Key Trees Combining Algorithm for Overlapping Resource Access Members

Amar Buchade, Rajesh Ingle
(Corresponding author: Amar Buchade)
Computer Engineering and Information Technology Department, College of Engineering, Pune
Wellesley Rd, Shivajinagar, Pune, Maharashtra 411005, India
(Email: amar.buchade@gmail.com)
(Received July 30, 2015; revised and accepted Sept. 30 & Oct. 13, 2015)

Abstract

In cloud computing environment, resources are accessed by multiple members. Resources may be considered as VM, CPU, Storage etc. Group key management required when multiple members in group accesses the resources securely. In existing group key management, separate key trees are formed even if members are common in another group to access the resources. The solution to this is to form the combined key trees for resources which containing overlapping members. Through the analysis it is observed that computational overhead is decreased by 22% if we combine the key trees than separate key trees for each resource.

Keywords: Group key management, resource, resource tree, security

1 Introduction

In Cloud computing, resources may be simultaneously accessed by multiple members. Here the resources may be considered as database, CPU, storage, applications. The members can be overlapped to access the above resources.

Other example HDTV, users can subscribe to various layers such as base layer, medium layer and enhanced layer channel. Users which subscribed medium layer can watch HDTV base layer as well as medium layer channel. User which subscribed enhanced layer can watch base layer, medium layer as well as enhanced layer channel.

In existing group key management, single separate key tree is built to form group key even if members are overlapped to access the resources. Member has to maintain keys for each key tree. The solution to this is to form the combined key trees for resources which containing common members.

To form the group key, TGDH protocol is used [3, 6, 7]. More specifically our contributions are

1) Combined key tree algorithm;
2) Algorithms: Single, batch join, single, batch leave;
3) Formulation of computational cost;
4) Computation cost analysis of resource key formation for separate key trees and combined key tree in terms of number of modulation exponentiation operations and sequential operations.

The paper is organized as follows. In Section 2, we present about resource key tree. Section 3 presents combining key trees algorithm, Section 4 covers Results and Analysis, Section 5 presents conclusion.

2 Resource

2.1 Initializations

Let Resource group $R = \{R_1, R_2, R_3, R_4, \cdots, R_n\}$. Consider two resources $R1$ and $R2$.

$R1 = \{m_1, m_2, m_3, m_4, \cdots, m_n\}$ be the members accessing resource $R1$.

$R2 = \{n_1, n_2, n_3, n_4, \cdots, n_n\}$ be the members accessing resource $R2$.

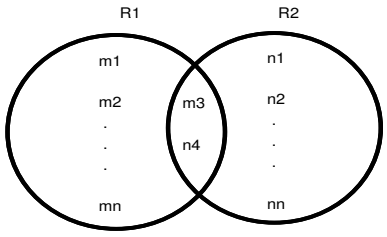

Figure 1: Resources, members representation

From Figure 1 it is observed that m3 and n4 are overlapped to access the Cloud resources. Assume $R1 \cap R2 = cm$, where cm is number of overlapping members which accesses the resources $R1$ and $R2$.

2.2 Resource Tree For Group Key Formation

Figure 2 shows resource tree with leaf nodes represents group members m_1, m_2, m_3, m_4, etc.

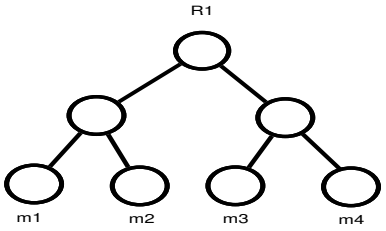

Figure 2: Resource key tree

In TGDH [1, 2, 7, 12, 13], group key is formed from bottom-up fashion. Following are the steps to form the group key.

