Secured and Efficient Authentication Scheme for Mobile Cloud

W Thamba Meshach
Assistant Professor
Department of Computer Science and Engineering
Prathyusha Institute of Technology and Management, Thiruvallur, Tamilnadu, India

K.S. Suresh Babu
Assistant Professor
Department of Computer Science and Engineering
Pillai Institute of Information Technology, Engineering
Media studies and research, Mumbai, Maharashtra, India

Abstract - Modern Mobile Architectures are architectures that support mobile device applications connected to decentralized infrastructure. Modern Mobile Architectures combine mobile technology and cloud computing. The biggest challenge in both cloud and mobile technologies right now is security; combine the two technologies and we have a security wormhole. In this paper, we presented the current state of mobile-cloud authentication technology and our proposed system Mobile Cloud Key Exchange (MCKE), an authenticated key exchange scheme that aims at efficient authentication. This scheme is designed based on randomness-reuse strategy and Internet Key Exchange (IKE) scheme. Theoretical analysis and simulation results are compared with the IKE scheme; the MCKE scheme can significantly improve the efficiency by dramatically reducing time consumption and computation load without sacrificing the level of security.

Keywords - cloud computing, Mobile cloud / computing, authenticated key exchange, security

I. INTRODUCTION

A. Background

Smart phones and other mobile devices are heavily used in today's world and still get even more important since the usage of mobile internet. The growth of the number of applications available for those devices in the last years has shown that there is a high demand for mobile applications [*]. However one common problem that all those devices share, still needs to be addressed: the limited capabilities of the devices regarding available resources like processor power, available memory and especially energy consumption.

A technology recently emerged in the IT industry offers an opportunity to solve those problems: Cloud computing (CC) gives its users the possibility to host and deliver services over the internet by dynamically providing computing resources [13].Cloud computing eliminates the requirement for users to plan ahead for acquiring different resources, such as storage and computing power, and therefore, is attractive to business owners. Moreover, enterprises can provide resources depending on service demand. Whereas, Cloud computing is emerged as the modern technology which developed in last few years, and considered as the next big thing, in the years to come. Since it is new, so it require new security issues and face new challenges as well [1]. In last few years it is grown up from just being a concept to a major part of IT industry. Cloud computing widely accepted as the adoption of virtualization, SOA and utility computing, it generally works on three type of architecture and these are: SAAS, PAAS, and IAAS. There are different issue and challenges with each cloud computing technology. The various security concerns and upcoming challenges are addressed in [2], [4], and also reviewed in terms of standards such as PCI-DSS, ITIL, and ISO-27001/27002. Until now there is no such standard is available regarding service or operational functioning, and its security is a major concern.

There are also the architectural security issues which are changing according to various architectural designs functioning over cloud computing [3]. Various surveys are in the market depicting the current scenarios such as the leading US research firm Gartner released a report "Assessing the security risk of cloud computing" in June 2008, this report raises the concern about risk in data storage, data recovery, data privacy, and data integrity [5]. Cloud computing providing services in layered medium, so there must be some SLA (Service Level Agreement)

or service management, must be applied over the layers, which eventually increase the confidence of the user. Data security over the cloud also a major concern and various methodologies are proposed [6], also privacy preserving auditing for the data storage security in cloud computing [7], raising the concern over the privacy related issues in data storage [8], such that no critical information can be intercepted as recently a case happened with Wikileaks, over the security of the data. Apart from data security, security management, and security risk management frameworks are also proposed [9], [10] addressing risk associated with cloud computing, and activities are planned in such a way ensuring that information is available and protected by applying Deming model or PDCA to curb the risk related to cloud computing security. Cloud computing works in layers as applying policies on these layers provide better security approach to manage the security concerns [11]. Cloud computing has given a new horizon to the data hosting and deploying services. The most important thing of cloud computing is that it enables customers a new way to increase capacity and add capability to their machines on the go. In last to utilize the cloud computing in Mobile and to take full advantage of it, users required optimized security solutions and assurance regarding the security issues and risk involved in information flow.

B. Mobile Computing

Here are lists of some of the things we have had to address in mobile computing:

1. Passwords are great, but they are hard to enter on a small form factor (i.e., mobile) device. Also, harder passwords are more secure, but they are also very difficult to input, and more likely to be "written down" somewhere.
2. Signature tokens are a great way to protect communication from a secure computing device to the cloud, but smart mobile devices are *always* in an uncontrolled environment. Because of this, token leakage is possible.
3. Any increase in security usually leads to a corresponding decrease in usability; since the smart mobile platforms lack the computational capabilities of a traditional laptop, our options are further limited.
4. I can't presently firewall my iPhone, monitor its network activity in-situ, or run intrusion detection on the device without a significant degradation in overall performance.
5. Unlike my credit card, if my data-plan is exceeded due to a compromise, or my cloud service is compromised due to a leaked signature key, there is no comfort in having a maximum liability amount. Both assume you will secure the asset and the responsibility lies solely and wholly with the customer.

