N-ary tree based key distribution in a Network as a Service provisioning model

Anand Kannan[1] and Gerald Q. Maguire, Jr[2]
School of Information and Communication Technology
KTH, Royal Institute of Technology
Stockholm, Sweden
E-mail: {anandk[1],maguire[2]}@kth.se

Ayush Sharma, Volker Fusenig, and Peter Schoo
Fraunhofer Research Institution for Applied & Integrated Security
Parkring 4, 85748 Garching, Germany
Email: {firstname.lastname}@aisec.fraunhofer.de

ABSTRACT
Cloud networking is a new technology which integrates network provisioning with the existing cloud service provisioning models. This integration allows service providers to provision network resources together with network performance guarantees as a part of their service offering. However, the introduction of multiple providers and service levels introduces many security challenges. One such challenge is identity management, especially authentication of different entities. This paper presents an analysis of a management scheme deployed in a simulated cloud network test bed. Our results show that this scheme is faster than binary and erasure encoding schemes. The scheme uses an N-ary approach and thus allows the placement of *n* entities at each level, unlike the binary scheme which is restricted to two entities.

Categories and Subject Descriptors
C.2.4, D.2.12, D.4.6, D.4.8, and E.3

General Terms
Algorithms, Design, Measurement, Performance, and Security

Keywords
Cloud networking, Security architecture, Authentication, Identity management, Trust management, Privacy

1. INTRODUCTION
The entire computing and information technology management service ecosystem, ranging from small and medium enterprises to large-scale conglomerates, has witnessed a paradigm shift in their service delivery and provisioning models during the last decade. Fuelled by the exponentially increasing costs involved in procuring and maintaining resources, organizations have shifted their resources into the "cloud". This has reduced the initial and maintenance expenditures for the information technology (IT) organizations, and revolutionized the entire IT ecosystem. Different flavors of clouds exist in the cloud ecosystem: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). However, each of these service provisioning models shares a common dependability problem due to the lack of guaranteed network resource provisioning between the end-user and the resources allocated to the cloud tenant.

The European Scalable and Adaptive Internet soLutions (SAIL) project [1] focuses on developing a new service provisioning model called Network-as-a-Service (NaaS) which shall ensure virtualized, elastic, dynamic, and on-demand network resources provisioned to the end-user/tenant. The SAIL project has developed a networking-as-a-service provisioning infrastructure utilizing a cloud network (CloNe) architecture [2], at its core to implement the NaaS model. Similar to other service provisioning models, CloNe also suffers from a number of security flaws (an exhaustive list of these is given by Schoo et al. [3]). It is important to carry out an in-depth security analysis of the cloud network architecture to compile a list of relevant security requirements and goals. One of the identified challenges for cloud network architectures is the integration of a secure and efficient key management module [29,30].

This paper explains the design, deployment, and analysis of an N-ary tree based key management algorithm which provides the underlying mechanism for an overall identity management function for the CloNe security architecture. This N-ary scheme allows the placement of *n* entities at each level, unlike the binary scheme [31] which is restricted to two entities. The main contribution of this paper is the deployment and analysis of the key management mechanism in our simulated cloud network test environment, which demonstrates the feasibility and performance of the proposed identity management function deployed in the overall CloNe architecture.

The remainder of this paper is organized as follows: Section 2 covers the related works pertaining to cloud security architecture and key management algorithms. Section 3 elaborates the CloNe architecture and the CloNe security architecture and depicts the interaction sequence between the supporting security functions. Section 4 explains the N-ary tree based key management algorithm, its placement in the CloNe security architecture, a sample interaction sequence, and our analysis results. Section 5 summarizes these results and compares them with the state of the art. Section 6 gives some and suggests future work.

