PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services

Yiannis Verginadis[1], Antonis Michalas[2], Panagiotis Gouvas[3], Gunther Schiefer[4], Gerald Hübsch[5] and Iraklis Paraskakis[6]

[1]*Institute of Communications and Computer Systems, National Technical University of Athens, Athens, Greece*
[2]*Security Lab, Swedish Institute of Computer Science, Stockholm, Sweden*
[3]*Ubitech Ltd., Athens, Greece*
[4]*Karlsruhe Institute of Technology, Karlsruhe, Germany*
[5]*CAS Software AG, Karlsruhe, Germany*
[6]*South East European Research Centre, Thessaloniki, Greece*

Keywords: Data Privacy, Security by Design, Context-aware Security, Symmetric Searchable Encryption, Cloud Computing.

Abstract: The valuable transformation of organizations that adopt cloud computing is indisputably accompanied by a number of security threats that should be considered. In this paper, we outline significant security challenges presented when migrating to a cloud environment and propose PaaSword – a novel holistic, data privacy and security by design, framework that aspires to alleviate them. The envisaged framework intends to maximize and fortify the trust of individual, professional and corporate users to cloud services. Specifically, PaaSword involves a context-aware security model, the necessary policies enforcement and governance mechanisms along with a physical distribution, encryption and query middleware, aimed at facilitating the implementation of secure and transparent cloud-based applications.

1 INTRODUCTION

Until recently, large-scale computing was available exclusively to large organizations with an abundance of in-house expertise. Cloud computing has changed that to the point where any user with even basic technical skills can obtain access to vast computing resources at low cost. In the technology adoption lifecycle, cloud computing has now moved from an early adopters stage to an early majority, where we typically see exponential number of deployments (Santos et al., 2009). Throughout the past few years, many users have started relying on cloud services without realizing it. Major web mail providers utilize cloud technology; tablets and smartphones often default to automatically uploading user photos to cloud storage and social networks; finally, several prominent CRM vendors offer their services using the cloud. In other words, the adoption of cloud computing has moved from focused interest to widely spread intensive experimentation and is now rapidly approaching a phase of near ubiquitous use.

Enterprises increasingly recognize the compelling economic and operational benefits of cloud computing (Micro, 2010). Virtualizing and pooling IT resources in the cloud enables organisations to realize significant cost savings and accelerates deployment of new applications, simultaneously transforming business and government at an unprecedented pace (CSA, 2013). However, those valuable business benefits cannot be unlocked without addressing new data security challenges posed by cloud computing.

Despite the benefits of cloud computing, many companies have remained cautious due to security concerns. Applications and storage volumes often reside next to potentially hostile virtual environments, leaving sensitive information at risk to theft, unauthorized exposure or malicious manipulation. Governmental regulation regarding data privacy and location presents an additional concern of significant legal and financial consequences if data confidentiality is breached, or if cloud providers move regu-

DOI: 10.5220/0005489302060213
In *Proceedings of the 5th International Conference on Cloud Computing and Services Science* (CLOSER-2015), pages 206-213
ISBN: 978-989-758-104-5

lated data across national borders (Paladi and Michalas, 2014). The contribution of this position paper is two-fold. First, we present a list of core security requirements and challenges that must be considered when migrating to a cloud environment. These security requirements were derived based on our experience with migrating existing applications to a private Infrastructure-as-a-Service (IaaS) cloud (Michalas et al., 2014). We extend this guide by discussing important attack vector characteristics for cloud environments that will pave the way for providing tighter security when building cloud services. Second, in order to tackle the critical cloud security challenges we present PaaSword, an envisaged framework that will maximize and fortify the trust of individual, professional and corporate users to cloud services and applications. PaaSword achieves that by providing storage protection mechanisms, which improves confidentiality and integrity protection of users' data in the cloud while it does not affect the data access functionality.

The rest of this paper is organized as follows. In Section 2, we further elaborate on the main data security challenges in cloud-enabled services and applications. In Section 3, we introduce a holistic, data privacy and security by design, framework enhanced by sophisticated context-aware access models and robust policy enforcement and governance mechanisms, aimed at facilitating the implementation of secure and transparent cloud-based applications. In Section 4, we briefly discuss relevant work while in Section 5, we conclude the paper by presenting the next steps for the implementation and evaluation of the proposed framework.

