A Modular Architecture and a Cost-Model to Estimate the Overhead of Implementing Confidentiality in Cloud Computing Environments

Mauro Storch[1], **Vinícius Meyer**[2], **Avelino Zorzo**[1], **Cesar A. F. De Rose**[1]

[1]*Pontifical Catholic University of Rio Grande do Sul (PUCRS)* - Porto Alegre, Brazil

[2]*University of Taquari Valley (UNIVATES)* - Lajeado, Brazil

mauro.storch@acad.pucrs.br, vinimeyer@univates.br

avelino.zorzo@pucrs.br, cesar.derose@pucrs.br

Abstract. *Cloud computing has become increasingly popular among organizations. As a result, security has become a major concern in the adoption of cloud computing environments. To ensure confidentiality and prevent data leakage, organizations have adopted various security measures, including sophisticated authentication methods and strong cryptography algorithms. However, implementing these measures generates additional overhead that could impact resource consumption and performance at user level. This paper proposes a modular architecture for a full-stack confidentiality cloud and a model to estimate implementation costs for each component that can be used as a blueprint to implement the needed confidentiality in a particular cloud scenario and estimate the resulting overhead. It contributes to the literature by enabling cloud administrators and users to leverage confidentiality based on their security needs and budget. Preliminary experiments show that our cost model achieves a high level of accuracy, up to 95%.*

1. Introduction

In recent years, the adoption of security mechanisms in public cloud computing environments has increased significantly, from small businesses to government agencies. Virtual Private Networks (VPNs) and HTTPS are now standard solutions for ensuring data confidentiality *in-transit*. For data *at-rest*, research has focused on embedded security using standard cryptography algorithms, while ciphering techniques have been developed for data *on-processing* [Kumbhakar et al. 2023].

Adding security layers to a software stack can significantly impact both software performance and resource allocation, particularly in cases where the confidentiality principle is a key concern. For example, when storing encrypted data in the cloud, cryptography must be adopted throughout the persistence flow, resulting in additional overhead for encrypting and deciphering data during access. Similarly, encrypting data during transmission through the cloud's networks also incurs a cost, as all transmitted data must be encrypted before being sent and deciphered after being received on the destination host. Furthermore, ensuring confidentiality during data processing is critical for increasing privacy levels and preventing data leakage, such as Cross-VM attacks that exploit in-memory data leakage in shared tenancies. To address these challenges, techniques such as querying over encrypted databases, outsourcing cryptography computations, and homomorphic

encryption have been developed to process data without disclosing it. However, implementing these techniques also adds overhead to overall cloud allocation. As such, it is essential to estimate the associated costs of adopting confidentiality in a computational environment, particularly when resources are shared, such as in public cloud providers.

This work focuses on estimating the impact of implementing data confidentiality in public cloud environments by examining the three fundamental components of a cloud stack: communication, storage, and processing. We propose a full-stack modular architecture that outlines the placement of security components to support confidentiality during data transmission, storage, and processing. To estimate the performance impact of adding security layers, we conduct a comprehensive analysis of the CPU overheads associated with the confidentiality principle. Our evaluation includes factors such as Virtual Private Networks for encrypting data *in-transit*, the overhead privacy costs of storage systems such as Cryptography File Systems [Blaze 1993], and the various complexities involved in processing encrypted databases, from hash comparisons to homomorphic operations. We model these components separately and use an OLTP (online transaction processing) benchmark (TPC-C) to reproduce real-world cloud usage scenarios. Finally, we use the observed variables to introduce a cost-model that predicts the overhead of implementing confidentiality foe each security layer of a cloud environment.

This paper is organized as follows. Section 2 presents related work regarding security and confidentiality overhead evaluation in computational systems. Section 3 proposes proposes a modular architecture for a full-stack confidentiality cloud. Section 4 presents the formal model for overhead estimation as a composed formula, built for communication, storage, and processing overhead. Section 5 describes both the validation of the intermediary models and the evaluation of the security overhead model using an OLTP benchmark. Finally, the conclusion and future work are discussed in Section 6.

