# CCBKE – Session Key Negotiation for Fast and Secure Scheduling of Scientific Applications in Cloud Computing

Chang Liu[1], Chi Yang[2], Xuyun Zhang[1] and Jinjun Chen[1]

[1]Faculty of Engineering and Information Technology
University of Technology, Sydney, Australia
Email: {changliu.it, xyzhanggz}@gmail.com, jinjun.chen@uts.edu.au

[2]School of Computer Science and Software Engineering
The University of Western Australia, Australia
Email: cyang@csse.uwa.edu.au

## Abstract

**Instead of purchasing and maintaining their own computing infrastructure, scientists can now run data-intensive scientific applications in a hybrid environment such as cloud computing by facilitating its vast storage and computation capabilities. During the scheduling of such scientific applications for execution, various computation data flows will happen between the controller and computing server instances. Amongst various quality-of-service (QoS) metrics, data security is always one of the greatest concerns to scientists because their data may be intercepted or stolen by malicious parties during those data flows, especially for less secure hybrid cloud systems. An existing typical method for addressing this issue is to apply Internet Key Exchange (IKE) scheme to generate and exchange session keys, and then to apply these keys for performing symmetric-key encryption which will encrypt those data flows. However, the IKE scheme suffers from low efficiency due to its low performance of asymmetric-key cryptological operations over a large amount of data and high-density operations which are exactly the characteristics of scientific applications. In this paper, we propose Cloud Computing Background Key Exchange (CCBKE), a novel authenticated key exchange scheme that aims at efficient security-aware scheduling of scientific applications. Our scheme is designed based on randomness-reuse strategy and Internet Key Exchange (IKE) scheme. Theoretical analyses and experimental results demonstrate that, compared with the IKE scheme, our CCBKE scheme can significantly improve the efficiency by dramatically reducing time consumption and computation load without sacrificing the level of security.**

***Keywords – hybrid environment; cloud computing; scheduling; efficiency; communication security; authenticated key exchange***

## 1. Introduction

Great changes are taking place in today’s IT and computing industry with the global popularity of hybrid computing environments with the concept of cloud computing[1]. 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 cloud for internal IT services. Cloud computing provides Infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS) and/or software-as-a-service (SaaS), so that users may have computation tasks accomplished on cloud in pay-as-you-go mode and won’t bother setting up and maintaining their own computing facilities [2], which is particularly a cost-saving solution in data- and computation-intensive applications such as applications in scientific research. Several cloud computing projects tailored for scientific users are already in existence, for example, projects Nimbus[3], Cumulus[4] and Aneka[5] have been proven to be quite resourceful. By utilising cloud computing service, the high cost in building and maintaining a supercomputing or grid computing environment for scientific applications is effectively reduced[6].

Security and privacy issues have been accompanying the popularisation and development of today's hybrid environments and the paradigm of cloud computing as one of the greatest concerns from people, ever since the start of their very presence [7-9]. Data in scientific applications particularly need to be carefully protected as they are invaluable intellectual properties which always directly relate to potential scientific discoveries and advanced technologies. The most security threats worried by the users are coming from inside the cloud, because that is where the users lose their direct control over the data. This is especially the case for a hybrid cloud system where part of its structure always lies in less-secure common or public domain. In regard to years of developments in cryptography, keeping data encrypted is the most common and safe choice to secure data in transmission, if encryption keys are managed properly.

Cloud IaaS services can be divided into cloud *storage* services and cloud *computing* services. For cloud *storage* services such as Amazon S3, users could just encrypt their data before putting them on cloud and decrypt them only when they finished using the service, as storage service providers won't need access to the contents of data [10]. Hence, there's no need in exchanging and managing the encryption keys over the risky communication environment in this case, as data are safe when transferring inside the cloud if users keep their encryption keys to themselves. Plus, given the high efficiency of symmetric cryptology algorithms such as stream cipher or even block cipher, encrypting/decrypting the data won't be a very heavy burden, even for entities with only lightweight computation capabilities. However, in cloud *computing* environments such as Amazon EC2, the cloud service providers need to apply computational operations on submitted user data and return the results. In scientific applications, these operations can be very complex. Therefore, the 'pre-encryption' strategy for cloud storage service is almost not applicable at all, as the resulting data will be no longer decryptable with the original key unless a fully homomorphic encryption scheme is used. Although Gentry[11] proposed a first-in-history fully homomorphic encryption algorithm in 2009 through the use of ideal lattices, there's still a long way to go until we have on earth an implemented fully homomorphic encryption scheme with reasonable efficiency. Current asymmetric-key cryptology allows multiple parties to communicate securely without exchanging keys. However, asymmetric-key encryption algorithms (such as RSA, ElGamal, Rabin, ECC, etc.) contain a number of exponential operations over a large finite field. Therefore, they work very slow with large datasets, which is why current asymmetric-key cryptology based infrastructure such as the Public-Key Infrastructure PKI [12, 13] wasn't incorporated into cloud implementations even if they are now very mature through years of development. To this end, exchanging a session key using asymmetric-key system and utilise it to encrypt data with symmetric-key encryption (we will term it a 'hybrid encryption' in the rest of this paper) is still the most efficient cryptological solution in protecting data confidentiality and integrity, especially in data-intensive scientific applications under the context of cloud computing.

For using hybrid encryption, Internet Key Exchange scheme Version 2 (IKEv2) [14] is a common choice of authenticated key exchange schemes to be used along symmetric-key encryption algorithms. IKE is a widely-adopted key exchange scheme which is being used along with IPSec as the default network-level data protection standard in TCP/IP Suite. Although symmetric-key algorithms (both stream ciphers and block ciphers such as Salsa20, AES and AES's predecessors DES/3DES) encrypt/decrypt very fast, the before-hand user authentication and key exchange steps in IKE are still tedious and time-consuming, especially we have at least thousands of independent service instances in the cloud running concurrently, where thousands of key-exchange operations are needed to be performed in every single round of communication.

In this paper we present Cloud Computing Background Key Exchange (CCBKE) scheme for security-aware scheduling in the background of cloud computing service providers. Our scheme is developed based on the widely adopted Internet Key Exchange (IKE) scheme [14]. By incorporating randomness-reuse strategy, our scheme reduces the number of modular exponential operations on cloud controller side, thereby reduces the computational load and execution time significantly compared to original IKE, which is the main research contribution of this paper.

