1 vs Vallum 2 . . . 73
5.4.2 Comparison with MariaDB: Performance Cost Analysis . . . 74
5.4.3 Considerations on the Trade-Off Between Security and Performance . . . 75
5.4.4 Throughput Performance Evaluation . . . 75
5.4.5 Throughput Analysis . . . 76
5.4.6 Comparative Performance for Throughput: Vallum 1 vs Vallum 2 77

6 CONCLUSIONS . . . 79

Bibliography . . . 86

1

Introduction

Ever since the emergence of advancements in cloud computing (ARMBRUST et al., 2010), the challenge of data management has intensified. In such environments, applications are commonly containerized and hosted on public infrastructure, which is beyond the direct oversight of both data and application proprietors. The data manipulated in these settings is frequently sensitive, necessitating stringent measures to prevent unauthorized access and thereby ensure privacy and confidentiality (FIRESMITH, 2003).

Many organizations provide and consume cloud-based services, seeking scalability, reduction of operational costs, redundancy, and security. However, there is a latent concern of data owners regarding security, reliability, and privacy, since data is physically distributed in unfamiliar locations and without any knowledge of who may have physical or logical access to the data. While there are confidentiality agreements and guarantees that cloud service providers must implement regarding security and privacy, data owners or managers cannot be sure that data will not be manipulated, transformed, or leaked. Moreover, even if they try to comply with the necessary regulations, the very nature of the cloud, technological updates, and different laws make compliance a very complex task. In fact, in addition to cyberattacks, people without proper authorization but with privileged access to communication, media, or physical environment are also potential risk factors (ZANDESH et al., 2019; RIAZI et al., 2020). Even users with legitimate access to the system can be points of vulnerability, since without proper protection and control, sensitive data can be exposed or exploited by

these users accidentally or on purpose.

While encryption is commonly used to protect data at rest, that is, data stored in cloud databases, it is still vulnerable when data needs to be temporarily stored in memory and in plain-text format for processing operations such as queries (ZUO et al., 2018). To address this issue, researchers have proposed database management systems that process queries over homomorphically encrypted data, allowing the execution of data operations while maintaining confidentiality (POPA et al., 2011; TU et al., 2013; WONG et al., 2014; PODDAR et al., 2016; ZHU et al., 2021; REN et al., 2022). However, these approaches often come with a time overhead due to the complexity of homomorphic encryption algorithms (DOUGLAS, 2019).

Another approach to protecting sensitive data is the use of *Trusted Execution Environments* (*TEEs*), which provides a secure area within a main processor, ensuring confidentiality and integrity for code and data. *TEEs*, such as *Software Guard eXtensions* (*SGX*), employ unique architectural security measures, including hardware-based memory encryption, to isolate specific application code and data in memory. Thus, in the context of data protection and privacy in databases, there are several research researches (BAJAJ; SION, 2011; ARASU et al., 2013; GRIBOV; VINAYAGAMURTHY; GORBUNOV, 2017; PRIEBE et al., 2018; WANG et al., 2017; SUN et al., 2021; FUHRY; JAIN; KERSCHBAUM, 2021), that utilize this hardware-based secure computation approach.

For instance, consider the protection of sensitive medical data stored and processed in a cloud environment. A healthcare provider may utilize a cloud-based system to store patient records, including diagnostic results and treatment histories. By leveraging *SGX*, the healthcare provider can ensure that sensitive medical information is encrypted both during storage and processing. When a doctor queries the system for patient data, *SGX* enclaves ensure that only the authorized query results are decrypted and returned securely, preventing unauthorized access by cloud administrators or malicious actors. This approach not only complies with strict healthcare data protection regulations such as HIPAA but also mitigates the risks of data breaches and ensures patient privacy.

Problem

The problem we address is the urgent need to secure sensitive data in cloud computing environments. As cloud computing gains widespread adoption, data owners are confronted with the formidable challenge of safeguarding the privacy and confidentiality of their sensitive information. This issue is exacerbated by the fact that data and applications are often hosted on public infrastructure, making them vulnerable to unauthorized access by malicious actors. Additionally, the cloud environment itself can serve as a potential vector for security attacks, further jeopardizing the integrity of sensitive data.

Hypothesis

The central hypothesis posits that achieving robust security and privacy for sensitive data in cloud environments is feasible through a multi-faceted approach. This involves the incorporation of access control systems, hardware-level security measures, and specialized algorithms for data anonymization. By rigorously examining the prerequisites for data confidentiality, pinpointing potential security vulnerabilities, evaluating various protective mechanisms, and assessing their impact on data access and manipulation performance, we aim to establish a secure and safeguarded ecosystem for sensitive information.

Objectives

Thus, considering this hypothesis, the main objective of this doctoral thesis is to propose effective strategies and solutions for the protection of sensitive data in cloud computing environments. The study seeks to address the challenges and threats associated with data security and privacy in cloud environments, with a focus on ensuring authorized access and preserving the confidentiality of sensitive information. Specifically, we have the following objectives:

- Investigate systems and mechanisms for protecting sensitive data in cloud environ-

ments: This entails researching and evaluating various techniques, technologies, and best practices for ensuring the security and privacy of data, including access control mechanisms, encryption schemes, secure hardware technologies, and data anonymization approaches;

- Analyze the impact of implementing confidentiality requirements on the performance of data access and manipulation in the cloud: This involves examining the trade-offs between security measures and performance efficiency, considering factors such as processing speed, latency, scalability, and resource utilization.

Overall, by focusing on these objectives, we aim to contribute to the development of effective strategies and solutions for securing and protecting sensitive data in cloud computing environments, thereby enabling organizations to confidently utilize cloud services while maintaining the privacy and confidentiality of their sensitive information.

Vallum

In view of the objectives, we designed, implemented and experimentally evaluated a platform that protects sensitive data in data management platforms called *Vallum* [1]. The platform isolates sensitive data from unauthorized access and ensures privacy during storage and processing. It leverages Intel *SGX* technology to ensure compliance with privacy restrictions and prevent leaks of sensitive system information.

Vallum fully supports the protection of sensitive data, the definition of privacy policies, and the enforcement of policy compliance for query results. All processing, data storage, and network traffic involving sensitive data are protected in a fully encrypted environment.

To meet different performance and cost needs, we designed two different versions of the platform. The first version involves full execution in *SGX* enclaves, ensuring full protection for sensitive data. However, this may affect response time and increase cost. Thus, the second version involves vertically partitioning the database and protecting only sensitive data in the *SGX* enclave. Non-sensitive data is processed in maximum

[1] Vallum is a Latin term that corresponds to a type of wall used by the Romans in fortifications.

performance settings and without the limitation of *Enclave Page Cache* (*EPC*), improving response time and reducing execution cost.

These two versions allow data owners to choose the approach that best suits their security and performance needs. The first version ensures full protection for sensitive data but may have an impact on performance and cost. The second version aims to improve performance and reduce costs, focusing only on the protection of sensitive data.

Our proposed solutions have been successfully implemented and disseminated to the academic community through publications in conferences and journals. These include Cloudscape Brazil (GUIMARAES, 2019), IEEE International Conference on Cloud Computing Technology and Science (CloudCom) (GUIMARAES et al., 2019a), European Conference on Computer Systems (GUIMARAES et al., 2019b), Future Generation Computer Systems (BLANQUER et al., 2020), and ACM International Conference on Information & Knowledge Management (GUIMARAES et al., 2020).

Thesis Organization

The rest of this thesis is structured as follows: Chapter 2 presents the background related to privacy protection and secure data management using hardware and software solutions and a discussion of the main proposals in the related literature. In Chapter 3 we present Vallum, a platform designed to protect cloud data using a reliable execution environment and data anonymization techniques. It offers two versions of architectures: one fully protected using *SGX* and the other protecting only sensitive data using *SGX*. Chapter 5 presents a set of experimental results on Vallum's performance evaluation, using various reference results and evaluating Vallum's overhead in practice, and Chapter 6 offers conclusions and shows future directions of our work.

2

Background and Related Work

This chapter analyzes existing work in the literature related to the object of the research. We focus primarily on the approaches associated with the *Trusted Execution Environments* and on solutions that aim to preserve privacy and confidentiality using mechanisms of access control and data anonymization. It also presents work related to the importance of sensitive data protection and solutions of how to process that data safely.

2.1 Background

A solution for managing sensitive data in the cloud must prioritize mechanisms to ensure properties such as security and privacy. For this, methods such as user authorization with various levels of granularity (FERRAIOLO; KUHN, 2009), encryption (ARNAUTOV et al., 2016), privacy for anonymity (PRASSER et al., 2020), and integrity imposed by the hardware (COSTAN; DEVADAS, 2016) have been applied.

Protecting confidential data is an important requirement for data management systems, especially with the emergence of data protection laws such as the European Union's *General Data Protection Regulation* (*GDPR*). This is even more critical for cloud-based environments where the user has control over neither the physical resources nor the cloud infrastructure itself. In these settings, a cloud provider can be itself considered as a serious attack vector (JANSEN, 2011). An adversary with access to the cloud infrastructure can, for instance, use main memory sniping techniques to capture

sensitive information from the cloud tenant. Moreover, there is no guarantee that the software stack used in the cloud environment is up-to-date with the latest security resources, potentially exposing it to a wide range of external attacks (CHEN et al., 2010).

End-to-end encryption is one of the most common approaches for protecting data at rest. However, to perform data processing operations, the data must temporarily reside in the main memory in plain-text format. This enables an attacker with privileged access to the hardware, especially the main memory or software to gain access to classified information. Hence, processing sensitive information in plain text in such environments must be strictly avoided as it may leak confidential information.

An approach to solve this problem is using database management systems that process queries over homomorphically encrypted data (POPA et al., 2011; TU et al., 2013). Although this approach seems to solve the problem of temporarily storing data as plain-text information in the main memory, query processing over homomorphically encrypted data adds a significant time overhead.

Moreover, homomorphic encryption still allows a malicious user to apply inference attacks (NAVEED et al., 2015; AKIN; SUNAR, 2014), thereby providing alternative means to access sensitive information.

In addition to the previously mentioned data security aspects, it is equally important to guarantee that only authorized parties have access to a particular data item. Moreover, in more complex scenarios, the rules that define which user has access to what information may grow exponentially, especially when data authorization is not constrained to relation-level access. The problem is not easy to solve as, even in the presence of access rules, a non-malicious user may inadvertently expose private data by performing queries that do not take into account the privacy of the individuals depicted in the results, thus releasing sensitive information to the public.

In this context of data management in cloud environments, data protection consists of two requirements: data access and privacy protection. Data access ensures that no unauthorized party may access sensible data in any form, while privacy protection ensures that personally identifiable information is removed from query results performed over sensible data through proper anonymization techniques.

To enforce privacy, data anonymization is typically carried out through the relaxation or obfuscation of query results to prevent the identification of subjects through the interpretation of quasi-identifiers attributes in the data. However, even in this process of modifying query results to guarantee that they cannot be used to identify an individual etc., the operations are carried out in the main memory, leading to the previously described problem of privacy-preserving data processing.

To prevent a malicious user with privileged access from reading another process's memory, Intel introduced *Software Guard eXtensions* in their architectures. This enables application developers to create so-called *enclaves*, i.e., *Trusted Execution Environments* (*TEE*) where the main memory is encrypted and thus unreadable by any outside user or attacker (COSTAN; DEVADAS, 2016). The deployment of such enclaves is a solution to avoid memory reading attacks in shared environments such as public clouds.

2.1.1 *Trusted Execution Environments*

To meet security requirements, *Trusted Execution Environments* (*TEE*) provide protection to run trusted binaries in potentially hostile remote environments, ensuring security at the kernel and hardware level of the code running on its protected memory regions. In this sense, several hardware manufacturers have developed technologies with different architectures, but with the same objective of creating safe environments for data processing. Among these technologies, the most popular include Intel *Software Guard eXtensions* (*SGX*), ARM TrustZone Technology, and *AMD* Memory Encryption (NING et al., 2018).

Specifically in this work, we adopted Intel *SGX* as a technology to support creation in a safe environment to protect sensitive data in the cloud. Intel *Software Guard eXtensions* (*SGX*) is a set of *CPU* instructions developed by Intel with the objective of protecting, at the hardware level, the processing of an application in a confidential and secure way(COSTAN; DEVADAS, 2016). It aims to create a reliable execution environment called an *enclave*, by defining private regions in physical memory (*DRAM*), collectively called the *Enclave Page Cache* (*EPC*). It guarantees the confidentiality and

integrity of data on the *EPC*, since they are encrypted and the *CPU* allows only the code processing inside the enclave to have access to it (COSTAN; DEVADAS, 2016; ARNAUTOV et al., 2016). However, there is a physical limitation on the size of the *EPC* per machine, reaching currently a maximum of 128 MB and effectively around 90 MB due to the space for managing the enclave (TIAN et al., 2019). Applications that require memory above this limit experience a time overhead, since the paging time between *EPC* and *DRAM* has a significant latency caused by data movement and the encryption and decryption process (HARNIK et al., 2018).

Although Intel provides a secure platform and an *SDK* to create secure applications using *SGX*, direct use of this *SDK* is expensive, requiring users to spend time learning about its features and how to manipulate enclaves, as well as generating additional and specific codes for this end. Considering this and to minimize this cost of implementation, we added an intermediate layer to our solution to facilitate and optimize the use of *SGX* technology, called *SCONE*.

SCONE(ARNAUTOV et al., 2016) is a secure container mechanism created to use *SGX* transparently, allowing applications to run in secure Docker containers without the need to manipulate the Intel *SDK*. In this way, it is possible to run applications in a secure environment that protects confidentiality and integrity, in which data and codes are always encrypted, even from root users. This protection is also extended to data communication, through *TLS* (*Transport Layer Security*). In terms of performance, *SCONE* optimizes memory access, caching data from encrypted applications and network buffers, reducing costly access to the enclave.

2.1.2 *Role-Based Access Control*

In addition to ensuring a secure and tamper-proof environment to protect data from unauthorized access, even by attackers with physical access to the hardware and network, including users with root permission. It is also important to ensure that authorized users have limited access to the permissions defined in an authorization and access control policy. Therefore, it is important that a proposed system to protect data and

privacy must include in its resources mechanisms to manage access to data in a way that confidentiality property is guaranteed.

Role-Based Access Control (*RBAC*) is a form of access control to a system, whose main objective is to protect the confidentiality and integrity of information based on the roles that a user can assume in the context of an organization and not in his identity (LOPEZ; RUBIO, 2018). Each role is assigned a set of types of transactions that can be performed on the system and users must be assigned to one of the roles in order to be granted access and privileges. These types of transactions are defined as a transformation process that involves a certain functionality of the system, as well as a set of data involved in this process. In this way, it is possible to guarantee that authorized users can carry out only the types of transactions defined for the function to which they belong, restricting the execution of any procedures outside its scope. The possibility of mapping contexts and organizational structures to access the control and authorization policy makes *RBAC* a flexible mechanism compatible with the constant changes that occur in an organization (FERRAIOLO; KUHN, 2009).

2.1.3 Privacy

Also in this context of protection of confidential data, one of the main challenges to guarantee the privacy of data holders of a system. To that end, there are three main mechanisms that help to solve this problem (BRITO FELIPE T. ; MACHADO, 2017):

1. **Encryption** is established with the premise of preventing unauthorized people from having access to specific, usually private, information. This process begins with data encryption which changes its values mathematically, generating a new unreadable value, but which can be returned to the original content with the use of an access key (decryption). Despite being widely used in computing, cryptography has some limitations when applied to privacy protection, since real searches on unreadable data have almost no utility or require very high computational usage, so it is not widely used for this purpose;

2. **Tokenization** involves the random transformation of sensitive data content into

symbols so that unauthorized people can see only meaningless values from the original content. Data recovery occurs from a mapping structure between the source value and the tokenized value, that is, there is no mathematical process to extract the original data, as in the encryption techniques. This mechanism is mainly used to protect Identifying attributes in third-party environments, as it is almost impossible to decipher the original data without the mapping table;

3. **Anonymization** is the transformation of data to prevent association with the original holder of that data. This technique results in obfuscated data with content similar to its original value, but it is not possible to establish a direct or indirect relationship with a specific individual. Therefore, anonymity maintains a certain level of data semantics that allows the sharing and analysis of information without the risk of compromising the privacy of data owners, however, it is important to note that this approach presents a condition in which usefulness and data privacy is configurable and inversely proportional.

For this research, we opted for a privacy protection approach based on anonymization, as it is a technique widely used for this purpose and that provides mechanisms for data sharing, maintaining a configurable level of data usefulness and preservation of privacy. In this way, we adopted the ARX library to provide support for this objective.

