International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 6 Issue: 3 165 - 169

Single Secret Key Crptosystem for Secure and Efficient Exchange of Data in Cloud

Dr. Hareesh K
Associate Professor
Department of Computer Science & Engineering
Government Engineering College Ramanagara
Karnataka, India
hareeshk.gec@gmail.com

Mr. Vishwas S
Assistant Professor
Department of Computer Science & Engineering
KVG College of Engineering
Sullia, Karnataka, India
vshcool@gmail.com

***Abstract*—**Nowadays users are storing their personal data on a cloud storage because of its numerous advantages. One of the important advantage in cloud storage is sharing of data between users or between organizations. In this paper we propose a simple, flexible, efficient and secure data sharing method for the cloud users. Here we are describing a special type of public key encryption scheme where public key, master-secret key, single secret key and cipher text sizes are constant. Single secret key can be obtained by combining number of secret keys. The sender can securely share multiple files with receiver by encrypting each file using a separate public key. Then the sender will combine all the public keys to form a single secret key which is exchanged with receiver by using Diffie-Hellman algorithm. Advantage of small single secret key is user can store this decryption key on a resource constraint devices like smart cards, smart cell phones or sensor nodes. Receiver can download the number of files by using single secret key.

Keywords-*Encryption;decryption;cloud;datasharing;singlesecretkey.*

*****

I. INTRODUCTION

Cloud computing has been envisioned as the next generation information technology architecture for enterprises, due to their long list of unprecedented advantages in the information technology history like ubiquitous network access, on demand self-service, rapid resource elasticity, independent location resource pooling, usage based transference of risk and pricing. The Encryption is the one most fundamental cryptographic function available in cloud. It has certain key management problems so that its practical adoption is often hampered. Due to rapid progress in technologies underlying multi networking made the development of many group oriented application, for example video conferencing, communal gaming, pay per view broadcast of sport events.

For the purpose of security group oriented communication which needs only authorized members having the privilege for group communications also known as access control. In access control using encryption key, contents are encrypted this is known as session key (SK). It can shared by all legitimate group members, also there are many key management schemes are existed example for such type of schemes are contributory scheme & centralized scheme. Multi-level access privilege which is supported by group access control mechanism is known as hierarchical access control. Multi communication has the problem of hierarchical access control it can be converted into a set of single set of single session access control problems.

Many group key management schemes which existed have already solved these problems managing the key for all members having different access privileges also known as multi group key management it achieves the forward and backward security. The time dependent encryption where a designated recipient cannot decrypt a cipher text before a semi trusted time server releases a trapdoor associated with the release time of encryptors choice is known as Timed Release Encryption (TRE). The traditional timed release encryption supports only the single recipient that seems undesirable as a message might be intended for several recipients simultaneously.

Many organizations due to less cost use cloud for large scale data storage. The members of an organization can share the data with other members by loading the data to the cloud, there are many organization uses benefits of the cloud storage and sharing service are numerous, example for such sharing service are the many employees who are part of the international enterprises, collaborative web application providers with a large user base, etc. The economic benefits can be achieved by making the out sourcing data as attractive. Security is the main key factor that makes the cloud wide development.

The data operation cannot be transparent for user in cloud, user have a huge concern on data integrity. In order to maintain the data integrity the user or the data owner can compute the verification of metadata in their data, and upload both of this to the cloud. Not only the user can verifies this but also public verifiers can be verified this. Assume that User A uploads all his private data (photos) on to the Google drive. User A is not relying on security provided by Google drive so user encrypts all his photos using key and then user uploaded photos to cloud. User A friend user B asked to share his photos in which user B appeared. User A can share photos with user B but the problem is how to transfer the decryption authority to user B. User A is having two methods to encrypt his photos 1) Encrypts all his data using symmetric key and deliver that key to user B 2) Encrypt the files with different keys and sends all keys to user B. Symmetric key method is not suitable because user B can able to access all data of user A. In the second method if user A wants to send hundred photos to user B then user has to send hundred keys which is not acceptable. Solution is user A has to encrypt the files by using different public keys but user has to send a constant-size single decrypt key to user B. Constant-size single decrypt key can be obtained by combining number of secret keys. Here encryption is done by using public key as well as ciphertext class

identifier. Key owner will have master secret key which is used to get number of secret keys. Here size of public key, master secret key, single secret key and ciphertext are constant. This paper is organized as, section II contains literature survey, section III contains background work and section IV deals with Implementation details, section V discusses on result analysis and section VI contains conclusion.

