ISSN 2229-5518

# An Efficient Authenticated Key Exchange Scheme for Security Aware Scheduling of Scientific Applications

N Ramadevi, Department of Computer Science and Engineering
Santhiram Engineering College, Nandyal

**Abstract**— Cloud computing is a pool of resources on network, that are dynamically provisioned on demand to its users based on pay-per-use policy. But it is facing certain issues of security, trust and efficiency when implemented on large enterprise such as scientific applications. In this paper I proposed a novel model Efficient Authenticated Key Exchange (EAKE) scheme that will provide solutions for the aforementioned issues. The proposed work mainly focuses on using the symmetric key cryptography for fast scheduling of tasks [13] and the Internet Key Exchange (IKE) scheme with randomness-reusability for key exchange.

**Keywords**— Cloud computing, CLC, Internet key exchange (IKE), Randomness-reusability, Authenticated key exchange.

## 1 Introduction

Cloud computing is a hybrid environment where it covers a large pool of resources deployed or implemented on heterogeneous platforms over network [1], [2]. The definition of cloud computing introduced by the U.S. National Institute of Standards and Technology (NIST) is cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing is a computing environment where the user will not bother setting up and maintaining their own computing resources, and can avail the infrastructural facilities in a pay-as-you-use mode [3], which is particularly a cost-saving solution in data and computation intensive applications such as scientific research [2], [ 4].

International IT corporations are advertising on their commercial cloud platforms such as Amazon EC2/S3, Microsoft Azure, Google App Engine and IBM SmartCloud, while a large number of enterprises and institutions have established their own private clouds. Fig.1 shows the typical architecture of a cloud computing.

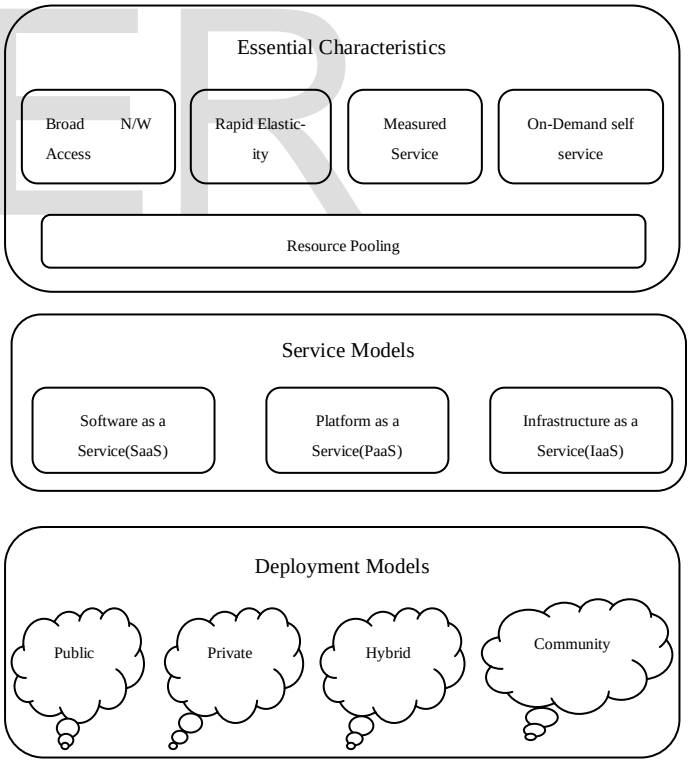


Fig.1. Typical architecture of a cloud computing

ISSN 2229-5518

### 1.1 SERVICE MODELS IN CLOUD

Cloud computing offers three fundamental service models [5]. The brief descriptions are as follows.