Is cloud computing in current scenario is providing confidentiality, integrity and being regulated by compliance like Data Protection Act. Through cloud computing the resource are centralized, so the exposure factor proportionally increase which results in risk. So it is necessary to put a countermeasure to mitigate the potential risk. According to the survey, some company says that due to cloud computing it become easier for bad guys to focus their effort and breach hundreds of thousands of record.[1] There is no security rating system in place for cloud computing, so business users can't rely on third party security mechanism. Risk factor with cloud computing are high because level of security provided by cloud provider

C. Cloud Computing

Further, when combined with what we had found in the cloud computing space:

1. Signature tokens must be protected, they are the main access point into most cloud environments, and used to enforce nonrepudiation at the cloud service provider.
2. Monitoring activity on a firewalled server inside a corporate server room is much easier than monitoring the same behaviour on someone's laptop in a coffee shop, or somebody's BlackBerry anywhere else.
3. Accessing protected assets from other cloud services requires the security protocol to be placed and managed outside of the firewall.

II. Proposed Algorithm

2.1 Present State Authentication Technologies Used in Mobile Cloud

2.1.1 Basic Authorization

Basic Authorization has been around for thousands of years - think "open sesame." It employs the authentication concept of a shared secret. If there is something only I and my communicating endpoint know, and we verify the knowledge of that secret, we have a substantial basis for authentication. There are a number of human factor issues with Basic Auth. The biggest comes from the fact that a password that is sufficiently complex to be secure is usually

hard to remember. There is a reason for that. Consider that the mechanisms we use for associativity are easily interrogated and analysed by computers trying to compromise an account. Combine this with the fact that sufficiently hard passwords are difficult to enter on a small form factor device and we have an authentication challenge for mobile devices using basic auth.

2.1.2 Signature Tokens

Signature tokens allow me to simultaneously authenticate and verify the integrity of the message. This combination is key to the nonrepudiation aspect of securing cloud and "X"aaS services where X is (P)latform, (S)oftware, (H)adoop, etc. In this signature token-based auth/auth model the user is responsible for protecting the secrecy of the signature token. Encrypting the tokens, utilizing them for only the duration of your cloud interaction, then completely destroying them by overwriting their memory location.

2.1.3 Open ID

Open ID has some attractive features, namely it and OAUTH both allow an application to authenticate once and authorize across cloud servers or other web-enabled services. This is a nice feature for mobile-cloud computing, and its biggest challenge is its token dependence. Its advantage over just a plain signature token is it ostensibly has a shorter lifetime of validity. Having an authorization or signature token is a legacy artifact of not having any objective information about a user that would identify that user uniquely. My laptop contains very little contextual information to form a context-aware authentication mechanism against, though there are technologies to enable this. A signature token in either system is a state management mechanism that lets the application know that this access has been previously authenticated. The storage of the authorization token is as essential in the open ID model as it is in the storage of the signature token in Signature Token auth/auth environments.

2.1.4 Open Auth

Open Auth overcomes the implicit service authentication issue by forcing mobile devices to support a multiphase authentication model where at one point the user may have to log in to an OAUTH providers service using a web interface. The provider will then make a call back to your application, at which point you are given an access token.

A typical OAUTH flow is this:

1. Apply to the provider for a consumer token.

2. Install the consumer token in the application(this has the same challenges presented by Signature Tokens and Open ID Access Tokens)

3. Using the consumer token, call to the provider to get a request token.

4. Once you have a request token, take the user to the page required by the provider for authentication.

5. The user is presented with a login page, the provider will perform a URL-based application callback after the user logs in with their username and password.

6. At this point you make another call to collect the user's Access Token.

7. The access token is used to authorize cloud service requests.

OAUTH's biggest weakness is similar to signature tokens and opens ID. It is the secure storage of the consumer token and the access token, but it also has the basic auth challenge of entering passwords on a small form factor device. Add to that the fact that it's hard to provide a URL-based call back solution in a mobile application, and this great technology shows its weaknesses in the modern mobile architecture space.

Present solutions that address the mobile computing shortcoming require an intermediary server or a notification process that interrupts the flow of user experience. You can see where while very highly adopted, OAUTH may not be an ideal solution for mobile cloud security.

