Hybrid Cloud Architecture for Software-as-a-Service Provider to Achieve Higher Privacy and Decrease Security Concerns about Cloud Computing

Paul Reinhold and Wolfgang Benn
Chemnitz University of Technology
Chemnitz, Germany
Email: {paul.reinhold@s2012,
wolfgang.benn@informatik}
.tu-chemnitz.de

Benjamin Krause and Frank Goetz
Qualitype GmbH
Quality Management Systems
Dresden, Germany
Email: {b.krause,f.goetz}@qualitype.de

Dirk Labudde
Hochschule Mittweida
University of Applied Sciences
Mittweida, Germany
Email: dirk.labudde@hs-mittweida.de

***Abstract*—Security and privacy concerns are still major issues during the adoption of cloud services. Software service developers face new challenges to solve this problem. To establish the software-as-a-service and to address privacy and security concerns of customers, providers can use the suggested hybrid cloud architecture to outsource the data persistence layer. By this approach, neither customer nor provider have to trust an arbitrary public cloud service provider. The approach offers a tradeoff between higher privacy and security for less flexibility and scalability in consideration of costs and therefore its application in practice. Test results of the implemented prototype demonstrate the practical suitability, but show the limitations as well. Besides focusing on functional requirements like privacy and scalability, also non-functional demands such as independency from special software or hardware needs and the minimal migration effort, have been considered.**

Keywords-Hybrid Cloud; Privacy; Cloud Security; Architecture; Software-as-a-Service; Key Management

I. INTRODUCTION

The increasing knowledge about cloud computing technology and its publicity leads to a growing number of service offerings over the Internet. Even small and medium sized companies are able to offer services for a large number of consumers through cloud computing concepts. Resources can be obtained easily from public cloud providers like Amazon Web Services [1], without limits regarding scaling and flexibility. Instagram is a typical example for modern cloud based application services [2]. Many of these cloud services are provided to the end users via the Internet in a form of Software-as-a-service (SaaS). SaaS can be understood as web or mobile applications with variable degree of complexity consumer can use on demand. For further information see [3].

The high acceptance of these services suggests that private consumers have a lower privacy demand than business users. The study of [4] indicated that in Europe, and especially in Germany the acceptance of public cloud services for business purposes is low. Typical reasons are security and privacy concerns. Companies do not want their critical business documents or customer data they manage in public clouds. In addition, the study showed that experiences with private cloud computing are, with 83%, mainly positive. It has to be pointed out that the size of the enterprise has a great influence on its experience with cloud computing. For instance, 60% of large companies already have private cloud experience, while small and medium-sized enterprises (SME) are more reserved.

Out of these concerns, we stress aspects of how a provider can develop and run a SaaS in which SaaS consumers get their privacy needs satisfied. Consumers might gain a higher acceptance to cloudbased SaaS so that cloud computing gets more attractive to SME-SaaS providers. Therefore, we propose a hybrid cloud architecture enhanced with an additional architecture layer between business logic and persistence layer. It has a minimal migration effort and reveals no information, except for meta data, about the outsourced data to the public cloud provider. For evaluation purposes, we implement a prototype using the suggested architecture to securely outsource unstructured (files) and structured data (databases) in a public cloud. The results show that our suggested architecture is very practical and an efficient/effective way for SaaS providers to use some of the advantages of public clouds.

The outline for the rest of the paper is as follows. Section II discusses a comparison of three common cloud delivery models with respect to privacy, costs and performance. In Section III, we describe the proposed hybrid cloud architecture. Section IV describes the implemented prototype, performance tests and resulting overheads. Section V provides a critical discussion of the results and gives an overview for future work. Section VI concludes the paper.

II. CLOUD MODEL COMPARISON

Our work focuses on SME-SaaS providers, which already run SaaS offerings or web applications and take new cloud offerings into account to become more cost-efficient. In addition, SaaS providers probably use own hardware to run their SaaS and plan to develop a new version or new service, which exceeds the current limit of their hardware. Another scenario could be that the provider needs to invest in new hardware to keep its services running and is looking for lower cost alternatives.