2. RELATED WORK
There are numerous key management and key distribution schemes described in the literature [9, 18-20]. In

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICACCI'12, August 3-5, 2012, Chennai, T. Nadu, India.
Copyright 2012 ACM 978-1-4503-1196-0/12/0010…$10.00.

many of these existing key management schemes, different groups of users obtain a new multicast key for every new session update. Among the various schemes for key distribution, the Maximum Distance Separable (MDS) [21] method utilizes error control coding techniques for distributing re-keying information. In MDS, the keys are obtained based on Erasure decoding functions [22] in order for each of the group members to compute the relevant session keys. In this method the Group center (GC) constructs a non-systematic MDS code *C* over the Galois Field GF(q) and a secure one-way hash function H(.) whose co-domain is GF(q). The GC generates *n* message symbols by sending the code words into an Erasure decoding function. The first message symbol is considered a session key, out of the n messages symbols, and the group members are *not* provided this particular key by the GC. Group members are given the (n-1) message symbols and they compute a code word for each of them. Each of the group members uses this code word and the remaining (n-1) message symbols to compute the session key. The main limitation of this mechanism is its computation and storage complexity. The computational complexity is $lr+(n-1)m$ where *lr* is the size of r bit random number used in the mechanism and m is the number of message symbols to be sent from the GC to group members. If lr=m=l, computation complexity is *nl*. The storage complexity is given by $[log2L]+t$ bits for each member. Where, L is the number of levels of the tree. Hence, GC has to store n ($[log_2L]+t$) bits.

The data embedding mechanism proposed by Trappe, et al. in [23] is used to transmit re-keying message by embedding the re-keying information in the multimedia data. In this mechanism, the computation complexity is O(log n). The storage complexity is directly proportional to the value of O(n) for the server machine and O(log n) for group members. This technique is used to update and maintain keys in secure multimedia multicast communication. The biggest limitation of this mechanism is that a new key called embedding key has to be provided to the group members in addition to the original keys, which increases the overhead.

Key management using key graphs as proposed by Wong, Gouda, and Lam [24] creates secure groups from basic key management graphs mechanism using a star based method and a tree based method. The mechanism is not scalable due to its excessive overhead. A new group keying method called the One-way Function Tree (OFT) algorithm has been proposed by McGrew and Sherman [25] uses one-way functions to compute a tree of keys. In this method keys are computed up the tree, from the leaves to the root reducing re-keying broadcasts to only approximately *log n* keys. The main limitation of this approach is its higher space complexity as compared to [23].

Trappe and Song proposed a Parametric One Way Function (POWF) [20] based binary tree key management scheme. In this scheme, a session key K_s is attached to the tree below the root node. Each node in the tree is assigned a Key Encrypting Key (KEK) which is an Internal Key (IK). Each user is assigned to a leaf and is given the IKs of the nodes from this leaf to the root node. If a balanced tree is complete, i.e., where all the leaf nodes have members associated with them, then it is necessary to generate a new layer of nodes when adding new members. However, when a user wants to join the group, the keys on the path from their assigned leaf node to the root and also the session key must be changed. These new keys are generated by GC. If a user departs from the group, then all the keys from this user's assigned leaf node to the root node become invalid. These keys must be updated and distributed using a bottom up or top down approach. The complexity of storage can be substantially reduced if the numbers of multiplications are reduced. Some of the key management schemes proposed in [26-28] are distributed key management approaches which are characterized by having no group controller. The group key can be generated either in a contributory fashion, where all members contribute their own share to computation of the group key, or generated by one member.

In this paper, we propose a customized key management mechanism which reduces the computational complexity of key generation and distribution; and at the same time increases the overall security by providing a larger key space. This mechanism is more apt for a cloud networking environment which needs to survive the failure of the GC.

3. CloNe ARCHITECTURE

The CloNe architecture is a multi-tier, multi-domain service provisioning model which provisions virtualized, elastic, dynamic, and on-demand network resources to the end-user/tenant. The virtualized network resource is referred to as a Flash Network Slice (FNS). Use of FNSs provides dynamic network resource provisioning capabilities in a heterogeneous multi-operator and heterogeneous network environment. CloNe has been designed to concretize the abstract requirements of a FNS and to ensure that the requested resource is correctly deployed on the underlying resource set. CloNe has an inherent three layer model consisting of its respective set of roles, a collection of interfaces which allow the participating entities and different security/management modules to communicate, and the various modules themselves. A detailed description of the CloNe architecture is given in [2]. Figure 1 shows an abstract view of this architecture.