A.System setup:

The system chooses a large prime integer to form a Diffie-Hellman group, and generator # of group $% &, i.e., # is a primitive root modulo. Normally is a Sophie Germain prime where is also prime, so that the group $%& maximises its resilient against square root attack to discrete logarithm problem. A certificate authority (CA) as in PKI is still needed in our security framework so that communicating parties can identify each other through exchanging verifiable certificates and !" , as the certificates contain public keys which can be used to verify the session partners' signatures, thereby their identities. Certificates are relatively long-termed data which are issued

to all participants of communication before the commencing of communication, and CA won't 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 (e.g. once a day) than key exchanging (e.g. reexchanging key in every new session), they won't affect the efficiency of a key exchange scheme for scheduling in general. Therefore, we will ignore all communications involving CA in our scheme and won't be discussing further details on issuing and revoking certificates.

B. MCKE Initial exchange:

Initial exchange is used when a new task is to be executed, because that is when CLC need 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 broadcast the following message to the domain of server instances S which contain n instances S1,...S:

Round1, c-> S: HDRc, SAc1, gx,Nc

where HDR and SA for algorithm negotiation, gx for Diffie-Hellman key exchange, and for freshness verification. The initiator of a normal IKE scheme will generate n secret x1, x2, x3... xn, then compute and send out gx1 , gx2 ....gxn ,either through multicast or one by one, to establish separated security channels with each receiver. In our scheme, although we still establish one for each server instance where i=1,2,..n, we are using only one single secret value0' for CLC in all * messages in order to reduce cost. We will further analyse security and cost reduction for this variation in section 4 and 5, respectively.

The session keys are now shared between CLC and each server instance for the use of encryption of later communications. Although the Diffie- Hellman key exchange is completed, the MCKE 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 " which are the signatures for these * messages, using its secret key from the key pair issued by CA and broadcast the following message to S:

Round 2, *C* -> *S*

HDR:

HDRc, {IDc, SAc2, Certc, CerReqs1 Sig}

gxy1 for i =1,2....n

The server instances can then verify the identity of the initiator of this conversation by using its session key #)7" 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 3, *S* -> *C*: HDRst{IDc,SAc2,Certc,Sig}gxy

for i=1,2,.....,n

where, similar to round 2 but only signed separately, Sig_{st} is signatures by Si to messages:

Mst = prf(prf(Nc|| Nst||gxy)||gyi||gx||IDsi)

for i = 1, 2, ...n

Note that this round involves * messages as well. After the identities of both CLC and server instances are authenticated through round 3 and 4, CLC will send to S1, S2,Sn, the split task data which are encrypted with session keys gxy1,..gxyn using symmetric encryption such as AES. After task execution, returns to CLC the results which are encrypted using S1, S2, Sn 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 200MB/s on today's desktop PCs) itself.

C. Rekeying:

In a multi-step task data need to be transferred back and forth in a multi-step task. In this situation it is not necessary to re-authenticate because of the high data dependency 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 and. As rounds 3 and 4 only contains fast operations such as signature and verification over short messages as well as symmetric-key encryption/decryption and HMAC functions, the computational overhead of rekeying process is almost identical to

the initial exchange from an efficiency prospect of view. We'll further analyse this in the section of performance evaluation.

III. Experiment and Result

3.1 Simulation Environment

U-Cloud is a cloud computing system in University of Technology, Sydney (UTS) on which we built and conducted our simulation environment. The computing facilities of this system are located in several labs in the Faculty of Engineering and IT, UTS. On top of hardware and Linux OS, We installed Xen Hypervisor which virtualises the infrastructure and allows it to provide unified computing and storage resources. Upon virtualised data centres, Hadoop is installed to facilitate MapReduce computing paradigm and distributed data management.

Furthermore, we installed Eucalyptus open source cloud platform [2, 17] which is responsible of global management, resource scheduling, task distribution and interaction with users.

3.2 Evaluation Method

It is clear that the actual efficiency improvement brought by our scheme highly depends on the size of dataset and the number of server instances in total. Experiments were conducted under multiple scenarios which will be demonstrated in the next section.

We will compare our scheme to the trivial scheme where CLC exchanges keys with each server instance using IKE in a separated fashion. Asymmetric-key cryptosystems are much slower (approximately 1000 times slower) than symmetric-key cryptosystems, thus large datasets are never encrypted with asymmetric-key cryptosystems in practice. Since our experiments are conducted on datasets of at least gigabytes in size, we will not compare our scheme to an asymmetric-key encryption based scheme because the results will be too predictably obvious.