The end-user of a cloud service shall be named cloud consumer or simply consumer [5]. In the consumers view, the provider offers SaaS over the Internet. Whether the service offered by provider's hardware or by third party resources is irrelevant for the customer, as long as service supply is ensured. However,

 ISBN: 978-1-61208-338-4

TABLE I. COMPARISON OF DIFFERENT CLOUD MODELS FROM CONSUMERS AND PROVIDERS POINT OF VIEW

View	Criteria	Private Cloud	Public Cloud	Hybrid Cloud
consumer	cost	high	low	medium
	privacy	medium	low	high
	data-at-rest encryption	yes	yes	yes
	key owner	provider	provider	consumer (and provider)
	key management	provider	provider	provider
	compliance	medium	low	medium - high
	governance	medium	low	medium - high
provider	cost	high	low	medium
	availability	medium	very high	high
	backup	medium	very high	very high
	hardware needs	high	very low	medium
	effort to run service	very high	low	medium - high
	achieve cost-efficiency	very difficult	easy	possible
	flexibility	high, but limited	very high	higher, but limited
	scaling	yes, but limited	yes	yes, but limited

the method of providing can be essential for the acceptance of the SaaS on consumers side.

According to the common cloud delivery model by Mell and Grance [6], the provider can run the SaaS in a private, public or hybrid cloud. In the private cloud, the provider runs its own cloud. He owns the hardware and has exclusive access to it. For cost-efficiency reasons the provider runs its hardware in form of a cloud, allowing flexibility and scaling effects. The public cloud is the most flexible and cost-efficient service, since the provider obtains resources and pays by use. The hybrid cloud is the third approach were the provider runs the service both in a private and public cloud.

Table I shows a comparison between the three cloud service models from both consumer's and provider's point of view. The costs factor is comprehensible and hardware expenses are usually passed to consumers. Privacy is low for public and medium for private cloud architecture. Private clouds often have strong authorization and access control concepts, but no special requirement to secure data with encryption against the provider itself [7]. Thus, the cloud provider often has access to customer data and their customers data, respectively. In a public cloud it is very costly or impractical to secure data and to keep them available for processing at the same time, like fully homomorphic encryption [8] can provide.

Another important aspect for consumers is data-at-rest encryption. Encryption is possible in all of the models, but strongly connected with key ownership and management. If the same instance encrypts data and stores the referring key, no trustable security can be guaranteed, because providers can encrypt data without users knowledge or permission. This has been recently documented for economically rational cloud providers [9]. Because of this circumstance the hybrid cloud model suggests a solution where consumers get more control over their data and the possibility for public cloud providers to access unencrypted files is eliminated. Compliance and governance depend directly on this solution.

If consumers care about their data security and privacy, the hybrid cloud model supposed to meet the needs best. Even if a consumer trusts its provider (with a private cloud) and consequently encryption is not needed, the hybrid solution is more economical.

A private cloud cannot provide the high availability, that is guaranteed by a public cloud. The hybrid model benefits from this fact by outsourcing parts of the architecture in a public cloud. Backup security underlays the same principle, in fact the backup process in hybrid model can be outsourced completely. The hardware needs and the effort to run the service are coherent. Lots of own hardware means not only to manage, but also to maintain and have environment settings (buildings, redundant broadband internet access) to run a private cloud. Because of this, it is much easier to create a cost-efficient SaaS in a public cloud than in a private cloud. The great advantages of cloud computing like flexibility and scaling are limited in private and hybrid cloud solutions. As a result of this comparison and questions, our aim is to combine the security of a private cloud with the flexibility, reliability and availability of a public cloud, creating a balanced solution. The hybrid approach offers a trade-off between increased security for decreased efficiency. We want to know if its worth doing this trade-off. In addition, questions we want to answer are:

- Is it possible to set up a practical solution, which does not reveal any information of the data, except for metadata, like size or structure?
- Is a support for both, unstructured data such as files and structured data such as databases, possible?
- Is it possible to do so with very low migration effort, to keep the acceptance high?

