KEY MANAGEMENT PROCESS ON THE HARDWARE CRYPTOGRAPHIC MODULE IN THE CLOUD COMPUTING

John Manuel Delgado Barroso, Luis Joyanes Aguilar
Universidad Pontificia de Salamanca nucleo Madrid, Paseo de juan xxiii, 3. 2804, Madrid, Spain

Pablo García Gundín
Universidad Pontificia de Salamanca nucleo Madrid, Paseo de juan xxiii, 3. 2804, Madrid, Spain

Keywords: Key Management, Cloud Computing, RSA PKCS, Cryptographic performance, Hardware Cryptographic Module (HSM).

Abstract: Cloud computing has multiple applications, from a standpoint of growth performance computing capabilities and data processing also offers a number of operational models, leaving out no items related to the scalability of the platform but with data security that it is managed and therefore the information in it are processed. The purpose of this paper is to implement a novel key management protocol on a platform of hardware cryptographic module to provide a solution to the management model, generation and use of cryptographic keys in the vicinity of cloud computing.

1 INTRODUCTION

Service providers can supply basic plans to ensure the development and service-based applications in the cloud, or they can leave all these protection measures at the discretion of the client.

As service providers in the cloud progress towards a stronger key management hosting, more work and research must be performed to overcome the potential obstacles in its adoption. Emerging standards should solve this issue in the near future, but they are still works in progress. Within the Cloud Computing, different issues and challenges related to key management must be resolved.

Access to key stores should be limited to entities that specifically require particular keys. Also, there should be policies that govern the key stores that use function separation to help access control. An entity that uses a particular key should not be the same entity which maintains it.

In relation to key backup and recovery, a key loss inevitably entails the loss of data which was protected by such key. Although this is an effective way to destroy data, an accidental loss of a key may result in the disappearance of protected data which was critical for the business attainment. In order to avoid this, solutions for data backup and restoring must be implemented in a safe manner.

There are a number of standards and information for guidance that are applicable to key management in the cloud. This paper proposes a prototype of key management supported over storage cryptographic devices that can be deployed on transactional or not-transactional Cloud Computing oriented processes.

2 CRYPTOGRAPHIC ACCESS SERVICE DEVELOPMENT

2.1 From the Command Process Point of View

A key generation has been implemented within an interoperable key management standard for the cryptographic service development. Such standard and attributes are shown in the following command:

Structure generation process consists of the following trace:

Manuel Delgado Barroso J., Joyanes Aguilar L. and García Gundín P..
KEY MANAGEMENT PROCESS ON THE HARDWARE CRYPTOGRAPHIC MODULE IN THE CLOUD COMPUTING.
DOI: 10.5220/0003142304930496
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2010), pages 493-496
ISBN: 978-989-8425-29-4
Copyright © 2010 SCITEPRESS (Science and Technology Publications, Lda.)

```
Object Type='00000002' (Symmetric Key),
Key Atributes:
Cryptographic Algorithm ='00000003' (AES),
Key length ='128'
Mask Use performed by the Key ='0000000C'

Tag: Request Message (0x420075), Type: Structure (0x01), Data:
  Tag: Request Header (0x420074), Type: Structure (0x01), Data:
    Tag: Protocol Version (0x420067), Type: Structure (0x01), Data:
      Tag: Protocol Version Major (0x420068), Type: Integer (0x02), Data:
0x00000001 (1)
      Tag: Protocol Version Minor (0x420069), Type: Integer (0x02), Data:
0x00000000 (0)
    Tag: Batch Count (0x42000D), Type: Integer (0x02), Data: 0x00000001 (1)
  Tag: Batch Item (0x42000F), Type: Structure (0x01), Data:
    Tag: Operation (0x42005A), Type: Enumeration (0x05), Data: 0x00000001
(Create)
    Tag: Request Payload (0x420076), Type: Structure (0x01), Data:
      Tag: Object Type (0x420055), Type: Enumeration (0x05), Data: 0x00000002
(Symmetric Key)
      Tag: Template-Attribute (0x42008E), Type: Structure (0x01), Data:
        Tag: Attribute (0x420008), Type: Structure (0x01), Data:
          Tag: Attribute Name (0x42000A), Type: Text String (0x07), Data:
Cryptographic Algorithm
          Tag: Attribute Value (0x42000B), Type: Enumeration (0x05), Data:
0x00000003 (AES)
        Tag: Attribute (0x420008), Type: Structure (0x01), Data:
          Tag: Attribute Name (0x42000A), Type: Text String (0x07), Data:
Cryptographic Length
          Tag: Attribute Value (0x42000B), Type: Integer (0x02), Data:
0x00000080 (128)
        Tag: Attribute (0x420008), Type: Structure (0x01), Data:
          Tag: Attribute Name (0x42000A), Type: Text String (0x07), Data:
Cryptographic Usage Mask
          Tag: Attribute Value (0x42000B), Type: Integer (0x02), Data:
0x0000000C (Encrypt, Decrypt)
```

