Secured Services in Cloud Computing Environment

Sahar Mohammed Abduljalil
Student
Department of Information Systems, Faculty of Computers and Information
Cairo University, Cairo, Egypt

Osman Hegazy
Professor
Department of Information Systems, Faculty of Computers and Information
Cairo University, Cairo, Egypt

Ehab E. Hassanein
Associate Professor.
Department of Information Systems, Faculty of Computers and Information
Cairo University, Cairo, Egypt

ABSTRACT

Securing data is always of vital importance and because of the critical nature of cloud computing and large amounts of complex data it carries, the need is even important. Cryptographic algorithms are one of the most important areas in security. They are processes that protect data by making sure that unwanted people can't access it. Unfortunately, when the dataset size is huge, both memory use and computational cost can still be very expensive. In addition, single processor’s memory and CPU resources are very limited, which make the algorithm performance inefficient. Security issues investigated in [1] requires that applications and services be capable of supporting a variety of security functionality such as authentications, authorization, and auditing and so on. Those mechanisms are likely to evolve over time because new mechanisms are developed and changed. So developer must avoid embedding security mechanisms statically and manually in order to adapt to changing requirements. The Cloud security service proposed casts those security functionalities in to a service. These strategy allows interfaces to be defined and permits an application to outsource security functionality from the cloud security service to fit it’s current need. We are addressing in this paper a clear separation of concerns between the “business logic” and the “security logic” in order for any service implementing the proposed security service to be considered a high level secured service. This model is targeting developers willing to write secured services without burdening the developer of continuously rewriting security routines, and only be concerned with the business logic of the service, on the other hand it targets the end user that need to use the service as it is, and get the result or output.

General Terms

Cloud Computing, Security, Cloud services.

1. INTRODUCTION

Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers to handle applications [2]. Cloud computing is a colloquial expression used to describe a variety of different types of computing concepts that involve a large number of computers connected through a real-time communication network. Cloud computing refers to computing with a pool of virtualized computer resources. A cloud can host different workloads, allows workloads to be deployed/scaled-out on-demand by rapid provisioning of virtual or physical machines, supports redundant, self-recovering, and highly-scalable programming models and allows workloads to recover from hardware/software failures and rebalance allocations. The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at datacenters. In a cloud computing system, there's a significant workload shift. Local computers no longer have to do all the heavy lifting when it comes to running application[3]. The network of computers that make up the cloud handles them instead. Hardware and software demands on the user's side decrease. The only thing the user's computer needs to be able to run is the cloud computing systems interface software, which can be as simple as a Web browser, and the cloud's network takes care of the rest [4] [5].

2. RELATED WORK:

NIST defines the term computer security as, “The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications).”Security is the mechanism by which information and services are protected from unintended or unauthorized access, change or destruction. Security in networking is based on Cryptography (a word with Greek origins, means “secret writing”), the science and art of transforming messages to make them secure and immune to attack [6]. Encryption is one of the principal means to guarantee security of sensitive information. Many encryption algorithms are widely available and used in information security. This paper explores some security algorithms named AES, RSA, and then finally digital signature, followed by a detailed explanation of the proposed model. Finally, implementation, evaluation, and experimental results in section 5 are reported. Conclusion and future work are described in section 6.

2.1 Symmetric Algorithm: Advanced Encryption Standard

The Rijndael algorithm is selected by National Institute of Standards and technology (NIST) as a new Advanced Encryption Standards (AES). It converts data to an unintelligible form called cipher text; decrypting the cipher text converts the data back into its original form, called plain text. AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. As the AES algorithm may be used with three different key lengths, these three different ‘‘flavors’’ are generally referred to as ‘‘AES-128’’, ‘‘AES-192’’, and ‘‘AES-256’’ [7].

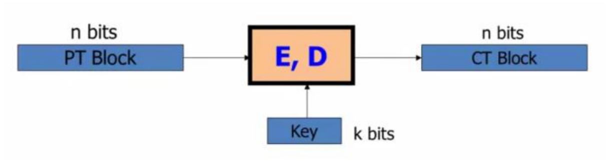

Figure 1: Block Cipher

The AES algorithm takes the Master Key K, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of 11 sub-key arrays of 16 words of 8 bits, denoted by w_i taking into account that the first sub-key is the initial key. To calculate every w_i (except w_0) the routine uses the previous $w_{i-1}$and two tables, RCon and S-Box [8].