2. Related Work

Despite the advantages of cloud computing environments, security concerns are pointed out as the main reason for companies with privacy requirements to avoid its adoption. This motivates security studies to consider cloud characteristics including multi-tenancy scenarios, where different users share the same virtualization stack to run their Virtual Machines (VMs). This scenario is susceptible to information leakage in several ways, such as cross-VM attacks [Giechaskiel et al. 2022] and the recent vulnerabilities found in processor widely used in cloud computing providers [Lipp et al. 2018]. In addition, many other threats related to communication [Vashishtha et al. 2023] and storage [Venkatesan and Chitra 2022], make managers consider cloud environments to be an unsafe solution. The issues and challenges of cloud computing security [Noor et al. 2016] have been investigated by researchers taking into account its repercussions for communication, storage, and processing of user's data.

These three main elements are evaluated in several works for building cloud instances with the aim of supporting confidentiality. To increase security in a computational environment, including clouds, some solutions apply hardware-based security. Cryptographic co-processors are used to support confidentiality during the runtime phase of an application. These techniques are also applied to the virtualization layer to guarantee the traceability of users' VMs. Such solutions using specialized hardware components

are based on a model called Trusted Computing [Group 2017]. However, the utilization of such hardware components demands physical modification of data centers, which can increase operating costs of cloud providers.

On the other hand, software-based architectures for confidentiality in cloud computing mainly cover issues and risks related to data in-transit and at-rest. Besides the well-known techniques for supporting confidentiality for communication using Virtual Private Networks and IPSec, the solutions for storage use encryption and decryption of data during the persistence flow. Such approach uses a single key, since solutions are based on symmetric cryptography algorithms, which is considered a single point of failure in case of key leakage. In order to guarantee trustful key distribution, some solutions do validation of operating system's images before launching VMs, ensuring they are not modified to allow key leakage [Paladi et al. 2017].

Regarding performance, some strategies do data classification and adopt cryptography for sensitive chunks only, reducing the overall impact of the encryption process [Tang et al. 2022]. Software-based solutions that apply confidentiality on the on-processing phase, have developed some novel cryptography techniques, such as Searchable Symmetric Encryption Schema (SSE) [Poh et al. 2017] and Homomorphic Encryption [Gentry 2009]. But just few works have introduced these concepts in cloud computing environments [Ali et al. 2024, Lopez et al. 2024]. Although these works support high protection level for sensitive data, they do not support confidentiality in communication, storage, and processing simultaneously, nor evaluate the resource allocation impact of adding cryptography solutions in a cloud environment, the two main contributions of our work.

3. Full-Stack Confidentiality Architecture

Confidentiality is a necessity for users with sensitive data in cloud computing environments, especially when using public providers. To be effective, a cloud computing architecture needs to consider confidentiality mechanisms for data in-transit, at-rest, and on-processing. Although some works in the literature have studied confidentiality in this domain, they lack in applying those mechanisms for all phases of the data workflow. Considering this scenario, it would be essential to support data confidentiality and to measure the impact of specialized mechanisms for communication, storage, and processing axes either isolated or combined, based on users' privacy requirements.

Security costs of cloud deployment are measured from different perspectives according to the chosen model: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), or Software-as-a-Service (SaaS). For the IaaS model, the security should be managed and deployed by the cloud users. Users are responsible for the cryptography and authentication mechanisms that will consume CPU cycles in their rented Virtual Machines. For PaaS and SaaS, the cloud provider delivers security as part of services such as databases, e-mail services, and e-commerce systems. In both cases, measuring the security overhead impact can help both users and providers to estimate resource allocation, availability, and costs.

On one hand, for PaaS or SaaS models, cloud providers offer security mechanisms like file encryption and private networks. Those services are also known as Security-as-a-Service (SECaaS) [Furfaro et al. 2014], where security is the responsibility of the cloud

provider. However, the user has lower control of the chosen algorithms and the key management. On the other hand, in the IaaS model, the user is responsible for deploying the security mechanisms to communicate, store and also process data. In such scenarios, security levels could be established following companies' security requirements. Both scenarios apply similar techniques to protect data, especially when supporting confidentiality through cryptography algorithms.

To provide confidentiality in the cloud, it is necessary to start thinking about the cloud as a computational system that is composed of data processing, storage, and communication axes. From that perspective, this work considers exploring the costs of the confidentiality principle in each of those axes.

