2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing

Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures

Wassim Itani Ayman Kayssi Ali Chehab
Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon
{wgi01, ayman, chehab}@aub.edu.lb

Abstract— **In this paper we present PasS (Privacy as a Service); a set of security protocols for ensuring the privacy and legal compliance of customer data in cloud computing architectures. PasS allows for the secure storage and processing of users' confidential data by leveraging the tamper-proof capabilities of cryptographic coprocessors. Using tamper-proof facilities provides a secure execution domain in the computing cloud that is physically and logically protected from unauthorized access. PasS central design goal is to maximize users' control in managing the various aspects related to the privacy of sensitive data. This is achieved by implementing user-configurable software protection and data privacy mechanisms. Moreover, PasS provides a privacy feedback process which informs users of the different privacy operations applied on their data and makes them aware of any potential risks that may jeopardize the confidentiality of their sensitive information. To the best of our knowledge, PasS is the first practical cloud computing privacy solution that utilizes previous research on cryptographic coprocessors to solve the problem of securely processing sensitive data in cloud computing infrastructures.**

Keywords-privacy;cloud computing; cryptographic coprocessors ;security

I. INTRODUCTION

Cloud computing has brought up major advancements to the IT industry. Building on its predecessors, namely, grid and utility computing, this new evolutionary model is witnessing a rapid expansion and proliferation. Today, clients are capable of running their software applications in remote computing clouds where data storage and processing resources could be acquired and released, almost, instantaneously. The virtualization layer on top of the commodity hardware in computing clouds is the driving force that allows cloud providers to "elastically" and promptly respond to client resource demands and requirements.

In spite of all the advantages delivered by cloud computing, several challenges are hindering the migration of customer software and data into the cloud. On top of the list is the security and privacy concerns arising from the storage and processing of sensitive data on remote machines that are not owned, or even managed by the customers themselves. With cloud computing, all the customer can see is a virtual infrastructure built on top of possibly non-trusted physical hardware or operating environments. We believe that data privacy should be provided to cloud customers as a service with minimal additional cost. Moreover, we believe that the cloud privacy model should be configurable and user-centric. That is the cloud customer should be able to flexibly control and manage the different privacy mechanisms necessary to protect sensitive data and achieve legal compliance. Customers should be aware through a secure privacy auditing process of all the operations carried out to secure the storage and processing of their sensitive information.

In this paper we present PasS; a set of security protocols for ensuring the privacy and legal compliance of customer data in cloud computing architectures. PasS allows for the secure storage, processing, and auditing of users' confidential data by leveraging the tamper-proof capabilities of cryptographic coprocessors. Using tamper-proof cryptographic coprocessors provides a secure and trusted execution domain in the computing cloud that is physically and logically protected from unauthorized access. PasS central design goal is to maximize users' control in managing the various aspects related to the privacy of sensitive data. This is achieved by implementing user-configurable software protection and data privacy categorization mechanisms. Moreover, PasS provides a privacy feedback process which informs users of the different privacy operations applied on their data. This aids in relieving users' security and privacy concerns, increasing the customers trust in the cloud service, and pushing the adoption of cloud computing in even the most privacy-demanding applications such as healthcare and financial applications. To the best of our knowledge, PasS is the first practical cloud computing privacy solution that utilizes previous research on cryptographic coprocessors to solve the problem of securely processing sensitive data in cloud computing infrastructures.

The rest of the paper is organized as follows: in Section 2 we present the system and trust models assumed in this work. Section 3 discusses the system design and architecture. In Section 4, we consider the PasS security protocols for ensuring the privacy of customer data. In Section 5, we continue with a discussion on a sample system implementation and an economic feasibility study. Related research and conclusions are presented in Sections 6 and 7 respectively.

II. SYSTEM AND TRUST MODELS

The system model assumed in this work is a typical cloud computing model with two main players:

1. A cloud provider which manages and operates a cloud infrastructure of storage and computing services.
2. A cloud customer or consumer that employs the cloud storage and computing resource facilities to remotely store and process data. The Internet is the main communication backbone for exchanging information between the cloud customer and the computing cloud.