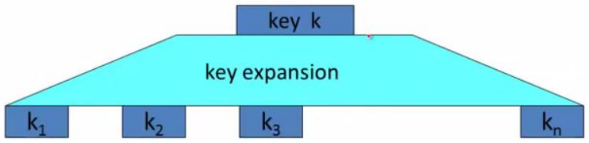

Figure 2: Key Expansion

The algorithm operates in eleven rounds. The first round performs only the AddRoundKey transformation, while the middle 9 rounds perform all four transformations. The final round performs the ByteSub, ShiftRow, and AddRoundKey transformations, omitting the MixColumn operation [9].
The algorithm requires 11 rounds. Each round operates on the state, a 4 x 4 matrix of 8-bit values. Each round involves up to four basic transformations [10]:

- Byte Substitution (ByteSub)
  The SubByte transformation is a non linear byte Substitution, using a substation table (s-box), which is constructed by multiplicative inverse and affine transformation.

- ShiftRow
  In the ShiftRow transformation, the bytes in the last three rows of the State are cyclically shifted over 1, 2 and 3 bytes, respectively. The first row is not shifted. The offset of the left shift varies from one to three bytes.

- MixColumn: operates on the State column-by column, treating each column as a four-term polynomial.

- AddRoundKey: The AddRoundKey Transformation performs an operation on the State with one of the sub-keys. The operation is a simple XOR between each byte of the State and each byte of the sub key. This transformation is its own inverse [11].

2.2 Asymmetric Algorithm: RSA public key Algorithm

The most popular public key algorithm is the RSA (named after their inventors Rivest, Shamir and Adleman).

-Public key algorithm that performs encryption as well as decryption based on number theory
-Variable key length; long for enhanced security and short for efficiency (typical 512 bytes)
-Varible block size, smaller than the key length
-The private key is a pair of numbers (d, n) and the public key also a pair of numbers(e,n)
-Choose two large primes p and q (typically around 256 bits)
-Compute n= pxq and z = (p-1) x (q-1)
-Choose a number d relatively prime to z
-Find e such that e x d mod (p-1) x (q-1) =1
-For encryption: C=Pe(mod n)
-For decryption: P=Cd(mod n) [12,13]

2.3 Digital Signature:

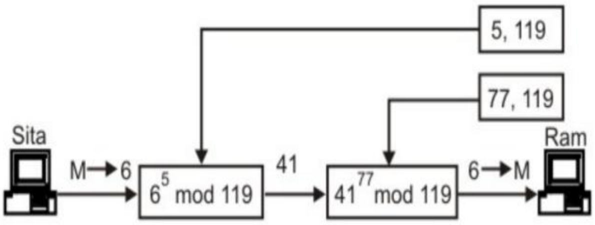

Figure 3: The RSA public key encryption technique.

A message digest (or hash) is a fixed-length value obtained on some message. This message digest value is always guaranteed to be the same for the same message. If we change the message even by a single bit, the message digest would change [14]. Hence, message digests can be used to ascertain the fact that a message has not been changed or tampered with, since it was created. Digital Signature can be classified into two processes:

Signing and encryption:

- Hashing: In this step small message digest is computed which is unique representation of the message. This evaluation ensures the message integrity. The digital signature is applied to this smaller message digest. This evaluation generates a unique code.
- Encryption: In this step message digest is encrypted using private key of the sender. It is used to sign the message digest. The original message can be recovered by decrypting the message signature using corresponding public key of sender. To obtain non-repudiation, Signing is performed.
- Packing: The plain message, message signature and the Public Key of the sender are packed together into a single packed unit.
- Encryption: The single packed unit of message which contains plain message and message signature along with the public of the sender is encrypted using receipt’s public key to form signed and encrypted message.

Decryption and verification:

- Decryption: In these steps the received message which is signed and encrypted is decrypted using the receiver’s private key to form a packed unit of message containing plain message, the signature and the public key of the sender.
- Unpack: The decrypted message in last step is unpacked into plain text message, message signature and the public key of the sender.