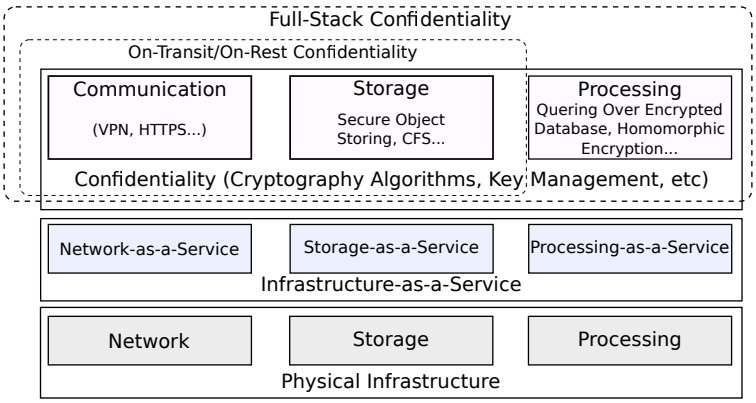

Figure 1. A modular confidentiality architecture for full-stack cloud environments.

Figure 1 shows a cloud architecture composed of services from a provider, which are managed by a confidentiality layer offering security services to cloud users. When using IaaS model, the deployment of this layer is the responsibility of cloud users. For PaaS and SaaS, this layer should be supported by cloud providers. The components can be combined to achieve a security level according to users' needs, *i.e.* supporting in-transit and at-rest confidentiality when choosing the communication and storage axes. A Full-Stack confidentiality design consists of adding confidentiality in all software operations from communicating to storing and processing. These three axes of a confidentiality cloud have independent implementations as follows.

The first axis considered is the communication among cloud nodes, internally and externally. To build confidentiality communication in the cloud, managers commonly create channels based on Virtual Private Networks, such as HTTPS and IPSec. The channels apply authentication protocols (using a password or digital certificates) and cryptographic algorithms. These algorithms can generate overload in the system, based on the chosen algorithm. Most cloud providers support confidentiality for communication as a standard service; however, in the IaaS renting model, the user is responsible for the security of communications among nodes, even internally in the cloud environment.

The second axis to be considered in a cloud environment is the storage layer used for data persistence. Besides the novel techniques proposed in the literature, one can also consider adopting Cryptographic File Systems (CFS) for data storage in public clouds. This technique, initially proposed by Blaze [Blaze 1993] in the '90s, consists of encrypt-

ing and deciphering data during the persistence flow, transparently supporting the confidentiality principle. Regarding feasibility, this technique can support the adoption of a strong algorithm with low impact on systems' performance. Similarly, the Object-Storage services provided by cloud software, such as OpenStack Swift [1] and AWS S3 [2], have introduced an encryption layer to support privacy for stored data.

The third axis is related to the processing layer in a cloud environment, specifically related to the *runtime* phase of applications. In the last few years, researchers have been proposing solutions to add confidentiality during the instruction execution inside the processors. The embedded cryptography instructions of Intel processors allow (de)ciphering data using hardware instructions. Those instructions handle only the registers' data, keeping data stored in the main memory encrypted. Although it could prevent data leakage from some cross-VM attacks by exploring the shared memory, the data would be in plain text in some manner inside the chip, theoretically allowing some leakage. Intel also introduced in some processors the SGX instructions, which created a *sandbox* for executing sensitive parts of applications. However, the application should be modified to add those instructions which add some overhead in its execution [Brenner et al. 2016]. Besides hardware-based solutions, to support confidentiality in the processing phase, it is also possible to consider algorithms that can handle encrypted data, such as operations over encrypted databases and homomorphic encryption. These techniques would ensure the computation of data without deciphering it entirely, and therefore avoid information leakage. The encrypted database querying technique [Arasu et al. 2014] considers handling enciphered data in the same manner as a Database Management System. Most encrypted databases also support homomorphic encryption for some operations such as SUM queries (SQL queries using SUM operations).

On top of such secure architecture, users' applications could leverage the benefits of cloud computing, such as resource elasticity and the pay-as-you-go fashion, without disclosing sensitive data [Meyer et al. 2022, Meyer et al. 2021]. These three axes can coexist, increasing the security level and avoiding information leakage of both application and data. Nowadays, public cloud providers already offer support to confidentiality for communication and storage in their IaaS products. Additionally, they have also been adopting security standards to support companies' PLAs (Privacy Level Agreements) in PaaS and SaaS products. Concerning feasibility, it is important for both users and providers to understand the impact of adding privacy in this environment. Each axis should be considered independently to produce a formulation to estimate the overhead added by the confidentiality mechanisms.