II. LITERATURE SURVEY

C. Wang et al., [1] proposed "Privacy Preserving Public Auditing for Secure Cloud Storage". This paper describes that external auditor could audit the user's cloud data without knowing the content of the data. The first most scheme to support scalable and efficient privacy-preserving public storage auditing in cloud. Multiple tasks that are delegated can be performed simultaneously in privacy-preserving manner by the external auditor or 'Third Party Auditor' (TPA). Security and performance of the proposed schemes are justified through concrete experiments and comparisons. Advantages of this paper is the TPA cannot build up a correct group of linear equations and therefore cannot access the user's data content, no matter if he/she collects same set of file blocks. This scheme makes use of a public key-based Homomorphism Linear Authenticator (HLA), to provide public auditability for the auditing protocol. The TPA does not maintain any secret keying material or states between audits, and no potential online burden on users. This ensures privacy for the user's data. Disadvantages of this paper is fear of leakage of user's outsourced data and there is a tradeoffs between storage and communication. The user has to choose between the storage/communication tradeoff parameters for their different system application scenarios.

B. Wang et al., [2] proposed "Storing Shared Data on the Cloud via Security-Mediator". Here the cloud data's integrity is verified by a public verifier. The verification metadata that ensures data integrity is not forgeable under adaptive chosen-message attack. The identity of a data owner is not revealed to a public verifier during the verification of data integrity. The communication requirement between a security mediator (SEM) and a data owner during signature generation should be small. Advantage is ease of support for dynamic groups that is new users can be added into a group and members existing can be revoked from a group. Disadvantage is data owner's anonymity is compromised.

J. Benaloh et al., [3] proposed "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records" is a secure and private storage of patients' medical records. The patient's record is divided into a chronicle structure, each portion of which is encrypted with a corresponding key. The patient stores a root secret key, from which a tree of subkeys is derived. The selected subkeys are distributed for decryption of various records. The design prevents unauthorized access to patient's medical data by data storage and healthcare providers, pharmaceutical and insurance companies, or others who have not been given the appropriate decryption keys. PCE allows the patient to generate subkeys which allows her delegates to search and access only specified parts of her record. Advantages are key revocation where patient has the option of changing keys by decrypting the portions of his record locally and re-encrypting with new subkeys. Emergency response where patients are given to wear or carry an enhanced medic-alert bracelet or similar devices which function for engaging an alarm. Patient Key Management where patients are allowed to keep a hardware device that stores a back-up of their root secret key. Usability where the system must be preset with several different options defining default hierarchies and sets of keys to issue to family members, doctors, devices, and so on. Disadvantages are reusability of the same key for the same category to generate many indexes, one obtains somewhat weaker security properties and there will be a delay between the time when documents are uploaded and are available for searching.

D. Boneh [4] proposed "Identity-Based Encryption from the Weil Pairing" where designing of leakage flexible cryptosystem using identity based encryption which allows efficient key delegation. Improving leakage allowed from each secret key, which is fraction of its size. Saving expensive secure storage without self-managing a hierarchy of delegation classes. Advantage is use of self-authenticating public keys. Identity-Based Encryption (IBC) schemes provide implicit and non-interactive pre-authentication among all network nodes. IBC schemes provide implicit public key validity checks. IBE schemes are affected by key attacks. There are two kinds viz., passive attacks by honest-but-curious Parameter Key Generation center (PKGs) and active attacks by dishonest PKGs. Active attacks cannot be fully prevented.