2 DATA SECURITY CHALLENGES IN THE CLOUD

According to the Cloud Security Alliance (Alliance, 2013), several top security identified threats refer to information disclosure and repudiation, rendering data security as realised through data protection, privacy, confidentiality, and integrity as top priorities. More precisely, the top four threats identified are: data leakage, data loss, account hijacking and insecure APIs. The externalized aspect of outsourcing can make it harder to maintain data integrity and privacy (IBM, 2011) and organizations should include mechanisms to mitigate security risks introduced by virtualization. Especially when they deal with sensitive data, such as health records, the protection of stored information comes as a top priority. Therefore, data security can be seen as the foundation upon which the entire transition to a cloud architecture should be based. Multiple risks must be addressed in order for an organization to guarantee the safety of users' records. One of the most important aspects is security of sensitive information. To this end, the deployment must ensure that all sensitive data is stored in encrypted form. Complementary to this, proper key management must ensure that encryption keys are not revealed to malicious users.

Based on this, it becomes evident that the most critical part of a modern cloud application is the data persistency layer and the database itself. As all sensitive information (including user credentials, credit card info, personal data, corporate data, etc.) are stored in these architectural parts, the database-takeover is the ultimate goal for every adversary.

The Open Web Application Security Project[1] foundation has categorized the database-related attacks (SQL injection) as the most critical ones. The importance of this attack vector is also reflected by respective incident reports. According to the Web Hacking Incidents Database [2], SQL injections represents 17% of all security breaches examined. These injections were responsible for 83% of the total records stolen, in successful hacking-related data breaches from 2005 to 2011. The criticality of the persistency layer is therefore evident. Most of the security fences that are configured in a corporate environment target the fortification of the so-called network perimeter (e.g. routers, hosts and virtual machines). Although existing intrusion detection systems (IDS) and intrusion prevention systems (IPS), try to cope with database-takeover security aspects (like Snort), the fact that, on the one side, automated exploitation tools (e.g. SQLMap) are widely spread, and, on the other side, IPS and IDS evasion techniques have become extremely sophisticated, denote that the risk of database compromise is greatest than ever. Moreover, by using mechanisms that rely on Web Application Firewalls (WAF) an organization can prevent various types of attacks but it is inadequate to protect against todays sophisticated SQL Injection and DoS attacks (Michalas et al., 2010). Additionally, internal adversaries in terms of cloud vendors or even unknown vulnerabilities of software platforms and security components widely adopted in cloud-based development may provide malicious access to personal and sensitive data. A recent example was the Heartbleed flaw[3] that constituted a serious fault in the OpenSSL cryptography library, which remained unnoticed for

[1]https://www.owasp.org/

[2]http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

[3]http://www.infosecurity-magazine.com/news/heartbleed-101/

more than two years and affected over 60% of Web servers worldwide. Additionally, regarding the post-exploitation phase, things are even worse in the case where a symmetric encryption algorithm has been employed to protect the application data. The already available cracking toolkits that utilize GPU processing power (e.g. oclHashcat) are able to crack ciphers using brute-force techniques with an attack rate of 162 billion attempts per second.

While most of the attack vectors are exposed in any Software-as-a-Service application by the system administrators misconfigurations, the database takeover and the post-exploitation of acquired data is under the sole responsibility of the application developer. The application developer is the one responsible both for sanitizing all HTTP-input parameters that could be used as attack vectors, and for reassuring that compromised data will be useless under the existing brute-forcing and reversing techniques. Nevertheless, even if the application developer follows strict guidelines, the mere utilization of an IaaS provider in order to host a Virtual Machine, or for a Platform-as-a-Service (PaaS) provider in order to develop a cloud application, may by itself spawn a multitude of inherent vulnerabilities. These vulnerabilities cannot be tackled effectively as they typically exceed the responsibilities of an application developer.