As the CLC performs key exchange operations with all server instances and encryptions/decryptions of the entire dataset, it is acting as the bottleneck of the performance of our scheme.By utilising square-and-multiply algorithm, the minimum complexity of a modular exponentiation operation. Therefore, as most operations in our scheme are of linear complexity (e.g. symmetric encryption, prf, etc), modular exponentiations are by far the most costly. According to the fast evolving capability of computation facilities, it is now unsafe to utilise in a typical application a Diffie-Hellman group with its size less than 1024 bits [13]. 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. Signing over a message could also be time-consuming if the message is long. In our scheme however, the message to be signed is of a fixed short length (typically 128 bits which is the output size of a HMAC function). In this regard, time consumption in signing messages is small compared to computations over those 1024-bit-at-least keying materials. Computations on server instances in key exchange processes can be completed almost instantly; Besides, data transfer takes almost no time as well because only kilobytes of data need to be transferred between CLC and server instances in order to complete key exchange. To this end, we will evaluate the performance of our scheme by evaluating the efficiency improvement on CLC because we can infer from the analysis above that time consumed elsewhere can be considered negligible. We will show that our scheme improved the efficiency of key exchange significantly by reduced the number of exponential operations on CLC during key material computation.

3.3 Simulation Result

Key exchange is to be utilised in combination with symmetric-key encryption for security. In traditional data-intensive scenarios its time consumption can be neglected compared to the heavy time consumption on encryption. For indicating the significance of research on efficient key exchange schemes in cloud computing, we first show that key exchange schemes are taking a large percentage of run time when running under distributed computing environments such as cloud computing.

A cloud computing infrastructure often employs thousands of server instances. We assume the dataset is split into 16MB blocks through MapReduce before being distributed and executed on server instances, which is a common case for a time-critical data-intensive scientific application. We assume the encryption algorithm

used for dataset encryption is AES/GCM with 64K tables. For the efficiency of encryption algorithm, we refer to the data from Crypto++ benchmarks, which indicate the speed of AES/GCM with 64K tables is 108MB/s.

It can be inferred that key exchange operations takes over 20% of time in security scheme, which means we will get significant overall performance improvement if a key exchange scheme with better efficiency is used.

For evaluating MCKE scheme, we implemented 2 key exchange schemes using Java: basic multi-user IKE and our scheme MCKE. We ran them under our cloud simulation environment, and tested total time consumption on CLC. For Diffie-Hellman key exchange we used 1024-bit MODP group [13] with a 1024-bit prime *p* and generator # 5. We used MD5 as pseudorandom function prf, and RSA algorithm for signature. The result is demonstrated in Figure 3.1.