III. HYBRID CLOUD ARCHITECTURE

The here proposed privacy-enhanced hybrid cloud architecture is illustrated in Figure 1. It applies typical security concepts in the cloud computing field using tier, logic and data partitioning described by Bohli et al.[10]. In contrast to the study of [10], we do not spread our tiers to various, non-collaborating cloud providers. As mentioned before, our approach is spread over a private and public cloud, performing all critical tasks in the private cloud and outsource only the data tier in public cloud. This makes it unnecessary to label tasks or data as critical in a manual or semi-automatic manner, like described by Zhang et al. [11]. The Figure 1 shows the data flow from the consumer via the SaaS provider to the public cloud. For simplification just ingoing traffic is displayed. The consumer uses a computer with Internet connection to access the SaaS (1). In addition, the consumer has a master key for encryption purposes. The initial login and identification

 ISBN: 978-1-61208-338-4

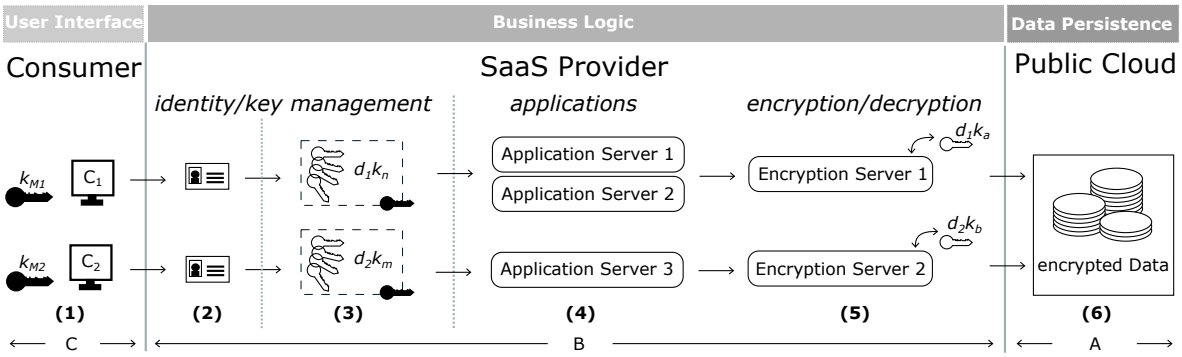

Figure 1. The hybrid cloud architecture for SaaS with an enhanced business logic layer by key management and encryption layer. Lines separate the actors and architecture layers. Dashed lines illustrate encrypted keys. Dotted lines represent a physical separation. Rounded and rectangular boxes represent virtual servers and data, respectively. A,B and C show the range of the thread scenarios.

procedure should be located on physically separated hardware or be outsourced to a trusted ID verification provider [12] (2). The key management system, storing encryption keys, is another security critical resource and should not be integrated in the private cloud (3). The private cloud structures contain the application server layer (4) and the encryption server layer (5). All communication pass these tiers, so they have to be highly scalable. The public cloud provider is illustrated in form of a persistence layer (6). The outgoing data flow differs slightly. The consumer's request is received directly by an application server, which asks on its part for the data. This request received by the encryption servers, sending a key request to the key management system. The resulting request to the cloud is encrypted by the encryption server with the received data key. The received data from the public cloud is decrypted with the same data key and afterwards send to the application server, that passes the data to the consumer.

A. Key Concept

The key concept is shown in Table II. Besides the hybrid cloud architecture, a hybrid encryption concept is used to provide a better performance. The master key is persisted by consumer and used to encrypt and decrypt the data keys. The data keys are persisted by the provider and used to encrypt and decrypt consumer's data. The transfer key pairs are generated during the customers registration and used for secure exchange of a temporary copy of the consumers' master key.

TABLE II. OVERVIEW OF EXISTING ENCRYPTION KEYS IN THE PROPOSED KEY CONCEPT.

Key	Type	Owner
master key k_M	symmetric	Consumer
data key $d_x k_y$	symmetric	SaaS Provider
transfer key k_T	asymmetric	Consumer/Provider

B. Security Overview

To evaluate the security of the architecture,we examined/constructed three threat scenarios that are indicated with A, B and C in Figure 1. All data traffic from the consumer to the SaaS provider and *vice versa* is encrypted through standards like Transport Layer Security (TLS) [13]. The consumer authenticates against the SaaS provider by multi-factor or strong authentication. After a successful login, the consumer allows the provider to decrypt the data keys $d_x k_y$ with its master key k_M. The data keys allow decrypting the data stored in the public cloud. The master key is solely persisted by the user. This prevents the SaaS provider to decrypt data from not logged in consumers. As a result, the consumer gains high control over the data. The public cloud provider only stores encrypted files and never gets access to encryption keys. Neither the consumer nor SaaS provider must trust the public cloud provider in this scenario.