PasS provides the cloud customer with a full control on the privacy mechanisms to be applied on the cloud data. The notion of customer trust in the services published by the cloud provider is based on the degree of sensitivity of the customer information. PasS supports three trust levels in the cloud service provider:

978-0-7695-3929-4/09 $26.00 © 2009 IEEE
DOI 10.1109/DASC.2009.139

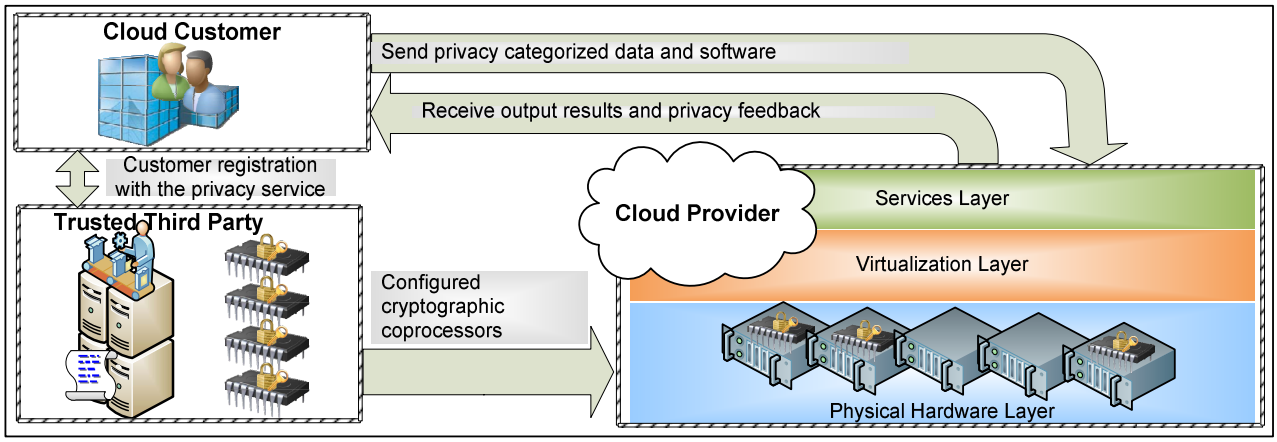

Figure 1. PasS Cloud-based System Model

1. Full trust: this level applies to insensitive data that can be safely stored and processed in the clear (without any form of encryption) on the computing cloud side. The provider is fully trusted for data storage and processing.
2. Compliance-based trust: This level applies to customer data that needs to be stored encrypted to support legal compliance regulations (such as the Health Insurance Portability and Accountability Act (HIPAA) for securing medical records and patient's information and the Gramm-Leach-Bliley Act for ensuring the confidentiality of financial records and banking transactions for any institution providing a financial service). In this level the customer trusts the cloud provider in storing her data encrypted using a provider-specific cryptographic key.
3. No trust: This level applies to highly-sensitive customer data that should be concealed from the cloud provider. This kind of sensitive information should be stored encrypted using customer-trusted cryptographic keys and should be processed in isolated cryptographic containers in the cloud. These isolated containers are configured, distributed, and maintained by a third-party that is trusted by the cloud customer as well as by the cloud provider. Figure 1 outlines the system model and sketches the interaction among the cloud customer, cloud provider, and the trusted third-party.

Note that other trust levels, such as those supported by role-based trust models, can also be considered.

It is worth mentioning here that a possible adversarial behavior is for a semi-honest cloud provider to emulate the protocol operation in software to reduce the required infrastructure costs imposed by relatively-expensive hardware coprocessors. However, as will be presented later in this paper, the PasS protocols are immune to such kind of adversary. This is due to the fact that the physically-secure coprocessor is configured by a trusted third-party with the cryptographic keying material necessary to execute and process encrypted customer software and data. With no access to these cryptographic structures, the cloud provider will not be able to emulate the operation of the coprocessor system.

III. SYSTEM DESIGN AND ARCHITECTURE

This section presents an overview of the system design and architectural components. It starts with a brief description on cryptographic coprocessors, discusses the coprocessor's configuration, distribution, and structuring, and presents the software division and data categorization mechanisms used to support the privacy of data storage and processing in the computing cloud.

A. *Secure Coprocessor Basics*

PasS relies on cryptographic coprocessors [1, 2, 3] to provide secure and isolated processing containers in the computing cloud. A cryptographic coprocessor is a small hardware card that interfaces with a main computer or server, mainly through a PCI-based interface. It is a complete computing system that is supported with a processor, RAM, ROM, backup battery, non-volatile persistent storage, and an Ethernet network card. For economical reasons, a crypto coprocessor is generally less capable in terms of processing and memory resources than the main server system it interfaces to. The main property that gives a crypto coprocessor its secure capabilities is the tamper-proof casing that encloses it and makes it resist physical attacks. A secure coprocessor tamper-resistance or tamper-responding mechanisms should reset the internal state of the coprocessor (RAM, persistent storage, processor registers) upon detecting any suspicious physical activity on the coprocessor hardware.