The rest of this paper is organised as follows. Section 2 discusses related work. Section 3 provides a motivating example in scientific applications as well as detailed analysis to our research problem. Section 4 describes our CCBKE key exchange scheme in detail based on a series of notations demonstrated at the beginning of this section, as well as providing security analysis. Section 5 evaluates the performance and efficiency of our scheme through experimental results. Section 6 concludes our work and points out future work.

## 2. Related Work

Cloud computing, the most popular among recently emerging hybrid environment, has a significant cost benefit compared to traditional distributed systems such as clusters and grids [2]. Scientific applications can utilise this advantage by migrating to the cloud [6], which is recently attracting a rapidly growing amount of research interest from research scholars. Recent cloud computing projects for scientific applications such as Nimbus[3], Cumulus[4] and Aneka[5] as well as some very recent research work [15, 16] are all aiming at the transformation from a traditional cluster or data centre to a cloud architecture. Since the advent of cloud computing, a number of scheduling algorithms have been developed for the picture of a cost-effective cloud computing environment. Several most recent examples are the work of Garg et.al [17] and Yuan et.al [18]; the former work assessed time and cost in cloud QoS, while the latter work investigated the trade-off between data storage, computation and economical cost in the cloud. However, none of them have yet taken into account the cost of security enhancement. Data flows are unprotected in their models which totally neglected the importance of data security. Some outstanding security issues in cloud computing were surveyed in [7-9]. Recently, much research work has also been done to address cloud security/privacy issues. Most of the approaches amongst them, however, were focusing on cloud storage service. Yao et.al [10] proposed a scheme to ensure cloud storage security by separating the encryption keys from the stored data which were encrypted by the keys. In [19], a privacy-preserving cloud data querying scheme is proposed from generally a data prospect of view, which aims to protect privacy-sensitive outsourced data. In [20] Wang et.al proposed a scheme to protect data integrity based on bilinear-pairing-based public-key cryptology, which allows a third-party authority to check the outsourced data on cloud service users' behalf. However, to our best knowledge, none has analysed data encryption on the backstage of cloud computing from an efficiency point of view. Our work starts to try and bridge the gap.

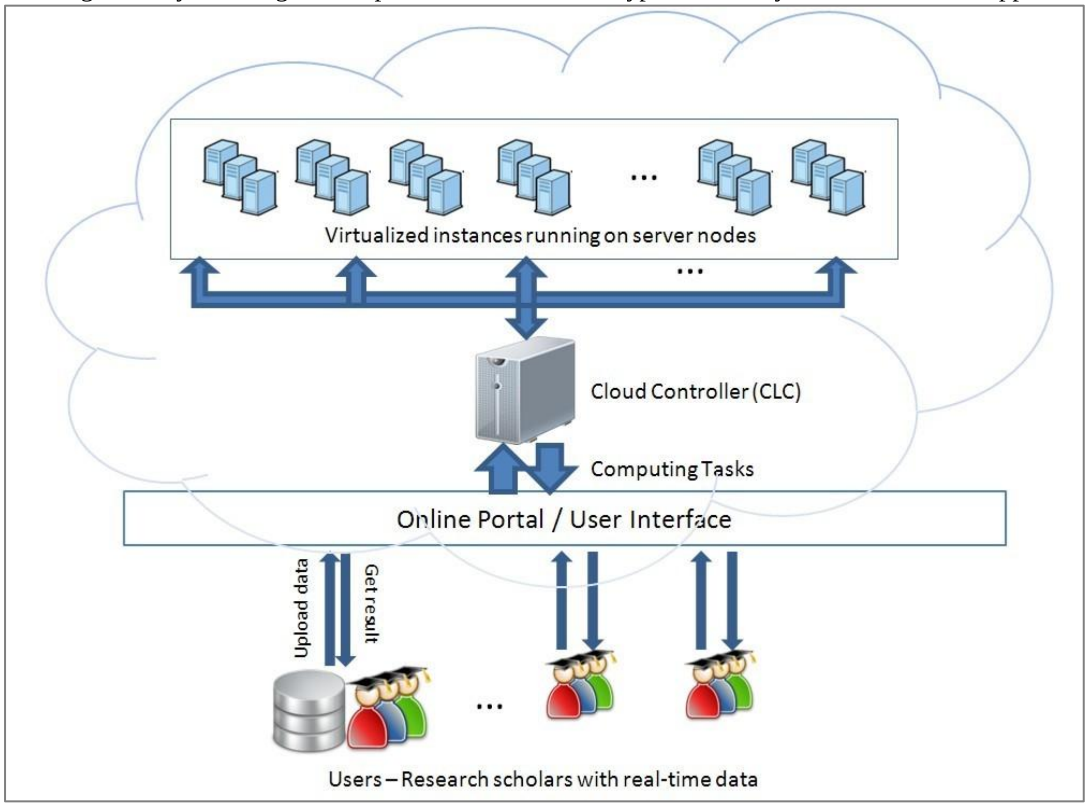
Encryption-based data security protection approaches for cloud *storage* services have been proposed, such as [10, 21]. However, in a cloud *computing* environment, original input data need to be processed on cloud side. Some encryption schemes allow processing over encrypted data, and it can still be decrypted using the same key thereafter. However, in these schemes only limited operations are allowed. For example the approach in [22] only allows *k*-NN (*k* nearest neighbours) algorithm to be applied over encrypted data. Fully homomorphic encryption [11] allows all operations on an encrypted dataset; However, no homomorphic encryption scheme with reasonable complexity has yet been published. Therefore, the efficiency of key exchange still remains as a major obstacle to the efficiency overall.