4. Modeling Cloud's Confidentiality Costs

As stated in previous sections, the cloud architecture can be seen as a computational system built over the axes of communication, storage, and processing. The addition of confidentiality in those axes can be evaluated in isolated models, allowing both users and providers to decide which ones are part of their stack and to understand the costs of supporting a required security level. On one hand, the user could be a company renting resources in a public cloud environment following the IaaS model, where it is necessary

[1] http://www.openstack.org/

[2] https://aws.amazon.com/en/s3/

to calculate the security impact in rented resources. On the other hand, the provider could be a cloud player (AWS, Azure) who offers confidentiality in their services in PaaS or SaaS models, where it is necessary to calculate the total costs of the security service.

4.1. Network Modeling

Network security in cloud computing environments considers communication among services, placed either inside or outside the cloud, for the models IaaS, PaaS, and SaaS. This communication is commonly established through the Internet, and the security requirements should be considered for compliance with companies' data exposure rules.

Regarding the application of confidentiality, tools such as IPSec and other VPN implementations apply cryptography algorithms for ciphering the payload of the protocol stack. The cipher used for this process is based on symmetric algorithms, which use a single key for encrypting data before sending and then decrypting it after receiving. This key is exchanged once, at the beginning of the communication during the authentication process. Some implementation also considers recreating the communication key using time intervals or communication events, increasing security by forcing an attacker to recalculate the target key. However, the overhead of these events is not significant compared to that of the encryption process.

Based on the data encryption, one can first define a formula considering the CPU time allocation for the cryptography spread over the communication time for a single cloud node as:

$$N_{(d)} = \frac{C_{time}(d)}{T(d)} \tag{1}$$

where C is the ciphering time consumption for a given d data amount, and T is the total transmission time. Regarding specific aspects of encrypting and sending, receiving, and deciphering data in a cloud node, the CPU time for each cryptography process needs to be considered in different perspectives for a specific amount of data, divided into sent and received amounts. So the formula could be rewritten as:

$$N_{(s,r,bs,br)} = \frac{E \times s + D \times r}{s/bs + r/br} \tag{2}$$

where E and D give the weight of cryptography for a certain data volume for encrypting and deciphering, respectively applied to each data amount. The sent and received volumes are defined by s and r, respectively. The total time is given, then, considering the bandwidth available per flow, expressed by bs and br. This formula produces a linear relationship between data volume and CPU allocation, and the weights need to be fitted according to the chosen algorithm.

Based on an application trace, it is possible to produce the values for fitting the formula. It is necessary to calculate the data volume transferred in a single node, the node's network capacity in terms of bandwidth, and the algorithm cost for encrypting (E) and deciphering (D) a certain data amount.

4.2. Storage Modeling

To support the confidentiality of data in a cloud environment during their at-rest stage, the persistence layer should also support the use of both authentication and cryptography

tools. Considering the IaaS cloud service model, in most public clouds, the VM's virtual disks are stored in shared spaces (i.e. managed by Linux Volume Manager, LVM), and data ciphering is often deployed by users. For PaaS and SaaS, providers support an encryption mechanism using standard cryptography algorithms, e.g. AES. In such implementation, data should be ciphered in every disk access, encrypting before writing and deciphering after reading.

Similarly to network modeling, there is also a linear relation considering the data volume persisted and the CPU allocation by the cryptography mechanism adopted in the persistence flow, such as in Cryptography File Systems. This relation is also observed in the function of the bandwidth capacity of the I/O subsystem, like in network capacity. In doing so, the formula to model the storage can be written as:

$$S_{(w,r,TP_w,TP_r)} = \frac{E \times w + D \times r}{w \times TP_w + r \times TP_r} \tag{3}$$

where w and r are total data written and read, respectively, and E and D are the weights for estimating the CPU time of encrypting and deciphering an amount of data. These values represent the time consumption cost for encrypting and decrypting a certain data amount, respectively. An essential aspect to be noticed in this model is the TP (throughput) variables that should consider the IO subsystem capacity, such as iowait Operating Systems variables. In [Storch and Rose 2017], the authors presented a detailed evaluation of the memory impact within the IO sub-system when using cryptographic file systems.

To apply this modeling, the application's storage data access should be traced concerning both reading and writing volumes. It is also necessary to measure the throughput capacity of the IO subsystem and the CPU time for the cryptography algorithm.