Y. Sun and K. J. Ray Liu [5] proposed a method "Scalable hierarchical access in Secure Group Communications" where many group communications require a security infrastructure that ensures multiple levels of access privilege for group members. Access control in hierarchy is prevalent in multimedia applications, which consist of users that subscribe to different quality levels or different sets of data streams. The proposed multi-group key management scheme achieves forward and backward security when users join the group communication with certain access privilege. Thus, it is necessary to develop group access control mechanism that supports the multi-level access privilege, which is referred to as the hierarchical access control. The advantages are useful in development of many group-oriented applications, such as video conferencing, pay-per-view broadcast of sport events and communal gaming. The disadvantage are 1) the hierarchical access control problem in multicast communications can be converted into a set of single-session access control problems, which have already been solved by many existing group key management schemes. By doing so, the key management for each data stream is performed separately. This method leads to inefficient use of keys and does not scale well when the number of data streams increases. 2) Compared with the tree-based contributory scheme, the multi-group contributory scheme significantly reduces the computation and latency associated with key establishment and update.

Ran Canetti Susan and Hohenberger [6] proposed a method "Chosen-Ciphertext Secure Proxy Re-Encryption" where in a proxy re-encryption (PRE) scheme, a proxy is given special information that allows it to translate a ciphertext under one key into a ciphertext of the same message under a different key. The proxy cannot, however, learn anything about the messages encrypted under either key. Encryption is one of the

most fundamental cryptographic functions, and its practical adoption is often hampered by key management problems. They address the problem of obtaining PRE schemes that are secure in arbitrary protocol settings, or in other words are secure against chosen ciphertext attacks. The concept of a CCA secure PRE scheme sounds almost self-contradictory, since on the one hand we want the ciphertexts to be non- malleable, and on the other hand we want to allow the proxy to "translate" the ciphertext from one public key to another. Still, they formulate a meaningful definition of CCA-secure PRE schemes, along with a construction that meets the definition in the standard model and under relatively mild hardness assumptions for bilinear groups. The advantages are it is much faster than asymmetric systems and it is hard to break if using a large key size. The disadvantages are key distribution- it requires secure mechanism to deliver keys properly and limited security

III. BACKGROUND

Cryptographic key assignment schemes [7, 8, 9 and 10] aim to store and manage the secret key in a cost effective manner. In tree structure if a key is assigned to a branch then the same key is used to derive the keys for its descendent nodes. Sandhu [11] proposed a method to generate symmetric keys in tree hierarchical fashion.

Consider a tree structure as shown in figure 1. In a tree every node represents secret key but end nodes (leaf nodes) represents the keys for the individual ciphertext classes. Darken circles represents keys for the classes to be delegated. Circles surrounded by dotted lines represents keys to be granted. By using keys of a non-leaf node it is possible to derive a keys of its descendent nodes.

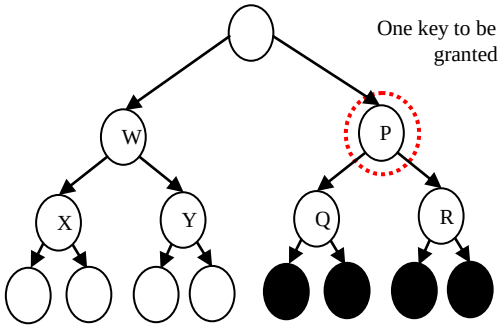

Figure 1(a): Compact key for tree structure

In figure 1(a), if user A wants to share all the files in 'P' category, user has to provide key for only 'P' node this itself will grant keys for all its descendent nodes ('Q' and 'R'). This is suitable when user wants to share more classes of the same branch so parent key is sufficient

Consider figure 1(b), if user A shares 'X1' ('W' → 'X' → 'X1') and Y1 ('W' → 'Y' → 'Y1') with another user B and that user is also having rights to view some of user A's 'P' data then user A is having more number of keys to share with user B. This results in increasing the key size. This approach is not suitable when user wants to share files from different sets. Generally hierarchical approach is solves the problem partially if a user wants to share the files under same branch. Here number of keys is directly proportional to the number of branches.