3 ENVISIONED FRAMEWORK

In this section, we present PaaSword, a framework that will allow cloud services to maintain a fully distributed and encrypted data persistence layer in order to foster data protection, integrity and confidentiality in the presence of malicious adversaries. To this end, we describe the need for a context-aware security model which will serve as the basis of a fine-grained access control scheme, one which allows the per-user management of access rights. In addition to that, we describe a physical distribution, encryption and query middleware that will be based on a searchable encryption (SE) scheme which will allow legitimate users to directly search on encrypted data, thus ensuring the confidentiality and integrity of stored data.

3.1 Context-aware Access Model

We envision a XACML-based[4] context-aware access model, which is needed by the developers in order to annotate the Data Access Objects of their applications. This context model should conceptualize the aspects, which must be considered during the selection of a data-access policy. These aspects may be any kind of information which is machine-parsable (Dey, 2001); indicatively they may include the user's IP address and location, the type of device that she is using in order to interact with the application as well as her position in the company. These aspects can be interpreted in different ways during the security policy enforcement. In particular, the context aware access model determines which data is accessible under which circumstances by an already-authenticated user.

Access control models are responsible for deciding if a user has the right to execute a certain operation on a specific object. Objects can be a server, an application, an entire database or even a single field in a table row. The user is considered as the active element and is called subject. A permission associates an object with an operation (e.g. read, write etc.). Access control models provide a list of permissions that each subject has on certain objects.

Commonly used access control models are the Mandatory Access Control (MAC), the Discretionary Access Control (DAC) and the Role-Based Access Control (RBAC) (Ferrari, 2010). In our approach, the process of granting/denying access will be based on dynamically changing parameters, thus our proposed model relies on a DAC model with groups. The context parameters are unique for every single user, so for granting access it is necessary to consider all information associated with a single user. Furthermore, an RBAC model would be inappropriate since for every change of a context parameter the role of each subject has to be changed.

To implement the dynamic change of context parameters in a static access control model, we will use the, so-called, context switches. Depending on the current context, a permission can be granted or denied (switched). This could switch dynamically with every change of the context. Context switches are responsible for managing operational permissions and object permissions. An operational permission gives the right to a subject to perform a specific operation while an object permission gives the right to perform an operation on a specific object.

3.2 Policies Access and Enforcement

Another important aspect of our proposed framework is a middleware that will encapsulate capabilities for maintaining the access policies model, for annotating and managing data access object annotations, for controlling their validity, for dynamically interpreting them into policy enforcement rules and for enforc-

[4]OASIS eXtensible Access Control Markup Language (XACML). https://www.oasis-open.org/

ing them. This envisaged middleware will provide: *(a)* a transparent key usage for efficient authentication purposes, related to authenticating the origin of the incoming access requests; *(b)* annotation capabilities in the form of a tool (can also involve an IDE plug-in) for allowing developers to declaratively create the minimum amount of rule-set that is needed for security enforcement purposes; *(c)* the dynamic interpretation of the data access object annotations into policy enforcement rules; *(d)* the governance and quality control of the annotations and their respective policy rules; and *(e)* the formulation and implementation of the overall policy enforcement business logic.

In terms of this middleware, we also consider the reuse and proper extension of technologies for developing an appropriate key management mechanism. This mechanism is necessary for the authentication of different parties that will be involved in the encryption and decryption of data. We aim at constituting the key-usage, transparent to the application usage. This involves the key propagation upon authentication of the user, directly to the security enforcement middleware. For efficiency, we will employ a hybrid encryption capitalizing upon the utilization of two different encryption functions. The inner layer will be encrypted with an algorithm that uses a symmetric encryption key K, while the outer layer will use an asymmetric encryption in order to encrypt the symmetric key K. Symmetric encryption allows more efficient schemes but privacy concerns are raised due to the fact that the involved parties must exchange the secret key. However, combining both techniques help to optimize the efficiency of the underlying protocols without sacrificing security. To this end, PaaSword will rely on both symmetric and asymmetric encryption in order to securely distribute K between legitimate users.

Additionally, we will also employ methods and mechanisms for governance and validity control of the data object annotations. More specifically, we will focus on the application of an ontology-driven governance approach for: *i)* the basic management of data object annotations (i.e. storage, retrieval, deletion, etc.), *ii)* validity checking of the data object annotations (e.g. rejecting any contradicting annotations made by the developer) and *iii)* dependency tracking among data objects annotations.