4.3. Processing Modeling

Even if the communication is made through a private tunnel, and data stored in the cloud is ciphered, it is also necessary to adopt privacy mechanisms during the processing phase since the cloud environment is a public and shared environment. Possible attacks include the cross-VM attack, which exploits cache memory shared among Virtual Machines. For instance, Querying over Encrypted Databases has gained attention as a solution for low transition issues since it follows structures such as the Relational Databases Systems [Xu et al. 2017] and uses standard SQL instructions. This solution handles data without disclosing it, using a set of complex operations. Unlike the linearity observed in communicating and storing models, the complexity of each operation for the processing axis needs to be considered separately, which demands the software's tracking to estimate the overall impact of privacy in the processing phase. For instance, a *Select equality*[1] operation is far cheaper than a *Select sum*[2] operation since the first one computes simple hash comparisons, while the second one requires Homomorphic operations [Popa et al. 2011].

The diversity of operations demands a formulation that considers each of them in isolation. The dataset is also observed per operation. The cost of processing data with privacy for this scenario is based on mapping the application's operations that can be replaced or wrapped by a secure instruction. For instance, the SQL instruction (insert,

[1] A SQL select with a where clause with an equals operation

[2] A SQL select with a SUM function adding values from a column

update, delete, select) can be replaced when using an encrypted database. The overhead is then evaluated per operation and then summarized. This cost can be modeled as:

$$P_{(O,d)} = O_1(d) + O_2(d) + \cdots + O_n(d) P_{(O,d)} = \sum_{i=0}^{n} O_i(d) \tag{4}$$

where n is the number of operations, and O is a matrix of cost functions over a certain dataset d. Every replaced operation should be mapped in the O matrix, and their costs are then added to P, which represents the overhead of processing with confidentiality.

4.4. Full-Stack Modeling

The three formulas could be applied independently, according to the required mechanisms adopted in the scenario. They could also be combined for scenarios, using two or three of the axes simultaneously. For cloud computing applications, one may consider, at least, combining the communication axis with secure storage as:

$$\begin{aligned} C_{(s,r,bs,br,w,r,TP_w,TP_r)} = \\ N(s,r,bs,br) + S(w,r,TP_w,TP_r) \end{aligned} \tag{5}$$

and finally adding secure processing costs as:

$$\begin{aligned} C_{(s,r,bs,br,TP_w,TP_r,O)} = \beta_0 + \beta_1 N(s,r,bs,br) + \\ \beta_2 S(s,r,TP_w,TP_r) + \beta_3 P(O, s+r) \end{aligned} \tag{6}$$

where O is the list of encrypted operations, which is processed over a dataset. The three formulas presented in this section aim to explain the CPU overhead according to the nature of the cryptography mechanism adopted for each component in its respective axis. The network and storage axes use standard cryptography algorithms, most of them based on symmetric algorithms. As mentioned, these algorithms impact the overall system linearly, according to the data volume that is encrypted and deciphered. The processing axis also uses cryptography algorithms, i.e. homomorphic encryption, but the data volume is not a single determinant for estimating the overhead since the complexity of the secure processing phase is variable.

This modeling evaluation and its applicability are presented in the next section for an OLTP benchmark running in a cloud environment.

5. Model Evaluation

The nature of the two kinds of algorithms (standard cryptography and processing encryption) drives the experiments that are demanded to produce an independent evaluation of each axis.

The model evaluation considers a benchmark execution over a cloud environment composed of VMs that communicate through a local network, with storage provided by the virtualization layer, in this case using Xen Server 6.2. The objective of this evaluation includes mapping the overhead added by communicating through a VPN, storing data in the cryptographic file system, and processing data into an encrypted relational database. After evaluating the network and storage models independently, the CPU load added by the cryptography mechanism is used for fitting a linear regression to predict the *On-transit/At-Rest Confidentiality* scenario depicted in Figure 1. Then, these values are used in combination within the processing axis model to evaluate the *Full-Stack Confidentiality* scenario.

The benchmark is a TPC-C [3] implementation producing a workload from one VM over a standard MySQL instance hosted into the second VM. For each execution, the database is re-populated, and both the communication and storage data volumes are profiled, as well as the CPU load percentages of the database process and the entire node. These metrics are produced in four execution scenarios: (1) without any cryptography, (2) with only Virtual Private Network (VPN), (3) with only the Cryptography File System (CFS), and (4) with both VPN and CFS. These four scenarios are also stressed using the benchmark, where the executions are from 1 to 10 simultaneous connections.