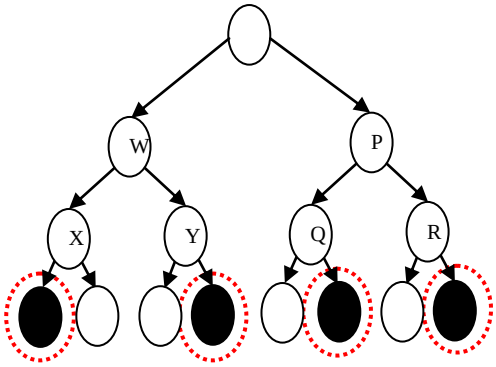

Figure 1(b): Compact key is not always a solution in static hierarchy

IV. IMPLEMENTATION

The first part of implementation focus on framework for single secret key and the second part focus on how to share encrypted data in cloud.

A. *Framework for Single Secret Key*

This scheme consist of six phases. The first phase is an Initialize phase where sender establishes the public system parameter. The second phase is a Key generation phase where public keys and master secret key is generated. Third phase is Encryption. Fourth phase is Aggregation of multiple keys into a single secret key. Fifth phase is transferring of single secret key. And last phase is Decryption.

- **Initialize (1^a, z):** This phase is executed by sender to create an account on untrusted server. 1^a is a security parameter and z indicates number of ciphertext classes.
- **Key Generation:** Sender will executes this phase to randomly generate both public key (pKey) as well as master secret key (mKey).
- **Encryption (*pKey, i, msg*):** Here message *msg* is encrypted by using public key *pKey* and ciphertext class identifier *i*. output of encryption phase is ciphertext cText.
- **Aggregation (*mKey, S*):** Aggregation combines the number of keys present in set *S* by using master secret key *mKey* and produces single secret key denoted as *sKey*.
- **Transfer:** Here the *sKey* is transferred to destination by using Diffie-Hellman key exchange approach.
- **Decryption (*sKey, S, i, cText*):** This phase is executed by destination who received single secret key *sKey*. Cipher text *cText* is decrypted with the help of *sKey*, a set *S*, and cipher text identifier *i* to get a original message *msg*.

B. *Sharing Encrypted Data in Cloud*

This scheme allows the sender to share their data in a confidential and selective way by using constant size single secret key. The figure 2 represents sharing of data in a cloud by using single secret key. Suppose user A wants to share his data $m_1, m_2, m_3, \ldots, m_n$. First user has to perform Initialize (1^a, z) to get parameters and then user will execute Key Generation to get

both public key (*pKey*) as well as master secret key (*mKey*). User A made public access to *pKey* and parameters obtained in Initialize phase. *mKey* is kept in a private mode. User A then encrypts the message m_n by using *pKey* and ciphertext indicator *i* to get ciphertext that is *cText = Encrption (pKey, i, m_n).* The encrypted data is uploaded into cloud. When a user B requested photos with user A, user will choose a set *S* of photos and calculate single secret key by executing Aggregation (*mKey, S*). Now user A is having a constant size single secret key called *sKey*. This is key is easy to send to user B by using Diffie-Hellman key exchange algorithm. When user B receives the single secret key then he will download the ciphertext cText. With a single secret key User B can decrypt the ciphertext using Decryption (sKey, S, i, cText) for each i € S.

C. *Discussion*

The single secret key and cipher text are constant size. The constant size is obtained from the system parameter of linear size. The system parameter can be stored in local storage, which is non-confidential or the service company provides a cache for storage. The trusted party may also generate the system parameter. The users trust parameter-generator for erasing the values used, securely.