1) Threat Scenario A: This scenario describes an attack against the public cloud provider. Even if the attackers have full (or physical) access to resources of the cloud provider, all data of the consumer are secure. The data stored by the public cloud will never be in plaintext, so the privacy and confidentiality of the data is guaranteed. This requires an appropriate encryption by the SaaS provider and careful handling of the corresponding encryption keys. Depending on the encryption techniques the SaaS provider uses, the public cloud provider can obtain meta information about the stored data such as size, structure or access pattern. In the simplest case, only data confidence can be provided by the SaaS. To secure integrity and availability, the SaaS provider can mirror and distribute files over different non-cooperation public clouds. The higher level of security results in higher costs for the SaaS provider and therefore for the consumer. So, this scenario gives in part of information revealing, a positive answer to the first question in Section II.

2) Threat Scenario B: Threat scenario B describes the offered security if the SaaS provider is attacked. The attacker gets no access to consumer data, but to the data keys of active consumers. Accessing data of the inactive user is impossible, even if the attacker has full access. The SaaS provider does not persist the consumers master key; therefore, it's no possibility for the attacker to decrypt the data keys. The security of logged out consumers is still guaranteed.

3) Threat Scenario C: Threat scenario C describes an attack against the consumer. If the attacker obtains the login credentials, factors and the master key, he can get full access to the consumer's data. To prevent the attacker getting easy access to other consumer's data, the SaaS provider should be multi-tenancy capable. In detail, application servers of different consumers should be at least virtually separated.

 ISBN: 978-1-61208-338-4

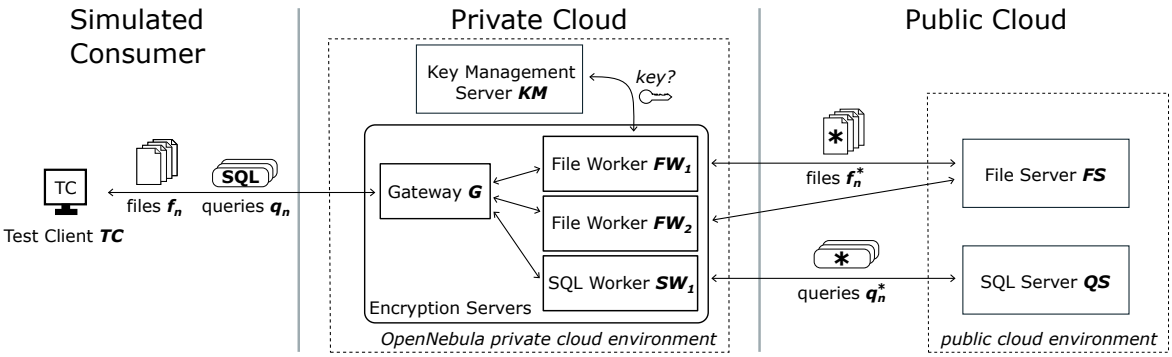

Figure 2. The implemented encryption server prototype as one core component of the proposed hybrid cloud architecture.

IV. Implemented Prototype

The developed prototype implements the encryption servers as one core component of the hybrid architecture. It is run in a private cloud environment based on OpenNebula 4.4 [14] powered by four physical hosts with 3 GHz Dual-Cores and 8 GB RAM. These machines consist of standard components to keep the hardware costs low.