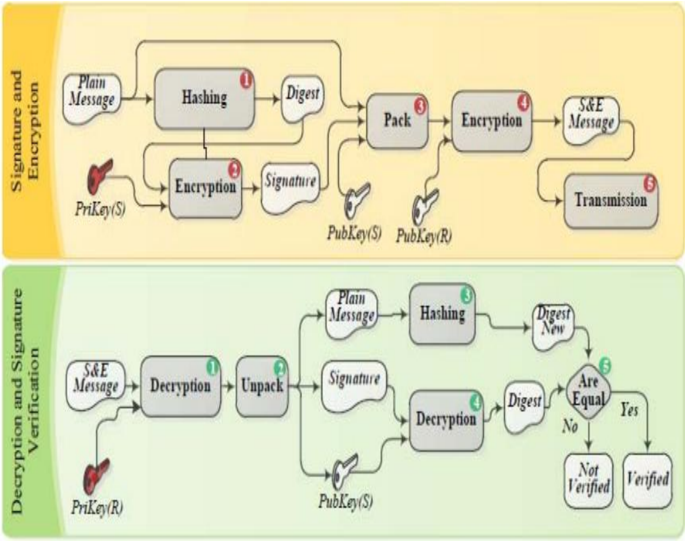

Figure 4: Digital Signature – Process at Sender's and Receiver's End/ adapted from [15]

2.4 Service Orchestration:

In the OASIS terminology as in (OASIS), SOAs are centered on *providers* that offer *capabilities* (sometimes called *enablers*). When a *consumer* invokes one of these capabilities through an *interaction* with the provider, some *real-world effect* occurs. As an example, when the consumer interacts with the hotel booking service, a room is booked a real world effect. A capability together with its *specification*, *contract*, and *real-world effect* is known as a *service*.

Providers have to make their services *visible* to allow potential consumers to *discover* them. They do this by publishing (exposing) a *service description* containing information about three aspects of the service: its *behavior*, its *interface*, and its *policies* and *contracts*. For example, the potential suppliers of weather forecasting services advertise the service description of their services, allowing consumers to discover them and interact with them. The interface description includes the specific protocols, commands, and information exchange by which actions are initiated that result in the real-world effects, as specified through the service functionality portion of the service description. Service providers and service consumers are together known as *service participants*.

Once a service has been made visible, it can be combined with other visible services to create a higher-level or *orchestrated* service that typically implements a particular *business process*, as shown in figure 5.

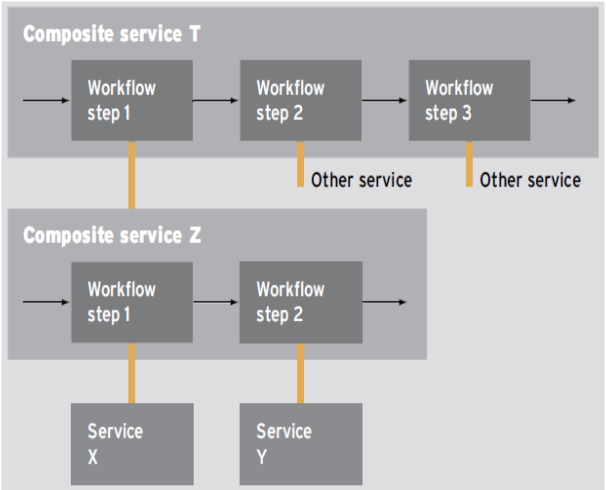

Figure 5: Composite Serivces/ adapted from [16]

3. PROPOSED WORK:

In this section, a new development model is proposed in order to write secured services. This will help the developer not to continuously rewrite security routines. The contribution is to promote a clear separation between the business logic of the cloud service which specifies the process by which users functional requirements are met, and the management logic security concern. In other way we are decoupling security concerns from the cloud service business logic. All mechanisms related for instance to security are confined and factored out into a cloud service and are not scattered over the business logic of the service**.** Cloud Security Manager Model is proposed based on the security algorithms, web services, and security approaches covered in section 2.1, 2.2, 2.3 and 2.4.