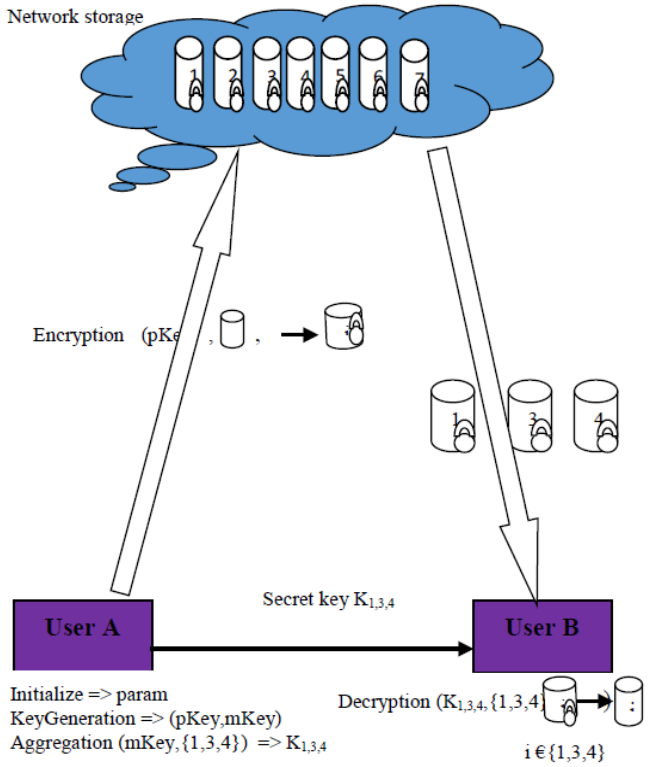

Figure 2: Data sharing in cloud by using single secret key

V. IMPLEMENTATION

For comparison consider the space requirements of tree based key assignment approach which is described in section III. The key hierarchy is complete binary tree with a height *h* so it supports up to 2^h ciphertext classes. Consider the figure 1(a), the user can be granted to access to 2^h classes by using only one key. To decrypt the ciphertext, sometimes user needs more number of keys as shown in figure 1(b). So we have to focus on number of symmetric keys (n_a) to be assigned in this approach.

A.

Consider there is 2^h ciphertext classes and r is the delegation ratio. Delegation ratio is ratio of the delegated ciphertext classes to the total classes. If delegation ration r=0 then n_a is zero that is all the classes are not accessible. If r=100% then n_a is as low as 1, that is all the classes are accessible.

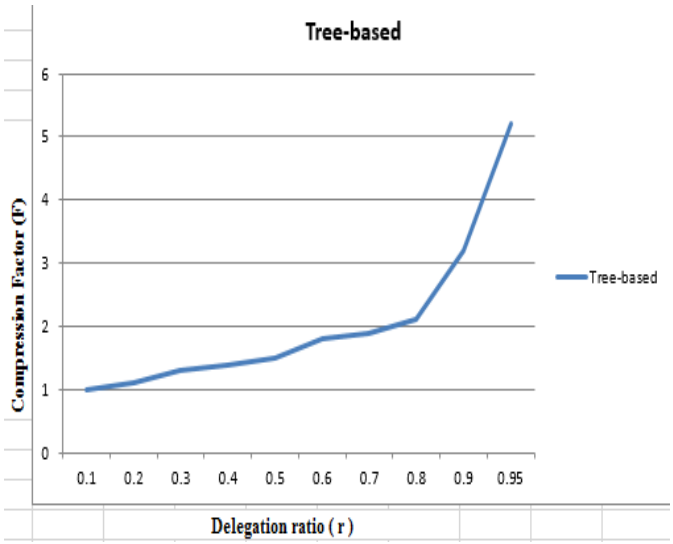

Figure 3: Compression factor v/s delegation ratio in tree-based approach

Compression factor (*F*) is the ratio of total number of delegated classes ($r2^h$) to the number of required granted keys (n_a). If the compression factor *F* is high then the granted key can able to decrypt more number of ciphertexts. Relationship between compression factor *F* and delegation ration *r* is shown in figure 3. For delegation ration *r = 0.90* the compression factor *F = 3.2* and for delegation ration *r = 0.95* the compression factor *F = 5.5*, which indicates that we get high compression ration only when *r* is very close to *1*.