Gateway G, File Worker FW_i, SQL Worker SQ_i and Key Management KM are virtual machines (VMs) and are part of an autoscaling service called *OneFlow*[15]. Figure 2 illustrates a minimal test setup with one gateway, three worker and the key management VMs. Load balancing is performed by the JBoss *mod_cluster 1.2.6* [16]. Therefore, G is a mod_cluster enabled Apache http-Server [17]. The implemented prototype is completely developed in Java. At the moment, the supported communication protocols are http and AJP [18]. The Test Client TC is able to simulate clients, which can imitate the behavior of application server(s) and consumers, respectively. That means, these simulated clients create http POST requests for file uploads and GET request for file downloads to test the very basic functionalities of file processing in a SaaS. The clients send also SQL Queries to test the database capability. The File Worker VMs FW_1, FW_2 were developed using Java Servlet and deployed on a JBoss AS7 Application Server [19]. To enable the servlet using different encryption methods, the installed JBoss AS7 was extended by a security provider module of Bouncycastle [20]. Files are encrypted with AES (256Bit) [21] in the private cloud and then uploaded to a public cloud server. WebDav [22] is used to provide a file server in the public cloud. The file servers is hosted by an European IaaS provider. The SQL Worker based on an CryptDB enabled mysql proxy server described by Popa et al. [23]. For storing this data a common MySQL database extended by CryptDB user defined functions is also hosted by the IaaS provider. Both, virtual server use minimal resources of 1 GHz with 1 GB RAM.

A. Test Setup

For the shown test results in Figure 3 and 4 the Test Client (TC) simulates four clients. Three clients, started with a delay of 20 s, send files, while one client sends SQL queries. Two file clients work after the following patterns. $ABABA$, where A stands for upload, download and delete (UDD) 12 files of 1 MB with a delay of two s and B stands for UDD 12 files of 1 MB with a delay of 10 ms. The third file client executes a CDC pattern, where C stands for UDD 5 files of 10 MB with a delay of 5 seconds and D stands for UDD 3 files of 10 MB with a delay of 10 ms. As long as these patterns are not finished yet, the SQL client repeats sending queries in form of 15 inserts, 10 selects and 15 deletes. To complete the test protocol, the simulated clients take 10 min and 23 s. Figure 3 shows the VM workloads of G and F_1, F_2 in 20 s intervals.

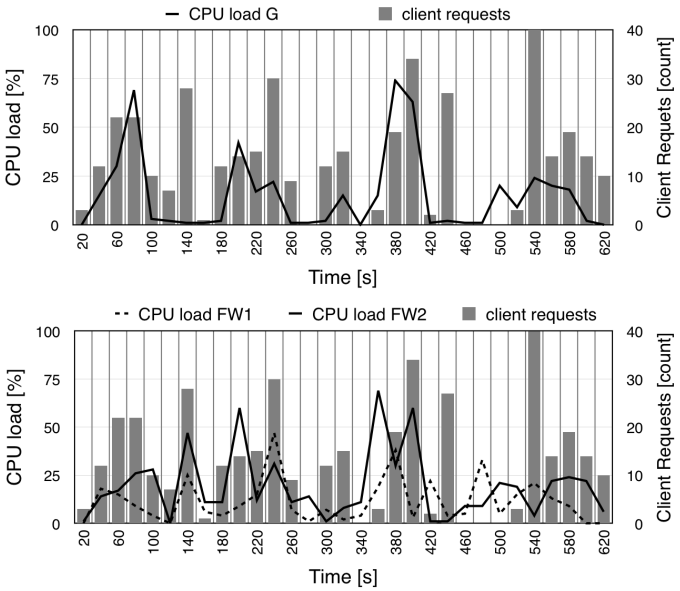

Figure 3. CPU load of G (0.2 vCPU, 512 MB RAM) FW_1, FW_2 (0.25 vCPU, 2 GB RAM) VMs depending on the number of client requests.

The load balancing metric configured in mod_cluster config in JBoss nodes combines CPU load, system memory usage and amount of outgoing/incoming requests traffic. Figure 4 shows the response times for processing the SQL queries.

B. Test results

The configured load balancing metric works very well, as shown in Figure 3. Especially, the timespan between 340 and 420 s is remarkable. The gateway recognize the high load of FW_2 sending client requests to FW_1. At time intervals of 380 s, it is inverse.