Another critical aspect of this middleware is the annotations interpretation mechanism. Such a mechanism will be used for dynamically generating access control policies, during application runtime, based on the interpretation of data object annotations. Such a mechanism will implement the essential decoupling between the access decisions and the points of use (i.e. Policy Enforcement Points (PEP) of the XACML specification). This interpretation is based on an XACML compliant context model and it can augment the offered functionality of any PaaS provider, with a security-as-a-service layer. To do so, we will use the OASIS XACML as it supports and encourages the separation of the access decision from the point of use.

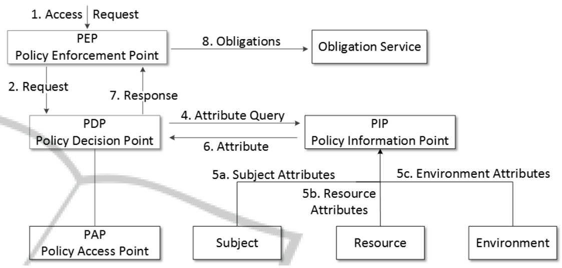

Figure 1: High level view of XACML Components.

3.3 Threat Model, Secure Storage & Query Middleware

In this sub-section, we provide a high level description of the protocol that will be used to effectively protect the stored data from malicious adversaries. To this end, we first describe the threat model under which a cloud application will be considered secure.

Threat Model. Similar to existing works in the area (Paladi et al., 2014; Santos et al., 2009), we assume a *semi-honest* cloud provider. In the semi-honest adversarial model, a malicious cloud provider correctly follows the protocol specification. However, she can intercept all messages and may attempt to use them in order to learn information that otherwise should remain private. Semi-honest adversaries are also called *honest-but-curious*.

Furthermore, for the rest of the participants in the protocol we share the threat model with (Santos et al., 2009), which is based on the Dolev-Yao adversarial model (Dolev and Yao, 1983) and further assumes that privileged access rights can be used by a remote adversary $\mathcal{ADV}$ to leak confidential information. The adversary, e.g. a corrupted system administrator, can obtain remote access to any host maintained by the provider. However, the adversary cannot access the volatile memory of any guest virtual machine (VM) residing on the compute hosts of the provider. This property is based on the closed-box execution environment for guest VMs, as outlined in Terra (Garfinkel et al., 2003) and further developed in (Zhang et al., 2011).

Secure Storage. A basic tenet of PaaSword is that sensitive data stored on untrusted servers must be always encrypted. This effectively reduces the privacy and security risks since it relies on the semantic security of the underlying cryptosystem, rendering the system relatively immune to internal and external attacks. Having this in mind, we propose a forward-looking design for a cryptographic cloud storage that will be based on a symmetric searchable encryption (SSE) scheme similar to the one proposed in (Kamara and Lauter, 2010). We plan to extend the previous work Cumulus4j (Huber et al., 2013) and MimoSecco (Gabel and Hübsch, 2014) in which an SSE scheme was presented and it was based on the IND-ICP security notion (Bösch et al., 2014) that hides relations between different data values of a data row and creates the base for secure database outsourcing.

An SSE scheme allows a user to search in encrypted data without learning any information about the plaintext data. Let $\mathcal{DB} = \{m_1, \ldots, m_n\}$ be a set of n messages (w.l.o.g $\mathcal{DB}$ can be considered as a database). For each $m_i \in \mathcal{DB}$ we extract a set of keywords which can later be used for executing queries. This set of keywords is denoted as $\mathcal{W} = \{w_1, \ldots, w_n\}$. For each $w_i \in \mathcal{W}$ we calculate $H(w_i)$, where $H(\cdot)$ is a cryptographically secure hash function under a secret key K'. Then, we encrypt the elements of $\mathcal{DB}$ with a secret key $\mathsf{K}'' \neq \mathsf{K}'$. By doing this, we create a searchable encrypted index I where each index entry, points to an encrypted list of rows that have a certain keyword. The client can use a trapdoor function to search the index and determine whether a specific keyword is contained in the index.