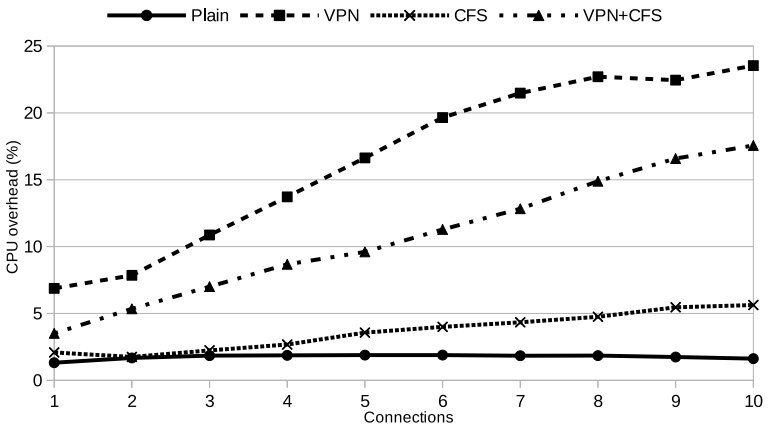

Figure 2. CPU load comparison among cryptography adoption modes: Plain, VPN-only, CFS-only, VPN+CFS

Figure 2 shows an accumulated CPU load for the entire node but discards the CPU load of the database process. These experiments aim to isolate the application and external mechanisms added to the host. It is possible to observe that with the increasing of simultaneous connections, the CPU load for scenario *(1) plain* does not have a significant difference. Otherwise, when adding some privacy mechanisms to the scenario, the CPU load was affected, and it is also possible to observe a linear relation within the data volume processed by the benchmark.

Regarding the fact that each experiment has different data volumes, it is possible to explain the CPU load as a function of the data volume as mentioned in Equations 2 and 3. Next, a multi-linear regression was calculated for Scenarios *(2) VPN* and *(3) CFS* with the aim to predict Scenario *(4) VPN+CFS*. For network overhead regression, the data volumes sent and received are added up to explain the extra CPU time consumption in communicating with privacy. For storage overhead regression, the volume written and read is then added up to explain the additional CPU load added by the cryptographic layer in the persistence flow. The independent regressions achieve an R-squared within 0.9991 and 0.9982, respectively. This evaluation drives the demonstration of the accuracy of Equation 5, where the predictions based on network and storage are added up to achieve the prediction of combined techniques.

Figure 3 shows the CPU overhead prediction for Scenario *(4) VPN+CFS*, based on the regression over data volume and CPU consumption of Scenarios *(2) VPN* and *(3) CFS*. The values demonstrate the application of Equation 5, with an accuracy close to 92% considering unstressed scenarios, where the CPU idle variable is higher than 20%, and an accuracy close to 95.7% for stressed cases, when CPU idle is lower than 15%, which was within the standard deviation.

After this validation, the same set of experiments for the four scenarios earlier described

[3]https://www.tpc.org/tpcc/

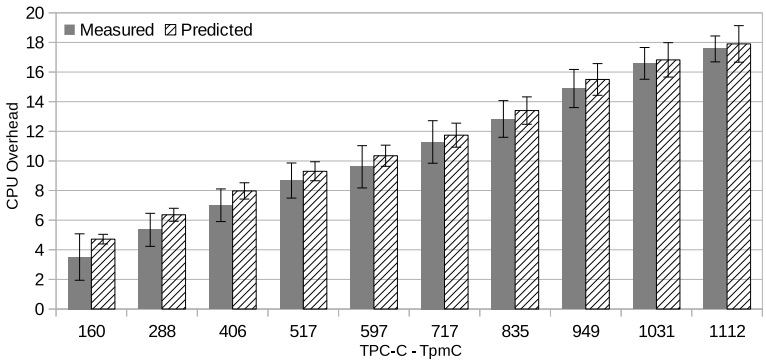

Figure 3. CPU Load prediction for TPC-C benchmark using Cryptography File Systems and Virtual Private Networks.