The Input/output access to the cryptographic coprocessor can be either done locally via the main server system bus, or remotely via the coprocessor network card.

B. *Coprocessor Configuration and Distribution*

The organizational unit responsible of configuring the crypto coprocessor and distributing it to cloud providers is a third-party entity that is trusted by the cloud provider and customer. In a cloud computing infrastructure, a crypto coprocessor should be installed on every physical server running a Virtual Machine (VM) for customers registered in the privacy service. To make the solution economically feasible, PasS allows the resources of the crypto coprocessor to be shared among more than one cloud customer. In fact, it is this sharing mechanism that necessitates the presence of a trusted third-party (*TTP*) to load the cryptographic data structures and keying material of more than one cloud customer on the crypto coprocessor. The *TTP* is viewed as a seller of a cloud privacy service in collaboration with the cloud provider. Technically, the main responsibility of the *TTP* is to load a set of private/public key pairs into the persistent storage of the crypto coprocessor. Every public/private key pair (PU_{CID}/PR_{CID}) is to be allocated to a single customer when the latter registers with the cloud privacy service. Upon registration, the cloud customer will securely receive a copy of her public/private key pair. This can

be achieved through a face-to-face transaction or through a secure electronic session.

The PU_{CID}/PR_{CID} key pair set can be remotely updated by the *TTP* even after the crypto coprocessor is installed in the computing cloud. This remote key update mechanism is very important to support the registration of new customers and the service revocation of existing customers. Moreover, this mechanism is essential for the system to cope with dynamic user workloads and resource requirements that may change over time. By remotely updating the keying material on the coprocessor, the *TTP* can dynamically upgrade or downgrade user resources as far as the number of coprocessors installed in the cloud and their hardware capabilities permit.

In addition to loading the customer's PU_{CID}/PR_{CID} key pair, the *TTP* also loads its own private key K_{TTP} into the persistent storage of the crypto coprocessor. This key is needed by the *TTP* to remotely authenticate to the crypto coprocessor and to securely execute commands against it.

C. Coprocessor Process Structure

We followed the ABYSS [4] processor model in structuring the processes of the cryptographic coprocessor. The main concept is to logically isolate the set of protected customer applications running on the coprocessor using a *root* highly-privileged process. We refer to this process as the *RP* daemon. The main responsibilities of the *RP* daemon are as follows:

1. Each customer application is divided into a secure part running in the address space of the coprocessor and a non-secure part running in the address space of the main server hosting the coprocessor (see Section III. D). The *RP* daemon ensures that each customer application runs in a separate protection domain, thus preventing any form of attack from one process to the other. Moreover, the *RP* daemon protects the interaction between an application process running on the main server and its protected part running on the crypto coprocessor.
2. The *RP* daemon authenticates software and data entering the coprocessor address space (See Sections IV. A and IV. B)
3. The *RP* daemon is the only process having access to the cryptographic keying material stored in the crypto coprocessor persistent storage. Therefore, it is responsible of the encryption/decryption operations required by the privacy enforcement mechanisms.
4. The *RP* daemon authenticates remote connections from cloud customers and the *TTP*.
5. The *RP* daemon participates in a secure privacy feedback process that is described in Section IV. C.

Due to the central role the *RP* daemon plays in the privacy service and to support a scalable security mechanism, it is recommended to execute multiple replicas of the same *RP* daemon in the crypto coprocessor.

D. Software Division

In the PasS security model, the cloud customer is responsible of configuring her software applications to support the security mechanisms enforced by the privacy service. To provide an economical and performance efficient security solution, the concept of software division is adopted. Based on this concept, the cloud customer classifies the set of logical components constituting the software application as protected and unprotected. The protected classification indicates that the logical component should be executed in a protected process in the address space of the crypto coprocessor. On the other hand, the unprotected classification indicates that the logical component can be executed in a traditional process on the main server. The set of protected processes running in the crypto coprocessor and belonging to different registered customers are isolated from each other and from the unprotected parts using the *RP* daemon. The protected application part should be stored encrypted (by the customer) on the provider's side. When this part is loaded into the crypto coprocessor, the *RP* process decrypts its contents and executes the resulting binary code.