ARX (PRASSER et al., 2020) is a software for data obfuscation that also offers its resources through a public *API*. It performs the obfuscation of private data in datasets according to several privacy models such as k-anonymity, ℓ-diversity, t-closeness, k-map, and differential privacy, among others. The attributes are obfuscated through domain generalization hierarchies that represent valid transformations of their values following certain levels, which can be configured by the user.

One of the main functionalities of ARX is its ability to calculate and evaluate measures of information loss and disclosure risk. It implements methods based on samples and methods based on populations to estimate the uniqueness of anonymized data. Some examples of measures implemented by ARX include the fraction of unique records in the sample, the average uniqueness of the sample, and k-anonymity. Additionally, ARX offers advanced features such as global and local recoding to control

the application of different anonymization models and measures of data utility and uniqueness (PRASSER et al., 2020).

Information loss is a crucial aspect of data privacy protection, as it measures the extent to which sensitive data is modified or perturbed during anonymization processes. It is important to minimize information loss while ensuring data privacy, as excessive perturbation can lead to inaccurate or invalid analysis results. Various measures have been developed to quantify information loss, including generic measures that analyze the impact of perturbation methods on the data (ESMEEL et al., 2020). However, for specific data uses, more fine-grained analysis is required, such as cluster-specific information loss measures (TORRA; LADRA, 2008). The balance between information loss and disclosure risk is a key consideration in data protection mechanisms, as the goal is to protect individuals' privacy while still allowing useful analysis of the data (TORRA; NAVARRO-ARRIBAS, 2014).

Furthermore, ARX provides additional features for privacy protection, such as syntactic privacy models (like ℓ-diversity and t-closeness), risk-based anonymization, differential privacy, and detection of HIPAA identifiers. It also supports the import of data from various sources, such as RDBMS databases, Excel spreadsheets, and CSV files. ARX is an open-source and cross-platform software library, which means it can be easily integrated into different environments and systems (PRASSER et al., 2020).

In summary, ARX is a powerful tool for addressing privacy and data anonymization issues. It offers a wide range of anonymization methods, as well as features for calculating measures of information loss and disclosure risk. ARX also provides additional resources for privacy protection and supports data import from various sources. It is a comprehensive and flexible solution for ensuring data privacy in different contexts and applications.

2.1.4 Global as View

Global as View (*GaV*) is a data integration approach that can be justified for use in a sensitive data protection architecture with vertical partitioning. This approach allows

for the integration of data from different sources into a single unified view, which can be beneficial in isolating sensitive data in a separate database. By using a declarative mapping language, *GaV* enables the specification of how each source relates to the unified view, allowing for the inclusion of only the necessary attributes and sensitive data in the isolated database. This approach can help enforce security policies and restrict data flow into a user's private universe, ensuring that sensitive data is protected. Furthermore, also supports schema refinements and iterative updates, making it suitable for dynamic environments (SELLAMI; HACID; GAMMOUDI, 2017; KATSIS; PAPAKONSTANTINOU, 2009; CALÌ et al., 2013).

Trino, formerly known as PrestoDB, is a distributed *SQL* query engine that allows for querying data from multiple sources in a unified manner. It can be presented as a solution to implement the *GaV* approach for vertically partitioning a database and treating sensitive data separately. By using Trino, one can define virtual views that represent the desired data partitions and apply security policies to restrict access to sensitive data. These virtual views can then be queried using standard SQL queries, providing a seamless and secure way to access the data. Trino's ability to handle large-scale data processing and its support for various data sources make it a suitable choice for implementing GaV with vertical partitioning (KARIMOV, 2020; GESSERT; WINGERATH; RITTER, 2020).

Query rewriting is a crucial technique in the context of *GaV* and Trino, enabling the integration of data sources and efficient query execution. It involves transforming queries into equivalent forms that can be executed using available data sources, addressing semantic heterogeneity. Ontologies and thesauri are used to identify connections between data schema sources, allowing for the rewriting of queries based on the selected schema and mappings between data sources and the global schema. Trino, as a distributed *SQL* query engine, plays a vital role in query rewriting within the *GaV* context, providing the necessary capabilities for query optimization and execution across multiple data sources. Its scalability and support for various data sources make it a suitable choice for query rewriting in *GaV* scenarios (BURON et al., 2020; CALVANESE; XIAO, 2018; CHORTARAS et al., 2016).

2.2 Related Work

There are several types of research that aim to create *Data Base Management Systems* (*DBMSs*) featuring resources for protection against leaks and unauthorized access, even when stored in the cloud. These solutions fall into two main categories (PRIEBE et al., 2018): systems that perform direct operations on encrypted data and systems that use reliable hardware for this purpose.

2.2.1 Secure *DBMSs* based on Homomorphic Encryption

A well-established solution to preserve *DBMS* confidentiality is the use of homomorphic encryption. This approach has the advantage of applying various operations directly to the encrypted data without having to decrypt it. However, this comes at a cost, as homomorphic encryption and decryption are expensive in terms of computation and therefore add considerable burdens to systems that rely heavily on this technique. Despite these problems, several works in this field have been published previously with the following results.

Systems like CryptDB (POPA et al., 2011), Monomi (TU et al., 2013), Arx (PODDAR et al., 2016), and SDB (WONG et al., 2014) base their respective solutions on the implementation of cryptographic schemes to perform operations directly on the encrypted data. This type of solution does not depend on specialized hardware, as data protection is provided at the algorithm level, but most have limitations related to the types of queries supported in the data set, in addition to the additional cost to process cryptographic operations.

CryptDB (POPA et al., 2011) works by intercepting all *SQL* queries to rewrite them and run under encrypted data. A proxy encrypts and decrypts all data, and changes operators, but without modifying query semantics, thereby ensuring that all private data is not in plain text in the *DBMS*. This proxy decrypts only data that the user involved in the operation has access authorization, based on the access control policy. For TPC-H, CryptDB can handle only four queries out of 22.

MONOMI (TU et al., 2013) is a system that performs analytical functions on

sensitive data in an unreliable *DBMS*. To do this, all data is encrypted and queries are performed on that protected data. The solution divides the workload between client and server, a scheduler module defines at runtime the best way to run a query which part can be run on an untrusted server, and which part can be executed by a safe client. It evaluates queries on encrypted data on the server, preventing the *DBMS* from accessing the keys for decryption. For efficiency purposes, non-sensitive data can be stored as plain text. The prototype had a performance of 1.24x, ranging from 1.03x to 2.33x compared to the same test being performed on an unencrypted database, but had some limitations to perform the benchmark because it was not possible performing three queries (13, 16, and 22) of the 22 proposals in TPC-H and for queries 17, 20, and 21 caused problems in the PostgreSQL optimizer, requiring that queries be modified to work around the problem.

Arx (PODDAR et al., 2016) proposes the implementation of strong cryptographic schemes, using almost exclusively the *AES*, and its queries are executed directly in the encrypted data. It presents two database indexes as data structures under *AES* and uses two proxies: a trusted client proxy that interacts with the application, encrypting the sensitive data and a server proxy that connects to the database server and converts the encrypted operations into queries for the *DBMS*. These proxies prevent changes to the application and database system use.

SDB (WONG et al., 2014) works by encrypting only the sensitive data using a secret-sharing method and the non-sensitive data is stored in plaintext. Uses an encryption scheme based on modular arithmetic so its operators are only applicable to integer values. It can support complex operations running on the server and performs all queries on the TPC-H benchmark.

FE-in-GaussDB (ZHU et al., 2021) combines software and hardware modes to support efficient query execution and *SQL* query processing in a complete scene, demonstrating its availability and effectiveness through a prototype and performance evaluation. The engine supports complete scene query processing, including matching, comparison, and other advanced computing functionalities, ensuring the security of operations on encrypted text data, maintaining query execution efficiency, and ensuring

data confidentiality. FE-in-GaussDB utilizes the Trusted Execution Environment (*TEE*) for secure operations on encrypted text data and uses an encryption driver on the client side to automatically encrypt data and send it to the server in encrypted text format, while the driver decrypts the query results returned from the database. Performance evaluation shows that FE-in-GaussDB introduces less than 5% overhead in operations on encrypted text columns while ensuring the security of operations on encrypted text data. However, the authors do not address the scalability of FE-in-GaussDB in terms of handling large data sets or high query loads, nor do they provide a comprehensive analysis of the trade-offs between security and performance in the presented solution. It is important to note that FE-in-GaussDB supports Intel *SGX* and ARM TrustZone but does not provide compatibility or integration with other database systems or frameworks.

HEDA (REN et al., 2022) focuses on the use of *Fully Homomorphic Encryption* (*FHE*) to support unlimited database aggregation queries, which involve filtering predicates and final aggregation, by leveraging the use of two types of *FHE* schemes, one for numerical values and one for binary values, and proposing a new encrypted text transformation mechanism to combine the encrypted values between these schemes. The HEDA system also introduces a new technique called secure aggregation, allowing multiple users to aggregate encrypted data without revealing their individual data to each other, which is important for applications where data is confidential and users do not trust each other. The authors implement the system and test it on only three TPC-H queries and one query on a real social media e-commerce database, and the evaluation results show the feasibility of using *FHE* to process *OLAP* queries. However, the proposed system is slower than plain-text processing and has room for improvement in terms of efficiency, as the performance on encrypted data is about 12 times slower than plain-text, and the storage requirements for the initialization key and the evaluation key are significant. The research focuses on unlimited aggregation queries and does not explore features such as GROUP BY, ORDER BY, or multiway joins, it also does not address the scalability of the system to larger databases and does not provide a comprehensive analysis of the performance impact of increasing the number of rows

or the complexity of the queries. Additionally, it does not establish a comparison with other encrypted or *FHE*-based database solutions in terms of performance, scalability, or security.

2.2.2 Secure *DBMSs* based on *TEEs*

While the previous solutions aim to secure data processing, trusted execution environments constitute more viable options for data protection in shared environments with hostile settings. Over the previous decade, exploratory work has been carried out to leverage the versatility of co-processors in contrast to the existing homomorphic encryption approach (BOUGANIM; PUCHERAL, 2002).

TrustedDB (BAJAJ; SION, 2011), Cipherbase (ARASU et al., 2013), StealthDB (GRIBOV; VINAYAGAMURTHY; GORBUNOV, 2017), EnclaveDB (PRIEBE et al., 2018), CryptSQLite (WANG et al., 2017), Enclage (SUN et al., 2021) and, EncDBDB (FUHRY; JAIN; KERSCHBAUM, 2021) perform data protection with specialized hardware support that provides *Trusted Execution Environments*. Data is deciphered only when run in these trusted environments, but this type of solution suffers from performance losses caused by the processing and storage limits that these architectures impose.

TrustedDB (BAJAJ; SION, 2011) is a prototype database system designed to protect sensitive data through reliable hardware and support for *SQL* operations. Uses the *IBM* 4764/5 with cryptographic coprocessors (*SCPU*), programmed to run components securely, but it does not limit the size of the supported database, using whenever possible unsafe resources of the servers to process public data. The attributes in the database are classified as: public or private, where private attributes are encrypted and can only be decrypted by the client or in the *SCPU*, this mechanism improves the overall performance of the solution.

Cipherbase (ARASU et al., 2013) is a *SQL* Server-based database system that uses reliable hardware and encrypted data operations to protect their confidentiality, especially in cloud environments. The system provides all the features that a standard *DBMS* has and fully supports *SQL* operations, combining its protection with resting

encryption, secure server, and fully homomorphic encryption, being implemented using *FGPA*s. Data owners can sort data according to the level of confidentiality so that the execution of that data is divided between client and server. On the client, an *ODBC* driver stores metadata and statistics, but the original metadata and statistics are stored encrypted on the server. On the server side, Cipherbase receives a client *ODBC* driver plan, interprets, applies changes, and performs the operation. The result is sent encrypted to client. When data is heavily encrypted, most processing of an operation is performed on the server using resources from an untrusted machine.

StealthDB (GRIBOV; VINAYAGAMURTHY; GORBUNOV, 2017) is an encrypted database management system that uses Intel *SGX* technology to ensure security, it is based on PostgreSQL and customized in C and C ++. StealthDB has a sophisticated architecture in which most of the *DBMS* runs outside of enclaves and only about 1500 lines of code implemented are effectively executed using the *SGX*, this prevents paging processes outside the *Enclave Page Cache* (*EPC*) that can raise the cost of processing from 3 at 1000 times. It provides a *DBMS* capable of processing large datasets and keeping them safe at rest and during query execution, integrating data encryption schemes and secure hardware. It also provides full support for *SQL* statements and mechanisms to authenticate users and check their authorizations. The evaluation was carried out between PostgreSQL in a standard installation and StealthDB with encrypted and unencrypted keys. And considering StealthDB latency, in the scenario in which all data is encrypted, including IDs, the median is 9.5 times greater than the result obtained with standard PostgreSQL.

EnclaveDb (PRIEBE et al., 2018) is a database engine that aims to ensure the protection of sensitive data using Intel *SGX* technology, it is based on Hekaton which is an in-memory database for *OLTP* workloads built into Microsoft *SQL* Server, modified and fully implanted in an enclave. Its architecture is divided into two parts: a *SQL* Server database server intended to store data that does not require protection, considered publics, and a *SGX* enclave that hosts the database engine, the purpose of which is to store sensitive data such as tables, indexes and metadata. The data in these columns is encrypted and integrity is maintained through memory encryption when it exits

the processor cache. The solution provides access control, in which only authorized users can perform operations on the database, as well as broader support for *SQL* query execution, but EnclaveDB does not support queries that simultaneously require data stored in the public environment and the safe environment.

CryptSQLite (WANG et al., 2017) uses Intel *SGX* architecture to ensure data reliability, it also uses symmetric encryption to protect data when it is at rest. It is based on SQLite, loaded into a *SGX* enclave to execute *SQL* operations securely. In tests performed under the TPC-H database, the proposed system is slower than an original SQLite by 2.4 to 15.36 times.

Enclage (SUN et al., 2021) is an encrypted storage mechanism for databases in trusted execution environments (*TEEs*) using Intel *SGX*. It incorporates native enclave designs such as page-level encryption, reduced enclave interaction, and hierarchical memory buffer, offering high-level security guarantees and high performance simultaneously. The authors present optimizations for encrypted database storage mechanisms in *TEEs*, such as deriving the optimal page size in the enclave and adopting delta decoding to access large data pages at a low cost, contributing to better utilization of limited enclave memory (*EPC*) and improving the overall performance of encrypted databases. Experimental results demonstrate that Enclage achieves a 13 times higher throughput and approximately 5 times storage savings. However, Enclage suffers from frequent enclave interaction, which negatively affects performance. Each individual comparison requires multiple enclave calls, resulting in significant performance overhead. The encryption schemes used in the proposed design also result in a high overhead of ciphertext in computation and storage. Furthermore, the decision to keep the internal states of the B-tree in plain text represents a serious risk of information leakage. Even an instantaneous adversary can learn the structure of the B-tree, compromising the security strength of the encryption scheme.

The EncDBDB (FUHRY; JAIN; KERSCHBAUM, 2021) is proposed as a high-performance encrypted cloud database that supports analytical queries on large datasets. It ensures data confidentiality through client-controlled encryption of column-oriented, in-memory databases and enables range searches using an enclave. EncDBDB utilizes

trusted execution environments, such as Intel *SGX*, to efficiently encrypt the data and offers nine encrypted dictionaries with different trade-offs in terms of security, performance, and storage efficiency. It allows users to choose the most suitable option for their needs. Range queries on datasets with millions of encrypted entries can be executed in milliseconds, with limited leakage and compacted encrypted data requiring less space than plaintext columns. However, the experiments do not provide a comprehensive security evaluation of the encrypted dictionaries used in EncDBDB, nor do they discuss its scalability in handling extremely large datasets or high query loads, which could be a potential limitation in a real-world scenario.

Compared to our results, encryption-based approaches such as CryptDB (POPA et al., 2011), Monomi (TU et al., 2013), Arx (PODDAR et al., 2016), and SDB (WONG et al., 2014) generally demonstrate superior performance. This advantage stems from the fact that these solutions are not constrained by the limitations inherent to systems relying on trusted hardware environments. However, they often fall short in adequately addressing privacy concerns and typically lack comprehensive support for *SQL* operations, which significantly restricts their flexibility in managing complex data processing tasks.