Each *infrastructure service user* makes an abstract resource request to the *infrastructure service* through the infrastructure service interface. The infrastructure service employs an infrastructure service controller at its core. This controller is responsible for carrying out the goal translation and the decision making. The goal translation module implements the translation of abstract user requests into concrete resource specifications which can be deployed by the infrastructure provider on their infrastructure. Moreover, each goal translation module is responsible for generating *multiple* plausible (pareto-optimal) [5] resource configurations while considering the multiple objectives specified by each participating entity in the CloNe infrastructure. A pareto-optimal solution is defined as a solution whereby none of the participating entities can experience a better result, without ensuring that at least one other entity experiences a reduction in their performance. Therefore, there is a clear need for a decision maker to select the best possible pareto-optimal solution which defines the actual resource configuration to be deployed on the resource set.

If the *infrastructure service* accepts a user's request, it carries out an intermediate goal translation with the help of supporting management and security modules, and then delegates the translated request to one or more *distributed infrastructure service* components by using the infrastructure service interface. Either the request can be completely fulfilled by a single *distributed infrastructure service* components or it will be distributed over multiple components. In the former case, the *distributed infrastructure service* component might decide to collaborate with additional *distributed infrastructure service* components, if it is unable to satisfy the resource request in its entirety. In such a case, the interactions would take place through the distributed control plane (DCP) interface.

Each *distributed infrastructure service* component operates within its individual administrative domain. Each domain is administered by an infrastructure provider, who has complete

control of all the resources in that domain. In order to assist the entire goal translation process, the infrastructure employs supporting management modules including a resource management and a fault management module. The resource management module keeps track of the usage and health of the underlying physical and virtual resources, and is responsible for monitoring their utilization and mapping the requested virtual resources onto the available physical resource set at its disposal. It is supported by a fault management module, which is responsible for monitoring faults, and providing the necessary inputs to the resource management and overall goal translation modules. However, as covered earlier, the cloud network architecture has its own set of problems [3], which has led to the development of a CloNe security architecture [4].