#### *1.1.1 Software-as-a-service (saas).*

It is the service that provides the consumer with the capability to use the provider's applications running on a cloud infrastructure [6]. The applications are accessible from various client devices, through a thin client interface, such as a web browser. (e.g. Google Docs, Salesforce.com, Microsoft Azure etc.,)

#### *1.1.2 Platform-as-a-service (paas).*

It is the service that provides the consumer with the capability to develop onto the cloud infrastructure; consumer creates the applications using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage but has control over the deployed applications and possibly application hosting environment configurations [6]. (e.g. Microsoft Azure service platform, Google App Engine etc.,)

#### *1.1.3 Infrastructure-as-a-Service (IaaS).*

It is the service that provides the consumer with the capability to provision processing, storage, networks, and other fundamental and run arbitrary software, which can include operating systems and applications [6]. The consumer has control over operating systems, storage, deployed applications, and possibly limited control of select networking components. (e.g. Amazon EC2,S3, Sun's cloud service etc.,)

### 1.2 DEPLOYMENT MODELS IN CLOUD

The four deployment models for cloud architecture are described below.

#### *1.2.1 Private cloud.*

It is owned or rented by an organization. The whole cloud resource is dedicated to that organization for its private use.

#### *1.2.2 Public cloud.*

It is owned by a service provider and its resources are sold to the public. End users can rent parts of the resources and can typically scale their resource consumption up (or down) to their requirements.

#### *1.2.3 Community cloud.*

It is similar to the private cloud, but where the cloud resource is shared among members of a closed community with similar interests.

#### *1.2.4 Hybrid cloud.*

It is the combination of two or more cloud infrastructures; these can be either private or public, or community clouds. The main purpose of a hybrid cloud is usually to provide extra resources in cases of high demand, for instance enabling migrating some computational tasks from a private cloud to a public cloud.

Security of data in cloud is one of the major issues which act as an obstacle in the implementation of cloud. To ensure adequate security in cloud computing, various security issues, such as authentication, confidentiality and integrity, and non repudiation, all need to be taken in to account.

## 2. MOTIVATING EXAMPLE AND PROBLEM IDENTIFICATION

In a typical cloud computing infrastructure, a central server (CLC) is employed for receiving and processing user requests in the front, and also responsible for scheduling and splitting tasks through MapReduce in the back [7], [8]. The virtualized server instances running on clusters of servers are responsible for processing the divided tasks in a parallel fashion and returning the results afterwards, and then CLC is capable of assembling the results and return to the user. Fig. 2 depicts the structure of a typical cloud system for scientific applications.

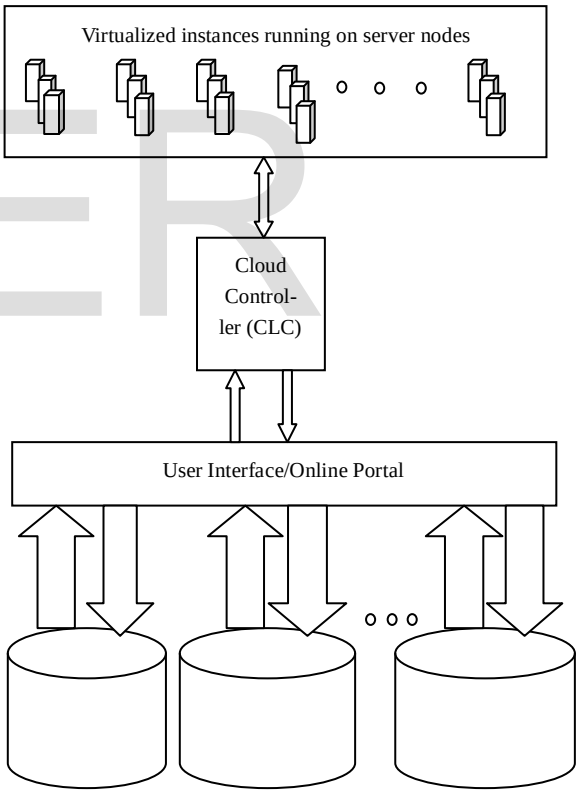


Fig. 2. A typical cloud computing environment for scientific applications

Scientific applications are often data and computation intensive, and are time critical. Deploying scientific applications in cloud computing is extremely cost saving, although there are still several challenges standing. First, data generated from scientific research are intellectual property

which may directly lead to scientific discovery, hence their value is inestimable. Therefore, data security, specifically confidentiality and integrity, are of extreme importance. Second, scientists usually need to access the results as early as possible, as a late-coming result may cause an enormous economical waste and loss of scientific discovery. Hence efficiency is also important in cloud scheduling for scientific applications. These two challenges constitute the main research work to tackle.