Figure 1: Input trace structure for key generation process.

And command to be sent is shown as:

```
42007501000001204200740100000038420067010000002042006802000000400
0000010000000042006902000000040000000000000000042000D020000000400
000010000000042000F01000000D842005A050000000400000001000000004200
7601000000C042005505000000040000000200000000042008E01000000A842000
801000000304200A07000000174372797074F67726170686963204l6C676F72
```

Figure 2: Input trace to the cryptographic key management service.

And the returned command can be defined by the following trace:

```
42007801000000C042007701000000484200670100000020420068020000000
4000000010000000042006902000000040000000000000000042008F09000000
08000000004AC0731C42000D0200000004000000010000000042000F01000
0006842005A0500000004000000010000000042007C050000000400000000
0000000420079010000004042005505000000040000000200000000042009107
0000002434366563613930612D346232302D343233632D623936382D37353
0663239396630636137000000000
```

Figure 3: Output trace from the cryptographic key management service.

Implementation model has been designed following the next class deployment:

es.openkmip.attributes: are comprised by attributes that the objects within the application should have. Many of them are closely linked with the key deployment in business environments.
es.openKmip.baseobject: These are the objects that comprise most of the commands integrated in the deployment of the key interoperability protocol. In this section, the credential display for the identification of who executes the action is implemented, as well as the associated cryptographic tasks permissions.
es.openKmip.ManagedObject: These are the templates that integrate objects which are managed in the protocol deployment.
es.openKmip.messajecontents: These are all necessary labels for providing attributes to the key generation processes.
es.openKmip.operations: All operation supported by key management protocol. These operations are designed for carrying out administrative tasks.
es.openKmip.pkcs11Bridge: This is a bridge that links attributes transmitted by the proprietary protocol commands, and are transferred to a generic interface (educational implementation) called IAIK.

This library has been designed for Java applications and is a cryptographic functions provider for Java classes working with information security. Among the many operations that IAIK performs, are included those related to key generation, both symmetrical and asymmetrical, data encryption and decryption, hash functions, certificate generation

3 SERVICE DEPLOYMENT PERFORMANCE STUDY

It has been performed a multi-thread server base on a Runnable interface programmed in Java language, this interface is implemented on a server that runs several tasks:

a) Command recognition function has been implemented.
b) Structural analysis model, based on the definition in the KMIP protocol has been also implemented.
c) Attributes provided from client have been associated to be set as attributes belonging to the RSA PKCS#11 key generations.

d) Request has been processed, generating a key based on the proposed request generation KMIP protocol model, development is completely service oriented and compatible with the key generation cloud computing scenario.