Figure 4 shows interesting results of the 18 rounds the SQL client executes its *'send 15/10/15 queries'* protocol. Although, the median of the response times is promising, there are lots

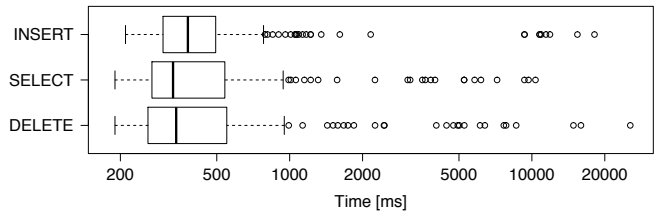

Figure 4. Logarithmic scaled response times for processing 234 INSERT, 180 SELECT and 234 DELETE queries. Whiskers maximums are 1.5 IQR.

of outliers. With regard to the logarithmic scale, a response time of ten to twenty seconds for a very basic query are unacceptable for practical use. Our best guess is the CryptDB proxy doing some internal recovery and key management operations. However, we bind the relevant folder via NFS to our key management KM, because all data in VMs is volatile, some file operations via the internal network should not last that long.

Table III displays our test results for different encryption algorithms, showing that AES works most efficient. For this reason, we use AES for encryption in the file workers.

In addition to the above-mentioned test, we have run some long term tests to get some impressions of efficiency and overall overheads. First of all, Figure 5a shows the percentage of times for encryption, communication with KM, and upload files to the cloud. It illustrates only four percent used for encryption, the rest of time the file worker are, roughly speaking, waiting. The same can be seeing on Figure 5b. It has to be noted that, these long term tests were done by one simulated client uploading, download, and deleting a 1MB file with a delay of 10 seconds. There is no waiting time, it is just the fact, as can be seen in Table III, that the encryption/decryption times compared to upload/download times are so small.

The overheads can be seen in Figure 6. In fact, the overhead to upload and download a file is around 51% and 28%, respectively. The difference can be explained by the required effort to store the encryption key and is also illustrated in Figure 5a. The overhead to delete a file is with around 126% very high. It is explainable with the additional roundtrip to the key management to delete the stored encryption key.

TABLE III. UPLOAD, DOWNLOAD, ENCRYPTION, AND DECRYPTION AVERAGE TIMES $\bar{t}$ IN SECONDS

encryption method	$\bar{t}_{up}$	$\bar{t}_{enc}$	$\bar{t}_{down}$	$\bar{t}_{dec}$
AES (265 bit)	1040.9	39.9	1116.4	51.45
DESede (168 bit)	1165.9	167.18	1239.4	135.83
Serpent (256 bit)	1180.9	57.18	1138.2	57.92
Twofish (256 bit)	1195.9	50.55	1160.4	50.45
CAST6 (256 bit)	1300.9	53.27	1037.6	40.09

V. DISCUSSION AND FUTURE WORK

Our test results showed that our enhanced hybrid cloud architecture is reasonable and applicable. With the statement of thread scenerio A, the first question can be answered positively, setting up the base for the more interesting questions two and three. Our proposed approach supports both, unstructured and structured data. However, our tests of the integration of the work of [23] show that database encryption is much more complicated than file encryption. This can be concluded from the fact that database encryption is not only a matter of data-at-rest encryption, but computation under encryption as well.

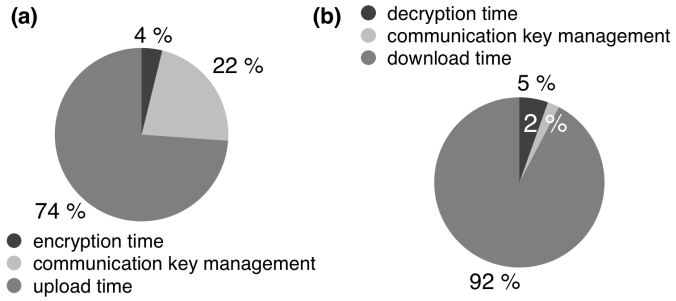

Figure 5. Percentage of time to encrypt and upload (a), decrypt and download (b), respectively.