It should be noted here that it is possible to run the entire software application in the secure coprocessor without applying any software division, however this would affect the performance and economic efficiency of the application which refutes the whole concept of a coprocessor.

E. Data Privacy Specification

Before uploading the data to be stored and processed in the computing cloud, the cloud customer classifies this data, based on significance and sensitivity, into 3 privacy categories:

1. No Privacy (*NP*): Data marked with this attribute is not sensitive and hence the provider is fully trusted to store it without any form of encryption. If network security is needed, the client can send the data over a secure SSL session.
2. Privacy with Trusted Provider (*PTP*): Data marked with this attribute is stored encrypted by a specific provider key. In this case the provider is trusted to encrypt the data using her own cryptographic key. This attribute is crucial for compliance with regulatory policies. To send *PTP* data to the provider, the customer is required to encrypt it over a secure SSL session to achieve network data confidentiality and integrity. The cloud provider is obligated by the contractual policy to extract the SSL secured data and store it encrypted.
3. Privacy with Non-Trusted Provider (*PNTP*): This data category is encrypted on the customer side by a customer-specific key (K_{CID}) shared with the cryptographic coprocessor (see Section IV. B). This kind of data is stored encrypted on the cloud storage facility and it cannot be accessed or viewed by the cloud provider (encrypted with a customer key). The *PNTP* data can only be processed in the address space of the trusted cryptographic coprocessor which shares the customer the possession of K_{CID}.

For each data category, the cloud provider allocates a logical storage partition that we refer to as a storage pool. The storage pool name is referenced by the name of the data privacy category. Thus, *NP*, *PTP*, and *PNTP* data is stored in the *NP*, *PTP*, and *PNTP* cloud storage pools respectively.

IV. Privacy Protocols

A. Data and Software Transfer Protocol

This section describes the protocol steps executed by the cloud customer to add the privacy enforcement structures to the software and data before transferring them to the computing cloud. Starting with the software privacy data structure, the customer configures the cloud software into 3 main components: the privacy tag, the protected software part S_{SID}, and the unprotected software part S'_{SID}. The privacy tag consists of the following elements:

CID: This is a unique customer identification number provided by the cloud provider upon customer registration with the cloud privacy service.

SID: This is a unique software identification number provided by the customer to each of her software applications.

$E(PU_{CID},K_{SID})$: firstly, K_{SID} is a symmetric key randomly generated by the cloud customer to encrypt the protected

software part S_{SID}. $E(PU_{CID},K_{SID})$ is a public-key encryption of K_{SID} by the customer's public key PU_{CID}. This encryption process ensures that no entity, other than the *RP* daemon on the crypto coprocessor, can extract K_{SID}.

DS: is a digital signature on the fields of the software package. Symbolically it is represented as follows: $DS = Sign(PR_{CID}, CID||SID||E(PU_{CID},K_{SID})||Timestamp||Identification||S_{SID}||S'_{SID})$. Where || is the concatenation operator.

Timestamp: this field represents the time just before the digital signature is created. It is included in the digital signature to protect against replay attacks.

Identification: This field includes general information about the software application such as its name, version, etc.

$E(K_{SID},S_{SID})$: This is the protected software part encrypted by K_{SID}.

S'_{SID}: This is the unprotected software part. It can be safely executed on the main server in the computing cloud.

Based on the above software configuration, the customer constructs the software packages and transfers them to the computing cloud.

Receiving the different software messages, the cloud provider checks their integrity and authenticity using the digital signature *DS* and stores them in the cloud storage.

Concerning the data transfer to the computing cloud, the customer executes the following protocol steps:

1. Classify the data according to the 3 privacy categories presented in Section III. E.
2. If the privacy category is *NP* or *PPT*, then the client sends this data as is to the provider. In the case of *PPT* data, the transfer should be secured by an SSL session. Receiving the data, the provider stores it in the appropriate storage pool.

If the privacy category is *PPNT*, the customer executes an authenticated version of the Diffie-Hellman key management protocol [25] with each crypto coprocessor this customer is registered with. The result of the Diffie-Hellman protocol execution is a shared secret K_{CID}. Using K_{CID}, the customer encrypts the *PPNT* data and sends it to the cloud provider to be stored in the *PPNT* storage pool. Note that the shared secret K_{CID} is securely stored by the *RP* daemon in the crypto coprocessor in an entry identified by *CID*. To ensure the integrity of data when stored in the cloud, Message Authentication Codes (*MACs*) using K_{CID} are applied on logically-separated data units. The selection of the logical unit is implementation-dependent. A unit could be a file, a set of related files, or any other customer-preferred integrity element.