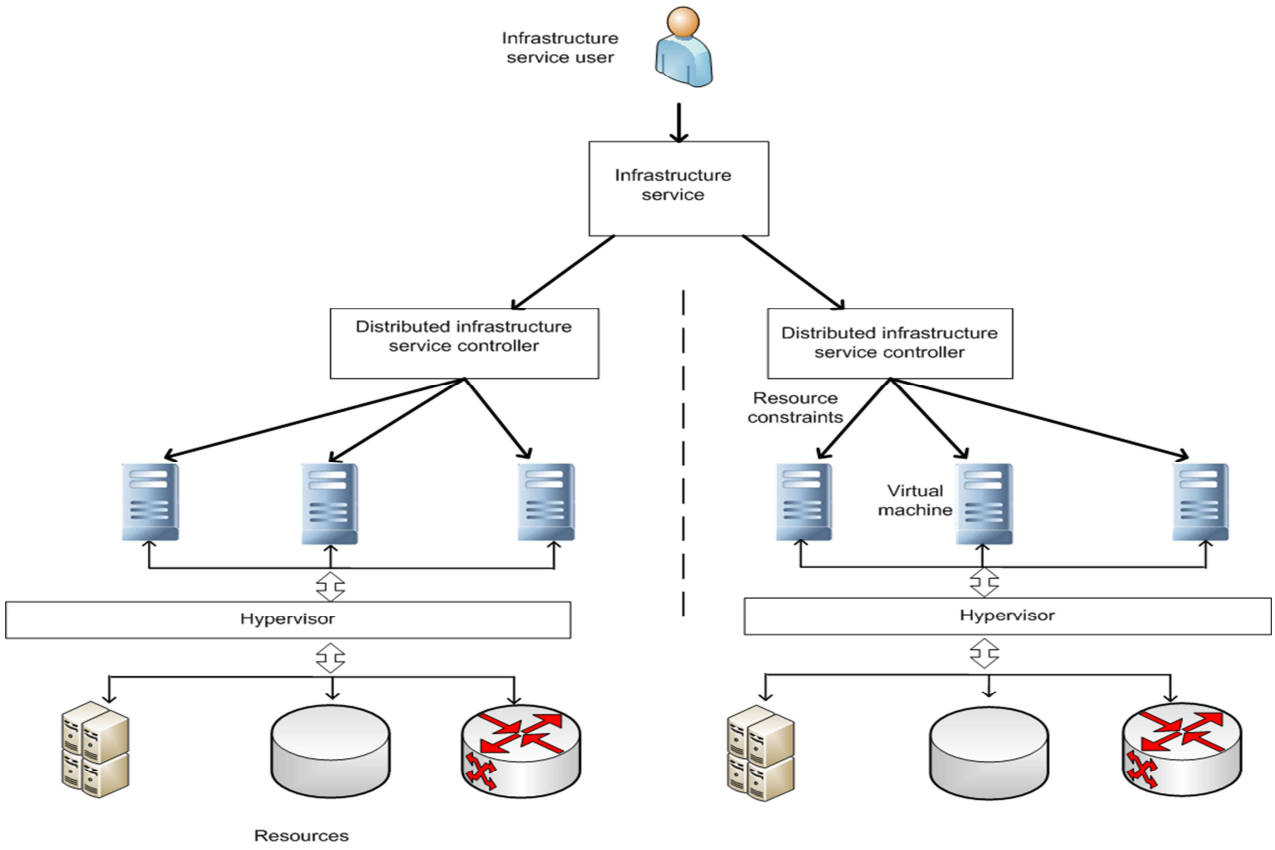

Figure 1: CloNe architecture

3.1 CloNe SECURITY ARCHITECTURE

The essential requirement of the security architecture is to translate the security requirements specified by the tenant into concrete resource constraints. Additional (security) requirements may also be provided by the different entities in the architecture. This security goal translation has been integrated in the overall goal translation function described by Bjurling et al. [6] and deployed in the CloNe architecture.

The resource configurations defined at the end of the translation process is deployed by the *infrastructure provider*, and the translation process is assisted by the different security modules depicted in Figure 2. The various security functions include an access control policy function, an auditing and assurance function, an identity management function, and the central security goal translation function (which forms the backbone of the overall security architecture). The access control policy function is responsible for determining access control policies for each *infrastructure service user*, and will require a suitable access control policy model to define those policies.

To support the access control policy function, an identity management function has been defined to perform the authentication checks of the varied entities in the infrastructure, and to ensure that only authorized parties are provided access to resources and/or services. Access control policies can only be successfully deployed in a system when the identities of the participating entities can be ascertained with a high probability. Therefore, a well- defined identity management function is indispensable to a system which wants to implement an access control policy model. Additional desirable features included in this identity management function include a compliance module, a federated identity management module, and an authorization and use profile management module.

A proposed improvement to the overall CloNe security architecture includes a backbone key management algorithm (to support the identity management function). The following Section 5 will cover design and deployment details concerning a key management algorithm. The core algorithms have been customized and deployed in the CloNe infrastructure, and its evaluation results and comparisons with other algorithms/mechanisms are described in the respective sections.

4. KEY MANAGEMENT

The respective security functions and their interactions with the security goal translation function are depicted in Figure 2. The access control policy function aids the different entities in