Key exchange over distrustful communication environment has been a extensively researched problem in public-key cryptography since Diffie and Hellman proposed the very first key exchange scheme [23] in 1976. Our problem here, essentially, is looking for an efficient key exchange scheme for exchanging different keys over participating parties. Although the topic of extending key exchange schemes in multi-party scenarios has been studied a lot in the past, to our best knowledge, this problem has not been well-addressed. This is probably because single key exchange process takes almost no time on modern hardware facility; therefore there was no requirement in doing research on this problem in the past. However, with thousands of independent instances executing different tasks in the cloud, now it is essential to develop an optimised key exchange scheme under this scenario if we are to use hybrid encryption to enhance cloud data security. Some existing key exchange schemes were trying to optimise the multi-party-same-key scenario with users join and leave dynamically, such as [24-27]. In [28, 29], extended security standards are formalised and researched for key exchange schemes for the basic 2-party scenario. Some of the most recent research on authenticated key exchange schemes are focusing on password-based key exchange[30, 31], which allows two parties to share a session key through exchanging a low-entropy password. These approaches are advantageous when humans are involved because a low-entropy password can be easily remembered. Unfortunately, no human action is required in our research problem. Kurosawa [32] and then Bellare et.al [33] studied the problem of asymmetric-key encryption in multi-user-different-data scenario with randomness reuse strategy. Although their schemes cannot be directly applied into cloud computing environments due to the low efficiency of asymmetric-key encryption/decryption over large datasets, this idea is a direct inspiration of our work. IKE is a widely-adopted authenticated key exchange scheme which is being used along with IPSec as the default network-level data protection standard in TCP/IP Suite. A latest updated description of IKE can be found in [14]. It has been known for its high security level [34], but lack efficiency in distributed environments, especially in cloud. This is the major motivation of this research.

## 3. Motivating Example and Problem Analysis

### 3.1 Motivating Example and Research Problem

In a typical cloud computing infrastructure, a central server is employed for not only receiving and processing user requests in the front, but also responsible for scheduling and splitting tasks through MapReduce in the back. This server is named cloud controller (CLC) in Eucalyptus system [35], we will use this denotation in this paper. Server instances running on clusters of servers are responsible of 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. A structure of additional hierarchical levels is often employed between CLC and end server instances as well, for more effective data management. However, we will ignore these levels in this paper since they just do task scheduling and "divide and deliver" on data and do not involve additional key management or encryption/decryption operations. Through minor variations, our scheme can easily be applied on a multi-layered cloud management system. Figure 1 depicts the structure of a typical cloud system for scientific applications.



...
Virtualized instances running on server nodes
...
Cloud Controller (CLC)
Computing Tasks
Online Portal / User Interface
Upload data
Get result
...
Users – Research scholars with real-time data

Figure 1. Basic Structure of A Typical Cloud Computing System for Scientific Applications

Scientific research applications are often data and computation intensive, quite a few among which are time-critical. Take astrophysics research for example, Australian astrophysics researchers operate a gigantic 64-metre Parkes telescope [36] which generates a large amount of data through constant observation. In pulsar searching workflow [18] the raw observation data are generated at a 1GB-per-second ratio. The beam file is generated for later pulsar searching through a local real-time compression operation on the raw data file. About 1 to 20GB beam file is generated from 4 to 60 minutes' observation, which is still a large dataset generated in a short time to be executed. We can infer from this example that due to its data-intensive nature, 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 properties which may directly lead to scientific discovery, hence their values are inestimable. Therefore, data security, specifically confidentiality and integrity, are of extremely 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. Another example in astrophysics is gravitational wave detection [37] which is especially time-critical. Due to its nature of real-time and streaming, delay in returning of a result of a task may lead to missing of an incoming gravitational wave. This will lead to loss of scientific discovery as the next wave would come after years. Moreover, thousands of hours of data-intensive computation was in vain, which is a terrible economical waste. Hence, efficiency is also important in cloud scheduling for scientific applications. These two challenges constitute the main research problem our work trying to tackle.

We will investigate the security challenge from communication security aspect of view. As cloud is a more open and distributed environment, it is essential to encrypt all data in transfer as well as authenticate each party's identity to ensure their confidentiality and integrity. As was discussed in Section 1, asymmetric-key encryptions are not suitable for our scenario because of their low efficiency. To utilise a symmetric encryption scheme, CLC must at first exchange a session key with each server instance through an authenticated key exchange scheme before distributing the divided data and tasks to server instances. Although there are multiple instances running on the same physical machine, it is better that each instance encrypts its communication with CLC with a distinct session key for risk management purpose. For example, if there's 100 instances each executing 10MB data on one machine and we use different session keys to encrypt this total 1GB data, then a malicious adversary will have access to only 10MB data other than that entire 1GB dataset if she/he gets to know one of the session keys through means such as side-channel attacks. Furthermore, if one of the server instances got hijacked, this adversary will reveal all the following data sent to the same physical machine until she/he get caught, as they are all encrypted with a same session key. The consequence may be hundreds of times more data leakage, depends on how many server instances are running on a single physical machine.

In a cloud computing systems there are always numerous server instances running inside. Small-to-medium-sized cloud systems such as an enterprise cloud or community cloud can have hundreds to thousands of server instances, and this number is even larger in large-scale cloud systems such as public cloud and commercial cloud. As a distinct session key is needed to be exchanged between CLC and each server instance, conducting key exchange between CLC and server instances becomes a tedious and time-consuming task in data-intensive applications such as scientific applications.

Based on the analysis above in this section, we can now see the research problem here - an efficient authenticated key exchange scheme is needed to ensure data security in cloud scheduling, especially for scientific applications. IKE[14] is one of the most efficient, secure, standardised and widely applied authenticated key exchange scheme for communications over an unsecure communication channel. However, when applied in our cloud computing scenario, the efficiency of IKE is very low. Unfortunately, existing research on authenticated key exchange schemes tend not to seek improvements on multiple-key-exchange schemes from an efficiency point of view. Previous research was mainly focusing on either strengthening security in two-party scenario [14, 28, 29], or multi-user-same-key scenarios [24-27]. Therefore, an authenticated key exchange (AKE) scheme for efficient cloud scheduling is on demand, especially for data- and operation-intensive scientific applications.

## 3.2 Problem Analysis

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 of $x^y \bmod p$ 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 of linear complexity (e.g. symmetric encryptions, pseudorandom functions, etc). 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 [14]. 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 identities authentication. Signing over a message could also be time-consuming if the message is long. In key exchange schemes however, messages to be signed are usually of a short fixed 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 those key exchange related computations over 1024-bit 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.

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

## 4. Randomness-reuse Based Authenticated Key Exchange – CCBKE

Similar to IKE, our CCBKE scheme consists of 3 independent components: system setup, initial exchange and rekeying. In this section we will give notations to some symbols for formalised description, demonstrate all these components in detail with those symbols, and provide a detailed security analysis for CCBKE.