Data security in cloud computing is more challenging than in traditional distributed systems due to cloud's unique characteristics of consolidation and multi-tenancy. Therefore it is essential to encrypt all data in transfer as well as to authenticate each party's identity to ensure their confidentiality and integrity. Asymmetric key encryptions cannot be directly applied to our scenario due to the low efficiency of asymmetric encryptions/decryptions over large data sets. To utilize a symmetric encryption scheme, CLC must at first exchange a session key with each server instance through an authenticated key exchange scheme. And this must be done before distributing the divided data and tasks to server instances. All virtualized server instances are isolated to avoid additional security threats brought by virtualization [9]. This can be considered for risk management purposes. For example if a malicious adversary gets to know one of the session keys through side-channel attacks or other methods, he or she will have access to only that particular server instance data.

In a cloud computing system there are always numerous server instances running on background servers, and this number is even larger in large-scale cloud systems such as public clouds and commercial clouds. As a distinct session key is needed to be exchanged between CLC and each server instance, conducting key exchange between CLC and each server instances becomes a tedious and time-consuming task, especially in data-intensive applications such as scientific applications.

Based on the analysis above, we can formulate the research problem here-with virtualization technology as an Efficient Authenticated Key Exchange (EAKE) scheme for security aware scheduling of scientific applications in cloud.

## 3.PROBLEM ANALYSIS AND PROPOSED SCHEME (EAKE)

As the CLC performs key exchange operations with all server instances and encryptions/decryptions of the entire data set, it is acting as the bottle neck of the performance of our scheme (EAKE). By utilizing the square-and-multiply algorithm, the minimum complexity of a modular exponentiation operation of xy mod p is $O((\log_2 x)^3)$ for integers x, y, p where x, y < p. Therefore modular exponentiations are by far the most costly, as most other operations in authenticated key exchange schemes are linear complexity (e.g. symmetric encryptions, pseudo random functions etc.,)

According to the fast evolving capability of computation facilities, it is now unsafe to utilize in a typical application a Diffie-Hellman group with its size less than 1024 bits [10], [ 11]. Hence it all comes down to Diffie-Hellman key exchange operations performed on CLC, which involves n times 1024 bit modular exponentiations for n server instances in each key exchange rounds, as n tends to be very large in typical cloud computing systems.

Digital signatures are always necessary in key exchange schemes for identity authentication, and could be time-consuming if the message is long. But in key exchange schemes, messages to be signed are usually of a short fixed length. In this regard time consumption in signing messages is small compared to those key exchange related computations over 1024 bit keying materials.

Computations on server instances in key exchange process can be completed almost instantly. Besides, data transfer takes almost no time as well because only kilobytes of data need to be transferred between the CLC and server instances in order to complete key exchange.

Based on the analysis above, we argue that the performance of the CLC is the bottle neck, mostly because of the amassed Diffie-Hellman operations performed on CLC. In our proposed scheme (EAKE), we will incorporate the randomness-reuse strategy [12] to reduce the number modular exponentiations on CLC, in order to mitigate the workload of CLC and to achieve superior overall efficiency over IKE in cloud computing environment.

Notations:

The following are the some notations used in the proposed scheme.

| | |
|---|---|
| S: | Server instances domain |
| Si: | The ith server instance in S |
| C: | Cloud Controller (CLC) |
| HDRi: | Header contains security parameter indexes. |
| CertReqi: | Certificate request, requesting i's certificate |
| Certi: | Certificate |
| Ni: | i's one-time nonce |
| SA: | Security associations, used in negotiating cryptographic algorithms |
| IDi: | Identity information |
| Sigi: | i's signature, can be verified using pre-defined algorithm |

ISSN 2229-5518

| | |
|---|---|
| | and public key within Certi |
| prf ( ): | Pseudo random function |
| {M}k: | Encrypt message M with session key k |

The proposed scheme works in two phases.