- Members m_1, m_2, m_3, and m_4 have $\alpha 1, \alpha 2, \alpha 3$ and $\alpha 4$ private keys respectively.
- Each member forms the public key called as blinded key. In this case, g is generator, p is prime number.
- Member blinded keys are $g^{\alpha 1} \bmod p$, $g^{\alpha 2} \bmod p$, $g^{\alpha 3} \bmod p$, $g^{\alpha 4} \bmod p$.
- Each member with its key and sibling blinded key forms intermediate key. For that path from leaf node to root node is traversed.
- Resource group key is formed as below.

$$g^{\alpha 1 \alpha 2 \alpha 3 \alpha 4} \bmod p.$$

Number of modular exponential operations required when four members are equal to 12; In general, when number of members are N, the modular exponential operations are equal to

$$N + N \log_2 N = N(1 + \log_2 N).$$

2.3 Resource Membership Matrix

Every member that accesses the resource, entry is made in resource membership matrix also for any member that joins/leaves in single or batch makes. Resource matrix contains the following entries.

Rows represents members $\{m_1, m_2, m_3, \cdots, m_n\}$ and columns represents resources $\{R_1, R_2, R_3, \cdots, R_n\}$

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots \\ 1 & 1 & \cdots & \cdots \\ 1 & 0 & \cdots & \cdots \\ 1 & \cdots & \cdots & \cdots \end{bmatrix}$$

It shows that there are $R1$, $R2$, $\cdots$, Rn resources. Member m_1 accesses resource $R1$ while member m_2 accesses resource $R1$ and $R2$, i.e. overlapped to access the resources $R1$ and $R2$.

3 Combining Resource Key Trees Algorithm

In existing key management algorithm [5, 8, 9, 10, 11, 14, 15] seperate key tree is built for each resource, even if same members are accessing multiple resources. Thus we can combine multiple resource key trees. Algorithm 1 illustrates combining resource key trees algorithm. Computation cost analysis is given in Section 4.

Algorithm 1 Combining resource key trees algorithm

1: Begin
2: Let R be the set of resources R_1, R_2, R_3, R_4, $\cdots$, R_n;
3: Let $M = \{m_1, m_2, m_3, \cdots, m_n\}$ be set of members;
4: Each member keep the track of members through resource access matrix;
Rows represents members $\{m_1, m_2, m_3, \cdots, m_n\}$ and columns represents resources $\{R_1, R_2, R_3, \cdots, R_n\}$:

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots \\ 1 & 1 & \cdots & \cdots \\ 1 & 0 & \cdots & \cdots \\ 1 & \cdots & \cdots & \cdots \end{bmatrix}$$

5: Identify the members which are overlapped to access multiple resources;
6: Build the key tree of overlapped members. Maintain the following entries in Table 1;
7: Identify the members which are not overlapped. Build the key tree of members which are not overlapped;
8: Combine the trees which are formed during Step 6 and Step 7;
9: End

Table 1: Resources containing overlapped members

Index	Resources	Overlapping members
1	$R1, R2$	m_2

3.1 Computational Cost for Group Key Formation

There can be multiple members overlapping to any resources. Let MEO denotes Modular Exponential Operations; RMM denotes Resource Membership Matrix. MEO after combining key trees is equal to MEO for separate key trees minus MEO due to overlapping members:

$$\begin{aligned} & MEO\_for\_separate\_key\_trees \\ = & \sum_{i=1}^{k} N_i(1 + \log_2 N_i) \end{aligned}$$

and

$$MEO\_due\_to\_overlapping\_member = \sum_{index=1}^{Tindex} (RTcount[index] - 1) \cdot (CM[index](1 + \log_2 CM[index]))$$

where

N is equal to number of members per key tree;

k is equal to total number of key trees;

$Tindex$ is equal to number of entries in Table 1;

$RTcount$ is equal to total resource count per entry;

CM is equal to number of members overlapped per entry.

It is observed that computation cost in terms of number of modular exponential for separate key tree is $O(N)$ while for key trees combined is $O(N - RCm)$.

Thus we can observe that number of modular exponential operations required in separate key trees is more, i.e. RCm compared to the combined key trees.

Table 2: Complexity

Best Case	Worst Case
$\Omega(N)$	O(2N)

Table 2 shown the complexity in terms of MEO:

1) Best case complexity in terms of MEO is when all members of resource groups are overlapped to access the resources.

2) Worst case complexity in terms of MEO is when members of resource groups are not overlapped to access the resources.

3.2 Single Member Join

Algorithm 2 explains algorithm for a member single join.

In single member join algorithm, it requires two messages:

1) Message from member for accessing the resource.

2) Message from sponsor to send the blinded key to form the group key.