B. Software Execution and Data Processing Protocol

This protocol describes the steps executed by the crypto coprocessor and the main server hosting the coprocessor to safely execute the customer cloud software. It also presents the privacy enforcement mechanisms that ensure the privacy of the customer sensitive data when processed in the computing cloud. The protocol steps are as follows:

1. The main server loads the unprotected software part and sends the software package to the crypto coprocessor.
2. The *RP* daemon on the crypto coprocessor reads the software privacy tag and ensures the authenticity and integrity of the software package by verifying *DS*. Moreover, the *RP* daemon checks the validity of the timestamp and extracts the key K_{SID} by decrypting $E(PU_{CID},K_{SID})$ with the customer's private key.
3. The *RP* daemon decrypts the protected software part $E(K_{SID},S_{SID})$ using K_{SID} and executes this part within the address space of the trusted coprocessor.
4. The *RP* process stores K_{SID} in an entry identified by (*CID*, *SID*) in the persistent storage of the coprocessor. This step avoids an expensive public-key decryption operation if the same software is loaded once again.
5. The unprotected part of the application can only process data with the *NP* and *PPT* privacy categories. Practically the unprotected part cannot access *PPNT* data since it does not have access to the ciphering key K_{CID} with which the data was encrypted. If the data requested belongs to the *NP* storage pool, then it is loaded immediately from the cloud central storage. On the other hand, if the data belongs to the *PPT* storage pool, the provider is responsible of decrypting the data before the application can load and process it. The main responsibility of the protected part of the application is to process sensitive customer data belonging to the *PPNT* privacy category (technically, the protected part can also process *NP* and *PPT* data in the same manner presented above). Data belonging to the *PPNT* privacy category is encrypted with the customer key K_{CID}. This key is safely stored in the persistent storage of the crypto coprocessor. To get access to the plaintext version of this data, the protected software part requests the *RP* daemon to load the data form the cloud central storage. The *RP* daemon, having access to K_{CID}, loads the encrypted data, decrypts it, checks its integrity, and presents it to the protected application part.

Output results produced by the customer software also belong to the 3 privacy categories *NP*, *PPT*, and *PPNT* (*PPNT* data is only produced by the protected software part). Data produced in the crypto coprocessor with the *NP* privacy category is sent by the protected application to the *RP* daemon. The latter sends it, as is, to the *NP* storage pool. *NP* data can be also produced by the unprotected software part. In this case the unprotected part is responsible of sending the data to the *NP* storage pool.

Data produced with the *PPT* privacy category in the crypto coprocessor is sent by the protected application to the *RP* daemon which in turn transfers it to a provider's encryption process running on the main server. This process encrypts the *PPT* data with a provider ciphering key and stores it in the *PPT* storage pool. Finally, data produced with the *PPNT* privacy category is sent to the *RP* daemon which encrypts it using K_{CID} and stores it in the *PPNT* storage pool. This data category can only be produced by the protected software part running in the crypto coprocessor.

C. Privacy Feedback Protocol

The privacy feedback protocol is an essential component that should be considered and planned thoroughly when designing privacy-aware cloud services. The main responsibility of this protocol is to inform users of the different privacy mechanisms applied on their data and to make them aware of any data leaks or risks that may jeopardize the confidentiality of their sensitive information.

The design of the *RP* daemon supports the operation of the privacy feedback process. This is due to the fact that the *RP* daemon represents a centralized processing entity that handles all the privacy enforcement mechanisms in the crypto coprocessor. The operation of the protocol is summarized as follows:

1. Whenever the *RP* daemon executes a privacy-related operation it creates a privacy audit record describing this operation. The type of the privacy operation and the contents of the privacy record are specified by the customer upon registration with the privacy service. The *RP* daemon encrypts the privacy record with K_{CID} to protect its confidentiality.