In contrast, solutions that use trusted hardware, such as TrustedDB (BAJAJ; SION, 2011) and StealthDB (GRIBOV; VINAYAGAMURTHY; GORBUNOV, 2017), are more aligned with our proposal, which is based on the use of *SGX*. These approaches ensure scalable support for large volumes of data, provide protection for data at rest, enable secure query processing, and implement authentication and authorization mechanisms for user credentials. Additionally, some of these solutions are capable of running all TPC-H benchmark queries without modification, a positive aspect in terms of usability and integration.

Our solution, based on the *SCONE* (ARNAUTOV et al., 2016) platform, stands out by enabling the continuous execution of services and applications in an encrypted form. It incorporates a more advanced authentication and authorization mechanism that controls permissions at the column level and enforces specific value restrictions, adapting user queries based on the policies defined by the data owner. While sharing common characteristics with the aforementioned solutions, our approach differentiates itself by

emphasizing privacy through the implementation of data anonymization techniques. Many of the solutions reviewed address privacy protection only through encryption, ignoring the threats that can arise in the absence of dedicated anonymization mechanisms. This gap is particularly relevant in critical contexts such as cloud environments, highlighting the need for a more comprehensive approach to data security.

Additionally, our solution offers a version that ensures full protection throughout the entire data processing flow, although it faces performance penalties when handling large volumes of information. We also provide an alternative version that minimizes the limits of *Enclave Page Cache* (*EPC*) by protecting only sensitive data, offering a more flexible and efficient option in situations that require a balance between security and performance.

3

VALLUM

In this chapter we present Vallum, a framework we designed, implemented, and evaluated as a first result of our research work. Its purpose is to protect data in cloud environments, ensuring its security and privacy. We developed two versions of this framework with different architectures, called Vallum 1 and Vallum 2. Vallum 1 was developed within the context of the *ATMOSPHERE* Project [1], and in this chapter, we will focus on detailing its features and functionalities. Vallum 2 will be discussed in Chapter 4.

3.1 Concepts and Assumptions

In our work, we assume a threat model that considers two different adversaries: user-level and infrastructure-level adversaries. Side-channel attacks are out of the scope of this work. This type of attack remains an obstacle for trusted execution environments. This includes, among others, side-channel attacks through page faults (SHINDE et al., 2016; XU et al., 2015), exceptions and interrupts (XU et al., 2015) and caching (BRASSER et al., 2017).

The adversary is assumed neither to be willing to disrupt the application's liveliness, i.e., disabling the system's network stack in the case of an infrastructure-level adversary nor to perform a denial-of-service attack in the case of a user-level adversary.

A user-level adversary is a remote user who maliciously uses credentials to gain

[1] More information available at: https://www.atmosphere-eubrazil.eu/

access to sensitive data. By default, this adversary has no access to the data unless it is granted as a result of the user's own submitted queries. This is an active and motivated adversary who can craft intentionally exploitative and disruptive queries, with high computational power at their disposal. The user is also capable of forging responses to requests from the server and, in general, controlling the client side of the application while having their credentials acknowledged by the system.

Although this adversary may have the highest role credentials issued by the system, valuable information would remain unavailable, thus keeping the adversary motivated to bypass data access restrictions. Therefore, these assumptions include the case of an adversary who is capable of escalating the validity of his credentials to the most powerful ones available.

An infrastructure-level adversary controls the entire software stack on the hosts where the data services and Vallum run, except for Vallum itself. Specifically, this means having control over the data engines and the operating system.

This adversary also has control over the complete software stack running on the physical host, such as the hypervisor or container engine in virtual systems, any orchestration engines utilized in the cloud, and the cloud platform itself, including storage volume resources.

The granularity of attacks is assumed to go down to bit-level manipulation of the main memory and disk states to arbitrarily large extents and all its consequences, including altering any data service-related files and data structures.

This adversary is assumed to be able to create and feed maliciously crafted packages to the enclave-protected application. The adversary has complete control over the network stack and software infrastructure and is capable of copying, dropping, replaying, and corrupting any package. However, they are not able to decrypt packages whose keys reside solely on the client side or within a process running in an enclave.

The infrastructure-level adversary is expected to have physical access to every host, being able to perform frozen-memory reads at any moment and to any location, including enclave areas. However, it is unexpected for the adversary to be able to use physical means to forcefully access data in processor registers or retrieve the *SGX* keys

for enclaves from the built-in chip extensions. The adversary being able to directly write to those registers to provoke a desired state is also not expected. Therefore, bugs were not considered in the implementation of *SGX* itself or software bugs in Vallum.

In general, it is assumed that this adversary is not capable of decrypting *TLS* connections as long as these connections are encrypted and decrypted inside enclaves or if one end is a remote client to whose machine the attacker has no access. This assumption is based on the idea that standard 2048-bit modulus *RSA* keys would be enough to discourage brute-force attacks.

3.2 Vallum Overview

Vallum is a layer for data access and protection to ensure that no adversary (malicious or not) can access sensitive data at any point in time. Vallum is designed to run as a proxy that completely isolates users from data engines that store and manage sensitive data. The platform ensures that data are securely processed in Intel *SGX* enclaves, guaranteeing confidentiality and integrity for data in the main memory during processing, while data at rest is encrypted and stored using a file system protection layer. In addition, Vallum also protects all internal and external communications through Vallum's network protection shield, which utilizes *TLS* connections to establish secure channels between the client, data engines, and all other components in a transparent manner. Vallum adds authorization primitives to the data management process, which are not typically provided out-of-the-box by most current data engines (e.g.,*DBMS*). Finally, to guarantee that all query results comply with the predefined access and privacy constraints set by the data owner, Vallum automatically obfuscates or removes sensitive parts from the result sets.

By implementing these features in two architecture versions for Vallum, we ensure its reliability through access control and privacy of sensitive data stored and processed in shared environments, such as public clouds, to which adversaries may have privileged access. Additionally, Vallum incorporates advanced authorization mechanisms, such as Role-Based Access Control (RBAC), to provide fine-grained control over

data access. Additionally, Vallum incorporates advanced authorization mechanisms, such as Role-Based Access Control (RBAC), to provide fine-grained control over data access.

The first architecture version that we designed, called Vallum 1, involves running completely securely using *SCONE* and Intel *SGX*, where all data are processed and handled in a secure and fully encrypted environment. However, this version is more expensive and more susceptible to the limitations of *EPC* in *SGX*.

The second version of Vallum's architecture that we designed, called Vallum 2, is constructed on the concept of vertical database partitioning and consists of isolating sensitive data from non-sensitive data, ensuring that only the sensitive data is processed within a secure environment using *SCONE*/Intel technology. In addition, the privacy guarantee process in the result is performed in this secure environment.

Vallum 2 offers a more cost-effective solution and is less prone to the limitations of *SGX* compared to the Vallum 1. By segregating confidential and non-sensitive data and processing them securely only when necessary, the system's performance is optimized. This is because the overhead of *SGX* is minimized, allowing Vallum 2 to provide a more efficient and practical solution for ensuring data security and privacy in cloud environments.

It is important to note that the choice between the two architecture versions of Vallum depends on the system requirements and available resources. Although Vallum 1 offers a higher level of security and the complete encryption of data, it also comes with a higher cost and may be subject to performance limitations imposed by *SGX*. On the other hand, Vallum 2 offers a more cost-effective solution that is less susceptible to *SGX* limitations while still ensuring the security and privacy of sensitive data.

3.2.1 RBAC and Sensitive Data Protection

Vallum utilizes Role-Based Access Control (RBAC) to enforce access restrictions and protect sensitive data. RBAC assigns permissions to roles based on predefined policies, ensuring users can only access data appropriate to their role.

For example, consider a healthcare database where Vallum manages access. Roles such as *Doctor*, *Nurse*, and *Administrator* are defined:

- **Doctor**: Has access to detailed patient records, including diagnoses and treatment plans, but cannot access administrative data such as billing information.
- **Nurse**: Can view basic patient information, such as names and prescribed medications, but is restricted from accessing sensitive diagnostic details.
- **Administrator**: Has access to billing and administrative records but is restricted from viewing patient medical details.

When a user queries the database, Vallum enforces these RBAC rules. For instance, a nurse querying patient data will receive obfuscated results for diagnostic information, ensuring compliance with access restrictions. This approach ensures sensitive data remains secure while allowing users to perform their tasks effectively within the confines of their roles.

3.3 Vallum 1

The architecture of Vallum 1 is illustrated in Figure 1. It shows the different modules where the previously mentioned features are encapsulated in the form of services such as *authentication*, *authorization*, and *privacy*. It is designed to support different *data engines*, ranging from *DBMS* and key-value stores to file systems.

3.3.1 Security

Since one of Vallum's main goals is to mitigate memory sniping attacks, it strongly relies on Intel *SGX*. Specifically, Vallum is based on *SCONE* (ARNAUTOV et al., 2016), a *Trusted Execution Environments* for running applications in Intel *SGX* enclaves without any additional implementation efforts. As shown in Figure 1, all the components of Vallum 1 are compiled using *SCONE*'s cross compiler to utilize the security features provided by Intel *SGX*, adding almost no implementation overhead for adaptation.

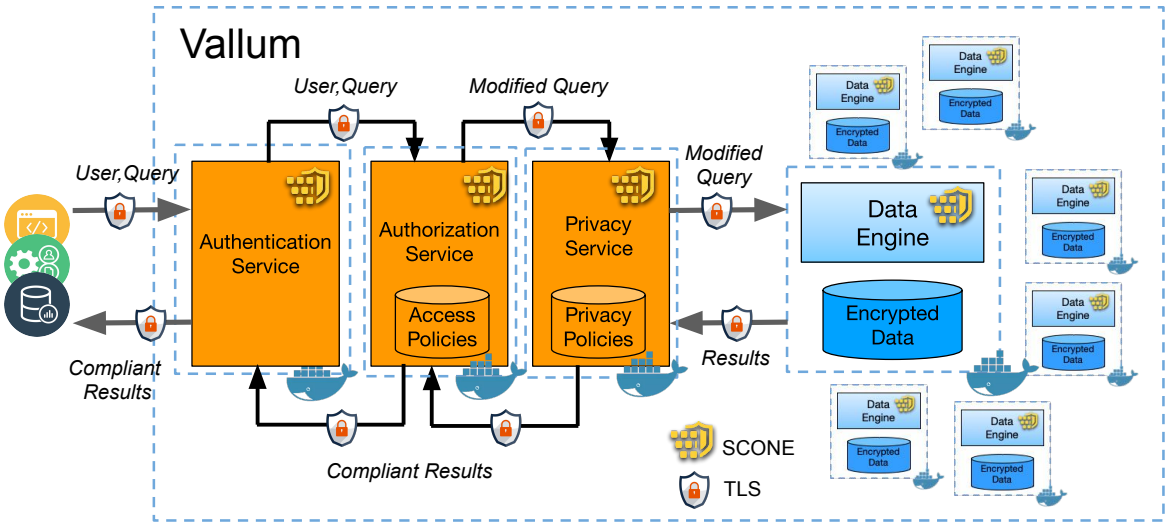


Figure 1 – Detailed architecture of Vallum 1, highlighting its components and their interactions for ensuring sensitive data protection in cloud environments.

This characteristic is essential to handle different types of data engines, including relational *DBMS*s such as MariaDB; non-relational systems such as MongoDB; key-value stores such as Redis; or even file systems such as *HDFS*. This approach allows us to use the security features provided by Intel *SGX* and easily integrate new data engines as needed.

As illustrated in Figure 1, secure communication in Vallum 1 is also enforced through *TLS* channels established between the different components as well as with the client to ensure that the data is encrypted at all times.

3.3.2 Authentication and Authorization

Authentication service of Vallum 1 is responsible for checking whether a specific user has valid credentials and, if so, loading a specific set of permissions for that particular user. This module also serves as an authentication instance for all the data engines. Therefore, all the engines can be accessed using the same credentials as those provisioned in Vallum 1. Users are authenticated through external attestation and key management mechanisms provided by *SCONE*.

The authorization service is responsible for checking and modifying queries to

ensure that they comply with the target data's access restrictions loaded during the authentication process. Vallum 1 was designed to grant or deny data access at different granularity levels, such as schema, table, column, record, file, etc. depending on the type of data engine. To achieve this, Vallum 1 first checks the credentials of the issuing user and parses their input query (in case of a *DBMS*) to verify whether it complies with the access schema restrictions defined by the data owner. If necessary, Vallum 1 modifies/rewrites the user query to comply with the constraints before sending a *modified query* to the target data engine or outright denying access to the data (see Figure 1).

In the case of relational databases, particularly, queries must not access tables and columns for which the user does not have the proper privileges. Thus, the authorization service module first parses the input query to verify if any unauthorized access attempt is being made. Next, to ensure tuple-level access control, the query is modified to comply with the constraints previously defined transparently. For instance, if a user queries a medical dataset and has access solely to patient data from a given hospital and patients over 18 years of age, the access control module adds these constraints directly to the query (for instance, where clauses in *SQL* queries) instead of simply rejecting it. This approach has the advantage of ensuring the user is not overly aware of the underlying access restrictions of the dataset.

3.3.3 Data Engine Support and Interaction

In the context of Vallum 1, a *data engine*, as illustrated in Figure 1, can be any system that performs the tasks of storing, managing, processing, and providing access to data, especially sensitive data. In principle, a data engine can be a *DBMS*, both relational or non-relational, a key-value store, or even a file system, among other options.

In Vallum 1, all data engines must be able to access data that are encrypted at rest. Although some modern data engines provide mechanisms for this, not all existing engines support it.

On the other hand, since keeping raw/plain data in the main memory represents

a possible attack vector, it is equally important to protect the data engine by using the same mechanisms as in Vallum; i.e., running data engines in *SGX* enclaves. Therefore, it is also proposed to run the data mechanisms in *SCONE* containers to take advantage of *SGX*'s memory protection while minimizing the adaptation costs for administrators. Note that it is possible to apply alternative approaches as long as they guarantee confidentiality.

Another measure to protect sensitive data stored in Vallum 1 is through the rejection of incoming connections other than from a trusted Vallum 1 instance. This mechanism prevents direct access attempts by malicious users, who may attempt to bypass Vallum 1 and connect to the data source directly, even with valid login credentials. To achieve this, external certificates are used, as well as key management mechanisms provided by *SCONE*.

3.3.4 Data Privacy

Another important component of Vallum 1 is the *privacy service*, which is responsible for preserving the data privacy constraints in the query results. This module performs two tasks: (1) verify whether a query result is compliant with all data privacy constraints, and (2) modify the query result to make it compliant.

Data privacy constraints can be defined in terms of k-anonymity (SWEENEY, 2002), ℓ-diversity (MACHANAVAJJHALA et al., 2007) and differential privacy (DWORK, 2011), among others. In the current implementation, Open source API ARX data anonymity was used (PODDAR et al., 2016), which not only supplies methods to verify whether a query result is compliant with many different privacy metrics but also provides algorithms to modify those results to adapt them to a given privacy level. Performance results were presented with the anonymity of k-anonimity in this work to show its impact on performance, but any other method, such as differential privacy, could be easily used.

The privacy service module also runs inside a *SGX* enclave, as the results that are checked and modified may be sensitive.

3.3.5 Extensibility

The architecture of Vallum 1 was conceived modularly to guarantee extensibility using plugins. For example, one of the intended future works, already under development, is an additional auditing module to generate logs and statistics about the received queries and their results. Other possible extensions are specific anonymization modules for other types of data such as raw text documents, images, and video streaming. Due to the architecture of Vallum 1, these modules can be easily integrated without any loss in trustworthiness, as long as the communication and the module itself utilize secure *TLS* channels and run in trusted execution environments such as Intel *SGX*.

The *SCONE Configuration and Attestation Service* (*CAS*) is utilized to verify the identity of the packets of the two modules and allow secure data exchange. This measure prevents adversaries from creating modules and impersonating modules to access modules directly, bypassing Vallum 1.

3.3.6 Monolithic vs. Microservices

Vallum 1 is designed to work with two possible architectural variations:

Microservices

Every module, that is, authentication, authorization, privacy, and data engines, runs as a single process on separate *SGX*-enabled nodes in the cloud using *SCONE* containers. This approach, in addition to being more flexible with the ease of adding new modules depending on application needs, has the advantage of minimizing the impact of paging, as each module can utilize the full $128\ MB$ *EPC* memory; however, this comes with the cost of needing extra communication between the modules.

Monolithic

All Vallum 1 modules, except for data engines, are part of the same process and run in a single *SCONE* container, communicating through function calls. This architecture has the advantage of reducing the overhead for inter-module communication as most calls are locally done through shared memory. However, due to the restrictions on the size of the *EPC*, this architecture may lead to paging and eventually slow down Vallum 1. Moreover, adding or replacing modules in this architecture requires a deeper understanding of Vallum's source code as well as recompilation.