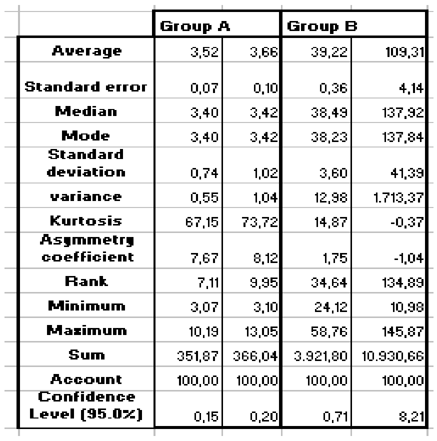
All previous tasks, designed to be concurrent processes, were made using Cryptoki provider library, therefore, executions are fully compatible with the provided standard client executions with their own performance measures.

For the performance test execution process, they have been structured four test operations:

a) 196 bits 3DES key length generation supported by a single execution thread in 100 iterations.
b) 196 bits 3DES key length generation supported by two simultaneous execution threads in 100 iterations.
c) 196 bits 3DES key length generation supported by 10 simultaneous execution threads in 100 iterations.
d) 196 bits 3DES key length generation supported by 50 simultaneous execution threads in 100 iterations.

Performance Remarks. The four selected behaviors have been separated in two great groups; the first one is related to performance oriented of non-transactional access, where sequential accesses can be evaluated and the second one, which evaluates concurrent transactional operations.

Table 1: Performance descriptive statistics for transactional group (A) and non transactional group (B) (data in ms).



	Group A		Group B	
Average	3,52	3,66	39,22	109,31
Standard error	0,07	0,10	0,36	4,14
Median	3,40	3,42	38,49	137,92
Mode	3,40	3,42	38,23	137,84
Standard deviation	0,74	1,02	3,60	41,39
variance	0,55	1,04	12,98	1.713,37
Kurtosis	67,15	73,72	14,87	-0,37
Asymmetry coefficient	7,67	8,12	1,75	-1,04
Rank	7,11	9,95	34,64	134,89
Minimum	3,07	3,10	24,12	10,98
Maximum	10,19	13,05	58,76	145,87
Sum	351,87	366,04	3.921,80	10.930,66
Account	100,00	100,00	100,00	100,00
Confidence Level (95.0%)	0,15	0,20	0,71	8,21

Considering the second group, responsible for implementing stress operations, oriented to transactional processes, is very important note that those operations are performed when Login process is done in the cryptographic device so, service only access to key generation functions, once key attributes have been assigned.

3.1 Considerations between Transactional and Non Transactional Operations

Average performances have been observed and those that those belonging to non-transactional group are slightly different, this proves the evidence that cryptographic device performance are constant indeed but it has not the ability to perform simultaneous operations directly.

From the point of view of the transactional group, such evidence is notoriously visible. Performance differences between cryptographic module access processes against simultaneous processes are clear. Simultaneous processes are greatly penalized key generation from the key management interoperability protocol. But regardless of tghe process is running, is guaranteed that those keys are being generated and stored in a secure environment, where all objects are encrypted by their cryptographic module.

Approximations have been made using high degree polynomials, in order to have a mathematical structure that represents the behaviour of the key generation performances because they need to be verified independently in which environments are deployed. The performance figures for non-transactional processes are:

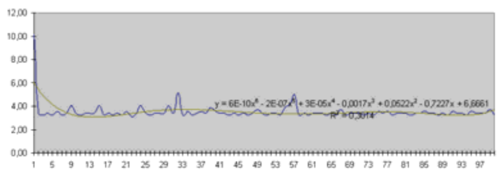


Figure 4: Table of performance returned with a thread in 100 iterations.

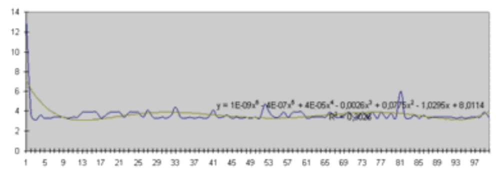


Figure 5: Table of performance returned with two simultaneous threads in 100 iterations.