### 3.1 SYSTEM SET UP:

The system chooses a large prime integer p to form a Diffie-Hellman group, and a generator g of group zp*, i.e., g is a primitive root modulo p. p is a Sophie German prime where (p-1)/2 is also prime, so that group zp* maximizes its resilient against square root attack to discrete logarithm problem. A certificate authority (CA) is needed in our security framework so that communicating parties can identify each other through exchanging verifiable certificates Certc and Certsi, as the certificates contain public keys which can be used to verify the session partner's signatures, there by their identities. Certificates are issued to all participants of communication by CA before the commencing of communication, CA will not be participating itself unless re-verification of identities and revocation and re-issuing certificates for participants are needed. As these should be done in a much lower frequency, they will not affect the efficiency of a key exchange scheme for scheduling in general.

### 3.2 KEY EXCHANGE:

Key exchange is used when a new task is to be executed, because that is when CLC needs to decide how to distribute this new task to be executed on existing computation infrastructure. i.e., which of the server instances are involved. CLC picks a secret value x < p, computes its public keying material gx in zp*, and broadcasts the following message to the domain of server instances S which contains n instances S1, S2 ,. . . , Sn;

Round 1, C → S: HDRc, $SA_{c1}$, $g^x$, Nc

Where HDR and SA for algorithm negotiation, $g^x$ for Diffie-Hellman key exchange, and N for freshness verification. The initiator of a normal IKE scheme will generate n secret values x1, x2, . . . , xn, then compute and send out $g^{x1}$, $g^{x2}$, . . . , $g^{xn}$, either through multicast or one by one to establish separated security channels with each server instance. In our scheme although we still establish one SA for each server instance Si where i=1, 2, . . . , n, we are using only one single secret value x for CLC in all n messages in order to reduce cost.

Upon receiving message 1, each server instance generates their secret value yi < p, compute key material $g^{yi}$, then respond within Round 2 as follows.

Round 2, S → C: $HDR_{Si}$, $SAS_{i1}$, $g^{yi}$, $N_{si}$, $CertReq_c$,
for i=1,2, … n.
After exchanging the first two rounds of messages, the session keys $g^{xy1}$, $g^{xy2}$, …., $g^{xyn}$ are computed for all parties as follows.

C: $g^{xy1} = (g^{y1})^x$, . . . , $g^{xyn} = (g^{yn})^x$
S1: $g^{xy1} = (g^x)^{y1}$
.
.
.
Sn: $g^{xyn} = (g^x)^{yn}$.

The session keys are now shared between CLC and each server instances for the use of encryptions. Although the Diffie-Hellman key exchange is completed, the initial exchange is not finished as the participants have to authenticate each other in order to prevent man-in-the-middle (MITM) attacks. Similar as in IKE, CLC generates signatures Sigci which are the signatures for these n messages, using its secret key from the key pair issued by CA.

$M_{ci} = prf(prf(N_c||N_{Si}||g^{xyi})||g^x||g^{yi}||SA_c||ID_c)$
For i=1,2,…n
And broadcast the following message to S:
Round 3, C→S: $HDR_c$, $\{ID_c\ SA_{c2}\ Cert_c\ CertReq_{S1}\ Sig_c\}_{gxy1}$ ||
$\{ID_c\ SA_{c2}\ Cert_c\ CertReq_{S2}\ Sig_c\}_{gxy2}$ || . . . ||
$\{ID_c\ SA_{c2}\ Cert_c\ CertReq_{Sn}\ Sig_c\}_{gxyn}$ ||,
for i=1,2, . . . , n.