Microservices vs Monolithic

In a previous study, presented in the paper *"Vallum: Sensitivity and Access Control for Sensitive Data in Cloud Environments"* during the *IEEE International Conference on Cloud Computing Technology and Science* (GUIMARAES et al., 2019a), held in Sydney, 2019, experiments were conducted to evaluate the performance of the monolithic and microservice-based versions of the Vallum system, including the privacy module. The results demonstrated that, regardless of the result set size, the microservice architecture achieved lower processing times compared to the monolithic approach.

The analysis of paging behavior conducted in that study identified the reasons for this performance difference. It was observed that the result set size significantly impacts the number of page faults in Vallum. For smaller sets, these faults were associated with general query processing. However, for 5MB result sets, most of the execution time was dominated by constant page faults occurring during the privacy module's processing.

In the monolithic version, a continuous stream of page faults was observed throughout the execution. In contrast, in the microservice architecture, the Authorization/Access module operated without page faults during privacy processing. Additionally, the privacy module in the distributed architecture experienced a shorter period of page faults compared to its monolithic counterpart. This difference is attributed to the greater availability of *EPC* memory in each module of the microservice architecture, which significantly reduced the number of page faults and improved overall

performance.

Although the paging behavior during the initial phase of database query initialization and preparation was similar across architectures, the microservice architecture demonstrated advantages by exhibiting fewer page faults in all scenarios analyzed. The most significant difference was observed in the final stage of processing, particularly for larger result sets, where the privacy module faced a high number of page faults due to the size of its working set exceeding the capacity of the *SGX* enclave's *EPC*.

The results highlighted the advantages of the microservice architecture in scenarios requiring intensive privacy processing. The distributed configuration enabled more efficient use of *EPC* memory, reducing the impact of page faults and optimizing performance.

3.4 Limitations of Vallum 1

The Architecture of Vallum 1 uses *SGX* to perform all processing in enclaves, which can significantly slow down the system as the result set grows. This is mainly due to the application of access and privacy policies, but the performance burden is also related to the use of *SGX* enclaves.

One of the primary reasons for this is the increased memory access latencies that occur when data is fetched from within the enclave. This happens because the data has to be encrypted and decrypted as it is transferred in and out of the enclave, which can significantly slow down the system. Another factor that can impact performance is the size of the enclave. The larger the enclave, the more memory it requires, which can lead to increased paging and thus slower performance.

The architecture of Vallum 1 can guarantee the security of data in the cloud and protect the privacy of a dataset according to the policies defined by the data owner. However, Vallum 1 can become increasingly slower in comparison to a standard *DBMS* configuration as the result set grows. Part of this burden is mainly related to the application of access and privacy policies, however, this architecture has a feature that further burdens its performance: performing all processing in *SGX* enclaves.

Despite ensuring a secure hardware-level environment, *SGX* imposes a reduced memory limit to process data with high performance, generating a large volume of paging between *EPC* and *DRAM*. This problem generates additional costs and increases Vallum 1's response time.

In response to the need to improve performance and resolve potential issues, we have implemented a second version of Vallum, called Vallum 2. This version is designed to minimize performance-related issues by making response time a more significant factor without compromising system security or data subject privacy.

To strike a balance between security and performance, we've identified that vertical fragmentation of the database is a promising strategy. This approach allows data processing to be segmented in a way that protection is confined to the treatment of sensitive data. As a result, only a portion of the processing will occur in an environment where performance is affected by the limitations of *SGX* enclaves.

In the next chapter, we will present a case study to provide a more concrete illustration of our solution for Vallum 2. This will be followed by a discussion of the architecture of Vallum 2.

4

VALLUM 2

In this chapter, we will continue to discuss Vallum, but we will focus on the details of Vallum 2's architecture. Vallum 2 supports transparent vertical fragmentation of a database. This version focuses on protecting the processing of sensitive attributes and the anonymization of the result. In other words, when Vallum 2 is processing sensitive data, this process is executed within the *SGX* environment. We begin by presenting a brief case study to motivate this new architecture.

4.1 Case Study

Consider a medical data recording system the data owner would define a schema that consists of an entity type *PATIENT* with attributes *SSN*, *NAME*, *SEX*, *BIRTHDAY*, *ZIP CODE* (*ZIPCODE*), and *PRE-EXISTING DISEASES* (*PRE_DISEASES*), as shown in Figure 1.

Not all attributes are considered sensitive and therefore do not require the same

PATIENT					
SSN	NAME	BIRTHDAY	SEX	ZIPCODE	DISEASES
111	Paul Smith	1990-05-21	M	123	Arthritis
222	Ana Backer	1995-10-02	F	456	Rheumatism
333	Elizabeth Apple	1990-07-23	F	789	Sinusitis
444	Peter Griffin	1990-12-09	M	321	Arthrosis
555	Lois Lane	2000-04-29	F	654	Scoliosis

Table 1 – Example of patient information.

SENSITIVE DATA			
VID	SSN	NAME	DISEASES
1	111	Paul Smith	Arthritis
2	222	Ana Backer	Rheumatism
3	333	Elizabeth Apple	Sinusitis
4	444	Peter Griffin	Arthrosis
5	555	Lois Lane	Scoliosis

NON-SENSITIVE DATA			
VID	BIRTHDAY	SEX	ZIPCODE
1	1990-05-21	M	123
2	1995-10-02	F	456
3	1990-07-23	F	789
4	1990-12-09	M	321
5	2000-04-29	F	654

Table 2 – Example of physical fragments.

level of protection. For instance, we could classify attributes such as *SSN*, *NAME* and *PRE-EXISTING DISEASES* (collectively referred to as *DISEASES*) as confidential, while *SEX*, *BIRTHDAY*, and *ZIP CODE* (collectively referred to as *ZIPCODE*) as non-confidential. This classification allows us to store the *PATIENT* entity data in two separate databases: one for sensitive data in a *Safe DBMS* (*SDBMS*), and another for non-sensitive data in a *Common DBMS* (*CDBMS*). To link these two bases of data, we need to create an attribute to identify a specific patient. This physical division can be visualized in Table 2 based on the classification of attributes.

The physical fragmentation of the database is not known to the data owner or users. They perceive the system as a single, unfragmented database. This means that client queries are generated based on this perceived general database, not the fragmented one. Therefore, this physical division is transparent to both administrators and clients.

Q	`select ssn, name, sex, birthday, zipcode, diseases from patient where sex = 'F'`
Q_1	`select vid, ssn, name, diseases from patient`
Q_2	`select vid, sex, birthday, zipcode from patient where sex = 'F'`

Figure 2 – Case study showcasing queries and their decompositions: Q represents the original query, while Q_1 and Q_2 demonstrate the partitioning of sensitive and non-sensitive data to enhance privacy during processing.

Let's consider a scenario where a client executes a query Q as shown in Figure 2. In Vallum's updated architecture, this query is split into two separate queries, Q_1 and Q_2, as depicted in the figure. Here, Q_1 is responsible for handling only sensitive attributes, while Q_2 manages the remaining attributes. Both Q_1 and Q_1 incorporate the attribute that enables their results to be joined. These queries are independently executed on each *DBMS*, depending on the level of protection required. By the end of this process, we will

have two distinct results: one containing sensitive data and the other with non-sensitive data. This process is illustrated in Figure 3.

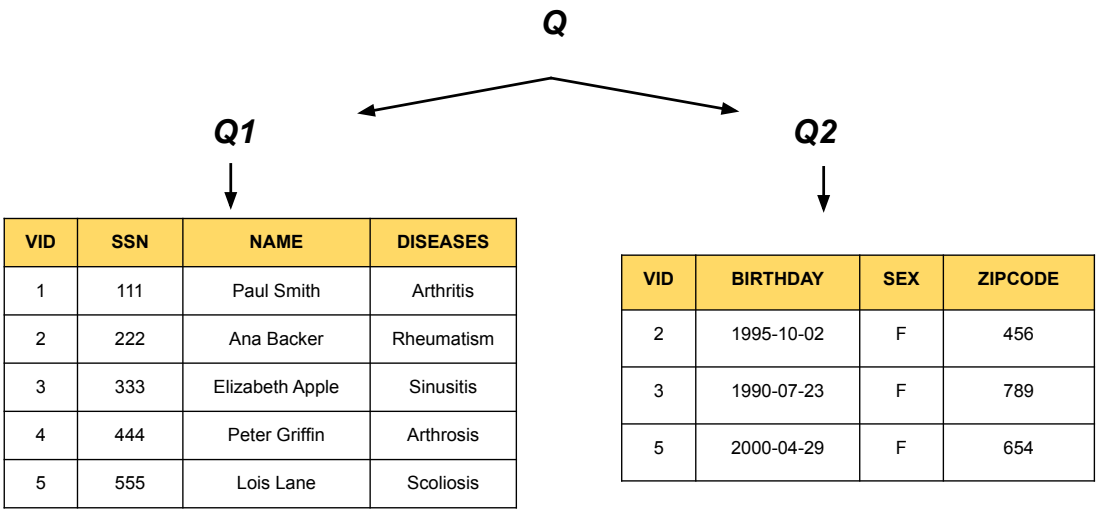
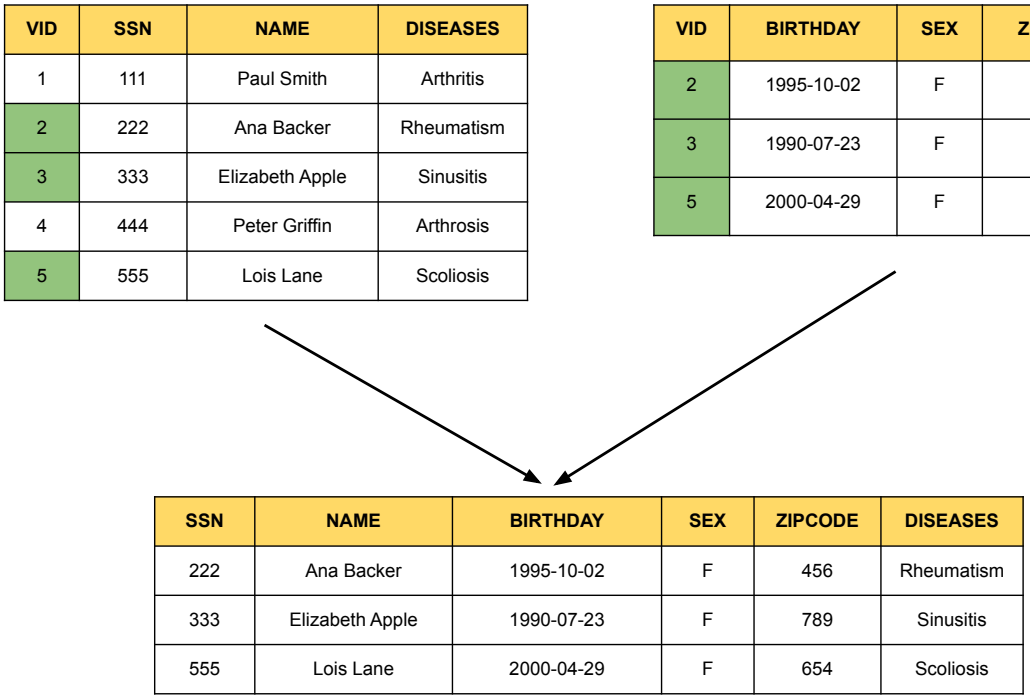


Figure 3 – Illustration of query division and corresponding results, showcasing the separation of sensitive and non-sensitive data during processing.

The results of each query must be properly combined based on the *vid* attribute, as depicted in Figure 4. It's important to note that since the joined result encompasses sensitive data, this process is conducted within a secure environment.

Now, consider that the data owner has defined a privacy policy for this dataset, such that attributes *SSN* and *NAME* are identifiers, *DISEASES* is sensitive, *BIRTHDAY*, and *ZIPCODE* are almost identifier and *SEX* is insensitive. In this case, the system must anonymize before releasing the result. In Figure 5 we present the anonymized result sent to the client, considering that we use the *k-anonymity* algorithm with parameter $k = 2$. Notice that, the anonymization process must also run in a safe environment.

Now, consider that the data owner has established a privacy policy for this dataset, designating the attributes *SSN* and *NAME* as identifiers, *DISEASES* as sensitive, *BIRTHDAY*, and *ZIPCODE* as quasi-identifiers, and *SEX* as insensitive. In this scenario, the platform performs an anonymization process before releasing the result. The anonymized result, sent to the client, is presented in Figure 5, taking into account the use of the *k-anonymity* algorithm with a parameter of $k = 2$. It's important to note that the anonymization process must also be conducted within a secure environment.



VID	SSN	NAME	DISEASES
1	111	Paul Smith	Arthritis
2	222	Ana Backer	Rheumatism
3	333	Elizabeth Apple	Sinusitis
4	444	Peter Griffin	Arthrosis
5	555	Lois Lane	Scoliosis

VID	BIRTHDAY	SEX	ZIPCODE
2	1995-10-02	F	456
3	1990-07-23	F	789
5	2000-04-29	F	654

SSN	NAME	BIRTHDAY	SEX	ZIPCODE	DISEASES
222	Ana Backer	1995-10-02	F	456	Rheumatism
333	Elizabeth Apple	1990-07-23	F	789	Sinusitis
555	Lois Lane	2000-04-29	F	654	Scoliosis

Figure 4 – Illustration of an inner join operation, demonstrating how data from multiple tables is merged based on common attributes to create a unified result set.

4.2 Vallum 2

Vallum 1 was designed to process the entire client request within a secure environment utilizing *SGX*, while also enabling the use of various types of *DBMS*s. If a client sends a

SSN	NAME	BIRTHDAY	SEX	ZIPCODE	DISEASES
222	Ana Backer	1995-10-02	F	456	Rheumatism
333	Elizabeth Apple	1990-07-23	F	789	Sinusitis
555	Lois Lane	2000-04-29	F	654	Scoliosis

SSN	NAME	BIRTHDAY	SEX	ZIPCODE	DISEASES
*	*	[1990-2000]	F	TEXAS	Rheumatism
*	*	[1990-2000]	F	TEXAS	Sinusitis
*	*	[1990-2000]	F	TEXAS	Scoliosis

Figure 5 – Illustration of anonymized results derived from the case study, showcasing the application of data anonymization techniques to protect sensitive information while preserving the utility of the dataset for analysis.

query, it is fully processed in *SCONE*-based containers and executed by a single *DBMS* also running inside a *SCONE* container.

Given the opportunity to protect only sensitive data and the fact that Vallum 1 was not designed to do so, we implemented an architecture that meets the requirements presented in the case study. In Figure 6, we present the architecture we implemented for Vallum 2. The focus is to answer requests for *SQL* queries with anonymized data, through an architecture based on microservices, in which each service is available in a different container. This strategy aims to achieve greater adherence to a cloud model given its scalability, as well as facilitating the split of the processing, mixing *SCONE/SGX* containers for services that handle sensitive data and containers with a standard configuration for cases that do not offer security and privacy risks.

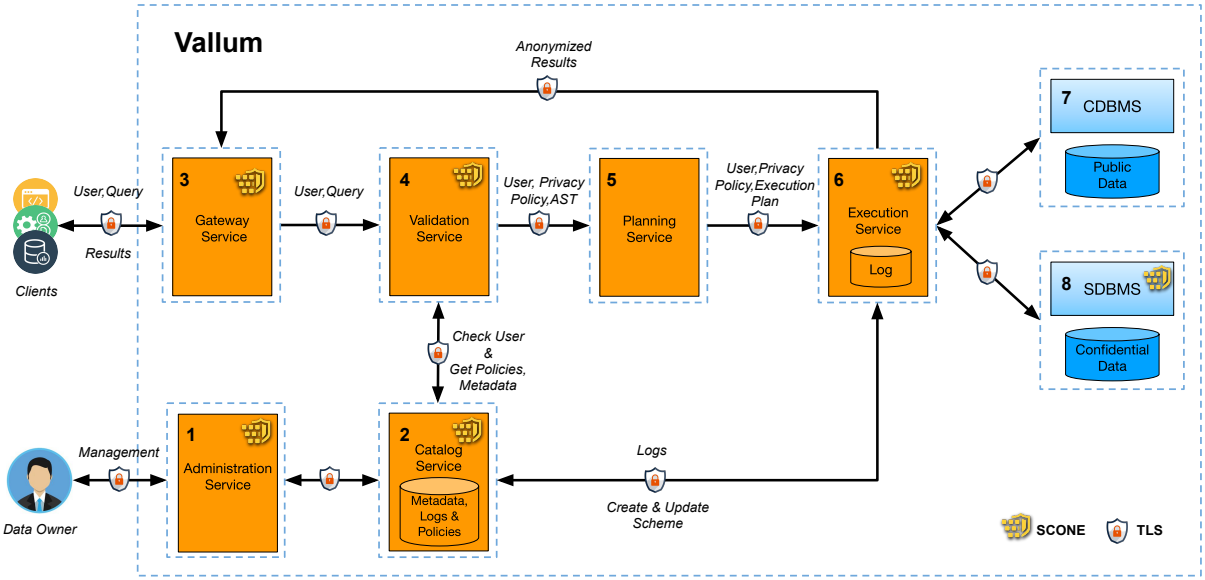


Figure 6 – Proposed architecture for the Vallum platform, highlighting its improved design to ensure enhanced protection of sensitive data while optimizing performance and scalability in cloud environments.