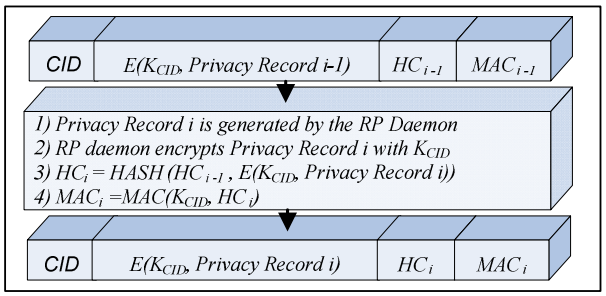

Figure 2. Confidentiality and integrity protection of the privacy log

2. The *RP* daemon constructs a hash chain data structure over the encrypted privacy record [22, 23]. This is illustrated in Figure 2. In this figure, HC_i is the hash chain constructed by hashing the i^{th} privacy record and the hash chain entry of the previous privacy record. Since HC_i includes HC_{i-1}, it is possible to verify the integrity of all previous privacy records by only authenticating *HCi.* Initially, HC_{-1} is given a default value to start the hash chain. To authenticate HC_i, *a* MAC_i field is added to the privacy record. This field represents the *MAC* of HC_i using K_{CID}.
3. The *RP* daemon stores the resulting secure record on the cloud storage facility.
4. A special customer application polls the contents of the privacy audit log every predefined period of time. After verifying the integrity of the privacy records using the hash chain and the respective *MAC* value, the application decrypts the contents of the privacy records using K_{CID}. The customer can analyze the decrypted privacy records and take actions based on this analysis.

V. IMPLEMENTATION AND ECONOMIC FEASABILITY

A prototype implementation of the PasS protocols is tested on a simple banking application. The prototype included, in addition to a sample implementation of the privacy protocols, an implementation of the data privacy categorization and software division mechanisms. The infrastructure used in the prototype emulates a standard cloud computing unit. The physical server used is an Intel Core 2 Duo machine running Windows Server 2008 Enterprise Edition. The 2 server processors run at 3.0 GHz and the system is supported with 4 GB of RAM. The virtualization layer is provided using VMware Workstation 6.5.2. The guest operating system runs Windows Server 2003 Enterprise Edition. To implement the functionality of the secure cryptographic coprocessor, we assume that one of the core CPUs is the physically secure coprocessor, while the other core is that of the main untrusted server. For the sake of testing the system functionality and security mechanisms, the current configuration provides a viable proof of concept.

The software division process is applied on the banking application by separating the application functions into a protected and unprotected process. The protected process carries out critical banking operations (account deposits, withdrawals, and transfers) and runs on the crypto coprocessor, while the unprotected process runs on the main server CPU. The application is developed using the C# programming language which is part of the Microsoft .Net framework. To programmatically set the processor affinity of the protected and unprotected processes, we used the *ProcessorAffinity* property of the *Process* class in the *System.Diagnostics* namespace.

A brief economic study shows that commercial cryptographic coprocessors range in price from several hundreds to several thousands U.S. Dollars. The cost of the coprocessor mainly depends on the processing and memory capabilities of the coprocessor, the degree of physical security and tamper-resistance supported, the compliance of the coprocessor with FIPS standards, and the crypto functionality (Hardware acceleration and cryptographic implementations) provided. We believe that the cost of the privacy solution presented can be greatly reduced based on a set of external economic factors as well as internal design choices related to the PasS protocol architecture itself. These factors are summarized in the following points:

1. The increase in demand on cryptographically secure facilities, particularly in computing clouds, and the emergence of open-source cryptographic processor designs will gradually result in a higher functionality/cost ratio.
2. The technological advancements in computing and memory hardware, as well as in physical packaging mechanisms, will result in delivering cost-effective crypto coprocessors.
3. The coprocessor sharing mechanism employed in PasS participates in splitting the cost of the privacy services among the cloud provider and the customers sharing the device. This sharing mechanism plays a major role in the cost-effectiveness of the security solution. To illustrate this, let *PSCP* denote the cost of the privacy service per processor, *PSC* the capital cost of the privacy service, *N* the number of crypto coprocessors installed in the cloud and *W* the number of customers sharing a crypto coprocessor. *PSCP* consists of the following cost components: The cost of the crypto coprocessor hardware, the coprocessor configuration and distribution, the coprocessor installation in the computing cloud, and the operational and maintenance costs including energy consumption and network bandwidth usage. Assume that *PSCP* is $ 8,000 and *N* is 15, then the capital cost of the privacy service *PSC* is $ 120,000. This capital cost is split among the cloud provider and the *W* cloud customers sharing each crypto coprocessor. A reasonable split percentage is for the cloud provider to incur 50% of the capital cost ($ 60,000). The remaining 50% is further divided among the *W* customers sharing the coprocessor. Assuming $W = 20$, then the cost incurred per customer is equivalent to $ 3000. This cost is very reasonable in return of the privacy service provided. From the cloud provider's perspective, we believe that the payback period will be very short due to the higher revenues resulting from the increase in customer base. This fact becomes more evident if we know that the main factor hindering the adoption of cloud computing, particularly in healthcare and financial applications, is the lack of privacy and compliance support in current cloud service implementations.
4. The software division mechanism implemented supports a better utilization of the crypto coprocessor and avoids any unnecessary loads on its resources. This aids in reducing the resource requirements, and hence the price, of the coprocessor.