## 4.1 Notations.

Before start demonstrating how the CCBKE scheme works, we first define some notations to be used for a formalised description.

| | |
|---|---|
| $S$: | Server instances domain |
| $S_i$: | The $i$th server instance in $S$ |
| $C$: | Cloud controller (CLC) |
| $HDR_i$ : | Header contains security parameter indexes |
| $CertReq_i$: | Certificate request, requesting for $i$'s certificate |
| $Cert_i$: | Certificate |
| $N_i$: | $i$'s one-time nonce |
| $SA$: | Security associations, used in negotiating cryptographic algorithms |
| $ID_i$: | Identity information |
| Sig$_i$: | $i$'s signature, can be verified using pre-defined algorithm and public key within $Cert_i$ |
| prf(): | Pseudorandom function |
| $\{M\}_k$: | Encrypt message $M$ with session key $k$ |

## 4.2 CCBKE System setup:

The system choose a large prime integer $p$ to form a Diffie-Hellman group, and a generator $g$ of group $\mathbb{Z}_p^*$, i.e., $g$ is a primitive root modulo $p$. Normally $p$ is a Sophie Germain prime where $(p-1)/2$ is also prime, so that the group $Z_p^*$ 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 $Cert_c$ and $Cert_{s_i}$ , 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. re-exchanging 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.

## 4.3 CCBKE 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 $g^x$ in $\mathbb{Z}_p^*$ , and broadcasts the following message to the domain of server instances S which contains $n$ instances $S_1, \dots, S_n$:

Round 1, $C$ -> $S$: $HDR_c$ , $SA_{c_1}$ , $g^x$, $N_c$

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 $x_1, x_2, \dots, x_n$ , then compute and send out $g^{x_1}, g^{x_2}, \dots, g^{x_n}$, either through multicast or one by one, to establish separated security channels with each receiver. In our scheme, although we still establish one $SA$ for each server instance $S_i$ where $i = 1,2, \dots, n$ , we are using only one single secret value $x$ for CLC in all $n$ messages in order to reduce cost. We will further analyse security and cost reduction for this variation in section 4 and 5, respectively.

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

Round 2, $S$ -> $C$: $HDR_{S_i}$ , $SA_{s_{i_1}}$ , $g^{y_i}$ , $N_{S_i}$ , $CertReq_c$ ,  for $i = 1, \dots, n$

Note that round 2 involves $n$ different messages sent from $S_1, \dots, S_n$ separately. After exchanging the first two rounds of messages, the session keys $g^{xy_1}, \dots, g^{xy_n}$ are computed for all parties as follows:

$$C: g^{xy_1} = (g^{y_1})^x, \dots, g^{xy_n} = (g^{y_n})^x$$

$$S_1: g^{xy_1} = (g^x)^{y_1}$$

$$\dots$$

$$S_n: g^{xy_n} = (g^x)^{y_n}$$

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 CCBKE 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 $\mathrm{Sig}_{c_i}$ which are the signatures for these $n$ messages, using its secret key from the key pair issued by CA:

$$M_{c_i} = \mathrm{prf}\big(\mathrm{prf}(N_c||N_{s_i}||g^{xy_i})||g^x||g^{y_i}||SA_c||ID_c\big), \text{ for } i = 1, \dots, n$$

and broadcast the following message to S:

Round 3, $C$ -> $S$:
$HDR_c$ , $\{ID_c, SA_{c_2}, Cert_c, CertReq_{s_1}, \mathrm{Sig}_c\}_{g^{xy_1}}||\{ID_c, SA_{c_2}, Cert_c, CertReq_{s_2}, \mathrm{Sig}_c\}_{g^{xy_2}}$
$|| \dots ||\{ID_c, SA_{c_2}, Cert_c, CertReq_{s_n}, \mathrm{Sig}_c\}_{g^{xy_n}}$ , for $i = 1, \dots, n$

The server instances can then verify the identity of the initiator of this conversation by using its session key $g^{xy_i}$ 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, $S$ -> $C$: $HDR_{S_i}$ , $\left\{ID_c, SA_{s_{i_2}}, Cert_{s_i}, \mathrm{Sig}_{s_i}\right\}_{g^{xy_i}}$ , for $i = 1, \dots, n$

where, similar to round 3 but only signed separately, $\mathrm{Sig}_{s_i}$ is signatures by $S_i$ to messages:

$$M_{s_i} = \mathrm{prf}\big(\mathrm{prf}(N_c||N_{s_i}||g^{xy_i})||g^{y_i}||g^x||SA_{s_i}||ID_{s_i}\big), \text{ for } i = 1, \dots, n$$

Note that this round involves $n$ messages as well. After the identities of both CLC and server instances are authenticated through round 3 and 4, CLC will send to $S_1, \dots, S_n$ the split task data which are encrypted with session keys $g^{xy_1}, \dots, g^{xy_n}$ using symmetric encryption such as AES. After task execution, $S_1, \dots, S_n$ returns to CLC the results which are encrypted using $g^{xy_1}, \dots, g^{xy_n}$ 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.

## 4.4 CCBKE Rekeying:

Rekeying is often accomplished by running initial exchange all over again. However, in the following cases, alternative strategies need to be applied. We'll also analyse in this section the efficiency of these strategies.

### *4.4.1 Failure Recovery*

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

### *4.4.2 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 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. Following the analyses in Section 3, 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 on CLC is almost identical to the initial exchange from an efficiency point of view.

## 4.5 Security Analysis for CCBKE

We will analyse the security of our scheme in two ways. We will show that our scheme is safe against both outside and inside attackers while maintaining perfect forward secrecy.

***4.5.1 Security Proof***

As cryptanalysis on symmetric-key encryption algorithms is outside the scope of this paper, the following discussions are under 2 standard cryptographic assumptions as follows.

*Assumption 1*: Any participant in our scheme cannot retrieve any data that was encrypted by any symmetric-key algorithm, unless it has the session key which was used to encrypt the data in the first place. I.e., cryptanalysis is beyond this security discussion.

*Assumption 2 (CDH assumption):* Given a cyclic group $G$ of order $q$, a generator $g$ of $G$ and two random integers $a, b \in \{0, \ldots, (q-1)\}$, retrieving $g^{ab}$ in polynomial time using only $\{g, g^a, g^b\}$ is computationally impossible.