Next, we describe the architecture of Vallum 2.

4.2.1 Database Management

Through an administration module, represented in the architecture as *Item 1*, the data owner can create and update a database in a catalog maintained by Vallum 2. This

catalog (*Item 2*) includes entities, attributes, roles, and users who can consult the data, security policies, privacy policies, logs, and statistics of the operations carried out. These tasks are performed through a web interface, with the back-end and catalog running in secure containers with *SCONE/SGX*, ensuring that they cannot be used as a point of attack, primarily because they store and manipulate user credentials.

Another important function of *Catalog Service* (*Item 2*) is the management of the database schema and data. It's worth noting that the new architecture includes two active *DBMS*s working in an integrated manner: one running securely in the *SCONE* container to store sensitive data, presented in the architecture as *Item 8*, and the other in the container with a standard configuration to store data considered non-confidential as *Item 7* of the architecture. In this way, all creation, modification, and removal of structures and data are carried out by Vallum 2 based on the information stored in this module, which is also responsible for providing the database's physical fragmentation strategy.

The fragmentation occurs at the entity level, which means that each entity is represented in the two *DBMS*s, but only contains the attributes defined according to its sensitivity type.

For example, in the medical data recording system of the case study, the data owner would create an entity for *PATIENT* with the attributes *SSN*, *NAME*, *SEX*, *BIRTHDAY*, *ZIP CODE* (*ZIPCODE*), and *PRE-EXISTING DISEASES* (*PRE_DISEASES*). Then, he would classify *SSN*, *NAME*, and *PRE-EXISTING DISEASES* (*PRE_DISEASES*) as sensitive attributes and *SEX*, *BIRTHDAY*, and *ZIP CODE* (*ZIPCODE*) as non-confidential. This classification will be used as a reference for Vallum to create an *ID* (*VID*) to identify each row, perform physical fragmentation and store sensitive data in the *SDBMS* (*Item 8*) with *SGX* and non-sensitive data in the *CDBMS* (*Item 7*).

The physical fragmentation of the database is unknown to the data owner, who for the example mentioned would use the Administration Service's Web interface to configure a single database for this medical system, and client queries are also produced considering only that general database, not fragmented, that is, this physical division is transparent to administrators and clients.

To define data access permissions, Vallum 2 maintains *Role-Based Access Control* mechanisms to ensure that users have access only to the data and operations configured for their role. This allows administrators to configure access to a role at the entity, attribute, and row level, as well as which operations (*INSERT, DELETE, UPDATE, and SELECT*) can be performed by that role on the entities and attributes that are authorized. That is, a given role can have access to only a few attributes in an entity, in addition to displaying a limited set of rows. To do this, we establish these permissions as follows: (1) Associate the entities to the role and define which operations can be performed; (2) Associate the function with the allowed attributes and what operations can be performed and (3) Define data permissions for the role, limiting the available lines and allowing the transformation of an *SQL* query, adding conditions to it.

Even after authorization, a result can improperly expose confidential information. Therefore, Vallum 2 configures criteria to ensure the protection of privacy. These policies are applied to each attribute of the entities, categorizing them as (a) Identification attributes with a high risk of re-identification and, therefore, are omitted in the query results; (b) Quasi-identifier attributes that can be combined for re-identification; (c) Sensitive attributes that establish properties to which the data subjects are associated and the improper disclosure of that data can cause damage; and (d) Insensitive attributes that pose no risk to privacy and can be maintained without modification. This classification of the level of privacy of the attributes is carried out in a personalized way for each role, allowing greater flexibility in the control of privacy according to the level configured.

Therefore, a given data set can present results with different patterns of anonymization and these patterns are defined by the data owner for each sensitive or quasi-identifying attribute, through domain generalization hierarchies (PRASSER et al., 2020) that define valid values at each transformation level of the attribute.

4.2.2 Authentication and Validation

After the data owner has established the access control, and privacy policies, and instantiated the database in Vallum 2, users associated with this database can issue queries through a secure connection (*TLS*) with Vallum 2. This connection involves sending the login, password, and the *SQL* query. This request is initially processed by the *Gateway Service*, which corresponds to *Item 3* of the Vallum 2 architecture, but the entire process is performed in a microservice architecture with different modules distributed in several containers.

Gateway Service module acts as a single point of access to the entire architecture, being protected by *SGX*, since it is responsible for handling all communication with the clients, receiving and responding to all requests involving sensitive data. This means that the client will only know the *Gateway Service* address, without communication with other Vallum 2 Modules.

Upon receiving the client's request, the *Gateway Service* forwards it to the *Validation Service* module (illustrated as *Item 4*). This allows the client to be authenticated and its query processed by Vallum 2. To perform this authentication, the data with the client's credentials are sent to the *Catalog Service*. If they are correct, this module sends the metadata of entities and attributes, the user's role, and the access control and privacy policies associated with that role to the *Validation Service*. If the credentials are incorrect, it sends the error message of the operation to be forwarded to the client by the *Gateway Service*.

If the user has valid credentials, Vallum 2 transforms the query text into an *Abstract Syntax Tree* (*AST*) as illustrated in Figure 7, checks the syntax of the *SQL* query, and validates its content with the metadata provided by the *Catalog Service*. Vallum 2 also checks if the user is allowed to access the requested attributes and entities and if he has any data restrictions defined by the data owner.

For example, in our case study, if a given role can only access the {*MANAUS*, *BELÉM*} values of the *CITY* attribute of the *PATIENT* entity, Vallum 2 will add conditions in the *AST* that meet this restriction and each attribute of the query will receive the type of confidentiality (confidential or non-confidential) and the classification regarding

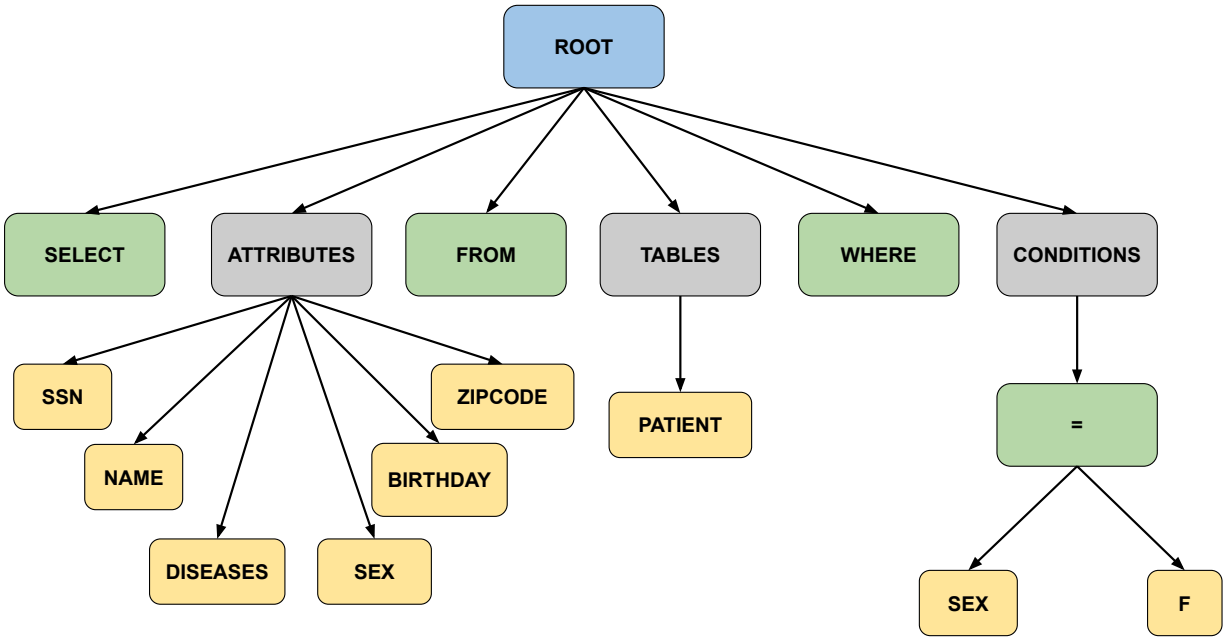


Figure 7 – Illustration of an *AST* generated from the case study, providing a detailed representation of the hierarchical structure and relationships within the analyzed data.

policy privacy (identifier, quasi-identifier, sensitive and insensitive) as shown in Figure 8. After this step, the request is sent to the Planning Service module with the query and the user.

4.2.3 Execution Planning

Vallum 2's architecture introduces a *Planning Service* (*Item 5*), designed with *Global as View* (*GaV*) concepts in mind. This module is responsible for receiving a *AST* from a global schema, which is partitioned vertically into two databases: one for sensitive data (*Item 8*) and the other for non-sensitive data (*Item 7*). This vertical partitioning necessitates the generation of a set of smaller sub-queries to fetch the data requested by the global schema query from the fragmented schemas. To perform this operation, the module employs the Trino library, which, based on a global schema, rewrites the query to distribute its execution between the two databases.

Contrary to other parts of Vallum 2, this module is not deployed in a protected *SCONE* container. This design decision is made to prioritize performance and reduce

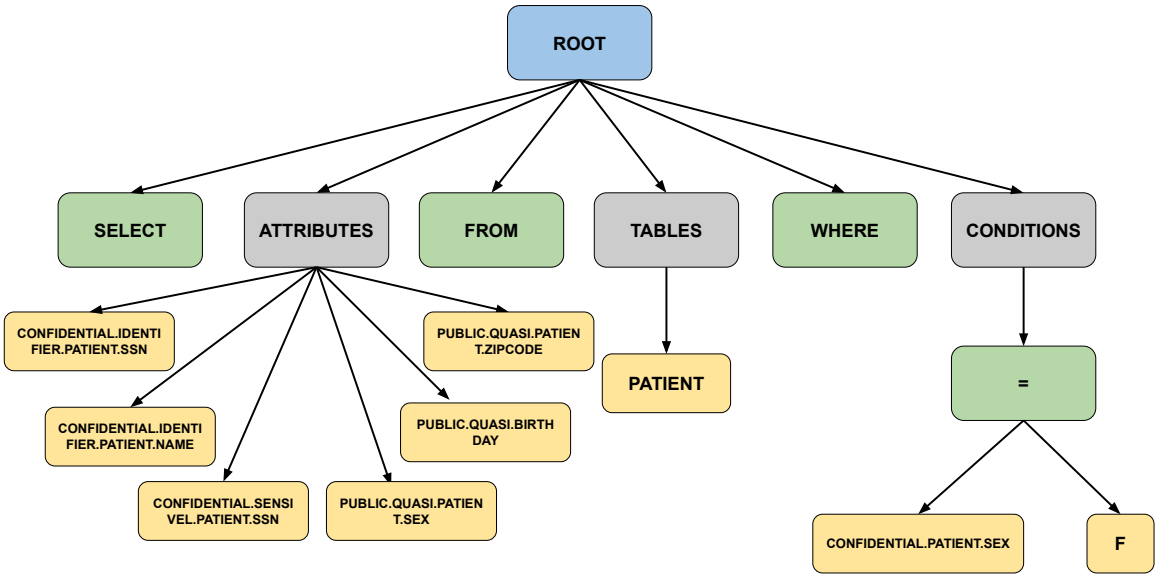


Figure 8 – Representation of the modified *AST*, illustrating the structural changes and enhancements made to optimize data parsing and analysis efficiency.

response time. As a result, this module is not subject to the limitations of *SGX*, and the scheme is potentially vulnerable. However, this impact is mitigated with a high level of protection for data classified as confidential.

The Trino library is responsible for generating these sub-queries and defining their execution plan. Once generated, they are sent to the *Execution Service* (*Item 6*), which is entirely designed by *SCONE*/*SGX*.

4.2.4 Query Execution and Privacy

The *Execution Service* module (*Item 6*) is responsible for interacting with *DBMS*s. It receives the execution plan and sends sub-queries to each *DBMS* according to its confidentiality level. In this way, queries involving sensitive data are directed to a *SDBMS* (*Item 8*) running in a container with *SCONE*, and queries involving non-sensitive data are sent to a *CDBMS* (*Item 7*) running with the default configuration. As we have previously mentioned, the goal is to minimize data processing using the *SGX* environment, thereby improving Vallum 2's response time.

After executing the queries in *DBMS*s and receiving the results, in this same

module, Vallum 2 performs a merge operation between these results using the *VID* attribute, as shown in Figure 4, Section refsec:case. Furthermore, the platform applies privacy policies to the final result, anonymizing the data and sending the anonymized result to the customer. As this module deals with sensitive data and is also responsible for applying privacy policies to the query result, it is necessary to run it in a container with *SCONE*. However, results that do not involve sensitive data will be processed in *DRAM* to reduce overhead on *EPC* and improve Vallum 2.

The ARX library ensures privacy by supporting the main privacy models. These models include k-Anonymit, k-Map, ℓ-Diversity, and Differential privacy.

4.3 Vallum 1 vs. Vallum 2

In this section, we describe the main differences between Vallum 1 and Vallum 2. For the sake of objectiveness, we present this comparison in Table 3.

Feature	Vallum 1	Vallum 2
Data Processing	Fully in *SGX*	Only sensitive data in *SGX*
Data Protection at Rest	Fully	Only on sensitive data
Data Protection in Transit	Yes	Yes
Database Fragmentation	No	Yes
Security Features	Yes	Yes
Privacy Features	Yes	Yes
Architecture	Monolithic/Microservices	Microservices

Table 3 – Summary of the main differences between Vallum 1 and Vallum 2

The architecture of Vallum 1 and Vallum 2 is designed with the same goal in mind: protecting data in cloud environments. However, they approach the problem differently, with Vallum 2 addressing some of the limitations identified in Vallum 1.

The key differences between Vallum 1 and Vallum 2 lie in their approach to data processing, database fragmentation, performance, privacy features, and architecture. In terms of data processing, Vallum 1 operates by processing all data within a secure *SGX* environment, regardless of the sensitivity of the data. On the other hand, Vallum 2 only processes sensitive data within the *SGX* environment, allowing non-sensitive data to be processed in a standard environment.

Regarding database fragmentation, Vallum 2 introduces a feature of transparent vertical fragmentation, separating sensitive and non-sensitive data into different databases. This feature, however, is not present in Vallum 1.

In summary, Vallum 2 is an evolution of Vallum 1, improving on the areas of performance, data processing efficiency, and privacy handling, while maintaining the core objective of secure data management in cloud environments.

Besides the qualitative comparison above, in Chapter 5 we present a comprehensive experimental comparison between the two architectures.

5

EVALUATION

This chapter describes the experiments conducted to evaluate the performance of different data protection approaches in database environments, comparing the MariaDB (baseline), Vallum 1, and Vallum 2 platforms. The analyzed scenarios involve the execution of TPC-H benchmark queries and the application of the k-anonymity algorithm, focusing on response time analysis and the efficiency of each solution.

5.1 Setup

For the experiments conducted in this research, we used five identical machines, all running Ubuntu 22. The hardware specifications of each machine were as follows:

- Processor: Intel® Core™ i7-7700 @ 3.60GHz (8 cores)
- RAM: 16 GB
- Storage: SSD
- Operating System: Ubuntu 22

Each machine was configured to perform a specific role within the experimental infrastructure, as described below.

The first machine was configured as the client, responsible for sending queries to the system and measuring the response times of the different tested solutions. This

machine simulated the end-user workload, making requests to the database and authentication/authorization services while collecting the necessary metrics for performance analysis.