While the above-mentioned scheme is implemented in previous works (Huber et al., 2013; Gabel and Hübsch, 2014) it has a limitation that we tend to cover in our proposed framework. More precisely, the current scheme follows a single write/single read (S/S) architecture, which makes it unrealistic for our cloud scenario. To overcome this limitation, we plan to build an SSE that will support multi write/multi read (M/M) meaning that a group of users based on access rights will be able to both read and write on the encrypted data. To this end, PaaSword will involve a key distribution algorithm that will extend S/S architecture to M/M. Additionally, a user revocation function will be implemented in order to exclude a user, which either acts maliciously or has no longer access rights. This is a crucial and challenging procedure, if we consider that many of the existing SSE schemes (Bösch et al., 2014) do not support user revocation and thus are susceptible to many attacks.

Query Middleware. In order to successfully support the SSE scheme described above, we aim to develop a persistency layer, called Virtual Database VB (Figure 2), and will be the intermediary that secures client data before it gets uploaded to the cloud. Additionally, this layer will be responsible for processing user queries. In our framework, the VB plays the role of a trusted third party. Consider, for example, the scenario where a user wants to search for a certain data in PaaSword secured databases. To do so, she will generate a query (q) containing a set of keywords that she is interested in and will send the request to the VB. Upon reception, the VB extracts the keywords from q calculates their hash values and queries the databases where the keywords w_i are stored. If the queries are successful and the keywords exist in one of the tables, the VB will obtain the row from the main table that contains the encrypted original data. Upon reception, the VB will reply to the users request by sending the acquired data.

3.4 Conceptual Architecture

The PaaSword compliant cloud applications that will be developed will inherit a fully physical distributed and totally encrypted data persistence layer, which will be able to determine on an ad-hoc basis whether an incoming data querying and processing request should be granted access to the target data during application runtime. The transformation process of a traditional application utilizing the PaaSword framework and the way the transformed application secures and protects the users' sensitive data is presented in Figure 2, which at the same time reveals high level architectural details of the framework.

In this framework, we consider applications that adopt and respect the Model-View-Controller (MVC) development pattern (Krasner and Pope, 1988). As seen in Figure 2 (step 1) the application developer imports an existing or creates a new MVC-based application in her favorite integrated development environment (IDE) for which an IDE-specific plug-in will be provided. During the second step of this process the application developer creates annotations at the DAO of the Controller referring to sensitive data that should be protected, according to the XACML-based model and defines the physical distribution, encryption and access rights scheme for each data object. In the third step, the DAO annotations will be checked for their validity and compiled with the overall application code. This will allow the transformation of the application's controller that has been enhanced with XACML-based DAO annotations, leading to the implementation of a PaaSword secure application. In the fourth step, the persistence layer of the application will be physically distributed and encrypted at the schema and instance level according to the incorporated DAO annotations, imposing the schema and driving query handling capabilities of the VB that augments the actual data persistence layer of the application. At application runtime (step 5), each query and processing request of the end-user is forwarded by the enhanced controller to the query handling mechanism that is responsible for the database proxy queries synthesis and aposynthesis. In step 6 and before the submission of the enhanced query to the VB, the query handling mechanism consults the policy enforcement mechanism to determine whether the incoming request should be granted or not. Upon policies enforcement and access permission, the query handling mechanism submits (step 7) the enhanced query to the augmented persistence layer (virtual database). The database proxy that is aware of the physical distribution scheme of the actual application database realizes the distributed query to the physically distributed and encrypted parts of the actual application database (step 8). Next, the federation of the respective encrypted data from the distributed parts of the database takes place (step 9). The federated data synthesis and ad-hoc decryption utilizing the key of the end-user that is transparently to the application, propagated to the query handling mechanism (step 10). Last, the query handling mechanism delivers the decrypted data to the application controller that forwards them to the end-user through the "view" component of the application.

According to the conceptual view (Figure 2), each end-user is equipped with a Hardware Security Module, such as USB stick or a smart-phone with digital rights management module, which contains a digital certificate (e.g. X.509). Part of the certificate includes keys that can be exported by the PaaS/IaaS provider. These keys upon export and verification will be transparently handed over to the query middleware which will be responsible for interacting with the VB, encrypting and decrypting the targeted data.