We propose a Cloud Security Manager model using the Service Oriented Architecture (SOA) to deliver a secured service. SOA and Web services covered in section 2.3 are one of the most important enabling technologies for cloud computing in the sense that resources (e.g., software, infrastructures, and platforms) are exposed in clouds as services. Secure operation requires that applications and services be capable of supporting a variety of security functionality, such as authentication, authorization, credential conversion, auditing, and delegation [17]. Interaction between services requires having a range of security requirements and mechanisms. These mechanisms and requirements are likely to evolve over time as new mechanisms are developed or policies change. According to the needs of the cloud users, they can appropriately choose the solutions available for their identity and access requirements, trust and privacy needs. Similarly, cloud providers can register to the proposed model and ensure their security requirements. The following section explains in details how the model works and renders the appropriate service to both the cloud service consumer and provider. In figure 4.1, When a developer needs to implement a system, they need to think about the logic and processes of the system then to implement the security logic. So if we are going to explain it as variables, lets say that any ERP system consist of the business logic (BL) and security logic (SL).

ERP1=BL1+SL
ERP2=BL2+SL
ERPn=BLn+SL

Where BL is the business logic varies according to the processes in the service, while the security logic remains constant, bearing in mind that security issues likely to evolve over time and new mechanisms are developed or policies change. As a result, security logic needs to be gathered as a separate service, instead of duplicating the security code over and over again in each service. In this way, we have tried to solve the problem of adaptability that concerns any software system. It became easier for security logic to be checked, updated and reviewed when gathering the security in one separate service. Moreover casting all security functionalities in a secured cloud service allows it to be located and used as needed by applications.

The working of the cloud security manager model involves four phases: Enrolment phase, Credential processing service phase, authorization service phase, and finally Service rendering phase.

-Enrolment phase: Users need to enrol themselves to the Cloud Security Service, this is, as the cloud service consumer, and cloud developer, but the enrolment procedure for both are totally different with respect to the data collected. A cloud developer will outsource and use some functionalities from the cloud security service as needed by the application, while the end user will send an input to the service, processing will be done on this input by the cloud security service, and finally output will be sent. So a registry is maintained for the service developer associated with the service ID that he is authorized to use. While for the cloud service user, login credentials data are collected, and a passphrase is needed for every user, moreover, all this login data will be stored encrypted by a generated one time public and private key, using RSA algorithm that have been elaborated in sections above.

-Credential Processing Service: This service validates the details of processing and validates if authenticated. In the proposed model, Challenge-Handshake Authentication Protocol (CHAP) was used for authentication. This protocol says that when a client demands data or any service of cloud computing. Service Provider Authenticator (SPA) first requests for client identity. The whole process is explained in figure 6 given below.

Authentication is performed in three steps:-

1. When client demands a service, Service Provider Authentication sends a "challenge" message to client.

2. Client responds with a value that is calculated by using one way hash function on the challenge.

3. Authenticator verifies the response value against its own calculated hash value. If the values match, the Cloud provider will give service, otherwise it should terminate the connection.

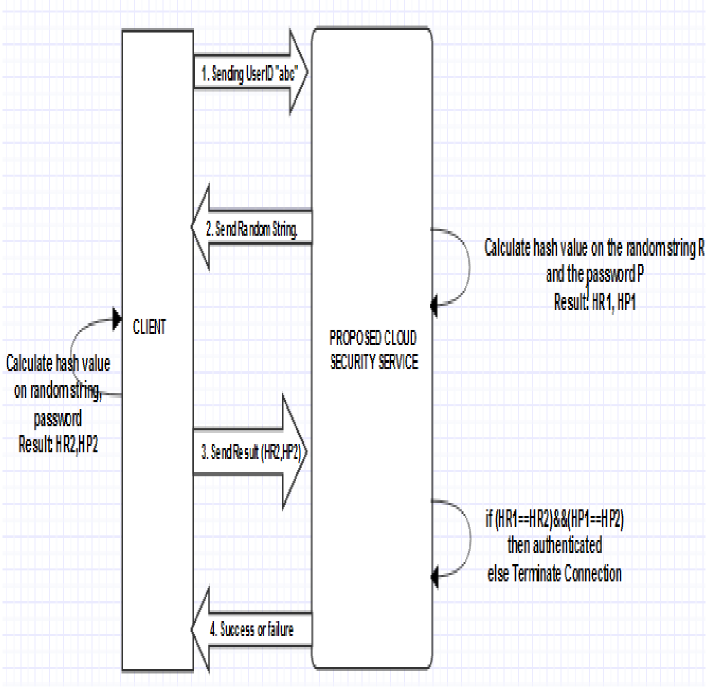
1. Sending UserID "abc"
2. Send Random String.
Calculate hash value on the randomstring R and the password P Result: HR1, HP1
CLIENT
PROPOSED CLOUD SECURITY SERVICE
Calculate hash value on randomstring, password Result: HR2,HP2
3. Send Result (HR2,HP2)
if (HR1==HR2)&&(HP1==HP2) then authenticated else Terminate Connection
4. Success or failure