The second machine hosted the Vallum 1 and Vallum 2 services, which control authentication and authorization. These services were executed within Docker containers, using Docker version `24.0.5` and configured to run in *SCONE*'s Simulation mode. Simulation mode was chosen to ensure greater stability, as the Hardware mode of *SCONE* exhibited instability issues during initial tests, which could compromise the reliability of the experiments. Therefore, the Simulation mode was adopted as a more stable alternative for running the Vallum services, ensuring more reliable performance for the analysis.

The third machine was dedicated to processing data with the k-anonymity algorithm. This machine, in the case of Vallum 2, was also responsible for planning and executing queries, considering the vertical partitioning strategy for the data. In the Vallum 2 model, the data was split between a database containing sensitive data and another with non-sensitive data, both processed within *SCONE* containers in Simulation mode. Vertical partitioning was used as a privacy protection strategy to ensure that sensitive data was handled separately from public data, minimizing the risk of improper exposure.

The fourth machine was dedicated to running MariaDB 10.2 with its default configuration, serving as the baseline for the experiments. This configuration provided a comparison point to assess the impact of privacy protection solutions, such as Vallum 1 and Vallum 2, on the database's performance.

Finally, the fifth machine was used to run MariaDB 10.2 within a *SCONE* container in Hardware mode, providing a more realistic scenario. The use of hardware mode was exclusive to the database, as it offers an additional layer of security by utilizing hardware enclaves, ensuring a safer execution environment that is closer to a real production environment. For the other solutions, such as Vallum 1 and Vallum 2, Simulation mode was chosen due to the instability of Hardware mode, as mentioned earlier.

Docker (v.24.0.5) was used to deploy all services in containers, including authentication, authorization, privacy protection, and database services. The containers were customized to work with *SCONE*, ensuring that all services ran within environments protected by security enclaves, as per the architecture proposed for protecting sensitive data.

Thus, the experimental setup consisted of a combination of different hardware configurations, *SCONE* operation modes (Simulation and Hardware), and task division among the machines according to the responsibilities of each service (authentication, privacy, and database), allowing for a detailed evaluation of the performance of the proposed solutions.

5.2 Experimental Scenarios

5.2.1 MariaDB (Baseline)

MariaDB was used as the reference system, representing the standard performance of a database without any data protection techniques. This configuration serves as the baseline for comparison with implementations that use *SGX* for data protection.

The choice of MariaDB as the baseline is justified by the fact that Vallum, in both versions 1 and 2, relies on a secure MariaDB database provided by SCONE for its operations. Using the standard MariaDB configuration as the baseline allows for a direct and relevant comparison between the unprotected database system and the secure implementations of Vallum. This ensures that the performance differences observed can be directly attributed to the additional security mechanisms applied in the SCONE-secured MariaDB environments. The baseline, therefore, serves as a meaningful point of reference to evaluate the impact of the data protection techniques employed in Vallum.

5.2.2 Vallum 1

The Vallum 1 scenario involves the complete protection of the database through *SGX* (Intel Software Guard Extensions), ensuring robust data security. However, due to the

limitations of *SGX*'s *Enclave Page Cache* (*EPC*) and the need for encrypted paging, Vallum 1 suffers from performance penalties, particularly with more complex queries.

5.2.3 Vallum 2

Vallum 2 adopts a hybrid approach, where the database is vertically partitioned into two parts: one protected by *SGX* and the other unprotected. Sensitive data is isolated and processed within the secure enclave, while non-sensitive data is managed by MariaDB. This allows for faster query execution and avoids the security overhead associated with protecting non-sensitive data.

5.3 Benchmark for Performance Evaluation

To measure the general overhead of the two Vallum versions, the well-established TPC-H benchmark (POESS; FLOYD, 2000) was used. This benchmark consists of 22 different queries, aimed at covering a variety of typical query processing loads. No privacy constraints were applied to the TPC-H queries to ensure that the results of this experiment remain on a scale comparable to other studies that use the same benchmark, making it easier to compare performance. The TPC-H benchmark was configured with a scale factor of 1 to create the dataset, which resulted in a total size of $1GB$.

5.3.1 Results for TPC-H Queries

The results for each query are summarized in Table 4, showing the execution times (in seconds) for MariaDB, Vallum 1, and Vallum 2. As expected, Vallum 1 and Vallum 2 exhibit higher query execution times compared to MariaDB, as both versions incorporate additional security mechanisms based on Intel *SGX*, which introduces overhead.

The results show the performance impact of the privacy-preserving features implemented in Vallum 1 and Vallum 2. For many queries, Vallum 1 shows a significant overhead compared to MariaDB, especially in more complex queries such as Q14 and Q9,

Query	**MariaDB (s)**	**Vallum 1 (s)**	**Vallum 2 (s)**
Q1	13.18	43.11	46.74
Q2	0.90	23.95	9.12
Q3	8.77	60.96	42.99
Q4	0.66	9.07	45.03
Q5	1.67	43.84	40.13
Q6	2.35	15.13	19.65
Q7	1.12	19.68	44.76
Q8	2.71	81.42	42.39
Q9	11.44	217.10	50.57
Q10	6.12	189.48	22.04
Q11	0.23	6.46	8.25
Q12	6.32	32.40	30.88
Q13	5.34	158.84	12.76
Q14	22.49	828.12	33.07
Q15	0.05	0.07	0.10
Q16	0.39	4.44	4.65
Q17	0.10	1.36	31.63
Q18	7.43	174.52	55.26
Q19	0.16	2.79	38.05
Q20	0.35	10.06	34.91
Q21	11.60	43.26	95.05
Q22	0.22	1.30	7.06

Table 4 – Execution times for TPC-H queries (rounded to two decimal places).

which likely result from the added security layers. Vallum 2, by contrast, demonstrates comparatively better performance due to the hybrid approach, which separates sensitive and non-sensitive data, enabling more efficient query execution for non-sensitive data.

These results suggest that while Vallum 1 and Vallum 2 provide enhanced data security through Intel *SGX*, the performance trade-offs must be carefully considered depending on the query complexity and the sensitivity of the data being processed.

5.3.2 Average Response Times

The analysis of the average response times for all TPC-H queries is presented below. The average values were calculated considering the response times of all 22 queries in each system (Figure 9).

- MariaDB: 7.25592 seconds
- Vallum 1: 61.98894 seconds
- Vallum 2: 36.75716 seconds

As expected, MariaDB showed the best performance, with an average response time significantly lower than the two *SGX*-based systems, Vallum 1 and Vallum 2. This can be attributed to the overhead imposed by the data protection technologies used in the two *SGX* platforms, particularly in the form of encryption and context switching associated with full protection (Vallum 1) and selective protection (Vallum 2).

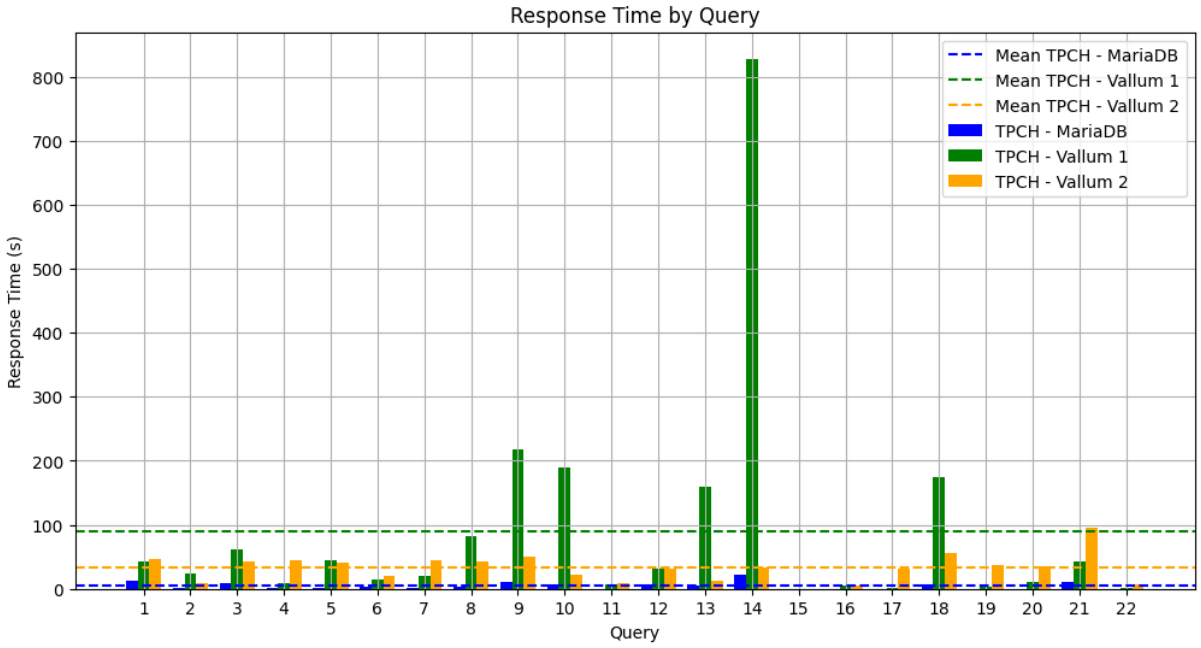


Figure 9 – Comparison of response times for TPC-H benchmark queries across different systems. The graph highlights the performance variations in seconds, emphasizing the impact of privacy-preserving mechanisms on query execution times.

These results indicate that Vallum 1 is approximately 8.55 times slower than MariaDB (61.98894 vs 7.25592), while Vallum 2 is approximately 5.06 times slower than

MariaDB (36.75716 vs 7.25592). However, Vallum 2 proved to be 40.71% faster than Vallum 1, highlighting the advantage of its selective protection approach.

5.3.3 Compatibility with TPC-H Queries

Both Vallum 1 and Vallum 2 were able to process all 22 TPC-H queries without failures, demonstrating the reliability and robustness of both systems in executing complex SQL operations. The implementation of data protection, whether full or selective, did not compromise the integrity of the query results, which is a positive indication for the application of security techniques in database management systems in high-performance environments.

5.3.4 TPC-H Analysis

A detailed analysis of the results reveals that the performance of Vallum 1 and Vallum 2 varies depending on the complexity of the queries, the volume of data processed, and the proportion of sensitive data involved. While Vallum 1, with its full protection, is more efficient in queries that heavily depend on sensitive data and require intensive processing within the enclave, Vallum 2 demonstrates better performance in queries that predominantly process non-sensitive data or perform complex operations on large data volumes outside the enclave.

Query Q14 is an example where Vallum 2 outperforms Vallum 1 due to the predominantly non-sensitive nature of the data being processed. This query calculates promotional revenue as a percentage of total revenue and involves aggregation operations applied to the `lineitem` and `part` tables. Below is the SQL representation of Query Q14:

```
SELECT
    100.00 * SUM(
        CASE
            WHEN p_type LIKE 'PROMO%' THEN
```

```
            l_extendedprice * (1 - l_discount)
            ELSE 0
        END
    ) / SUM(l_extendedprice * (1 - l_discount)) AS promo_revenue
FROM
    lineitem,
    part
WHERE
    l_partkey = p_partkey
    AND l_shipdate >= DATE '1994-09-01'
    AND l_shipdate < DATE '1994-09-01' + INTERVAL '1' MONTH;
```

In Vallum 1, all operations are performed within the *SGX* enclave, including joins and conditional aggregations. This requires large volumes of data to be transferred to the protected memory, resulting in overhead caused by *EPC* limitations and constant encryption/decryption operations. In contrast, in Vallum 2, only sensitive data is processed within the enclave, while non-sensitive data, such as prices and discounts, is handled outside the enclave. This hybrid approach significantly reduces the volume of data processed in a protected environment, allowing Vallum 2 to achieve superior performance in this query.

In contrast, Query Q21 illustrates a case where Vallum 1 has an advantage over Vallum 2, despite its higher complexity. This query involves multiple joins across the `supplier`, `lineitem`, `orders`, and `nation` tables, as well as conditional subqueries dependent on sensitive data. Below is the SQL representation of Query Q21:

```
SELECT
    s_name,
    COUNT(*) AS numwait
FROM
    supplier,
    lineitem l1,
    orders,
```

```
    nation
WHERE
    s_suppkey = l1.l_suppkey
    AND o_orderkey = l1.l_orderkey
    AND o_orderstatus = 'F'
    AND l1.l_receiptdate > l1.l_commitdate
    AND EXISTS (
        SELECT *
        FROM lineitem l2
        WHERE
            l2.l_orderkey = l1.l_orderkey
            AND l2.l_suppkey <> l1.l_suppkey
    )
    AND NOT EXISTS (
        SELECT *
        FROM lineitem l3
        WHERE
            l3.l_orderkey = l1.l_orderkey
            AND l3.l_suppkey <> l1.l_suppkey
            AND l3.l_receiptdate > l3.l_commitdate
    )
    AND s_nationkey = n_nationkey
    AND n_name = 'INDIA'
GROUP BY
    s_name
ORDER BY
    numwait DESC,
    s_name
LIMIT 100;
```

In Vallum 1, the entire query is processed within the enclave, avoiding the

need to synchronize data between protected and unprotected environments. This simplifies the execution flow and improves efficiency, particularly for subqueries with `EXISTS` and `NOT EXISTS` clauses, which require access to and comparison of large sensitive datasets. In Vallum 2, the separation between sensitive and non-sensitive data necessitates synchronization between the two environments, introducing additional overhead. Furthermore, many conditions in Query Q21, such as `l_receiptdate > l_commitdate`, directly involve sensitive data, limiting the benefits of the hybrid approach.

Additionally, when comparing the average response times for all 22 queries, the following can be observed:

- Vallum 1 is approximately 8.55 times slower than MariaDB (61.98 vs 7.25).
- Vallum 2 is approximately 5.06 times slower than MariaDB (36.75 vs 7.25).
- Vallum 2 is 40.71% faster than Vallum 1 in terms of average response time.

Both implementations demonstrated the capability to meet the demands of the TPC-H queries, validating the feasibility of data protection approaches.

These observations demonstrate that the performance of each solution is influenced by the proportion of sensitive and non-sensitive data, as well as the total volume of data processed. Queries like Q14, where non-sensitive data predominates, favor Vallum 2, which leverages its hybrid architecture to reduce the impact of *SGX*. Conversely, queries like Q21, which require full protection and intensive processing of sensitive data, highlight the advantage of Vallum 1, whose integral protection eliminates the costs associated with synchronization between environments. This alignment between the characteristics of the queries and the protection architectures underscores the importance of adapting solutions to the specific demands of each scenario

5.4 Privacy and Throughput Performance Evaluation

A dataset from the Brazilian federal government was used to evaluate privacy performance. This dataset contains detailed information about public servants, such as

salaries, travel tickets, and travel expenses for each trip. During the experiments, various parameters were adjusted to assess their impact on privacy performance:

(i) The final query result size, which consisted of 415,118 rows, totaling approximately 109 MB;

(ii) The number of quasi-identifier attributes, varying between 7 and 13;

(iii) The application of the k-anonymity constraint, with values of k chosen from $\{2, 5, 10\}$.

Additionally, to test the throughput of the solutions, different configurations of simultaneous connections and dataset sizes were evaluated. These measurements considered 5, 10, and 15 concurrent connections, as well as datasets with sizes of approximately $500MB$, $1GB$, and $1.5GB$. These metrics allowed us to assess the scalability of the proposed solutions under higher load scenarios, in terms of both the number of concurrent connections and the volume of data being processed.

The results obtained were generally consistent, regardless of the value of k or the number of quasi-identifiers used. All performance measurements were obtained by averaging the results of 10 executions for each query, ensuring a precise and robust evaluation of the different scenarios.

The results obtained were generally consistent, regardless of the value of k or the number of quasi-identifiers used. All performance measurements were obtained by averaging the results of 10 executions for each query, ensuring a precise and robust evaluation of the different scenarios. The detailed values of the results can be found in Table 5, which presents the response times for each scenario across different values of k.

The results demonstrate that Vallum 1, which uses full protection through *SGX*, exhibits higher response times compared to Vallum 2, which adopts a vertical data partitioning strategy to protect only data classified as confidential. For example, for $k = 10$, Vallum 1 had a response time of 46.08 seconds, while Vallum 2 achieved 43.09 seconds. This 6.5% difference in favor of Vallum 2 indicates superior efficiency in query execution, reflecting the system's ability to process non-sensitive data more quickly.

Scenario	K	Response Time (s)
MariaDB	10	30.17
Vallum 1	10	46.08
Vallum 2	10	43.09
MariaDB	5	56.61
Vallum 1	5	72.97
Vallum 2	5	66.78
MariaDB	2	46.77
Vallum 1	2	57.85
Vallum 2	2	56.91

Table 5 – Response times for each scenario across different values of K (rounded to two decimal places).