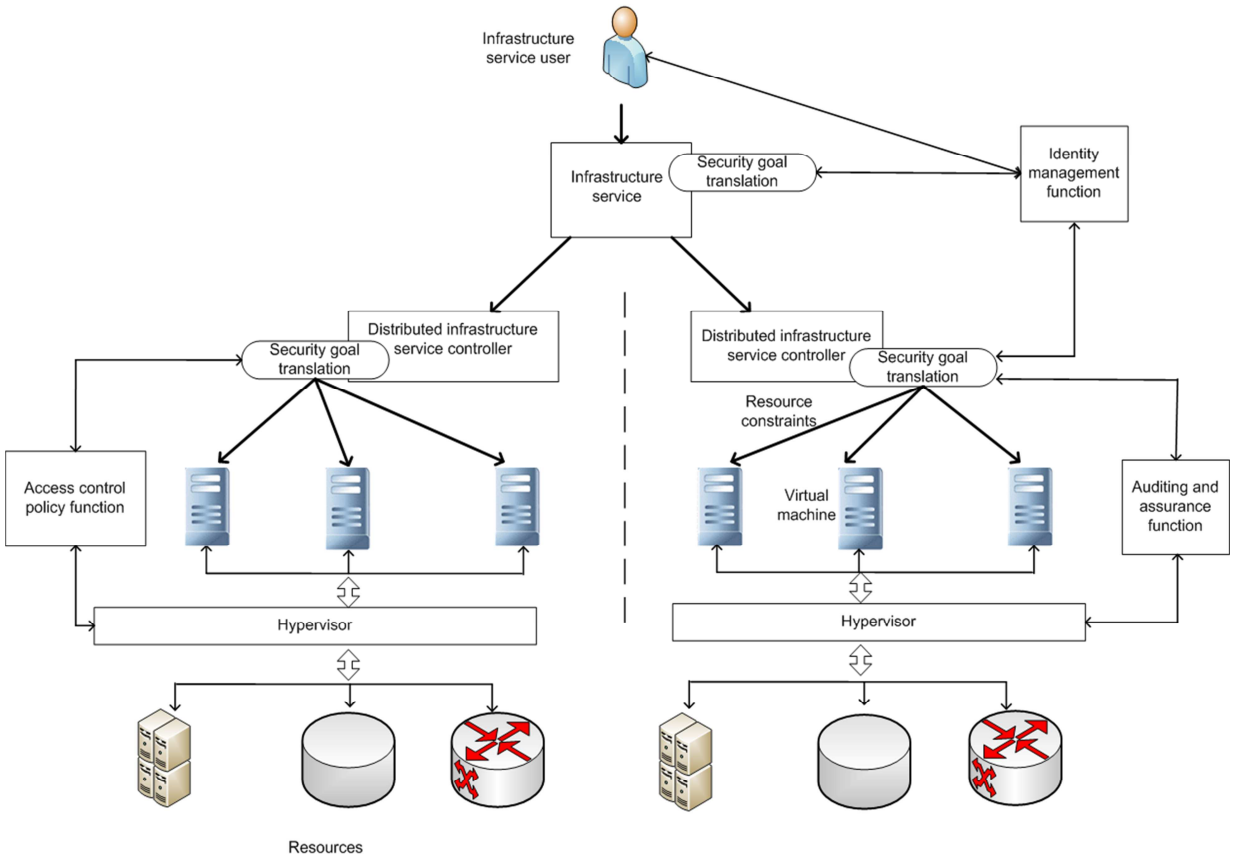

Figure 2: CloNe security architecture

the CloNe environment to set and implement access control policies on the underlying resources, with respect to each *infrastructure service user*. The access control policies may either be directly specified by entities with specific roles (such as the tenant or infrastructure service user, infrastructure service or the infrastructure provider) or could be indirectly derived from the security goals specified by any of these entities described above.

The auditing and assurance function checks whether the parameter constraints, which have been defined by the goal translation function and need to be realized on the underlying hardware resources, have indeed been fulfilled or not. The auditing mechanism is periodically executed, but could also be invoked upon request and/or need. The participating entities might want to verify whether all the security mechanisms functioned properly during a specific interval of time, especially in the event of a security breach. The assurance function is responsible for assuring the infrastructure service user or other entities of the properties of entities/resources it is communicating with.

The identity management solution provides five functions to support the overall security goal translation function. The functions are: identity provisioning, authentication, federated identity management, authorization and user profile management, and compliance. Identity provisioning promotes the secure and efficient management of provisioning and deprovisioning of user identities. Authentication allows credential management, strong authentication and optionally a choice of the desired strength of authentication on the fly, delegated authentication, and managed trust across all entities involved in the architecture.

Federated identity management empowers the cloud tenant to authenticate themselves using their desired identity provider. Therefore, an exchange of identity attributes takes place between identity providers and service providers. Authorization and user profile management is useful for setting up access control policies and trusted user profiles. Information regarding access control policies has to be decided between the *infrastructure service*, *identity provider* (someone who manages the identities of *infrastructure service users* and authenticates them as and when needed), and sometimes the infrastructure service user. The *identity provider* maintains user profiles in tandem with the *infrastructure service user* himself. The policy information is then decided upon between the service provider and tenant. Finally, compliance ensures that the CloNe architecture is compliant with the regulations specified by different organizations/regions and that it satisfies the enterprise and/or country audit and compliance requirements.