Table 3 shows the single join. Table 4 represents the message while joining the group for accessing the resource. The member which wants the access of resource broadcast the message containing message id, its originating address, list of resource membership, request for which resource. This helps to each member to make the entry in resource membership matrix.

Algorithm 2 Single member join

1: Begin
2: Joining member broadcasts for resource access with its message contains whether it is already a member of other resource.
3: Each member of resource/s including sponsor notices it and makes this entry in resource membership matrix.
4: Each member looks into resource membership matrix.
5: Each member builds its own key tree by considering overlapping members and non overlapping:
   1) Joining member which is not going to be overlapped, it is added as per TGDH.
   2) Joining member which is already accessing other resource (overlapping) becomes the sponsor.
6: Joining member which is overlapped forms key graph of resources.
7: Non overlapping member maintains its own key tree.
8: Sponsor computes the blinded key and broadcasts it.
9: Group key establishment as in Section 2.2.
10: End

3.3 Batch Join

There can be members which simultaneously access the resources [4]. The following things can be happened,

1) Some members in group may access single resource;

2) Some members in group may access the multiple resources at particular instant of time.

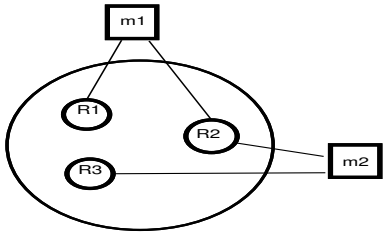

Figure 3: Members join

Figure 3 shows that member m_1 accesses the resources $R1$, $R2$. Member m2 accesses the resources $R2$ and $R3$.

In existing key management, seperate/isolated resource key tree is formed for members that accessing multiple resources. Algorithm 3 explaining when multiple members requests for accessing the resources.

3.3.1 Finding Suitable Position for Common Members in Tree

After building subtree of the overlapping members,it is inserted at the root of the tree to minimize the height of the tree.

Table 3: Single join

Messages	Unicast	Broadcast	MEO
2	0	2	$(N+1)(1+\log_2(N+1))$

Table 4: Message from joining member

Message Id	Source Address	Broadcast Address	Resource Membership	Resource Request

Algorithm 3 Batch join

1: Begin
2: Joining members broadcasts for resource with its message contains whether it is already member of other resource.
3: Each member of resource/s including sponsor notices it and makes the entries in resource membership matrix.
4: Joining members can be categorized as:
   1) Some members newly requesting resource.
   2) Some members already accessing of resources and require to access other resource.
5: Build key graph as per Algorithm 1.
6: Sponsor computes the blinded key and broadcasts it.
7: Group key establishment as in Section 2.2.
8: End

3.3.2 Sponsor Selection for Batch Join

Sponsor is overlapped member which accessing the resources. Otherwise it is selected as TGDH approach [7].

Table 5 shows when members join for the resources access. Here n represents number of members currently added for the resources access.

3.4 Single Leave

Algorithm 4 explains algorithm for a member single leave.

Algorithm 4 Single member leave

1: Begin
2: Leaving member broadcasts that it is leaving from particular resource.
3: Each member of resource/s including sponsor notices it and makes this entry in resource membership matrix.
4: Each member builds its own key tree by considering overlapping members and non overlapping members.
5: Sponsor computes the blinded key and broadcasts it.
6: Group key establishment as in Section 2.2.
7: End

In single member leave algorithm, it requires two messages:

1) Message from member for non-access the resource/s.
2) Message from sponsor to send the blinded key to form the group key.

Table 6 shows the single leave analysis. Table 7 represents the message while leaving from the group. The member which leaves from the group, broadcast the message containing message id, its originating address, non-access of resource. This helps to each member to make the entry in resource membership matrix.

3.5 Batch Leave

There can be members which simultaneously access the resources and completes access of resources. In these members,

1) Member may finishes access of single resource;
2) Members may finishes access of the multiple resources.

Algorithm 5 explains algorithm for members Batch leave. Table 8 represents the batch leave analysis.