5.4.1 Privacy: Vallum 1 vs Vallum 2

The analysis of response times reveals that, in terms of efficiency, Vallum 2 outperforms Vallum 1. Specifically, for $k = 10$, Vallum 1 had a response time of 46.08 seconds, while Vallum 2 achieved 43.09 seconds, resulting in a 6.5% difference in favor of Vallum 2. This difference indicates greater efficiency in query execution by Vallum 2, mainly due to the system's ability to process non-sensitive data more quickly.

When analyzing the configurations for $k = 5$ and $k = 2$, the trend of better performance by Vallum 2 remains consistent. For $k = 5$, Vallum 1 recorded a time of 72.97 seconds, while Vallum 2 achieved 66.78 seconds, resulting in an 8.5% difference. For $k = 2$, the recorded times were 57.85 seconds for Vallum 1 and 56.91 seconds for Vallum 2, representing a smaller difference of only 1.6%, but still favoring Vallum 2. These results suggest that the vertical partitioning approach adopted by Vallum 2 not only preserves the security of sensitive data but also optimizes the overall system performance and reduces response time.

5.4.2 Comparison with MariaDB: Performance Cost Analysis

When comparing the response times of Vallum 1 with MariaDB, which serves as the baseline, both systems showed higher response times. For $k = 10$, Vallum 1 showed a 52.6% increase over MariaDB, meaning its response time was approximately 1.53 times greater. For $k = 5$, the increase was 28.9%, with the response time being about 1.29 times greater. For $k = 2$, Vallum 1 recorded a 23.6% increase, making its response time about 1.24 times greater than MariaDB (Figure 10).

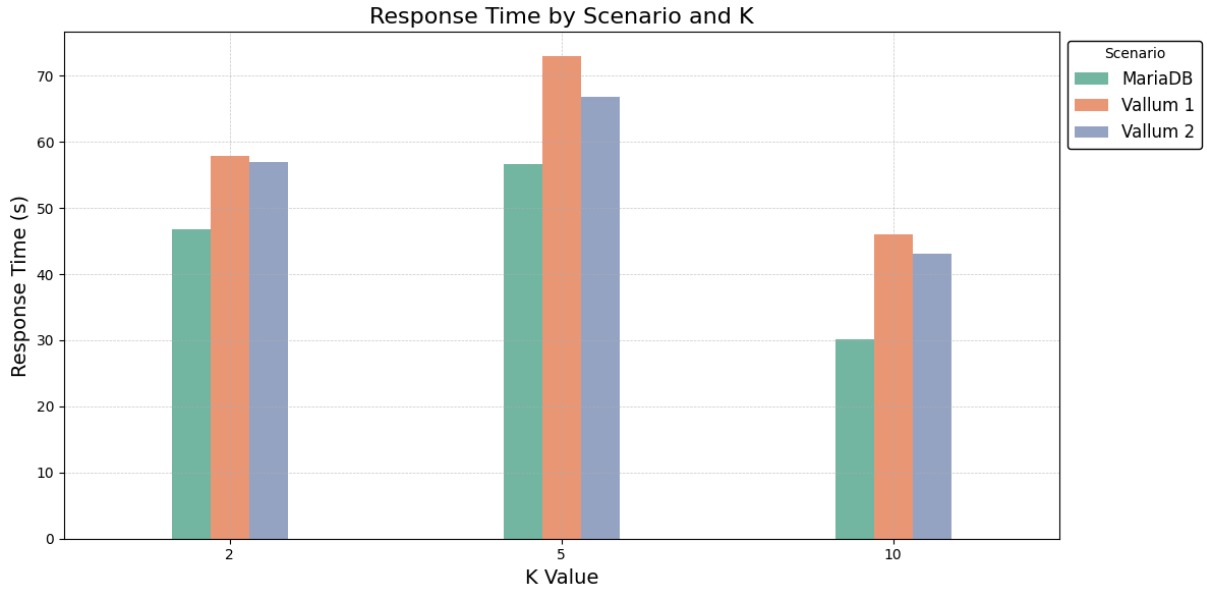


Figure 10 – Analysis of response times across privacy-preserving approaches for various values of K. This visualization emphasizes the performance differences between systems and highlights the efficiency of selective protection strategies in handling sensitive data.

Vallum 2, in turn, showed increases of 42.7% (for $k = 10$), 17.9% (for $k = 5$), and 21.7% (for $k = 2$) compared to MariaDB. These increases correspond to response times approximately 1.43, 1.18, and 1.22 times greater, respectively. These results highlight the performance cost associated with implementing security measures in both systems, emphasizing the importance of considering the trade-off between security and performance, especially in systems with requirements for the protection of sensitive data.

5.4.3 Considerations on the Trade-Off Between Security and Performance

The consideration of the trade-off between security and performance is an important factor, particularly in systems that handle sensitive data in cloud environments. While response times were higher for both platforms (Vallum 1 and Vallum 2), the protection of sensitive data provides a significant benefit in terms of privacy and security. The implementation of techniques such as *SGX* in Vallum 1 and the *SCONE* security layer in both platforms ensures data protection, albeit with performance costs.

5.4.4 Throughput Performance Evaluation

To conclude the experiments, we conducted tests with Vallum 1, Vallum 2, and the MariaDB database, focusing on the performance evaluation in terms of the amount of data or transactions each system can process within a given period.

We provide a comparative performance evaluation of the MariaDB, Vallum 1, and Vallum 2 systems under data protection conditions using the k-anonymity privacy algorithm with $k = 5$. The main goal was to investigate how different security approaches impact data processing capacity in scenarios with progressively increasing workloads, considering variations in the number of simultaneous connections and the volume of data processed.

The tests were conducted in scenarios with different configurations of concurrent connections: 5 connections processing a total of approximately $521.5MB$, 10 connections processing a total of $1GB$, and 15 concurrent connections handling $1.56GB$ of data. In addition to the gradual increase in the number of connections and the size of the processed data, all scenarios included data protection through the k-anonymity privacy algorithm, with a k value of 5. The results of these tests are presented in the Table 6.

Scenario	Size	Response Time (s)
MariaDB	521.5 MB	69.99
Vallum 1	521.5 MB	131.11
Vallum 2	521.5 MB	104.71
MariaDB	1 GB	128.24
Vallum 1	1 GB	311.85
Vallum 2	1 GB	184.90
MariaDB	1.56 GB	191.90
Vallum 1	1.56 GB	443.97
Vallum 2	1.56 GB	275.59

Table 6 – Throughput performance results for MariaDB, Vallum 1, and Vallum 2 (rounded to two decimal places).

5.4.5 Throughput Analysis

MariaDB was established as the baseline for performance comparison. In the tests, the system showed response times of 69.99 s for a total of $521.5MB$, which increased to 128.24 s when processing $1GB$, and 191.90 s for $1.56GB$. These results highlight optimized performance in an environment without data protection, reflecting the system's efficiency in handling large volumes of information.

In contrast, Vallum 1, which implements full protection through *SGX*, showed significant performance degradation. The response time for $521.5MB$ was 131.11 s, representing a slowdown of approximately 1.88 times compared to MariaDB. For $1GB$, the time increased to 311.85 s, resulting in a degradation of 2.43 times. With $1.56GB$, the response time reached 443.97 s, indicating a slowdown of 2.32 times compared to the baseline. This performance degradation is attributed to the need to process all operations securely, constrained by the *SGX Enclave Page Cache* (*EPC*), which compromises efficiency in queries involving larger result sizes.

On the other hand, Vallum 2 presents an approach that combines security and performance. In this scenario, the data is partitioned vertically, resulting in a response time of 104.71 s for $521.5MB$, which corresponds to a degradation of 1.49 times compared to MariaDB. For $1GB$, the response time was 184.90 s, representing a degradation

of 1.44 times. Finally, for $1.56GB$, the response time was 275.59 s, with a degradation of 1.44 times compared to the baseline. The partitioning structure allows non-sensitive operations to be processed without the security overhead, thus mitigating the negative impacts of *SGX*'s *EPC*.

5.4.6 Comparative Performance for Throughput: Vallum 1 vs Vallum 2

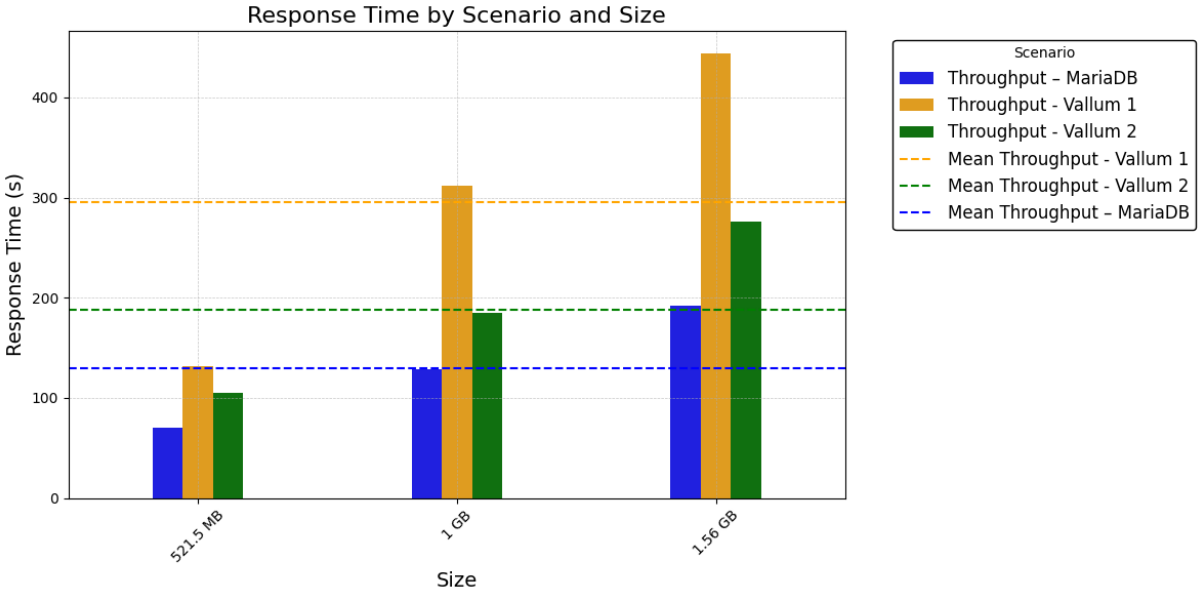


Figure 11 – Comparison of response times for various privacy-preserving approaches across different dataset sizes. This figure highlights the scalability and efficiency of each approach, demonstrating the trade-offs between data protection and system performance.

The comparative analysis between Vallum 1 and Vallum 2 reveals that the increase in the volume of processed data impacts the performance of Vallum 1 more significantly. In contrast, Vallum 2, which adopts a data partitioning strategy, shows a more controlled performance degradation. This suggests that the partitioning approach contributes to faster response times, especially in scenarios where not all data needs to be protected.

To illustrate these differences, we present a graph (Figure 11) that shows the response times in different scenarios, varying according to the size of the processed

data.

The results show that while data protection is important, the choice of architecture and security techniques directly affects performance. The comparison between Vallum 1 and Vallum 2 demonstrates that more efficient approaches, such as the vertical partitioning used by Vallum 2, can enhance platform performance, even as the volume of data increases. However, it is important to note that it is not always practical to leave some data unprotected. In such cases, Vallum 2 does not meet this requirement, while Vallum 1, with its full protection, offers greater flexibility in this regard.

6

Conclusions

This chapter revisits the core aspects of this research, connecting the findings to the initial problem, hypotheses, and objectives outlined at the beginning of the study. It begins by addressing the central issue of protecting sensitive data in cloud environments, emphasizing the challenges posed by the migration of operations to the cloud and the risks associated with shared infrastructure. It then evaluates the hypothesis of balancing robust security and performance, examining how the proposed platforms —particularly Vallum 1 and Vallum 2 — respond to these challenges. Finally, it discusses how the research objectives were achieved, with a particular focus on the development and impact of the Vallum in addressing data privacy, sensitivity, and performance in cloud environments.

The central problem addressed by this research was the increasing need to protect sensitive data in cloud computing environments, a context that has become more relevant as organizations and individuals have migrated their operations to the cloud. This scenario is characterized by shared infrastructure, where data and applications are often hosted in public environments, exposing them to a higher risk of malicious attacks. Furthermore, the cloud environment itself can serve as a vector for security breaches, jeopardizing the integrity and sensitivity of sensitive data.

The analysis conducted in this work shows that the challenge of protecting sensitive data is not only related to the implementation of effective security systems but also to the ability to balance these mechanisms with system performance. The research revealed that total protection of data in the cloud, as provided by SCONE/*SGX* in

Vallum 1, can significantly impact performance, particularly in scenarios with large data volumes. On the other hand, approaches such as Vallum 2, which employs selective protection, demonstrated better performance without significantly compromising security, especially when only sensitive data needs to be protected.

The central hypothesis of this research posited that robust security and privacy for sensitive data in cloud environments could be achieved through a multi-faceted approach, incorporating access control systems, hardware-level security measures, and specialized data anonymization algorithms. This hypothesis was partially validated, as the Vallum 1 solution, utilizing *SGX* for full protection, demonstrated that combining strong security with specialized hardware is feasible, but it also revealed a considerable performance overhead.

In contrast, the selective protection approach used in Vallum 2 proved to be more efficient in terms of performance, without compromising the security of sensitive data. Moreover, by implementing data partitioning, the Vallum 2 version was able to mitigate the negative effects of *SGX*'s security overhead, validating the hypothesis that a hybrid approach can be an efficient solution to balance security and performance in database systems in the cloud.

The research aimed to achieve two main objectives, as outlined at the beginning of the work:

- **Investigate systems and mechanisms to protect sensitive data in cloud environments:** The study evaluated a variety of protection approaches, including encryption, hardware-based security (such as SGX), and data anonymization techniques. The development of the Vallum 1 and Vallum 2 solutions was a direct response to this objective, as both offer ways to protect sensitive data without compromising the overall integrity of the system. The *SGX* implementation in Vallum 1 represents a robust security solution but with clear limitations in terms of performance. In contrast, Vallum 2, with its selective protection approach, demonstrated a more efficient solution that combined security and performance more effectively.

- **Analyze the impact of implementing sensitivity requirements on the perfor-**

mance of data access and manipulation in the cloud: The research thoroughly examined the impacts of implementing security mechanisms such as SGX on response time and throughput. It was found that while SGX provides high protection, it imposes a significant penalty in performance, especially with large data volumes. Vallum 2, by adopting selective protection, showed a much smaller performance impact, validating the idea that, in cloud environments, an adaptive protection strategy based on data sensitivity is more effective in terms of performance.

The results obtained confirm that robust security in cloud environments can be achieved, but with significant trade-offs in performance, particularly when hardware-based secure solutions like *SGX* are used. Vallum 1, with its full protection, ensured the sensitivity of data but at the cost of considerable response time and scalability penalties. The Vallum 2 version, by adopting selective protection, was more scalable, maintaining a good level of security while delivering better performance, especially when handling large volumes of data.

These findings reinforce the relevance of a hybrid approach, where sensitive data receives full protection, while non-sensitive data can be processed without the additional security overhead. In cloud environments, where scalability and processing efficiency are critical, the Vallum 2 solution appears to be a promising option to ensure privacy and security without sacrificing performance.

Contributions

This work contributes to the existing literature on cloud security by proposing a hybrid solution that combines the advantages of full protection with *SGX* and the efficiency of selective protection. The data partitioning approach in Vallum 2 represents an effective alternative to protecting sensitive data while minimizing performance overhead, an aspect that is still not widely explored in existing solutions.

In addition to the theoretical contributions, this research also led to significant practical results. The first version of Vallum was implemented within the context of

a major international project called *ATMOHPERE*, which involved various research institutions and companies in Brazil and Europe. This project played an important role in the development and validation of the Vallum in real-world scenarios.

Furthermore, several academic contributions have been made as a result of this research:

- **Vallum: Framework for Access & Privacy Protection** - Cloudscape Brazil - Demo Session, Belém, 2019;
- **Vallum: Database Privacy, Sensitivity, and Access Rights for Sensitive Data in Cloud Environments** - European Conference on Computer Systems - Poster Session, Dresden, 2019;
- **Vallum: Sensitivity and Access Control for Sensitive Data in Cloud Environments** - IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Sydney, 2019;
- **Federated and Secure Cloud Services for Building Medical Image Classifiers on an Intercontinental Infrastructure** - Future Generation Computer Systems, v. 110, p. 119-134, 2020;
- **Vallum-Med: Protecting Medical Data in Cloud Environments** - CIKM '20: The 29th ACM International Conference on Information and Knowledge Management - Demo Session, New York, 2020.