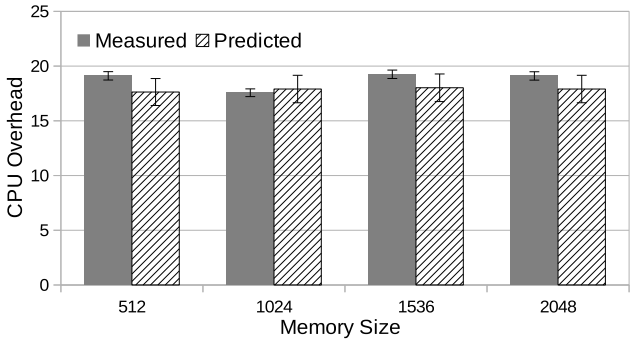

Figure 4. CPU Load prediction for TPC-C benchmark running in virtual machines with different memory sizes.

was reproduced for different memory sizes in the VM, with the aim of re-validating the relation of data volume and CPU load. The same fitted values obtained with the multi-linear regression were used to predict the CPU overhead. The predicted values for different memory size setups achieved accuracy close to 94%. Figure 4 shows the comparison for these experiments.

Following the validation of the modeling, Equation 4 presents a sum of the overhead of the different complexities for encrypted operations. The application of this formula requires a profile of the cloud software. For these experiments, the set of operations executed by the TPC-C benchmark was mapped, and the overhead for each one was calculated according to CryptDB evaluation [Popa et al. 2011].

Besides the overhead added by the techniques of Querying over Encrypted Databases, the CryptDB also changes the data volume persisted in the server side. This issue increases the database volume by 4.5 times, which demands extra CPU for persisting it into a Cryptography File System. However, no data volume increase is noticed in the communication.

To demonstrate the application of Equations 4 and 6, the CPU overhead is re-calculated in a scenario where the CryptDB overhead reduces the number of processed operations. By reducing the number of operations, a multi-linear function is calculated to achieve new values for reading and writing variables based on data volumes considering a CFS. These new values are finally used to recalculate the CPU overhead. Figure 5 shows the predicted overhead of VPN and CFS when the CryptDB is adopted in the environment. It is also possible to observe a reduction in the number of operations of up to 10% in average.

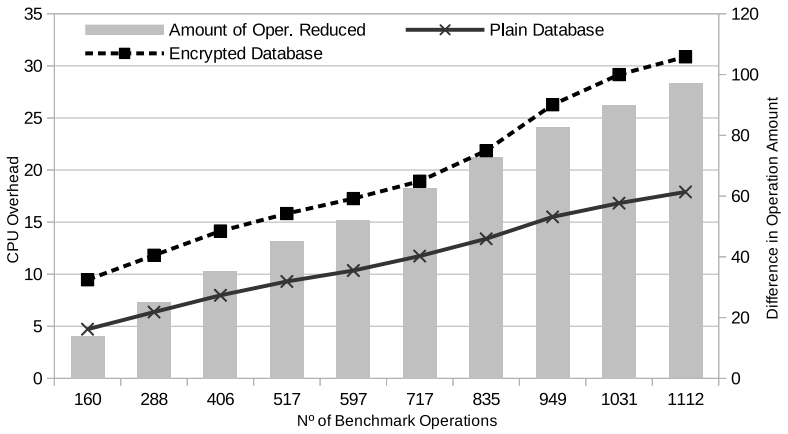

Figure 5. CPU Load overhead recalculated in function of CryptDB overhead and its increased persisted data volume.

6. Conclusion and Future Work

In recent years, companies and government agencies have increased their security demands to achieve privacy in cloud computing environments. However, the security mechanisms that support the confidentiality principle are commonly not part of either software sizing estimations or cloud pricing. This may lead to under-provisioning and, consequentially, performance loss at user-level resulting in additional investments to maintain the service. A better understanding of the trade-offs involved when privacy techniques and mechanisms are used will help users and providers to build tools for estimating the extra overhead of adding confidentiality to the communication, storage, and processing of sensitive data in cloud environments. Both the modular full-stack architecture and the cost-model proposed in this paper are a first step in this direction, and this work already considers the data volume, both transferred and persisted in the cloud, as well as the diverse complexity of private operations, such as the homomorphic operations, computed inside virtual machines in public clouds. The evaluation presented in this paper compares the execution of a database benchmark with and without privacy mechanisms, which includes Cryptography File Systems and Virtual Private Networks, and the model could predict the confidentiality overhead with an accuracy close to 95%, enabling cloud administrators and users to leverage confidentiality based on their security needs and budget constrains. In future work, we will expand the scope of our experiments, validating the proposed cost model with applications from other domains, like batch processing with Hadoop[1], as well as streaming and signal processing applications, like Flink[2].