Similar to most security analyses to public-key communication protocols, we now define the capabilities of an outsider attacker and an inside attacker.

***Definition 1*** (*cloud outside attacker*): A malicious outside attacker $M_o$ is an adversary who is capable of monitor, intercept, and change all communication traffic in the whole cloud background structure, in order to gain access to protected data in transit. However, $M_o$ does not have access to any of the node machines, and its identity is not legit, i.e. $M_o$ cannot obtain a valid certificate to let itself be authenticate by CLC or any server instance.

***Definition 2*** (*cloud inside attacker*): A malicious inside attacker $M_i$ is an adversary who can be authenticated by CLC and act as a legitimate server instance of the communication. However, $M_i$ does not have access to any other legitimate participants', including server instances' and CLC's, private information.

We now prove that our scheme is secure against both these types of attackers.

***Theorem 1***: A *cloud outside attacker* $M_o$ cannot retrieve in polynomial time any exchanged session key $g^{xy_i}$ in CCBKE.

*Proof:* Following *Definition 1*, we know that an outside attacker $M_o$ can gain access to all public keying materials $g, p, g^x, g^{y_1}, \ldots, g^{y_n}$ by monitoring the entire network, but $M_o$ cannot get secret information such as $x, y_1, \ldots, y_n$. Considering the computational hardness of computational Diffie-Hellman (CDH) problem, we know that $M_o$ cannot compute any $g^{xy_i}$ where $i = 1, \ldots, n$, in polynomial time, with $g^x, g^{y_1}, \ldots, g^{y_n}$.

***Theorem 2***: Assume $k = g^{xy_m}$ is the session key negotiated between a *cloud inside attacker* $M_i$ and CLC. $M_i$ cannot retrieve in polynomial time any session key $g^{xy_i}$ other than $k$, unless a negligible probability.

*Proof:* According to *Definition 2*, $M_i$ has its own secret value $y_t, k$ in addition to information controlled by $M_o$. Since all secret values $y_1, \ldots, y_n$ are generated by individual server instances themselves instead of allocated, there's a probability that a same secret value is generated and used for key exchange by different server instances, which we call a 'collision'. For example, when a collision between $M_i$ and a legitimate server instance $S_l$ (with its secret value $y_l$) happens, we have $y_l = y_m$. Therefore $M_i$ can easily retrieve data sent by $S_l$ using its own key $g^{xy_m}$. Since $y_1, \ldots, y_n$ are randomly chosen over $\mathbb{Z}_p^*$, the probability $\epsilon$ for a collision happen will be:

$$\epsilon = \prod_{i=1}^{n-1}\left(1 - \frac{i}{p}\right) \approx \prod_{i=1}^{n-1} e^{-\frac{i}{p}} = e^{-\frac{n(n-1)}{2p}}$$

This is a similar situation to the famous birthday attack where $\epsilon$ depends on $n/\sqrt{p}$ as well. In our CCBKE scheme, $p$ is commonly 1024-bit and $n$ (number of server instances) is usually several thousand which can be considered negligible compared to $2^{512}$. To this end, $n/\sqrt{p}$ is very close to 0. Thus, the probability for a collision to happen is negligible, which means an inside attacker cannot retrieve any of others' session key except a negligible possibility. Combining this conclusion with *Theorem 1*, we have finished proving of *Theorem 2.*

Since all identity information are encrypted with the session keys in authentication rounds 3 and 4 and both outside attackers and inside attackers cannot retrieve others' session key, we have the following lemma.

***Lemma 1*** Any pretended participant will fail authentication in rounds 3 and 4.

Once the authentication of rounds 3 or 4 failed, the communication will then be terminated. A man-in-the-middle attack or any kinds of identity forgery attack to our scheme will hence not be successful.

***4.5.2 Perfect Forward Secrecy***

Same as in IKE, session keys used for encrypting communications are only used once until they are expired and destroyed. Thus, a previously used session key or secret keying material is worthless to a malicious opponent even a previously-used key or a secret keying material is somehow exposed. This is one of the major advantages of using a key exchange scheme in hybrid encryption, which is why we did not choose to simply encrypt the session key with an asymmetric-key encryption algorithm, even though it can be easily done through PKI considering the fact that we have adopted a CA in our CCBKE architecture.

## 5. Experiment and Evaluation

The CCBKE scheme proposed in this paper is generic. It can be deployed with varioius scientific application and on various cloud computing platforms to improve efficiency and reduce the cost in guaranteeing communication security.

According to the analyses in section 3.2, time consumption of operations outside CLC can be considered negligible. In CCBKE, Rounds 1 and 3 are carried out on CLC, which both contained $n$ modular exponential operations. Since the CLC used only one randomness $x$ instead of $\{x_1, \dots x_n\}$ to conduct Diffie-Hellman key exchange with $\{S_1, \dots S_n\}$, there are only 1 instead of $n$ modular exponential operations left. Hence, approximately half of the modular exponential operations are cut out. In this regard, intuitively speaking, the CCBKE scheme can have vast efficiency advantage over IKE [14]. We now further conduct experiments to quantitatively demonstrate that CCBKE, when applied in cloud, can indeed improve the efficiency of IKE significantly.

### 5.1 Experiment Environment

U-Cloud is a cloud computing system in University of Technology, Sydney (UTS) on which we built and conducted our experimental 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 [38] which virtualises the infrastructure and allows it to provide unified computing and storage resources. Upon virtualised data centres, Hadoop [39] is installed to facilitate MapReduce programming paradigm and distributed data management. Furthermore, we installed Eucalyptus open source cloud platform [35, 40] which is responsible of global management, resource scheduling, task distribution and interaction with users. The structure of U-Cloud system is depicted in Figure 2.