The server instances can then verify the identity of the initiator of this conversation by using its session key $g^{xyi}$ to decrypt its own part of this message. Signatures can be verified through the public key contained in the certificate. Similarly server instances will send out their own encrypted ID, signature and certificate to CLC for verification.

Round 4, S→ C: $HDR_{Si}$, $\{IDc\ SA_{Si2}\ Cert_{Si}\ Sig_{Si}\}_{gxyi}$, for i=1,2, . . . ,n

Where SigSi is signature by Si to messages: $M_{Si} = prf(prf(N_c||N_{Si}||g^{xyi})||g^{yi}||g^x||SA_{Si}||ID_{Si})$, for i=1,2, . . . ,n.

After the identities of both CLC and server instances are authenticated through round 3 and round 4, CLC will send to S1, S2, . . . , Sn the split task data which are encrypted with session keys $g^{xy1}$, $g^{xy2}$, . . . , $g^{xyn}$ using symmetric encryption such as AES. After task execution S1, S2, . . . , Sn returns to CLC the results which are encrypted using $g^{xy1}$, $g^{xy2}$, . . . , $g^{xyn}$ as well. The prf function is often implemented as an HMAC function such as SHA-1 or MD5, which outputs a fixed length short message (commonly 128 bits), and has high efficiency (around 200 MB/s) itself.

Rekeying is often accomplished by running initial key exchange all over again. However in the following cases, alternative strategies need to be applied.

ISSN 2229-5518

*Failure recovery:*

If any message that constitute the initial exchange fails to arrive, the CLC simply start a one-to-one IKE key exchange session with this specific instance. As this is an accidentally happening situation and can be tackled on-the-run, this additional time consumption can be considered negligible.

*Multi-step tasks:*

In a multi-step task data need to be transferred back and forth. In this situation it is not necessary for the participants to re-authenticate each other after the successful authentication in the first round of data transfer because of the high dependency of data in a same task. Therefore only rounds 1 and 2 are needed to be performed with new keying materials and minor changes to SA and HDR fields.

## 4. EXPERIMENTAL RESULTS

The experiments were performed for both IKE and proposed scheme. The cloud computing infrastructure is made using OpenNebula cloud environment with Linux operating system, and a KVM hypervisor [15] on which Hadoop simulator is used for performing MapReduce [14], [16] programming. Furthermore, we installed the OpenStack [17] cloud environment which is responsible for global management, resource scheduling, task distribution and interaction with users.

It is clear that the efficiency depends mainly on the size of the data sets used and the number of server instances. The experiments were conducted on large number of data sets with different number of server instances. For evaluating our proposed scheme, we implemented two key exchange schemes using java: A basic multi-user IKE scheme and our proposed scheme. For parameters of the Diffie-Hellman key exchange we used a 1024 bit mod p group[10] with a 1024 bit prime p and generator g=2. We used SHA-1 for pseudo random function, RSA algorithm for signature and AES for message encryptions.

As was analyzed in section 3, CLC run-time is the only predominant factor in comparing the efficiency of an authenticated key exchange scheme for scheduling in cloud computing. Based on the time consumption on CLC, the efficiency of the IKE and proposed scheme (EAKE) are demonstrated in Fig. 3