Figure 6: Authentication process acquired from Cloud Security Service

-Authorization Service: Different security mechanisms are classified as cryptography, biometric based, Audibility, Digital signature, Authentication services, so we need a service that says which service is authorized to be used by which user. But as for the implementation part, I have used cryptography, digital signature, and authentication.

-Service rendering:

The service rendering phase explains how the cloud users are significantly leveraging the benefits of the Security Management Service for cloud security. The registered users can now avail the service.

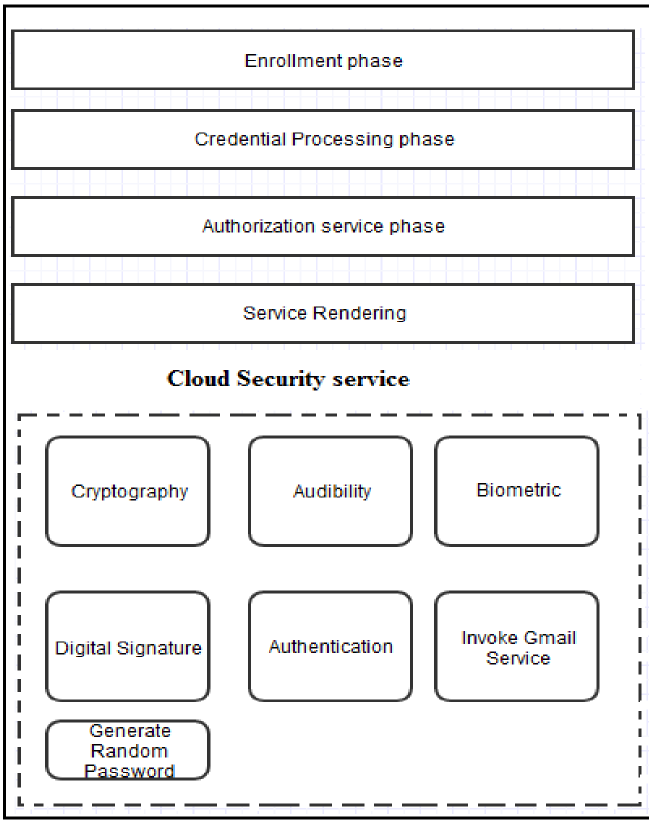

Figure 7: Working of proposed cloud security service.

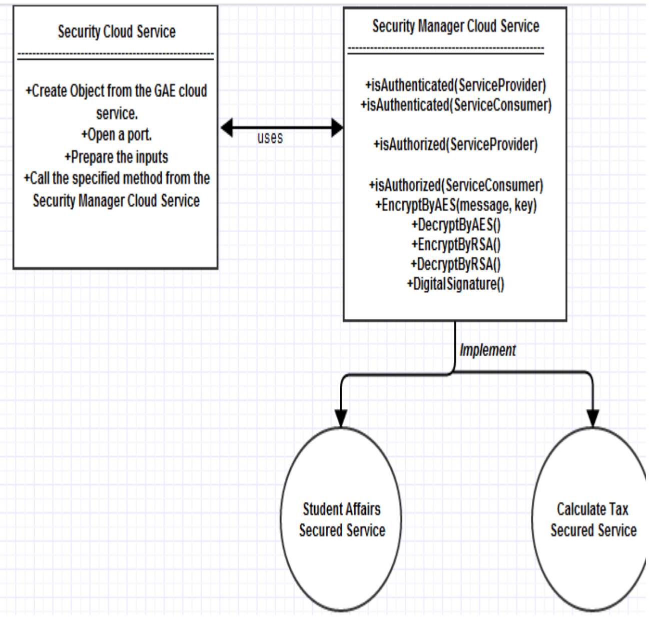
Security Cloud Service
+Create Object from the GAE cloud service.
+Open a port.
+Prepare the inputs
+Call the specified method from the Security Manager Cloud Service
uses
Security Manager Cloud Service
+isAuthenticated(ServiceProvider)
+isAuthenticated(ServiceConsumer)
+isAuthorized(ServiceProvider)
+isAuthorized(ServiceConsumer)
+EncryptByAES(message, key)
+DecryptByAES()
+EncryptByRSA()
+DecryptByRSA()
+DigitalSignature()
Implement
Student Affairs Secured Service
Calculate Tax Secured Service