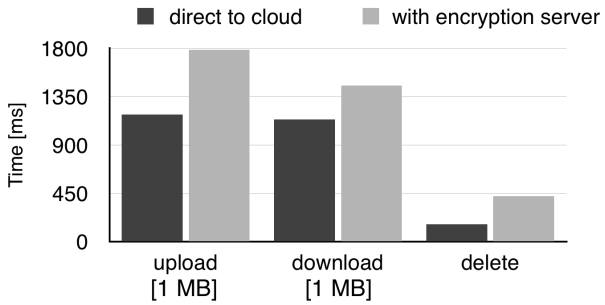

Figure 6. Diagram of average times to upload, download, and delete files with and without encryption

However, the support of databases is limited in a way not all queries can be supported; details can be seen in [24]. As a result the answer of question two is positive. Our test results confirm the mentioned fact in [23] that the implementation of CryptDB is highly prototypical. As the own implementations of Google and SAP show [25], more development effort - e.g. towards full support of JDBC - is necessary. As Figure 6 shows the overheads for file encryption are acceptable. Even the high overhead in deleting files, considering, the response time is still under half a second, the usability of the SaaS would be influenced in a very small way. An integration with low migration effort, as question three asks, is very realistic. In fact, as the gateway behaves like a file server/ database, the only change will be switching from old servers to the gateway server.

Figure 5 points out that the encryption workload of file worker is not high. This leaves space for additional functionalities like file compressing, for faster up- and downloads, or file indexing for possible searches over the encrypted files. The latter is mentioned in [26]. Also, the integration of a secure identity and key management system, e.g. Kerberos [27], is required to provide a SaaS solution with focus on the customers privacy. Attribute-based encryption concepts like [28][29] could be an interesting option for open questions like: How to integrate SaaS access rights in the key management system.

Moreover, the tests show that implementations of failure and backup routines are absolutely necessary. Despite, the different focus in this first implementation, we want to point out that security is not only about protecting data from unauthorized access or viewing, but also issues of auditing, data-integrity,

and reliability should be concerned too. These points will be addressed in future works.

VI. Conclusion

This paper described a hybrid cloud architecture model for SaaS providers with special consideration of service consumers' privacy and security aspects. Section II compares the three models, private, public, and hybrid cloud from which SaaS providers can choose. The private cloud model is the most inflexible and cost intensive option. Probably being an option for large companies owning much hardware resources, it is not suitable for SME-SaaS providers. The public cloud model is the preferred choice for SaaS providers if the application has no particular high privacy or security requirements. Especially for private end-user applications the public cloud model is cost efficient with low time to market and high scalability. Public cloud services can be recommended to start-ups because of the low investment costs and great flexibility. The hybrid cloud architecture offers compromise for multiple reasons. First, the solution addresses SME with experience in SaaS and own hardware infrastructure. Second, this model offers a higher security level and lowers privacy concerns of consumers. Albeit the cost efficiency is not as high as for the public model, it is clearly higher than for the private model. Despite these advantages, the hybrid approach incurs efficiency penalties in form of a trade-off between increased security for decreased efficiency, flexibility and scalability of public cloud solutions. Besides developing cost-efficient hybrid and secure SaaS solutions, it is highly complex and needs lots of expertise. The hybrid model offers significantly improved security compared to a public cloud architecture and neither the consumer nor the SaaS provider have to trust the public cloud provider. Of course, the consumer has to trust its SaaS provider. However, this is more reasonable than to trust a public cloud provider with an obscure number of third parties.

The prototype includes scalable and flexible encryption servers, a minimal key management system, a public file and database server. Test results showed that a hybrid architecture SaaS extended with encryption servers is a practical solution. In addition, the results illustrated that on-the-fly encryption and decryption is not only a matter of fast encryption methods, but a matter of high network throughput as well. To be applicable in productive systems, improvements of performance and more research are necessary.

The implemented prototype shows that the suggested hybrid architecture is a first step to achieve a higher acceptance of cloud-based SaaS, where providers address the consumers' concerns of privacy and security.

Acknowledgment

The published work has been financially supported by the European Social Fund (ESF) and the EU. We would like to thank the anonymous reviewers for helpful comments.

References

[1] AWS. Amazon web services. [Online]. Available: http://aws.amazon.com [retrieved: April, 2014]

[2] Instagram. Instagram website. [Online]. Available: http://instagram.com [retrieved: April, 2014]

[3] P. Buxmann, T. Hess, and S. Lehmann, "Software as a service," Wirtschaftsinformatik, vol. 50, no. 6, 2008, pp. 500–503.