4 RELATED WORK

In an attempt to reinforce the security of remote service accesses, researchers introduced the concept of location-aware access control (LAAC), which allows a system to grant, or deny, access to users based on their physical location. LAAC models typically extend the three basic access control models DAC, MAC and RBAC (Decker, 2011). Even though LAAC protocols have been studied extensively (Cleeff et al., 2010), there is a clear lack of schemes that determine user access not only on the basis of the users physical location and credentials, but also on the additional pertinent contextual information.

The work reported in (Covington et al., 2001) was the first to introduce the notion of context-aware access control (CAAC), motivated by applications for intelligent homes. More precisely, the authors introduced a set of services which are enabled based on the location of objects or subjects. The main drawback of the proposed model is the fact that it does not support dynamically generated context, whilst it fails to address important requirements such as multi-granularity of position. Other existing CAAC models are predominantly based on RBAC (Kayes et al., 2013) and typically target a specific domain (Costabello et al., 2012).These models, however, have not been designed to provide fine-grained data access control, e.g. by providing the ability to specify different access rules for different rows of a database.

Regarding the policy management, as shown by a recent survey of methods in contemporary open source registry and repository systems (Kourtesis and Paraskakis, 2012), a major weakness is the lack of proper separation of concerns. The policy definition and policy enforcement are entangled in the implementation of a single software component – the policy checker. The rules that a policy comprises are typically encoded in an imperative manner, as part of the same code that checks for potential policy violations. This has a number of negative repercussions among which is the lack of portability and the lack of explicit representation of policy relationships.

The data distribution and encryption algorithms are also important aspects towards trusted cloud services and applications. In (Gentry, 2009), C. Gentry presented the first fully homomorphic encryption scheme that enables semantically secure outsourcing to the cloud. The cloud provider operates blindly on the encrypted data and yields the correct, encrypted result. Nevertheless, its practicality is in question as the latest implementations are still orders of magnitude slower than just downloading all encrypted data, decrypting, processing and encrypting it locally and finally uploading it again. In another interesting approach (Popa et al., 2011), the concept of onions is used. Onions are managed monolithically by a proxy, acting as an adapter between the user and the storage back-end. Each attribute in a relational table is initially asymmetrically encrypted. If certain queries for an attribute are issued, layers of the onion are peeled off, resulting in another, less secure onion. CryptDB uses a novel scheme for order preserving encryption that leaks no information about the data besides or-

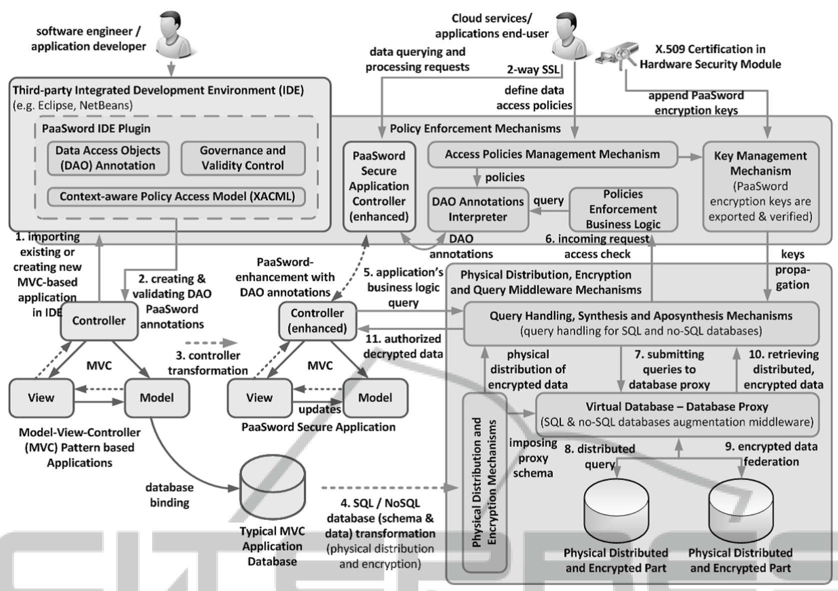
software engineer / application developer
Third-party Integrated Development Environment (IDE) (e.g. Eclipse, NetBeans)
PaaSword IDE Plugin
Data Access Objects (DAO) Annotation
Governance and Validity Control
Context-aware Policy Access Model (XACML)
1. importing existing or creating new MVC-based application in IDE
2. creating & validating DAO PaaSword annotations
Controller
MVC
View
Model
Model-View-Controller (MVC) Pattern based Applications
3. controller transformation
PaaSword-enhancement with DAO annotations
Controller (enhanced)
updates
PaaSword Secure Application
database binding
Typical MVC Application Database
4. SQL / NoSQL database (schema & data) transformation (physical distribution and encryption)
Cloud services/ applications end-user
data querying and processing requests
2-way SSL
define data access policies
X.509 Certification in Hardware Security Module
append PaaSword encryption keys
Policy Enforcement Mechanisms
PaaSword Secure Application Controller (enhanced)
Access Policies Management Mechanism
policies
DAO Annotations Interpreter
query
Policies Enforcement Business Logic
Key Management Mechanism (PaaSword encryption keys are exported & verified)
DAO annotations
6. incoming request access check
keys propagation
5. application's business logic query
11. authorized decrypted data
Physical Distribution, Encryption and Query Middleware Mechanisms
Query Handling, Synthesis and Aposynthesis Mechanisms (query handling for SQL and no-SQL databases)
physical distribution of encrypted data
7. submitting queries to database proxy
10. retrieving distributed, encrypted data
Virtual Database – Database Proxy (SQL & no-SQL databases augmentation middleware)
Physical Distribution and Encryption Mechanisms
imposing proxy schema
8. distributed query
9. encrypted data federation
Physical Distributed and Encrypted Part
Physical Distributed and Encrypted Part