VI. RELATED WORK

A large amount of research work has dealt with the design and implementation of secure cryptographic coprocessors. The secure cryptographic processor concept was firstly introduced in [1]. In this paper, Best presents how crypto coprocessors can be utilized to enforce software copyright protection and prevent software piracy. Popular crypto coprocessor designs included Citadel [5], µABYSS [6], Luna-340 [7], and CCProc [8]. The considerable advancement in physical security mechanisms and packaging technology [9] and the assortment of secure applications that can be implemented on top of

physically secure coprocessors was a major driving force to a prosperous commercial market. IBM was the leader on this front by providing a set of successful implementations meeting the strictest FIPS 140 security standards [10, 11]. Moreover [12] presented a general-purpose open-source cryptographic coprocessor that provides competitive performance and higher functionality compared to commercial products at one to two orders of magnitude lower cost.

Research on crypto coprocessors also targeted the implementation of a wide set of secure applications leveraging the tamper-proof capabilities of these coprocessors. Of these we can mention the Dyad project and its applications [3, 13] by Tygar and Yee. In [14] Kelsey and Schneir presented a solution for remotely authenticating software outputs using a trusted cryptographic coprocessor. Their solution employed an ABYSS-based [4] software division mechanism similar to the one presented in PasS. The main assumption in the paper, which makes the solution feasible to implement, is that the secure part of the application running on the trusted coprocessor can determine the output of the whole software program. [24] utilized cryptographic coprocessors for implementing trusted virtual machine monitors on top of untrusted operating systems. The trusted virtual machine monitor allows the partitioning of the crypto coprocessor into a set of isolated virtual machines with different security requirements and configurations. In [15] secure coprocessors are used to protect the privacy of data mining and sharing in Anti Money Laundering (AML) and credit rating application areas.

Research on data privacy in cloud computing is still in its early stages. Good reports on the topic are presented in [16] which discusses the risks imposed by the adoption of cloud computing on data privacy and legal compliance, and [17] which emphasizes on the need to develop a sound digital identity infrastructure to support tackling privacy and security concerns in computing clouds. [18, 19] present a comprehensive set of guidelines on designing privacy-aware cloud services. [18] summarizes the privacy patterns in 6 recommended practices: "(1) minimizing customer personal information sent to and stored in the computing cloud; (2) protecting sensitive customer information in the cloud; (3) Maximizing user control; (4) Allowing user choice; (5) Specifying and limiting the purpose of data usage; (6) providing the customer with privacy feedback". Note that tips 2 to 6 are addressed in this paper. The first recommendation is not addressed since we believe that it does not comply with the cloud computing spirit. We believe that the customer should be able to safely send any kind of sensitive data to the cloud and to secure it by providing user-configurable privacy enforcement mechanisms. It should be noted here that Homomorphic encryption [20] techniques for allowing specific algebraic operations on encrypted data are still theoretical and lack any pragmatic implementation. A practical fully-homomorphic cryptosystem is still an open research topic [21].

VII. Conclusion

This paper presented PasS, a set of security protocols for ensuring the privacy of customer data in cloud computing infrastructures. The security solution relies on secure cryptographic coprocessors for providing a trusted and isolated execution environment in the computing cloud. The paper discussed the PasS protocols and described the privacy enforcement mechanisms supported by them. The paper also presented a description of a proof of concept implementation of the privacy protocols. Future extensions will: (1) consider a variety of design choices including those that do not rely on the presence of a trusted third-party, (2) investigate alternative key management and distribution mechanisms, (3) research the development of standard patterns to systematically support the software division process, and (4) provide detailed analysis and evaluation of the system implementation.

Acknowledgment

The authors would like to thank and acknowledge the support of the Lebanese National Council for Scientific Research and the AUB University Research Board.