Both figures presents polynomial representation structures in order to replicate performance models, unfortunately, R2 study has provided a very low trust level in relation to the predictive model. This phenomenon can be credited to the initial warming process, more specifically at the time of opening port process in its initial phase. Once this initial connectivity mechanism has been made, port connections model is involved in a pool of active connections, and initial exchange is not needed. Other important elements performed at the time of the KMIP service startup process, is that initializes the cryptographic storage model, as well as module authentication process , leaving it with an active session, so that KMIP protocol commands are sent and directly processed, no authentication commands that makes impossible unassisted operations are not required in this case.

Transactional processes performance figures are:

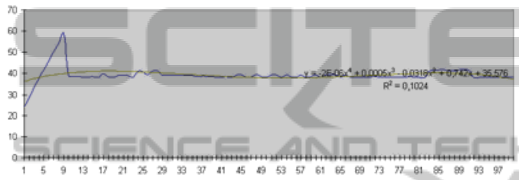


Figure 6: Table of performance returned with 10 simultaneous execution threads in 100 iterations.

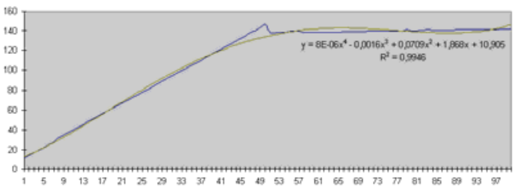


Figure 7: Table of performance returned with 50 simultaneous execution threads in 100 iterations.

By homologous to the non transactional performance study done, we can note that study performed to obtain a mathematical structure that provides the way to estimate the performance for a specific number of simultaneous threads; the analysis for 10 simultaneous threads has obtained a low R2, while for 50 simultaneous threads it can be obtained a structure with an approximate capacity of 99,46%, being this one of the most important elements of this study because it can be sized and predict performances based on the most transactional study case reviewed in this research project.

4 CONCLUSIONS

This research project has presented a set of criteria and guidelines, which have provided the following original results; first study about performance of the interoperable key accessibility (KMIP-OASIS), where it can be seen, by doing a division of substantiated evidences based on the transactional nature of each scenario and each performance returned. Develop a high degree polynomial, in which R2 is minimal in order to have an approach model of the accessibility key protocol conditioning to a specific number of concurrent connections and integrate a recent key management technology using a standard cryptographic hardware access protocol.

REFERENCES

Marchini R. 2010, Paper, Cloud Computing: a practical introduction to the legal issue, BSI Standards. Avaliable at: http://www.whsmith.co.uk/CatalogAndSearch/ProductDetails.aspx?ProductID=9780580703225, Last Update: Undefined.

Martin Luther. 2008, Article, Key Management Infrastructure for Protecting Stored Data, IEEE Computer. Avaliable at: http://www.tschofenig.priv.at/wp/wp-content/uploads/2008/05/r6secu1.pdf, Last Update: Undefined.

Mather T, Kumaraswamy S. 2009. Book, Cloud security and Privacy, O’Reilly Media.

Oasis, 2009. Oasis Members Form Key Management Interoperability Protocol (KMIP) Technical Committee, Available: www.oasis-open.org, Last Update: Undefined.

Oasis, 2009, KMIP: Key Management Interoperability Protocol., WP. Avaliable at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip Last Update: Undefined.

Fajardo Ed. And R. Marin-Lopez, 2009 , RFC 5609 - State Machines for the Protocol for Carrying Authentication for Network Access (PANA), Network Working Group., Available: http://datatracker.ietf.org/wg/pana/charter/,. Last update: undefined.

Rittinghouse J,. 2009, Cloud Computing: Implementation, Management and Security, CRC Press, Available: http://www.crcpress.com/product/isbn/9781439806807 , Last update: undefined.

Rungren A,. 2008, KeyGen2 Presentation, V0.31,, Available: http://webpki.org, Last update:Undefined.