Figure 2: PaaSword Framework Conceptual Architecture.

der and thus allows sorting encrypted data securely. The main drawback of CryptDB is the lack of security guarantees to the client. More precisely, the only guarantee is that an untrusted server will learn only the information that is necessary to process the query. This may cause every attribute to be reduced to the plain text in the worst case. Also, peeling off layers cannot be reversed, so a single query is sufficient to lower the security forever.

5 CONCLUSIONS

In this position paper, we proposed the PaaSword framework that can be exposed as a service at the level of PaaS. This framework can tackle the identified cloud security requirements and challenges that should be considered in order to enhance data protection, integrity and confidentiality in the presence of malicious adversaries. The envisaged PaaSword goes beyond the state-of-the-art and allows cloud services to maintain a fully distributed and encrypted data persistence layer. Our framework involves a context-aware security model, the necessary policies enforcement mechanism along with a physical distribution, encryption and query middleware.

Future work involves the design and implementation of the proposed framework into a fully functional solution which will be validated through the following five pilots in various industrial contexts: *i)* Encrypted persistency as a service in a PaaS provider, *ii)* Intergovernmental secure document and personal data exchange, *iii)* Secure sensors data fusion and analytics, *iv)* Protection of personal data in a multi-tenant CRM, *v)* Protection of sensible enterprise information in multi-tenant ERP. These pilots will allow us to test PaaSword and validate its added value in a variety of heterogeneous cases.

Finally, an area that will benefit from PaaSword framework is the so called participatory sensing (Michalas and Komninos, 2014). The evolution of this field is driven by the introduction of sensors into mobile devices. The openness of such systems and the richness of user data they entail raise significant concerns for their storage and processing. Protocol designers by having PaaSword framework in hands will be able to incorporate secure cloud computing techniques in order to facilitate the storage and processing of the vast amount of collected data.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644814, the PaaSword project (www.paasword.eu) within the ICT Programme ICT-07-2014: Advanced Cloud Infrastructures and Services.

REFERENCES

Alliance, C. S. (2013). The notorious nine – cloud computing top threats in 2013.

Bösch, C., Hartel, P., Jonker, W., and Peter, A. (2014). A survey of provably secure searchable encryption. *ACM Comput. Surv.*, 47(2):18:1–18:51.

Cleeff, A. v., Pieters, W., and Wieringa, R. (2010). Benefits of location-based access control: A literature study. In *Proceedings of the 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM '10, pages 739–746, Washington, DC, USA. IEEE Computer Society.