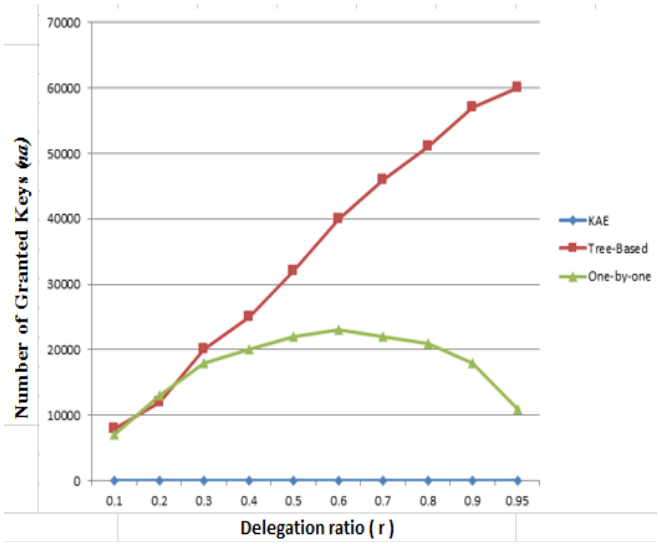

Figure 4: Number of granted keys (n_a) required for different approaches in the case of 65536 classes of data

Figure 4 represents number of granted keys between proposed single secret key, tree-based approach and one-to-one approach. As in the figure 4, in one-to-one approach the number of granted keys is directly proportional to number of delegated ciphertext classes. In tree-based approach, number of granted keys saved according to the delegation ratio. In the proposed single secret key approach, delegation of decryption can be efficiently implemented with the fixed size single secret key.

In the proposed approach the Compression factor (*F*) is considered as a tunable parameter at the cost of *O(n)* sized system parameter where *F = n.* At a constant time encryption is achieved and decryption is achieved in *O(|S|)* group multiplications with two pairing operations, where *S* indicates set of ciphertext classes which is decryptable by granted single secret key. The experimental results are shown in Table 1. The execution times for *Initialize, Key Generation, and Encryption* are independent of delegation ratio *r.* Time complexities of *Aggregation* and *Decryption* are linearly proportional to delegation ratio *r.* The experiment is conducted for 65536 number of classes.

VI. CONCLUSION

In cloud storage, protection of user's data privacy is a challenging issue. Cryptographic techniques are more vulnerable to attacks due to usage of advanced mathematical tools. This necessitates usage of more number of keys for an application. The proposed scheme facilitates framing of single secret key by combining multiple public keys. This approach reduces transmitting more keys over the network in between communicating parties. The proposed scheme is advantageous as it can combine public keys from independent classes. Regardless of number of public keys, the combined secret key formed will have same size of any given public key.

REFERENCES

[1] C. Wang, S. S. M. Chow, Q. Wang, K. Ren and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," IEEE Transactions on Computers, vol. 62, no. 2, pp. 362–375, 2013.

[2] B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," in International Conference on Distributed Computing Systems - ICDCS 2013. IEEE, 2013.

[3] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," in Proceedings of ACM Workshop on Cloud Computing Security (CCSW '09). ACM, 2009, pp. 103–114.

[4] D. Boneh and M. K. Franklin, "Identity-Based Encryption from the Weil Pairing," in Proceedings of Advances in Cryptology – CRYPTO '01, ser. LNCS, vol. 2139. Springer, 2001, pp. 213–229.

[5] Y. Sun and K. J. R. Liu, "Scalable Hierarchical Access Control in Secure Group Communications," in Proceedings of the 23th IEEE International Conference on Computer Communications (INFOCOM '04). IEEE, 2004.

[6] R. Canetti and S. Hohenberger, "Chosen-Ciphertext Secure Proxy Re-Encryption," in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07). ACM, 2007, pp. 185–194.

[7] G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys," in Proceedings of Advances in Cryptology – CRYPTO '89, ser. LNCS, vol. 435. Springer, 1989, pp. 316–322.

[8] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 14, no. 1, pp. 182–188, 2002.

[9] G. Ateniese, A. D. Santis, A. L. Ferrara, and B. Masucci, "Provably-Secure Time-Bound Hierarchical Key Assignment Schemes," J. Cryptology, vol. 25, no. 2, pp. 243–270, 2012.

[10] R. S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," Information Processing Letters, vol. 27, no. 2, pp. 95–98, 1988

[11] S. G. Akl and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Transactions on Computer Systems (TOCS), vol. 1, no. 3, pp. 239–248, 1983.

[12] Cheng-Kang Chu, Shreman S. M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage," IEEE Transactions on Parallel and Distributed Systems, vol. 25, issuue 2, 2014.