References

[1] RM Best, "Preventing Software Piracy with Crypto-Microprocessors", in *Proceedings of IEEE Spring COMPCON 80*, pp 466-469.
[2] L C. Guillou, M. Ugon, and JJ. Quisquater, "The smart card: A standardized security device dedicated to public cryptology", In Gustavus J Simmons, editor, *Contemporary cryptology: The science of information integrity*, IEEE Press, Piscataway, NJ, 1992.
[3] J. D. Tygar and B. Yee, "Dyad: A system for using physically secure coprocessors", In Proc. of *IP Workshop*, 1994.
[4] S.R. White and L. Comerford, "ABYSS: An Architecture for Software Protection", *IEEE Transactions on Software Engineering,* vol. 16, No. 6, June 1990, pp. 619-629.
[5] S.R. White, S.H. Weingart, W.C. Arnold, E.R. Palmer, Introduction to the Citadel architecture: security in physically exposed environments, Technical Report RC 16672, Distributed Systems Security Group, IBM T.J. Watson Research Center, March 1991.
[6] S.H. Weingart, Physical security for the mABYSS system, IEEE Computer Society Conf. on Security and Privacy, 1987.
[7] CHRYSALIS-ITS HOME PAGE: HTTP://WWW.CHRYSALIS-ITS.COM
[8] Theodoropoulos, D., et al., "Cproc: An efficient Cryptographic Coprocessor", in *Proceedings of the 16th IFIP/IEEE International Conference on Very Large Scale Integration*, 2008.
[9] S. Weingart, "Physical security for the mABYSS system", *in Proc. of the IEEE Computer Society Conference on Security and Privacy*, pp. 52-58, 1987.
[10] J. Dyer, M. Lindemann, R. Perez, R. Sailer, S. Smith, L. van Doorn, and S. Weingart, "Building the IBM 4758 secure coprocessor", *IEEE Computer*, 34:57–66, October 2001.
[11] T.W. Arnold, L.P. Van Doorn,"The IBM PCIXCC : A new cryptographic coprocessor for the IBM eServer", IBM Journal of Research and Development, Vol 48, May 2004.
[12] P. Gutmann, "An Open-source Cryptographic Coprocessor", in *Proceedings of the 9th USENIX Security Symposium*, pages 97–112, 2000.
[13] B.S. Yee, J.D. Tygar, "Secure coprocessors in electronic commerce applications", in *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, July 1995.
[14] B. Schneier and J. Kelsey, "Remote Auditing of Software Outputs Using a Trusted Coprocessor", *Journal of Future Generation Computer Systems*, v.13, n.1, 1997, pp. 9-18.
[15] B. Bhattacharjee, N. Abe, K. Goldman, B. Zadrozny, V. R. Chillakuru, M. del Carpio, and C. Apte, "Using secure coprocessors for privacy preserving collaborative data mining and analysis", *in DaMoN '06*, 2006.
[16] Robert Gellman, "WPF REPORT: Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing", February 23, 2009.
[17] A. Cavoukian, "Privacy in the clouds", in *Springer Identity in the Information Society*, Published online: 18 December 2008.
[18] Pearson, "Taking Account of Privacy when Designing Cloud Computing Services", in *Proceedings of* ICSE-*Cloud'09*, Vancouver, 2009.
[19] S. Pearson and A. Charlesworth, "Accountability as a Way Forward for Privacy Protection in the Cloud", HP Labs Technical Report, HPL-2009-178, http://www.hpl.hp.com/techreports/2009/HPL-2009-178.pdf (2009)
[20] C. Gentry, "Fully homomorphic encryption using ideal lattices", in *Symposium on Theory of computing*, ACM, 2009, pp. 169–178.
[21] http://www.schneier.com/blog/archives/2009/07/homomorphic_enc.html
[22] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics", *ACM Transactions on Information and System Security* 2(2):159-196, May 1999.
[23] W. Itani, A. Kayssi, and A. Chehab, "PATRIOT – a Policy-Based, Multi-level Security Protocol for Safekeeping Audit Logs on Wireless Devices," *in Proc. of IEEE SecureComm'05)*, September 2005, Athens, Greece.
[24] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *Proc. 19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, 2003.
[25] W. Diffie, P.C. van Oorschot, and M.J. Wiener, "Authentication and authenticated key exchanges", Designs, Codes and Cryptography 2 (1992), 107-125.