Costabello, L., Villata, S., and Gandon, F. (2012). Context-aware access control for rdf graph stores. In Raedt, L. D., Bessire, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P. J. F., editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 282–287. IOS Press.

Covington, M. J., Long, W., Srinivasan, S., Dev, A. K., Ahamad, M., and Abowd, G. D. (2001). Securing context-aware applications using environment roles. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, SACMAT '01, pages 10–20, New York, NY, USA. ACM.

Decker, M. (2011). Modelling of location-aware access control rules. In *Handbook of Research on Mobility and Computing: Evolving Technologies and Ubiquitous Impacts*, pages 912–929. IGI Global.

Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7.

Dolev, D. and Yao, A. C. (1983). On the security of public key protocols. *Information Theory, IEEE Transactions*, 29(2):198–208.

Ferrari, E. (2010). *Access Control in Data Management Systems*. Morgan and Claypool Publishers.

Gabel, M. and Hübsch, G. (2014). Secure database outsourcing to the cloud using the mimosecco middleware. In Krcmar, H., Reussner, R., and Rumpe, B., editors, *Trusted Cloud Computing*, pages 187–202. Springer International Publishing.

Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., and Boneh, D. (2003). Terra: A virtual machine-based platform for trusted computing. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 193–206.

Gentry, C. (2009). *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA. AAI3382729.

Huber, M., Gabel, M., Schulze, M., and Bieber, A. (2013). Cumulus4j: A provably secure database abstraction layer. In Cuzzocrea, A., Kittl, C., Simos, D. E., Weippl, E., Xu, L., Cuzzocrea, A., Kittl, C., Simos, D. E., Weippl, E., and Xu, L., editors, *CD-ARES Workshops*, volume 8128 of *Lecture Notes in Computer Science*, pages 180–193. Springer.

IBM (2011). Security and high availability in cloud computing environments. Technical report, IBM SmartCloud Enterprise, East Lansing, Michigan.

Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage. In Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J., Sako, K., and Seb, F., editors, *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer Berlin Heidelberg.

Kayes, A. S. M., Han, J., and Colman, A. (2013). An ontology-based approach to context-aware access control for software services. In Lin, X., Manolopoulos, Y., Srivastava, D., and Huang, G., editors, *WISE (1)*, volume 8180 of *Lecture Notes in Computer Science*, pages 410–420. Springer.

Kourtesis, D. and Paraskakis, I. (2012). A registry and repository system supporting cloud application platform governance. In *Proceedings of the 2011 International Conference on Service-Oriented Computing*, ICSOC'11, pages 255–256, Berlin, Heidelberg. Springer-Verlag.

Krasner, G. E. and Pope, S. T. (1988). A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49.

Michalas, A. and Komninos, N. (2014). The lord of the sense: A privacy preserving reputation system for participatory sensing applications. In *Computers and Communication (ISCC), 2014 IEEE Symposium*, pages 1–6. IEEE.

Michalas, A., Komninos, N., Prasad, N. R., and Oleshchuk, V. A. (2010). New client puzzle approach for dos resistance in ad hoc networks. In *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference*, pages 568–573. IEEE.

Michalas, A., Paladi, N., and Gehrmann, C. (2014). Security aspects of e-health systems migration to the cloud. In *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pages 212–218. IEEE.

Micro, T. (2010). The need for cloud computing security. In *A Trend Micro White Paper*.

Paladi, N. and Michalas, A. (2014). "One of our hosts in another country": Challenges of data geolocation in cloud storage. In *Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE), 2014 4th International Conference on*, pages 1–6.

Paladi, N., Michalas, A., and Gehrmann, C. (2014). Domain based storage protection with secure access control for the cloud. In *Proceedings of the 2014 International Workshop on Security in Cloud Computing*, ASIACCS '14, New York, NY, USA. ACM.

Popa, R. A., Redfield, C. M. S., Zeldovich, N., and Balakrishnan, H. (2011). Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA. ACM.

Santos, N., Gummadi, K. P., and Rodrigues, R. (2009). Towards trusted cloud computing. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA. USENIX.

Zhang, F., Chen, J., Chen, H., and Zang, B. (2011). Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 203–216. ACM.