[4] BITOM and KMPG, "Cloud-monitor 2013 cloud-computing in deutschland – status quo und perspektiven," KMPG Study, Februar 2013.

[5] F. e. a. Lui, "Nist cloud computing reference architecture," National Institute of Standards and Technology, Tech. Rep., 2011.

[6] P. Mell and T. Grance, "The nist defintion of cloud computing," National Institute of Standards and Technology, Tech. Rep., 2011.

[7] M. A. Rahaman. How secure is sap business bydesign for your business. [Online]. Available: http://scn.sap.com/docs/DOC-26472 [retrieved: April, 2014]

[8] C. Gentry, "A Fully Homomorphic Encryotion Scheme," Ph.D. dissertation, Stanford University, 2009.

[9] M. a. a. van Dijk, "Hourglass schemes: How to prove that cloud files are encrypted," in Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, pp. 265–280.

[10] J.-M. Bohli, N. Gruschka, M. Jensen, L. Lo Iacono, and N. Marnau, "Security and Privacy Enhancing Multi-Cloud Architectures," IEEE Transactions on Dependable and Secure Computing, 2013, pp. 1–1. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6487347

[11] K. Zhang, X. Zhou, Y. Chen, and X. Wang, "Sedic : Privacy-Aware Data Intensive Computing on Hybrid Clouds Categories and Subject Descriptors," 2011, pp. 515–525.

[12] D. Hühnlein, G. Hornung, H. Roßnagel, J. Schmölz, T. Wich, and J. Zibuschka, "Skidentity–vertrauenswürdige identitäten für die cloud," DA-CH Secur, 2011, pp. 296–304.

[13] T. L. Security. Tls documentation. [Online]. Available: http://tools.ietf.org/rfcmarkup/5246 [retrieved: April, 2014]

[14] OpenNebula. Opennebula website. [Online]. Available: http://docs.opennebula.org/4.4/release_notes/ [retrieved: April, 2014]

[15] ONEFlow. One flow doc. [Online]. Available: http://docs.opennebula.org/4.4/advanced_administration/application_flow_and_auto-scaling/appflow_configure.html [retrieved: March, 2014]

[16] ModCluster. mod-cluster website. [Online]. Available: http://www.jboss.org/mod_cluster [retrieved: April, 2014]

[17] Apache. Apache 2.2 website. [Online]. Available: http://httpd.apache.org/docs/2.2/en/ [retrieved: April, 2014]

[18] AJP. Ajp website. [Online]. Available: http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html [retrieved: April, 2014]

[19] JBossAS7. Jboss website. [Online]. Available: http://www.jboss.org/jbossas/ [retrieved: April, 2014]

[20] Bouncycastle. bouncy castle website. [Online]. Available: http://www.bouncycastle.org/java.html [retrieved: April, 2014]

[21] J. Daemen and V. Rijmen, The design of Rijndael: AES-the advanced encryption standard. Springer, 2002.

[22] WebDAV. Webdav website. [Online]. Available: http://www.webdav.org [retrieved: April, 2014]

[23] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB : Protecting confidentiality with encrypted query processing Accessed Citable Link Detailed Terms CryptDB : Protecting Confidentiality with Encrypted Query Processing," 2011.

[24] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in Proceedings of the 39th international conference on Very Large Data Bases. VLDB Endowment, 2013, pp. 289–300.

[25] C. impact. Cryptdb website impact section. [Online]. Available: http://css.csail.mit.edu/cryptdb/#Impact [retrieved: April, 2014]

[26] S. Kamara, C. Papamanthou, and T. Roeder, "Cs2: A searchable cryptographic cloud storage system," Microsoft Research, TechReport MSR-TR-2011-58, 2011.

[27] Kerberos. Keberos documentation. [Online]. Available: http://tools.ietf.org/html/rfc4120 [retrieved: April, 2014]

[28] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007, pp. 321–334.

[29] C.-P. A.-B. Encryption. Advanced crypto software collection website. [Online]. Available: http://hms.isi.jhu.edu/acsc/cpabe/ [retrieved: April, 2014]

 ISBN: 978-1-61208-338-4