Algorithm 5 Batch leave

1: Begin
2: Leaving members broadcasts that it is leaving from particular resource as overlapping members in resource groups or as per TGDH if not overlapping exists.
3: If the leaving member is itself sponsor, sponsor selection.
4: Each member of resource/s including sponsor notices it and makes these entries in resource membership matrix.
5: Build the key graph as per Algorithm 1.
6: Sponsor computes the blinded key and broadcasts it.
7: Group key establishment as in Section 2.2.
8: End

3.5.1 Sponsor Selection in Batch Leave

If the sponsor is leaving member, sponsor is selected as one of the overlapping member. If no overlapping member exists, sponsor is selected as per TGDH [7].

Table 5: Batch join analysis

Messages	Unicast	Broadcast	MEO
2	0	2	$(N+n)(1+\log_2(N+n))$

Table 6: Single leave analysis

Messages	**Unicast**	**Broadcast**	**MEO**
2	0	2	$(N-1)(1+\log_2(N-1))$

Table 7: Message details in single member leave

Message Id	Source Address	Broadcast Address	Non-access of which Resource

Table 8: Batch leave analysis

Messages	**Unicast**	**Broadcast**	**MEO**
2	0	2	$(N-n)(1+\log_2(N-n))$

4 Results and Analysis

Analysis is done by taking resources, varying members size. From Figure 4, it is observed that when No. of resources are 2, Total number of Members=250 and overlapping members in resources are increased, number of exponential operations are decreased (22.41%) when key trees are combined.

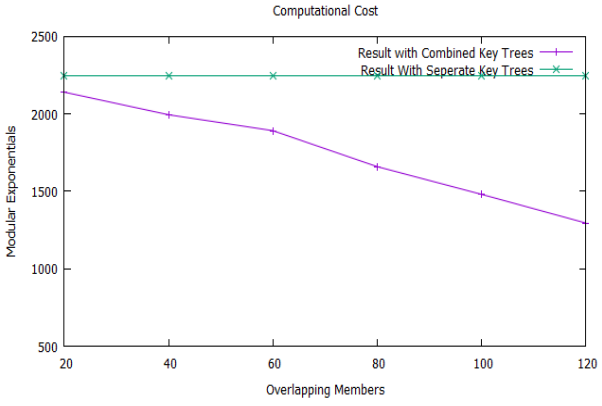

Figure 4: Computational cost, number of resources = 2, total members = 250

From Figure 5, it is observed that average computation cost with separate key trees increased by 21.82% comparing with combining key tree.

From Figure 6, it is observed that when Group size = 100, overlapped members = 10 and as we increase the number of resources, modular exponential operations are decreased when key trees are combined.

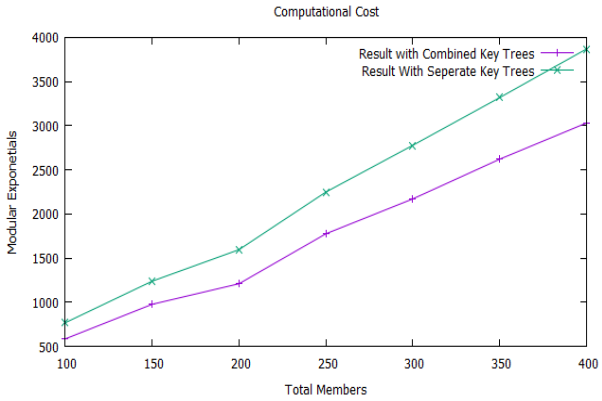

Figure 5: Average computational cost, number of resources = 2

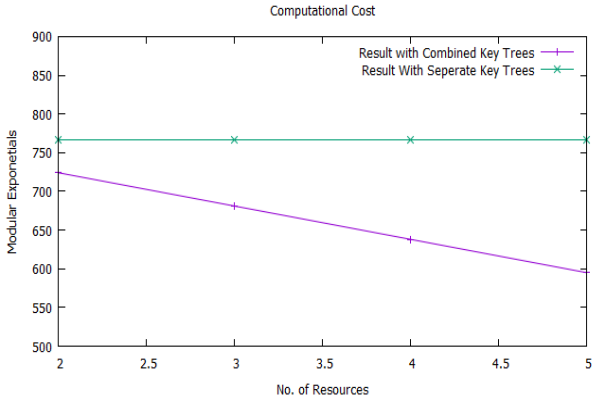

Figure 6: Average computational cost, total members = 100, overlapping members = 10

5 Conclusions

In Cloud computing, resources such as CPU, VM, database, storage and applications are simultaneously accessed by multiple members. The members can be overlapped to access the above resources. Group key is formed by contributing share of each members with TGDH approach. In existing group key management, single separate key tree is built to form group key even if members are overlapped to access the resources.