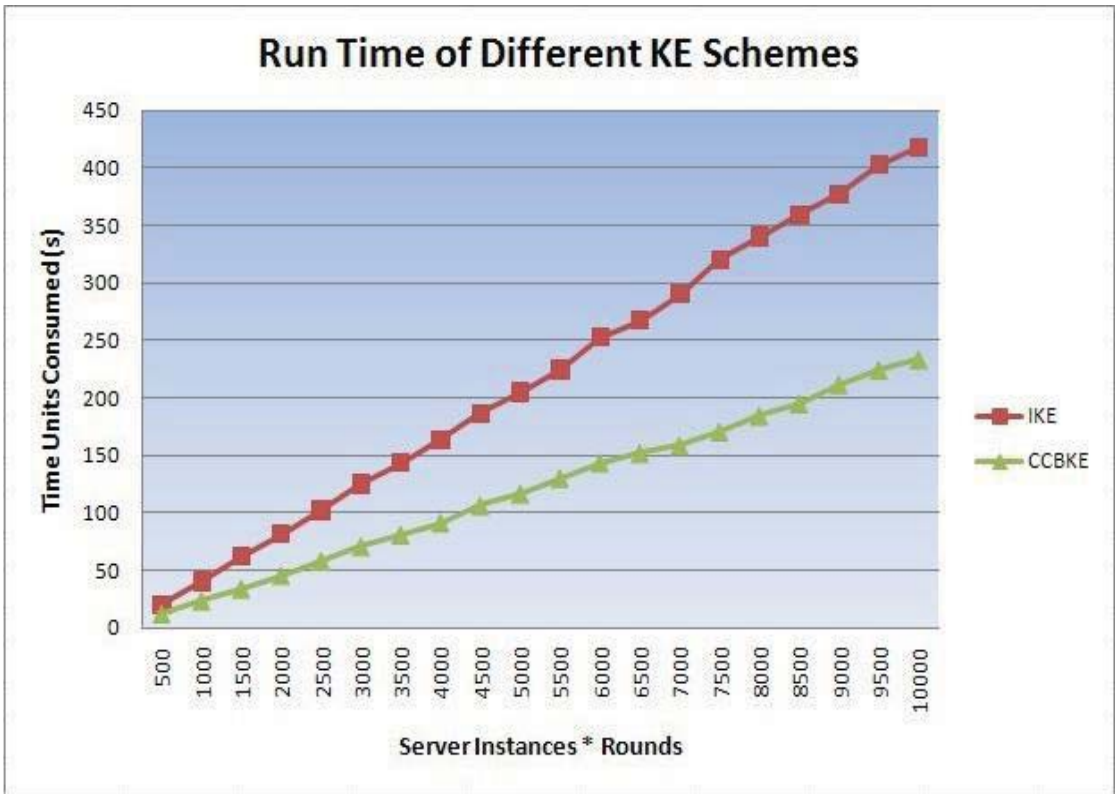

Figure3.1. Performance of Our SchemeCompared to IKE in Efficiency

We can see that our scheme consumes almost half of the run time on CLC compared to IKE. In time-critical multi-round applications where thousands of server instances are involved, a total of hours of time consumption can be reduced which means much faster results and a much lesser chance of missing a scientific discovery.

IV. CONCLUSION AND FUTURE WORK

In this paper the proposed authenticated key exchange scheme, namely Mobile Cloud Key Exchange (MCKE), which aimed at efficient security-aware scheduling of scientific applications? Our scheme has been designed based on the commonly-used Internet Key Exchange (IKE) scheme and randomness-reuse strategy. Both theoretical analyses and simulation results have demonstrated that, compared with the IKE scheme, our MCKE
scheme has significantly improved the efficiency by dramatically reducing time consumption and computation load with the same level of security.

While we have proposed an efficient key exchange scheme which can significantly reduce time consumption, efficiency of security-aware scheduling for scientific applications in the cloud can be further pushed forward by incorporating more cost-efficient encryption which, as the succeeding stage after Key Exchange, still takes a large partition of time consumption in cloud computing for scientific applications. Therefore, in the future, we will further investigate new strategies to improve the efficiency of symmetric-key encryption towards more efficient security-aware scheduling

REFERENCES

[1] G. S. E. Deelman, M. Livny, B. Berriman, J. Good, , in: , "The cost of doing science on the cloud: the montage example," in *ACM/IEEE Conference on Supercomputing (SC '08)*, Austin, Texas, 2008, pp. 1–12.

[2] Balachandran reddy etal Meiko Jensen etal On technical security issues in cloud computing, , 2009

[3] Cong Wang, Ensuring Data Storage security in cloud computing etal, 2010

[4] John harauz, Data security in the world of cloud computing, etal, 2010

[5] JASON CHRISTENSEN *Part 1: Mobile Cloud authentication and authorization,Cloud Computing Journal*

[6] Kemp, R., N. Palmer, T. Kielmann, and H.Bal, Cuckoo: a Computation Offloading Framework for Smartphones, in Proceedings of the Sixteenth annual conference of the Advanced School for Computing and Imaging 2010. 2010: Veldhoven, the Netherlands. p. 70-77.

[7] Mei, C., J. Shimek, C. Wang, A. Chandra, and J. Weissman, Dynamic Outsourcing Mobile Computation to the Cloud. 2011, Department of Computer Science and Engineering, University of Minnesota: Twin Cities.

[8] Mell, P. and T. Grance, NIST SP 800-145. The NIST Definition of Cloud Computing (Draft). Recommendations of the National Institute of Standards and Technology. 2011.

[9] K.-W. Park, S. S. Lim, and K. H. Park, "Computationally Efficient PKIBased Single Sign-On Protocol, PKASSO for Mobile Devices," *IEEE Transactions on Computers,* vol. 57, pp. 821 - 834, 2008.

[10] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific Cloud Computing: Early Definition and Experience," in *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, Dalian, China, 2008, pp. 825 - 830.

[11] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic, "TrustStore: Making Amazon S3 Trustworthy with Services Composition," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '08)*, Melbourne, Australia, 2010, pp. 600-605.

[12] D. Yuan, Y. Yang, X. Liu, and J. Chen, "On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems," *Journal of Parallel and Distributed Computing,* vol. 71, pp. 316-332, 2011.

[13] J. Zhao and D. Gu, "Provably secure authenticated key exchange protocol under the CDH assumption," *Journal of Systems and Software,* vol. 83, pp. 2297-2304, 2010.

[14] Zhang, Q., L. Cheng, and R. Boutaba, Cloud computing: state-of the-art and research challenges. Journal of Internet Services and Applications, 2012, p. 7-10

[15] http://www.cioedge.com/content/statecloud- computing-security