Figure 8: Detailed Proposed Model

4. IMPLEMENTATION AND EVALUATION:

Google App Engine has been introduced, and the proposed model has been migrated to and evaluated on the Cloud in terms of security, number of line of codes, and execution time. Results show efficiency and capability to run on the Cloud. Google App Engine is a PaaS solution, which presents its developers with a platform to run web applications that support Java, Python and Google Go languages.

- A SOAP based web service for ***Security Manager*** class is created with webmethods 'encryptAES', 'decryptAES', 'encryptRSA', 'decryptsRSA', 'digitalSignature', 'generateRandomPaaword', 'digestbyMD5', and 'invokeGmailService'.

- ***Security*** web service was implemented and it consumes the Security Manager web service. So it is a service consumer, and provider, where it consumes the Security Manger webservice, and provides a direct service for any other services to be a securely deployed webservice.

- Testing the proposed model by implementing ***SecuredExamService*** that *implement* Security Service.

- Experimental evaluation was done on eclipse-SDK-App Engine 1.8.0. Also, each one was run on different input sizes: 10kb, 100kb, 500kb, 1000kb and 2000kb. The comparison (uniprocessor) running time and running time on the cloud was done by calculating the Speed-Up Ratio. Speed-Up Ratio is defined as the ratio of mean processing time on a single processor to the mean processing time on the cloud. Each algorithm was run multiple times with each input size and the mean value was used for calculations in each case.

Table 1: Mean Processing Time when requesting AES and RSA from Cloud Security Service

Input Size	AES (LOCAL)	AES (CLOUD)	RSA (LOCAL)	RSA (CLOUD)
10 kilobytes	100 ms	88 ms	1200 ms	958 ms
100 kilobytes	300 ms	90 ms	30975 ms	1366 ms
500 kilobytes	1499 ms	91 ms	823847 ms	935 ms
1000 kilobytes	3047 ms	89 ms	3406315 ms	2125 ms
2000 kilobytes	6568 ms	92 ms	13708830 ms	1672 ms