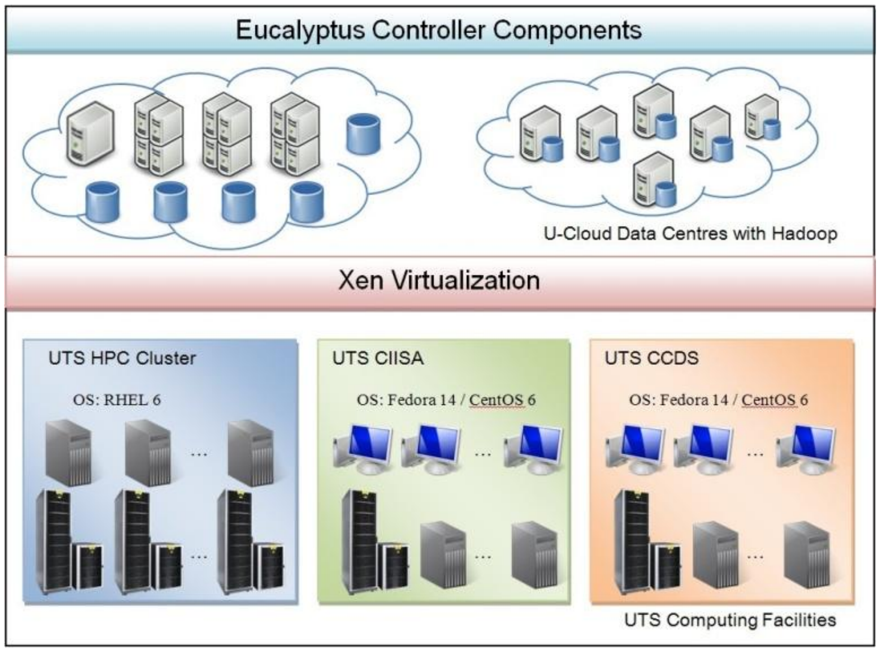


Figure 2. U-Cloud Structure

## 5. 2 Experiment Process

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 on multiple datasets with different number of server instances involved, which will be demonstrated in the next sub-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 the hybrid encryption scheme to an asymmetric-key encryption scheme because the results will be too predictably obvious.

For evaluating CCBKE scheme, we implemented 2 key exchange schemes using Java: basic multi-user IKE scheme and our CCBKE scheme. For parameters of Diffie-Hellman key exchange we used 1024-bit MODP group [14] with a 1024-bit prime $p$ and generator $g = 2$. We used MD5 as pseudorandom function prf, and RSA algorithm for signature. We ran the two key exchange schemes under our cloud environment with a few astrophysics datasets, and tested total time consumption on CLC.

According to the analysis in Section 3, we will evaluate the performance of our scheme by evaluating the efficiency improvement on CLC because we can infer from the analysis that time consumed elsewhere can be considered negligible. We will show in the next sub-section that our scheme improved the efficiency of key exchange significantly.

## 5.3 Experimental Results and Analysis

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, which indicates the significance of research on efficient authenticated key exchange schemes. When applying hybrid encryption to traditional data-intensive applications, key exchange schemes are always being utilised in combination with symmetric-key encryption to ensure data security. In these scenarios, time consumption of key exchange schemes can be neglected compared to the heavy time consumption on encryption. However, the situation is different in cloud computing, which we demonstrate the difference. A cloud computing infrastructure often employs thousands of server instances. For a time-critical data-intensive application such as scientific applications, the GB dataset is split into MB blocks and then distributed and executed on server instances through MapReduce. We use IKE time consumption data from our experiment to represent the efficiency of key exchange scheme. For symmetric-key encryption algorithms, we included two algorithms used for dataset encryption: the most widely-recognized block cipher, *AES*, in Galois Counter Mode (GCM) with 64K tables, and *Salsa20/12*, a stream cipher which is a more popular kind in encrypting large datasets because of its high efficiency. Both of the two algorithms were proven to be secure against various kinds of cryptanalysis. For the efficiency of encryption algorithms, we refer to the performance result from Crypto++ benchmarks [41] which indicates the speed of *AES/GCM* with 64K tables is 108MB/s, and the speed of *Salsa20/12* is 643MB/s for data encryption. Experiments are conducted on several datasets taken from astrophysics research; results are listed in Tables I and II:

TABLE I. ONE-ROUND RUNTIME COMPARASION ON TIME CONSUMPTION OF KEY EXCHANGE AND AES ENCRYPTION ON CLC

| **Dataset Size (GB)** | **2** | **8** | **12** | **15** | **32** |
|---|---|---|---|---|---|
| Server Instances Involved | 100 | 500 | 1000 | 1500 | 4000 |
| Data Block Size (MB) | 20 | 16 | 12 | 10 | 8 |
| AES/GCM Encryption Time (s) | 18.52 | 74.07 | 111.11 | 138.89 | 296.31 |
| IKE Key Exchange Time (s) | 4.04 | 20.48 | 41.77 | 61.92 | 163.10 |
| Key Exchange Take Percentage of (%) | **17.91** | **21.66** | **27.32** | **30.84** | **35.50** |

TABLE II. ONE-ROUND RUNTIME COMPARASION ON TIME CONSUMPTION OF KEY EXCHANGE AND SALSA ENCRYPTION ON CLC

| **Dataset Size (GB)** | **2** | **8** | **12** | **15** | **32** |
|---|---|---|---|---|---|
| Server Instances Involved | 100 | 500 | 1000 | 1500 | 4000 |
| Data Block Size (MB) | 20 | 16 | 12 | 10 | 8 |
| Salsa20/12 Encryption Time (s) | 3.11 | 12.44 | 18.66 | 23.33 | 49.77 |
| IKE Key Exchange Time (s) | 4.04 | 20.48 | 41.77 | 61.92 | 163.10 |
| Key Exchange Take Percentage of (%) | **56.50** | **62.21** | **69.12** | **72.63** | **76.62** |

which proved that in hybrid cloud computing environment, key exchange operations indeed take a large percent of time consumption in a hybrid security scheme[1]. This means the overall performance will be significantly improved if a key exchange scheme with better efficiency is used.