From the result and analysis, it is observed that there is reduction of computation cost more than 22% when resource key trees are combined.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[2] X. Gu, Y. Zhao, and J. Yang, "Reducing rekeying time using an integrated group key agreement scheme," *Journal of Communications and Networks*, vol. 14, no. 4, pp. 418–428, 2012.

[3] H. R. Hassen, A. Bouabdallah, H. Bettahar, and Y. Challal, "Key management for content access control in a hierarchy," *Computer Networks*, vol. 51, no. 11, pp. 3197–3219, 2007.

[4] R. Ingle and G. Sivakumar, "Tunable group key agreement," in *32nd IEEE Conference on Local Computer Networks (LCN'07)*, pp. 1017–1024, 2007.

[5] S. Jarecki, J. Kim, and G. Tsudik, "Flexible robust group key agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 879–886, 2011.

[6] D. H. Je, J. S. Lee, Y. Park, and S. W. Seo, "Computation-and-storage-efficient key tree management protocol for secure multicast communications," *Computer Communications*, vol. 33, no. 2, pp. 136–148, 2010.

[7] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 60–96, 2004.

[8] I. Lam, S. Szebeni, and L. Buttyán, "Invitation-oriented tgdh: Key management for dynamic groups in an asynchronous communication model," in *41st IEEE International Conference on Parallel Processing Workshops (ICPPW'12)*, pp. 269–276, 2012.

[9] D. Li and S. Sampalli, "A hybrid group key management protocol for reliable and authenticated rekeying," *International Journal of Network Security*, vol. 6, no. 3, pp. 270–281, 2008.

[10] V. S. Naresh and N. V. E. S. Murthy, "Elliptic curve based dynamic contributory group key agreement protocol for secure group communication over ad-hoc networks," *International Journal of Network Security*, vol. 17, no. 5, pp. 588–596, 2015.

[11] M. Rajaram and D. Thilagavathy, "An interval based contributory key agreement," in *IEEE International Conference on Wireless Communication and Sensor Computing (ICWCSC'10)*, pp. 1–6, 2010.

[12] Y. Sun and K. J. Liu, "Hierarchical group access control for secure multicast communications," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1514–1526, 2007.

[13] G. Wang, J. Ouyang, H. H. Chen, and M. Guo, "Efficient group key management for multi-privileged groups," *Computer Communications*, vol. 30, no. 11, pp. 2497–2509, 2007.

[14] H. Xiong, X. Zhang, W. Zhu, and D. Yao, "Cloudseal: End-to-end content protection in cloud-based storage and delivery services," in *Security and Privacy in Communication Networks*, pp. 491–500, Springer, 2012.

[15] K. Xue and P. Hong, "A dynamic secure group sharing framework in public cloud computing," *CIEEE Transactions on loud Computing*, vol. 2, no. 4, pp. 459–470, 2014.

Amar Buchade Amar Buchade is pursuing Ph.D.(Computer Engineering) from College of Engineering Research Centre, Pune under Savitribai Phule Pune University. He has received B.E. and M.E. in CSE from Walchand College of Engineering, Sangli in 2002 and 2005 respectively. His research area is Distributed System, Cloud computing and Security.

Rajesh Ingle is adjunct Professor at Department of Computer Engineering, Government College of Engineering Pune. He is Professor in Department of Computer Engineering, Pune Institute of Computer Technology, Pune. He has received Ph.D. CSE from Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Powai. He has received the B.E. Computer Engineering from Savitribai Phule University of Pune, and M.E. Computer Engineering from Government College of Engineering, Savitribai Phule Pune University. He has also received M.S. Software Systems from BITS, Pilani, India. He is a senior member of the IEEE, IEEE Communications Society, and IEEE Computer Society. His research area is Distributed system security, Grid middleware, Cloud security, Multimedia networks and spontaneously networked environments.