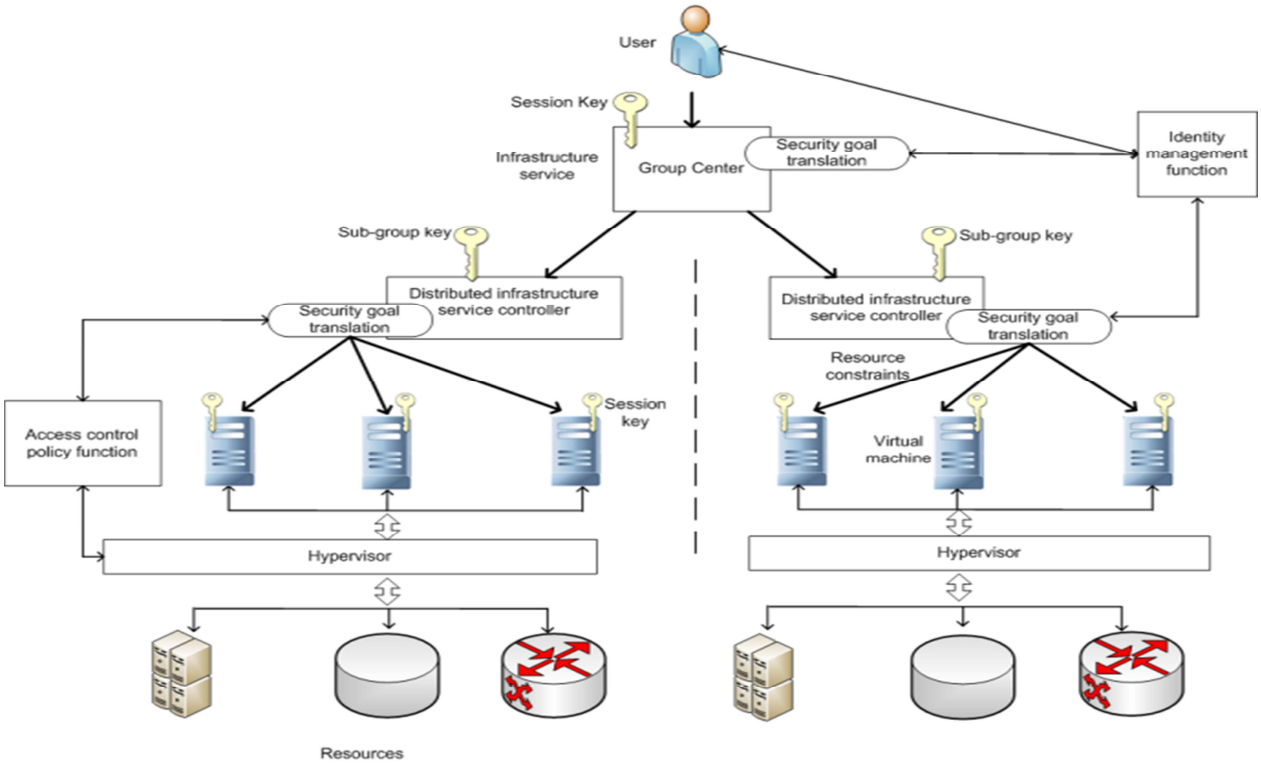

Figure 3: Key management function in CloNe security architecture

The remainder of this section describes the deployment of the key management scheme proposed by Vijaykumar et al [9]. This scheme aids in authenticating the participating entities, namely *infrastructure service*, *distributed infrastructure service* components across multiple infrastructure provider boundaries, and the *infrastructure providers*. The scheme offers a more efficient method of authenticating the participating entities (as it has lower time complexity than binary and erasure encoding methods) in the architecture, and it utilizes a larger key space which improves the overall security of the identity management function.

4.1 Simple test bed

We have created a simple test bed using two computers with similar hardware (CPU: Intel Pentium i7-2720QM (6 m cache, 2.20Ghz), RAM 8 GB, NIC: Intel Ultimate-N 6300 (802.11 a/b/g/n) Half Mini Card, Hard Drive: Seagate 500GB) connected with a cross over cable. Each physical computer hosted five virtual machines with Proxmox VE 1.8, an Open source virtualization environment. Each VM was running Ubuntu 10.10. We used IPv4 as the communication protocol stack. Proxmox VE uses a bridged networking model. These bridges are similar to physical network switches, but implemented in software on the underlying Proxmox VE host. All VMs share a single bridge, thus it was as if virtual network cables from each guest were all plugged into a single physical switch. To avoid cross VM communication, VLANs (implementing IEEE 802.1q) are used to separate the networks as if each VM were separately connected to the underlying physical system. Quagga, a network routing software suite was installed in each physical machine to enable inter-VLAN communication. The Apache Hadoop 1.0.0 framework was also installed in each VM. Each VM can act as a master or a slave depending on the deployed application. The master node can use resources of one or more slaves at any given time. Each VM has to authenticate every other VM before sharing resources. Even if a VM belongs to the same logical rack, they authenticate each other and communications are routed through the virtual router whose routing daemon is running on the underlying physical machine.