As was analysed 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. Compared by time consumed on CLC, the efficiency of the two key exchange schemes is demonstrated in Figure 3:

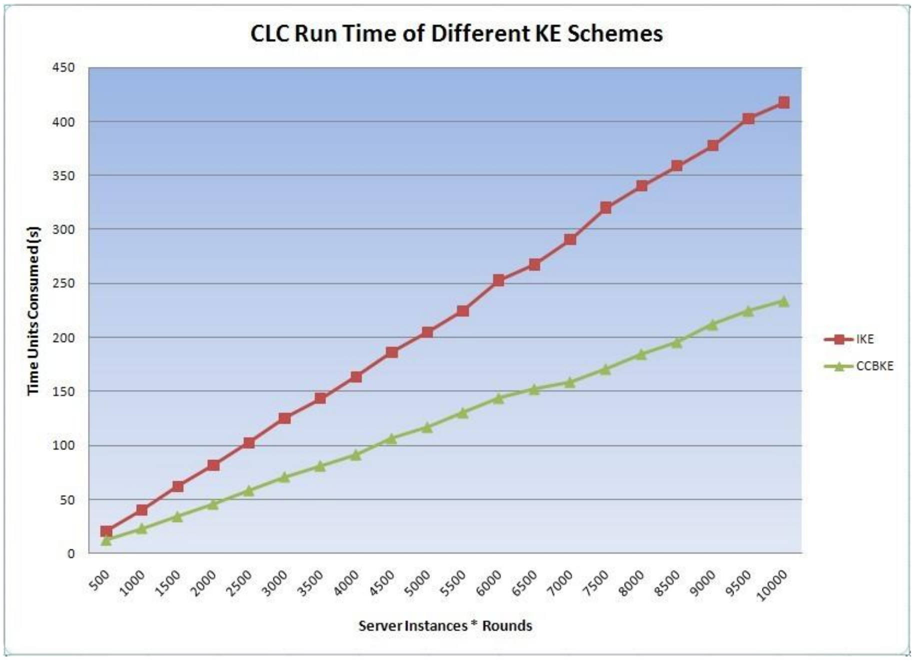


Figure 3. Performance of Our Scheme Compared in Efficiency to IKE

We can see that our scheme only consumes about half of the run time on CLC compared to IKE, which is a significant improvement. 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.

## 6. Conclusions and Future Work

In this paper we have proposed a novel authenticated key exchange scheme, namely Cloud Computing Background Key Exchange (CCBKE), which aimed at efficient security-aware scheduling of scientific applications in hybrid computing

[1] This percentage will be even higher in real-world scenario since KE efficiency depends heavily on scheduling algorithm and network status while encryption time stays relatively stable.

environments such as cloud computing. Our scheme has been designed based on the commonly-used Internet Key Exchange (IKE) scheme and randomness-reuse strategy. Both theoretical analyses and experimental results have demonstrated that, compared with the IKE scheme, our CCBKE 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 hybrid environments can be pushed further forward by incorporating more cost-efficient encryption which, as the succeeding stage after Key Exchange, still takes a large partition of time consumption, even 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.

## Acknowledgment

This research work is partly supported by Australian Research Council under Linkage Project LP0990393. We are also grateful for the useful discussions with researchers from Swinburne University of Technology on astrophysics research.

## 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) 599-616.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing, Technical Report No. Ucb/Eecs-2009-28, University of California at Berkeley. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf, accessed on 24 October, 2011.

[3] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, M. Tsugawa, Science Clouds: Early Experiences in Cloud Computing for Scientific Applications, in: First Workshop on Cloud Computing and its Applications (CCA '08), Chicago, USA, 2008, pp. 1 - 6.

[4] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, 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.

[5] C. Vecchiola, R. N. Calheiros, D. Karunamoorthy, R. Buyya, Deadline-Driven Provisioning of Resources for Scientific Applications in Hybrid Clouds with Aneka, Future Generation Computer Systems 28 (2012) 58-65.

[6] 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.

[7] S. Subashini, V. Kavitha, A Survey on Security Issues in Service Delivery Models of Cloud Computing, Journal of Network and Computer Applications 34 (2010) 1-11.

[8] T. Mather, S. Kumaraswamy, S. Latif, Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. O'Reilly Media, Sebastopol, 2009.

[9] D. Zissis, D. Lekkas, Addressing Cloud Computing Security Issues, Future Generation Computer Systems, in press, doi:10.1016/j.future.2010.12.006 (2011)

[10] J. Yao, S. Chen, S. Nepal, D. Levy, 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.

[11] C. Gentry, Fully Homomorphic Encryption Using Ideal Lattices, in: Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09), Bethesda, USA, 2009.

[12] A. Boldyreva, M. Fischlin, A. Palacio, B. Warinschi, A Closer Look at Pki: Security and Efficiency, in: Proceedings of the 10th international conference on Practice and theory in public-key cryptography (PKC '07), Beijing, P.R.China, 2007, pp. 458–475.

[13] K.-W. Park, S. S. Lim, K. H. Park, Computationally Efficient Pki-Based Single Sign-on Protocol, Pkasso for Mobile Devices, IEEE Transactions on Computers 57 (2008) 821 - 834.

[14] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen. Internet Key Exchange Protocol Version 2 (Ikev2). The Internet Engineering Task Force Request for Comments (IETF RFC) 5996, September 2010. Available: http://tools.ietf.org/html/rfc5996, accessed on 24 October, 2011.

[15] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, D. H. J. Epema, Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing, Parallel and Distributed Systems, IEEE Transactions on 22 (2011) 931-945.

[16] S. N. Srirama, P. Jakovits, E. Vainikko, Adapting Scientific Computing Problems to Clouds Using Mapreduce, Future Generation Computer Systems 28 (2012) 184-192.

[17] S. K. Garg, S. K. Gopalaiyengar, a. R. Buyya, Virtual Machine Provisioning Based on Analytical Performance and Qos in Cloud Computing Environments, in: Proceedings of the 40th International Conference on Parallel Processing (ICPP '11), Taipei, Taiwan, 2011.

[18] D. Yuan, Y. Yang, X. Liu, J. Chen, On-Demand Minimum Cost Benchmarking for Intermediate Dataset Storage in Scientific Cloud Workflow Systems, Journal of Parallel and Distributed Computing 71 (2011) 316-332.

[19] N. Cao, Z. Yang, C. Wang, K. Ren, W. Lou, Privacy-Preserving Query over Encrypted Graph-Structured Data in Cloud Computing, in: IEEE International Conference on Distributed Computing Systems (ICDCS '11), 2011, pp. 393 - 402.

[20] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing, IEEE Transactions on Parallel and Distributed Systems 22 (2011) 847 - 859.