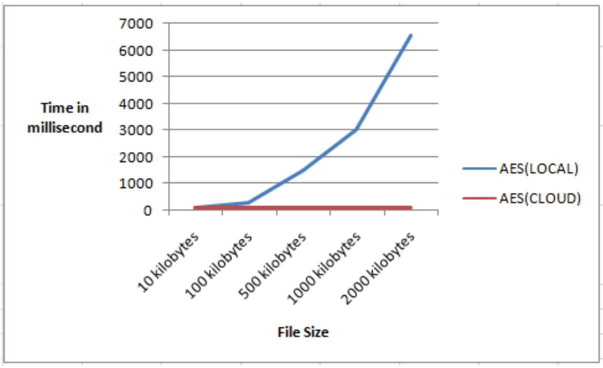

Figure 9: Mean Processing Time when requesting AES from Cloud Security Service

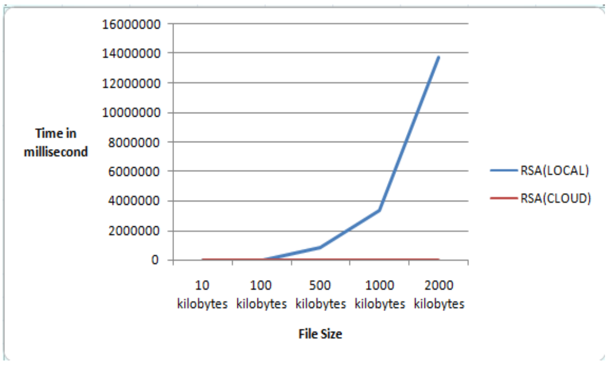

Figure 10: Mean Processing Time when requesting RSA from Cloud Security Service

5. CONCLUSION

In this thesis, a model has been proposed, where security workload is totally separated from the service logic. This research found that it is possible to separate security functions from business activities, encapsulate them into services, a reusable secured service, and a business service, then combine or merge them to achieve secured services without burdening the developer from coding the security code over and over again, so the developer of the service will concentrate on the business logic of the service itself not the security issues. To sum up, one time security service is done, but used several times with many services independent from each other. So proposed model is satisfying and targeting developers who are building cloud-based services where they can implement directly the security logic in order to get a secured cloud services. The scope of the security service was including AES file encryption system, RSA system for secure communication, Onetime password to authenticate users and MD5 hashing for hiding information. However, as the number of users using the security service increase, accessing this centralized service can experience delays. This separation of security logic from the business logic has given us some merits where multiple security mechanisms can be added and updated in the security management service directly. Moreover, dynamic creation of services, users must be able to create new services dynamically without administrator intervention.

Using Eclipse IDE, java run time environment, and GAE we have implemented the model in the form of encryption, and decryption algorithms which have been discussed above.

6. REFERENCES

[1] Sahar Mohammed Abduljalil, O. H. (May 2013). A Novel Approach for Handling Security in Cloud Computing. *International Journal of Computer Applications* , 9-14.

[2] http://www.webopedia.com/TERM/C/cloudcomputing.htmlcited on Aug 21, 2013

[3] Strickland, J. (n.d.). *How Cloud Computing Works*. Retrieved November 20 , 2012, from HowStuffWorks: http://computer.howstuffworks.com/cloud-computing/cloud-computing.htm

[4] https://bluelabelhost.com/whatisthecloud, cited on Aug 21,2013

[5] Gurpreet Kaur, M. M. (2013). Analyzing Data Security for Cloud Computing Using Cryptographic Algorithms. *Journal of Engineering Research and Application* , 782-786.

[6] Gurpreet Singh, S. (2013). A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security. *International Journal of Computer Applications* , *67*, 33-38.

[7] Amanpreet Kaur, G. R. (March 2013). Secure Broker Cloud Computing Paradigm Using AES And Selective AES Algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering*

[8] Seyed Hossein Kamali, M. H. (2010). A New Modified Version of Advanced Encryption Standard Based Algorithm for Image Encryption. *International Conference on Electronics and Information Engineering*

[9] Mehboob Alam, W. B. (2002). A Novel Pipelined Threads Architecture for AES Encryption Algorithm. *IEEE* , 296 - 302.

[10] Abha Sachdev, M. B. (April 2013). Enhancing Cloud Computing Security using AES Algorithm. *International Journal of Computer Applications* , *67*.

[11] G. Jai Arul Jose, C. S. (2011). Implementation of Data Security in Cloud Computing. *International Journal of P2P Network Trends and Technology-* , 2249-2615.

[12] Esh Narayan, M. M. (Sep. 2012). To enhance the data security of cloud in cloud computing using RSA algorithm. *International Journal of Software Engineering.*

[13] Eman M.Mohamed, H. S.-E. (2012). Enhanced Data Security Model for Cloud Computing. *The 8th International Conference on INFOrmatics and Systems*

[14] Kahate, A. (2007, july 11). *What are Digital Signatures? Compute and Verify a Digital Signature Using Java*. Retrieved January 4, 2012, from Indic Threads: http://www.indicthreads.com/1480/what-are-digital-signatures-compute-and-verify-a-digital-signature-using-java/

[15] Ravneet Kaur, A. K. (2012). Digital Signature. *International Conference on Computing Sciences, IEEE* .

[16] Hobbs, C. a. (2006). Time-sensitive Service-Orient. *Nortel Technical Journal* .

[17] Jean Bacon, D. E. (2010). Enforcing End-to-end Application Security in the Cloud. *International Conference on Middleware*, (pp. 293-312).