4.2 Key computation protocol

The key management algorithm proposed by Vijaykumar et al. [9] can be deployed in both hierarchical and distributed scenarios. In the hierarchical scenario, the value of *n* needs to be fixed, as this defines the number of child nodes each node can have. This is an improvement over a binary tree-based key management scheme, where the value of *n* is fixed as 2. The hierarchical scenario includes a GC, which can have up to *n* child nodes. The GC will in turn have up to *n* sub-group heads. Each sub-group head will have up to *n* child nodes (virtual machines). Each virtual machines creates their own public-private key pairs according to the following formulae as described in [9]:

$$PK_i = y^{\varphi(Ki)} mod\ p \qquad (1)$$

In equation 1, node *i* choses a private key K_i and creates the corresponding public key PK_i. $\varphi(Ki)$ specifies the Euler's totient value of the private key K_i. Additionally, *y* and *p* are public parameters of the chosen group over which the cryptographic operations are carried out.

After the computation of the public key, each node exchanges their public key with the other group members, and together they create the private key for the group. Creating a

public key pair for the group (sub-group head or GC) is not necessary, as there are no group based signatures (hence no need for an asymmetric key pair) required in this method. The messages exchanged between the group members only need to be encrypted with the same group private key being used both for encryption and decryption. As a result this method of encrypting and decrypting group messages will be faster than schemes that utilize an asymmetric key management method [33].

The public key exchange as described in [9] is used to createa group key for an N-ary tree with *n* equal to 2. The value if *n* is set to 2 to keep the example simple. We will assume that the GC is denoted by node *k*. If node *i* has a public key K_i and node *j* has a public key $K_,$, then node *i* sends its public key to node *j* and vice versa. Equation (2) explains how equation (1) can be used by node *i* to compute the public key PK_j.

$$GK_k = PK_j^{\varphi(K_i)} = (y^{\varphi(K_j)})^{\varphi(K_i)} \bmod p \qquad (2)$$

Node *i* receives the public key (PK_j) of node *j* and it knows its own private key, namely K_i. Similarly, equation 3 describes the creation of the group key by node *j*.

$$GK_k = PK_i^{\varphi(K_j)} = (y^{\varphi(K_i)})^{\varphi(K_j)} \bmod p \qquad (3)$$

Node *j* receives the public key of node *i*, while it knows its own private key, namely K_j. Clearly the value obtained from equations (2) and (3) must be same. Thus, this step can be repeated throughout the entire N-ary tree, and then the child nodes can collaborate to compute the group key.

For the distributed scenario, there is no GC or sub-group head. Therefore, the group key will be created by the group members by exchanging their public keys, and there is only a single level of child nodes.

4.3 Sample interaction

The key management function is integrated with the identity management function, and allows the latter to carry out authentication, authorization, and compliance. The access control policy function sets and implements access control policies for the individual entities participating in the CloNe infrastructure, but the majority of access control policies focus on defining access control for the infrastructure service users. The key management algorithm generates, distributes, and resets keys amongst the participating entities, namely the *infrastructure service*, distributed *infrastructure service*, and virtual machines. Multicast communication is an effective routing technology that reduces network traffic and improves application throughput, especially in data center networks [34, 35, 36]. For this reason, it is important to deploy a key management system which supports the authentication of CloNe entities (virtual machines and data center resources) involved in multicast communication. As we will show later the key management scheme proposed by VijayKumar et al. is faster than binary and erasure encoding schemes with the same key space, and it is an excellent choice for multicast (as described in subsection 4.2). This key management scheme allows the different CloNe entities to send encrypted messages to authorized parties. This is extremely important for a scenario whereby virtual machines from two different administrative domains are involved in the same provisioned service. In such a scenario, it is extremely important that virtual machines from either administrative domain which are *not* involved in the provisioning should not be able to decrypt these messages. Moreover, multicast offers an appropriate communication mechanism for two of the proposed use cases for CloNe, namely video distribution and enterprise in the cloud.