[21] K. P. N. Puttaswamy, C. Kruegel, B. Y. Zhao, "Silverline: Toward Data Confidentiality in Storage-Intensive Cloud Applications," presented at the 2nd ACM Symposium on Cloud Computing (SOCC '11), Cascais, Portugal, 2011.

[22] W. K. Wong, D. W.-l. Cheung, B. Kao, N. Mamoulis, Secure Knn Computation on Encrypted Databases, in: Proceedings of the 35th SIGMOD international conference on Management of data (SIGMOD '09), Providence, USA, 2009, pp. 139-152.

[23] W. Diffie, M. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory 22 (1976) 644 - 654.

[24] E. Bresson, O. Chevassut, D. Pointcheval, Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions, in: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT '02 ), Amsterdam, Holland, 2002.

[25] Z. Zhou, D. Huang, An Optimal Key Distribution Scheme for Secure Multicast Group Communication, in: IEEE Conference on Computer Communications (INFOCOM '10), San Diego, USA, 2010.

[26] J. Katz, J. S. Shin, Modeling Insider Attacks on Group Key-Exchange Protocols, in: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS’05), Alexandria, USA, 2005, pp. 180-189.

[27] J. Katz, M. Yung, Scalable Protocols for Authenticated Group Key Exchange, Journal of Cryptology 20 (2007) 85 - 113.

[28] R. Küsters, M. Tuengerthal, Computational Soundness for Key Exchange Protocols with Symmetric Encryption, in: Proceedings of the 16th ACM conference on Computer and communications security (CCS '09), Chicago, USA, 2009, pp. 91-100.

[29] J. Zhao, D. Gu, Provably Secure Authenticated Key Exchange Protocol under the Cdh Assumption, Journal of Systems and Software 83 (2010) 2297-2304.

[30] A. Groce, J. Katz, A New Framework for Efficient Password-Based Authenticated Key Exchange, in: Proceedings of the 17th ACM conference on Computer and communications security (CCS '10 ), Chicago, USA, 2010, pp. 516-525.

[31] J. Katz, V. Vaikuntanathan, Round-Optimal Password-Based Authenticated Key Exchange, in: Proceedings of the 8th conference on Theory of cryptography (TCC'11), Providence, USA, 2011, pp. 293-310.

[32] K. Kurosawa, Multi-Recipient Public-Key Encryption with Shortened Ciphertext, in: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems: Public Key Cryptography (PKC '02), Paris, France, 2002, pp. 321–336.

[33] M. Bellare, A. Boldyreva, J. Staddon, Randomness Re-Use in Multi-Recipient Encryption Schemeas in: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC '03), Miami, USA, 2003.

[34] R. Canetti, H. Krawczyk, Security Analysis of Ike's Signature-Based Key-Exchange Protocol, in: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '02), Santa Barbara, USA, 2002, pp. 143-161.

[35] Eucalyptus Open Source Cloud Platform. Available: http://open.eucalyptus.com/, accessed on 24 October, 2011.

[36] Australia Telescope, Parkes Observatory. Available: http://www.parkes.atnf.csiro.au/, accessed on 24 October, 2011.

[37] D. Kawata, R. Cen, L. C. Ho, Gravitational Stability of Circumnuclear Disks in Elliptical Galaxies, The Astrophysical Journal 669(1) (2007) 232-240.

[38] Xen Hypervisor. Available: http://xen.org/, accessed on 24 October, 2011.

[39] Hadoop Mapreduce. Available: http://hadoop.apache.org, accessed on 24 October, 2011.

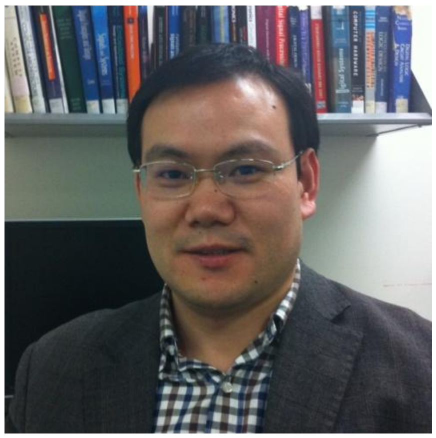
[40] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, The Eucalyptus Open-Source Cloud-Computing System, in: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09), 2009, pp. 124-131.

[41] Crypto++ Benchmarks. Available: http://www.cryptopp.com/benchmarks.html, accessed on 24 October, 2011.

**Chang Liu** was born in Jinan, China. He received his B.Eng. degree in computer science in 2005 and M.Sc. degree in information security in 2008, both from Shandong University, Jinan, China. He is currently a Ph.D. student in the Faculty of Engineering and Information Technologies at University of Technology Sydney, Australia. His research interests include cloud computing, scheduling and resource management, cryptography and data security.

**Chi Yang** received his B.S. in computer science from Shandong University, China, in 2004. He received his M.S. (by Research) in computer science from Swinburne University of Technology, Melbourne, Australia, in 2007. Currently, Chi Yang is a Ph.D student at the University of Western Australia, Perth, Australia. His major research interests include the XML data stream query processing and optimization, the scientific workflow, cloud computing, green computing and data processing techniques over wireless sensor networks.

**Xuyun Zhang** received the BS degree and ME degree in computer science from Nanjing University, China, in 2008 and 2011, respectively. He is currently working towards the PhD degree at the Faculty of Engineering & IT, University of Technology, Sydney. His research interests include cloud computing, service computing, QoS, privacy and security.



Dr **Jinjun Chen** is an Associate Professor from Faculty of Engineering and IT, University of Technology Sydney, Australia. He holds a PhD in Computer Science and Software Engineering (2007) from Swinburne, a master degree in engineering (1999) and a bachelor degree in applied mathematics (1996) from Xidian University, China. Dr Chen's research interests include cloud computing, social computing, green computing, service computing, e-science/e-research, workflow management. His research results have been published in more than 100 papers in high quality journals and at conferences, including the ACM Transactions on Software Engineering and Methodology (TOSEM), IEEE Transactions on Software Engineering (TSE), and the International Conference on Software Engineering (ICSE). He received Swinburne Vice-Chancellor's Research Award for early career researchers (2008), IEEE Computer Society Outstanding Leadership Award (2008-2009), IEEE Computer Society Service Award (2007), Swinburne Faculty of ICT Research Thesis Excellence Award (2007).