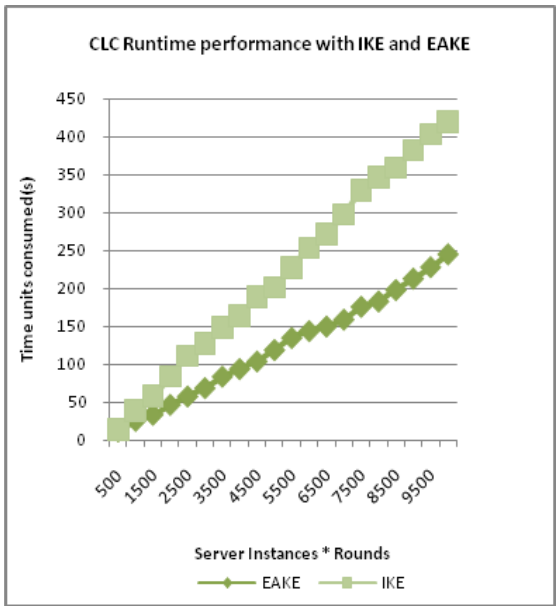


Fig. 3. Performance of EAKE compared to IKE

We can see that our scheme only consumes about half of the runtime on CLC compared to IKE, Which is a significant improvement, and which meets the characteristics of the scientific applications.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a novel authenticated key exchange scheme, which aimed at efficient security aware scheduling of scientific applications in cloud computing. Both theoretical analyses and experimental results have demonstrated that, compared with the IKE scheme, our proposed scheme (EAKE) has significantly improved the efficiency by dramatically reducing time consumption and computation load with the same level of security.

In future new strategies can be investigated to improve the efficiency of symmetric key encryption towards more efficient security aware scheduling.

## REFERENCES

[1] R.Buyya, C.S yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud Computing and Emerging It Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems 25(2009) pp. 599-616.

[2] L. Wang, G.V. Laszewski, A.J Younge, X. He, M. Kunze, J. Tao, C. Fu , Cloud Computing: A Perspective Study, New Generation Computing 28 (2010) pp.137-146.

[3] M. Armbrust, A. Fox, R. Griffith, A.D Joseph, R.H Katz, A. Konwinski, G. Lee, D.A Patterson, Rabkin, I. Stoica, M. Zaharia, Above the Cloud: A Berkeley View of Cloud Computing, Technical report no.UCB/EECS-2009-28, University of California at Berkeley.

[4] L. Wang, M. Kunze, J. Tao, G.V. Laszewski, Towards Building a Cloud for Scientific Applications, Advances in Engineering software 42 (2011) pp. 714-722.

[5] C.Rong , Son T. Nguyen, Martin Gilje Jaatun, Beyond lightning: A Survey on Security Challenges in Cloud Computing. Computers & Electrical Engineering (2012).
[6] L.B.A Rabai, M. Jouini , AB. Aissa, A. Milli, A Cyber Security Model in Cloud Computing Environments, Journal of king saud university Computer and Information Sciences (2012).
[7] Eucalyptus Open Source Cloud Platform. Available : http:// ope.eucalyptus.com/
[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. soman, L. youseff, D. Zagorodnov, The Eucalyptus Open-Source Cloud Computing System, in: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGR ID'09,2009,pp.124-131.
[9] S. Subashini, V. Kavitha, A Survey on Security Issues in Service Delivery Models of Cloud Computing, Journal of Network and Computer Applications 34 (2010) pp. 1-11.
[10] W. Diffie, M. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory 22(1976) pp. 644-654.
[11] C. Kanfman, P. Hoffman, Y. Nir, P.Eronen, Internet Key Exchange Protocol Version 2(IKEv2), in: The Internet Engineering Task Force Request for Comments, IETF RFC,vol.5996,September 2010.
[12] M. Bellare, A. Boldyreva, J. Staddon, Randomness Reuse in Multi-Receipient Encryption Schemes, in: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography, PKC'03,Miami, USA,2003.
[13] R. Kusters, M. Tuengerthal, Computational Soundness for Key Exchange Protocols with Symmetric Encryption, in: Proceedings of the 16th ACM conference on Computer Communications Security, CS'09. Chicago, USA, 2009, pp.91-100.
[14] S.N. Srirama, P. Jakovits, E. Vainikko, Adapting Scientific Computing Problems to Clouds using Mapreduce, Future Genera tion Computer Systems 28 (2012) pp.184-192.
[15] KVM Hypervisor. Available: www. linux-kvm.org/
[16] Hadoop MapReduce. Available: http:// hadoop.apache.org/
[17] Openstack Open Source Cloud software. Available: http://openstack.org/
[18] Crypto++ Bench Marks. Available:http://www. cryptocom/benchmarks.html