These contributions highlight the academic and practical impact of this research in the field of cloud security, particularly in the areas of privacy protection, access control, and data sensitivity in cloud environments.

Future Work

As a next step, it would be interesting to investigate how the proposed approach can be scaled to distributed systems, where multiple protected nodes could collaborate in

executing queries without compromising security. This would allow the solution to be applied in more complex environments, such as data clusters and large-scale systems.

Additionally, future studies could explore the integration of homomorphic encryption and differential privacy techniques to protect data in cloud environments. This would expand security options, enabling the protection of sensitive data even in more challenging scenarios, such as executing queries on large data volumes in the cloud.

Another promising direction for future research would be the integration of Artificial Intelligence (AI) techniques to enhance both the efficiency and security of query processing. For example, *Vallum 2* could be optimized with AI algorithms capable of rewriting queries to push execution as close as possible to the Database Management System (DBMS), thus minimizing the amount of sensitive data that needs to be transferred and improving both performance and security.

Automating the application of privacy policies could also be a significant advancement. AI techniques could be used to analyze and filter query results based on specific privacy rules, ensuring that sensitive data is effectively protected in an efficient manner.

Finally, another interesting area of research would be the use of machine learning techniques to predict data access patterns and optimize the placement of data in the DBMS. This would not only improve system efficiency but also increase security by reducing unnecessary exposure of sensitive data during query execution.

Challenges Faced

Throughout the development of this research, several challenges were encountered that impacted both the execution and completion of the work. These difficulties arose not only from technical and methodological aspects but also from external factors that significantly influenced the research process.

The COVID-19 pandemic posed substantial challenges during the research. The closure of the Federal University of Amazonas (UFAM) led to a shift to remote classes and activities, which disrupted access to essential on-campus resources such as labora-

tories, specialized software, and direct collaboration with colleagues and advisors. This abrupt transition increased the reliance on remote communication tools, which, while helpful, often lacked the immediacy and depth of in-person interactions. Additionally, the pandemic brought increased anxiety and uncertainty, affecting both focus and productivity. Prolonged periods of isolation and the added stress of adapting to a remote environment exacerbated these challenges, contributing to difficulties in maintaining steady progress.

From a technical perspective, one of the most significant challenges was implementing vertical partitioning in the database. This technique required dividing data into sensitive and non-sensitive parts while ensuring that the system could seamlessly handle queries across these partitions. This partitioning was further complicated by the need to maintain data consistency and minimize the performance impact of separating sensitive information.

Another major hurdle was the implementation of the Global as View (GaV) technique to ensure transparency for users interacting with vertically partitioned data. The GaV approach required creating unified virtual schemas that allowed users to query the database without needing to understand the underlying partitioning. Designing and implementing this abstraction layer involved integrating multiple components and ensuring compatibility between the secure and standard database environments. Debugging and optimizing the query rewriting and execution processes within this framework proved to be a particularly demanding task.

The pressures of balancing academic work with the personal and societal challenges brought on by the pandemic also led to health issues. Increased anxiety and stress resulted in periods of reduced productivity, necessitating adjustments to timelines and work methodologies. Maintaining mental and physical health while managing academic responsibilities required significant effort and resilience.

Despite these difficulties, the persistence and adaptability demonstrated throughout the research process allowed these challenges to be addressed effectively. Collaboration with advisors and peers, even in remote settings, played a critical role in overcoming technical obstacles. Moreover, incremental progress in implementing the partitioning

and GaV techniques, combined with rigorous testing, ensured the delivery of functional and efficient solutions.

The lessons learned from navigating these challenges not only contributed to the completion of this work but also provided valuable insights into handling adversity and complex problem-solving, both academically and personally. These experiences underscore the importance of resilience and adaptability in achieving long-term goals, even in the face of unforeseen circumstances.

BIBLIOGRAPHY

AKIN, I. H.; SUNAR, B. On the difficulty of securing web applications using cryptdb. In: *IEEE 4th Intl. Conf. on Big Data and Cloud Computing*. [S.l.: s.n.], 2014. p. 745–752. 23

ARASU, A. et al. Orthogonal security with cipherbase. In: *CIDR*. www.cidrdb.org, 2013. Disponível em: <http://dblp.uni-trier.de/db/conf/cidr/cidr2013.html#ArasuBEKKRV13>. 18, 33

ARMBRUST, M. et al. A view of cloud computing. *Communications of the ACM*, v. 53, n. 4, p. 50–58, 2010. 17

ARNAUTOV, S. et al. Scone: Secure linux containers with intel sgx. In: KEETON, K.; ROSCOE, T. (Ed.). *OSDI*. USENIX Association, 2016. p. 689–703. Disponível em: <http://dblp.uni-trier.de/db/conf/osdi/osdi2016.html#ArnautovTGKMPLM16>. 22, 25, 36, 42

BAJAJ, S.; SION, R. Trusteddb: a trusted hardware based database with privacy and data confidentiality. In: *ACM SIGMOD Intl. Conf. on Management of data*. [S.l.: s.n.], 2011. p. 205–216. 18, 33, 36

BLANQUER, I. et al. Federated and secure cloud services for building medical image classifiers on an intercontinental infrastructure. *Future Generation Computer Systems*, Elsevier BV, v. 110, p. 119–134, set. 2020. Disponível em: <https://doi.org/10.1016/j.future.2020.04.012>. 21

BOUGANIM, L.; PUCHERAL, P. Chip-secured data access: Confidential data on untrusted servers. In: *28th Intl. Conf. on Very Large Data Bases*. [S.l.: s.n.], 2002. p. 131–142. 33

BRASSER, F. et al. Software grand exposure: SGX cache attacks are practical. *arXiv preprint arXiv:1702.07521*, p. 33, 2017. 38

BRITO FELIPE T. ; MACHADO, J. C. Preservação de privacidade de dados: Fundamentos, técnicas e aplicações. In: *36o JAI - Jornadas de Atualização em Informática*. [S.l.: s.n.], 2017. p. 91–130. 26

BURON, M. et al. Obi-wan. *Proceedings of the VLDB Endowment*, Association for Computing Machinery (ACM), v. 13, n. 12, p. 2933–2936, ago. 2020. Disponível em: <https://doi.org/10.14778/3415478.3415512>. 29

CALÌ, A. et al. Data integration under integrity constraints. In: *Seminal Contributions to Information Systems Engineering*. Springer Berlin Heidelberg, 2013. p. 335–352. Disponível em: <https://doi.org/10.1007/978-3-642-36926-1_27>. 29

CALVANESE, D.; XIAO, G. Semantic technologies for data access and integration. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018. Disponível em: <https://doi.org/10.1145/3269206.3274272>. 29

CHEN, Y. et al. *What's new about cloud computing security*. 2010. UC Berkeley Report No. UCB/EECS-2010-5. 23

CHORTARAS, A. et al. Reformulating ontological queries using materialised rewritings. In: *Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics*. ACM, 2016. Disponível em: <https://doi.org/10.1145/2912845.2912867>. 29

COSTAN, V.; DEVADAS, S. Intel sgx explained. *IACR Cryptology ePrint Archive*, v. 2016, p. 86, 2016. Disponível em: <http://dblp.uni-trier.de/db/journals/iacr/iacr2016.html#CostanD16>. 22, 24, 25

DOUGLAS, J. *Querying Over Encrypted Databases in a Cloud Environment*. Tese (Doutorado), 2019. Disponível em: <https://doi.org/10.18122/td/1520/boisestate>. 18

DWORK, C. Differential privacy. *Encyclopedia of Cryptography and Security*, Springer, p. 338–340, 2011. 45

ESMEEL, T. K. et al. Balancing data utility versus information loss in data-privacy protection using k-anonymity. In: *2020 IEEE 8th Conference on Systems, Process and Control (ICSPC)*. IEEE, 2020. Disponível em: <https://doi.org/10.1109/icspc50992.2020.9305776>. 28

FERRAIOLO, D. F.; KUHN, D. R. Role-based access controls. *CoRR*, abs/0903.2171, 2009. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr0903.html#abs-0903-2171>. 22, 26

FIRESMITH, D. G. Security use cases. *Journal of object technology*, v. 2, n. 3, 2003. 17

FUHRY, B.; JAIN, H. A. J.; KERSCHBAUM, F. EncDBDB: Searchable encrypted, fast, compressed, in-memory database using enclaves. In: *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2021. Disponível em: <https://doi.org/10.1109/dsn48987.2021.00054>. 18, 33, 35

GESSERT, F.; WINGERATH, W.; RITTER, N. The NoSQL toolbox: The NoSQL landscape in a nutshell. In: *Fast and Scalable Cloud Data Management*. Springer International Publishing, 2020. p. 175–190. Disponível em: <https://doi.org/10.1007/978-3-030-43506-6_8>. 29

GRIBOV, A.; VINAYAGAMURTHY, D.; GORBUNOV, S. Stealthdb: a scalable encrypted database with full SQL query support. *CoRR*, abs/1711.02279, 2017. Disponível em: <http://arxiv.org/abs/1711.02279>. 18, 33, 34, 36

GUIMARAES, R. P. Vallum: Framework for access privacy protection. In: *Cloudscape Brazil*. [S.l.: s.n.], 2019. 21

GUIMARAES, R. P. et al. Vallum: Privacy, confidentiality and access control for sensitive data in cloud environments. *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, p. 103–110, 2019. 21, 47

GUIMARAES, R. P. et al. Vallum: Database privacy, confidentiality and access rights forsensitive data in cloud environments. In: *European Conference on Computer Systems - Poster Session*. [S.l.: s.n.], 2019. 21

GUIMARAES, R. P. et al. Vallum-med. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. ACM, 2020. Disponível em: <https://doi.org/10.1145/3340531.3417421>. 21

HARNIK, D. et al. Securing the storage data path with sgx enclaves. *CoRR*, abs/1806.10883, 2018. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1806.html#abs-1806-10883>. 25

JANSEN, W. Cloud hooks: Security and privacy issues in cloud computing. In: *Hawaii Intl. Conf. on System Sciences*. [S.l.: s.n.], 2011. p. 1–10. 22

KARIMOV, E. Trie data structure. In: *Data Structures and Algorithms in Swift*. Apress, 2020. p. 67–75. Disponível em: <https://doi.org/10.1007/978-1-4842-5769-2_9>. 29

KATSIS, Y.; PAPAKONSTANTINOU, Y. View-based data integration. In: *Encyclopedia of Database Systems*. Springer US, 2009. p. 3332–3339. Disponível em: <https://doi.org/10.1007/978-0-387-39940-9_1072>. 29

LOPEZ, J.; RUBIO, J. E. Access control for cyber-physical systems interconnected to the cloud. *Computer Networks*, Elsevier BV, v. 134, p. 46–54, abr. 2018. Disponível em: <https://doi.org/10.1016/j.comnet.2018.01.037>. 26

MACHANAVAJJHALA, A. et al. ℓ-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowledge Discovery from Data*, v. 1, n. 1, 2007. 45

NAVEED, M. et al. Inference attacks on property-preserving encrypted databases. In: *ACM SIGSAC Conf. Computer and Communications Security*. [S.l.: s.n.], 2015. p. 644–655. 23

NING, Z. et al. Preliminary study of trusted execution environments on heterogeneous edge platforms. In: *SEC*. IEEE, 2018. p. 421–426. ISBN 978-1-5386-9445-9. Disponível em: <http://dblp.uni-trier.de/db/conf/edge/sec2018.html#NingLZS18>. 24

PODDAR, R. et al. Arx: A strongly encrypted database system. *IACR Cryptology ePrint Archive*, v. 2016, p. 591, 2016. 18, 30, 31, 36, 45

POESS, M.; FLOYD, C. New tpc benchmarks for decision support and web commerce. *ACM SIGMOD Record*, v. 29, n. 4, p. 64–71, 2000. 65

POPA, R. A. et al. Cryptdb: protecting confidentiality with encrypted query processing. In: *ACM Symp. on OS Principles*. [S.l.: s.n.], 2011. p. 85–100. 18, 23, 30, 36

PRASSER, F. et al. Flexible data anonymization using ARX—current status and challenges ahead. *Software: Practice and Experience*, Wiley, fev. 2020. Disponível em: <https://doi.org/10.1002/spe.2812>. 22, 27, 28, 56

PRIEBE, C. et al. Enclavedb: A secure database using SGX. In: *EnclaveDB: A Secure Database using SGX*. [S.l.: s.n.], 2018. 18, 30, 33, 34

REN, X. et al. HEDA. *Proceedings of the VLDB Endowment*, Association for Computing Machinery (ACM), v. 16, n. 4, p. 601–614, dez. 2022. Disponível em: <https://doi.org/10.14778/3574245.3574248>. 18, 32

RIAZI, M. S. et al. HEAX. In: *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. [S.l.: s.n.], 2020. 17

SELLAMI, M.; HACID, M.-S.; GAMMOUDI, M. M. An incremental approach to data integration in presence of access control policies. In: *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. [S.l.: s.n.], 2017. p. 187–190. 29

SHINDE, S. et al. Preventing page faults from telling your secrets. In: *ACM Asia Conf. on Comp. and Comm. Security*. [S.l.: s.n.], 2016. p. 317–328. 38

SUN, Y. et al. Building enclave-native storage engines for practical encrypted databases. *Proceedings of the VLDB Endowment*, Association for Computing Machinery (ACM), v. 14, n. 6, p. 1019–1032, fev. 2021. Disponível em: <https://doi.org/10.14778/3447689.3447705>. 18, 33, 35

SWEENEY, L. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, v. 10, n. 5, p. 557–570, 2002. 45

TIAN, D. J. et al. A practical intel sgx setting for linux containers in the cloud. In: AHN, G.-J. et al. (Ed.). *CODASPY*. ACM, 2019. p. 255–266. ISBN 978-1-4503-6099-9. Disponível em: <http://dblp.uni-trier.de/db/conf/codaspy/codaspy2019.html#TianCHTB19>. 25

TORRA, V.; LADRA, S. Cluster-specific information loss measures in data privacy: A review. In: *2008 Third International Conference on Availability, Reliability and Security*. IEEE, 2008. Disponível em: <https://doi.org/10.1109/ares.2008.87>. 28

TORRA, V.; NAVARRO-ARRIBAS, G. Data privacy: A survey of results. In: *Studies in Computational Intelligence*. Springer International Publishing, 2014. p. 27–37. Disponível em: <https://doi.org/10.1007/978-3-319-09885-2_3>. 28

TU, S. et al. Processing analytical queries over encrypted data. *Proceedings of VLDB Endowment*, v. 6, n. 5, p. 289–300, 2013. 18, 23, 30, 36

WANG, Y. et al. Cryptsqlite: Protecting data confidentiality of sqlite with intel sgx. In: *NaNA*. IEEE Computer Society, 2017. p. 303–308. ISBN 978-1-5386-0604-9. Disponível em: <http://dblp.uni-trier.de/db/conf/nana/nana2017.html#WangLSMWYSLZD17>. 18, 33, 35

WONG, W. K. et al. Secure query processing with data interoperability in a cloud database environment. In: DYRESON, C. E.; LI, F.; ÖZSU, M. T. (Ed.). *SIGMOD Conference*. ACM, 2014. p. 1395–1406. ISBN 978-1-4503-2376-5. Disponível em: <http://dblp.uni-trier.de/db/conf/sigmod/sigmod2014.html#WongKCLY14>. 18, 30, 31, 36

XU, Y. et al. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: *IEEE Symp. on Security and Privacy*. [S.l.: s.n.], 2015. p. 640–656. 38

ZANDESH, Z. et al. Legal framework for health cloud: A systematic review. *International Journal of Medical Informatics*, v. 132, p. 103953, 2019. 17

ZHU, J. et al. Full encryption: an end to end encryption mechanism in gaussdb. *Proceedings of the VLDB Endowment*, Association for Computing Machinery (ACM), v. 14, n. 12, p. 2811–2814, jul. 2021. Disponível em: <https://doi.org/10.14778/3476311.3476351>. 18, 31

ZUO, C. et al. Fine-grained two-factor protection mechanism for data sharing in cloud storage. *IEEE Transactions on Information Forensics and Security*, Institute of Electrical and Electronics Engineers (IEEE), v. 13, n. 1, p. 186–196, jan. 2018. Disponível em: <https://doi.org/10.1109/tifs.2017.2746000>. 18