References

Ali, S., Wadho, S. A., Yichiet, A., Gan, M. L., and Lee, C. K. (2024). Advancing cloud security: Unveiling the protective potential of homomorphic secret sharing in secure cloud computing. *Egyptian Informatics Journal*, 27:100519.

Arasu, A., Eguro, K., Kaushik, R., and Ramamurthy, R. (2014). Querying encrypted data. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1259–1261, New York, NY, USA. ACM.

Blaze, M. (1993). A cryptographic file system for unix. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 9–16, New York, NY, USA. ACM.

[1]http://hadoop.apache.org
[2]https://flink.apache.org

Brenner, S., Wulf, C., Goltzsche, D., Weichbrodt, N., Lorenz, M., Fetzer, C., Pietzuch, P., and Kapitza, R. (2016). Securekeeper: Confidential zookeeper using intel sgx. In *Proceedings of the 17th International Middleware Conference*, Middleware '16, pages 14:1–14:13, New York, NY, USA. ACM.

Furfaro, A., Garro, A., and Tundis, A. (2014). Towards security as a service: On the modeling of security services for cloud computing. In *2014 Int. Carnahan Conf. on Sec. Tech.*, pages 1–6.

Gentry, C. (2009). *A fully homomorphic encryption scheme*. PhD thesis, Stanford University.

Giechaskiel, I., Tian, S., and Szefer, J. (2022). Cross-vm covert- and side-channel attacks in cloud fpgas. *ACM Trans. Reconfigurable Technol. Syst.*, 16(1).

Group, T. C. (2017). Trusted computing.

Kumbhakar, D., Sanyal, K., and Karforma, S. (2023). An optimal and efficient data security technique through crypto-stegano for e-commerce. *Multimedia Tools and Applic.*, 82(14).

Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., and Hamburg, M. (2018). Meltdown. *ArXiv e-prints*.

Lopez, L. J. R., Millan Mayorga, D., Martinez Poveda, L. H., Amaya, A. F. C., and Rojas Reales, W. (2024). Hybrid architectures used in the protection of large healthcare records based on cloud and blockchain integration: A review. *Computers*, 13(6).

Meyer, V., da Silva, M. L., Kirchoff, D. F., and De Rose, C. A. (2022). Iada: A dynamic interference-aware cloud scheduling architecture for latency-sensitive workloads. *Journal of Systems and Software*, 194:111491.

Meyer, V., Kirchoff, D. F., Da Silva, M. L., and De Rose, C. A. (2021). Ml-driven classification scheme for dynamic interference-aware resource scheduling in cloud infrastructures. *Journal of Systems Architecture*, 116:102064.

Noor, T. H., Sheng, Q. Z., Maamar, Z., and Zeadally, S. (2016). Managing trust in the cloud: State of the art and research challenges. *Computer*, 49(2):34–45.

Paladi, N., Gehrmann, C., and Michalas, A. (2017). Providing user security guarantees in public infrastructure clouds. *IEEE Transactions on Cloud Computing*, PP(99):1–1.

Poh, G. S., Chin, J.-J., Yau, W.-C., Choo, K.-K. R., and Mohamad, M. S. (2017). Searchable symmetric encryption: Designs and challenges. *ACM Comput. Surv.*, 50(3):40:1–40:37.

Popa, R. A., Zeldovich, N., and Balakrishnan, H. (2011). Cryptdb: A practical encrypted relational dbms. Technical report, MIT Libraries.

Storch, M. and Rose, C. A. F. D. (2017). Cloud storage cost modeling for cryptographic file systems. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 9–14.

Tang, X., Liu, Z., Shao, Y., and Di, H. (2022). Side channel attack resistant cross-user generalized deduplication for cloud storage. In *ICC 2022 - IEEE International Conference on communications*, pages 998–1003.

Vashishtha, L. K., Singh, A. P., and Chatterjee, K. (2023). Hidm: A hybrid intrusion detection model for cloud based systems. *Wireless Personal Communications*, 128(4):2637–2666.

Venkatesan, B. and Chitra, S. (2022). Retracted: An enhance the data security performance using an optimal cloud network security for big data cloud framework. *International Journal of Communication Systems*, 35(16):e4854.

Xu, G., Ren, Y., Li, H., Liu, D., Dai, Y., and Yang, K. (2017). Cryptmdb: A practical encrypted mongodb over big data. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6.