In the remainder of this section we will consider a sample interaction in which the user makes a resource request of the *infrastructure service*, and the latter carries out a (security) goal translation, and then delegates the translated resource configurations to the set of *distributed infrastructure service* components. In such a case, a session is initiated and keys are generated for each user of the group, in this case the *infrastructure service*, *distributed infrastructure service* components, and the virtual machines required for the service provisioning. Keys are generated *only* for the group-users that are required in satisfying this service provisioning request. However, ensuring that only the required virtual machines have been allotted keys is a responsibility of the hypervisor deployed at the data centers of each *infrastructure provider*, and this has not been analyzed in this paper. The leaf nodes (virtual machines) generate their respective session keys, and together collaborate using the mechanism described in subsection 4.2 to create a sub-group key (for the *distributed infrastructure service*). The sub-group head (i.e., the *distributed infrastructure service*) collaborates with the leaf nodes to create the key for the GC (in this case the *infrastructure service*) using the mechanism described in subsection 4.2. Therefore, each lower layer node will know the keys of all the nodes lying between it and the GC. If a node leaves the group, then the group should compute new keys for all members lying between the deleted member and the GC. Properly removing departing nodes is an important requirement, because virtual machines might be switched off and restarted due to technical faults and/or overload. In addition, new virtual machines may be assigned for the service (i.e., provisioned), and thus new keys have to be generated during the group join and leave operations in order to ensure forward and backward secrecy. Therefore, as soon as a virtual machine leaves the group, the remaining group members (i.e., the remaining virtual machine instances) will compute new keys for the sub-group head (*distributed infrastructure service*) and the GC (*infrastructure service*). Similarly, when a new member joins an existing service provisioning instance, new keys will be also generated. As noted earlier in this subsection, authenticating the identity of authorized virtual machines is the responsibility of the individual hypervisor instances and has not been covered in this paper.

4.4 Analysis results

In addition to the deployment details of the key management algorithm, it is equally important to describe the performance of the key management algorithm in the CloNe environment. The time taken for generating keys for various key distribution methods with respect to the chosen key management algorithm are covered in [9]. In our work we have carried out additional tests to determine the key computation times with varying group sizes, when the algorithm was deployed in the simple test bed of the CloNe infrastructure described in subsection 4.1. The tests were carried out using a binary tree-based key management scheme, an N-ary tree-based key management scheme, and an erasure encoding scheme all implemented using JAVA programs in order to obtain results consistent with the original simulation results presented in [9]. In the N-ary tree-based key management scheme as the key size increases in terms of the number of bits, the key space also increases. Consider the case of an N-ary tree when the key size is 1 byte, the intruder needs 100 attempts to decipher the message without knowing the secret key. While in a binary tree based key management system, only 10 attempts are needed. Therefore, the N-ary key management system provides 10 times greater security than the binary key management system. This is explained in more detail along with a mathematical proof in [9]. Table 1 shows a comparison between binary tree based key management and N-ary

based key management method, where the key size is taken as the major parameter.

Table 1: Comparison of key space

Binary tree based key management		N-ary based key management method	
Key size (in bits)	Key space	Key size (in bits)	Key space
8	1-10	8	1-100
16	1-100	16	1-1000
24	1-1000	24	1-10000
32	1-10000	32	1-100000
64	1-100000	64	1-1000000

Table 2 shows the number of multicast messages which need to be sent from the GC to the different group members, in order to recover the sub-group key and session key. These results prove that the N-ary method takes fewer re-keying messages, in comparison with the binary key management schemes. A mathematical proof is described in [9]. For the results in Table 2 we have chosen n as 3 , in general for g messages the group size is n^g. However, a disadvantage of the N-ary based key management scheme is that the value of n can not be modified without constructing a new tree using the new value of n. Thus, if $n = 3$, then at each level a node can only have a maximum of 3 children nodes.

Moreover, the graphical results shown in figures 4 and 5 provide additional performance results for the different key management algorithms. Figure 4 compares the key computation results obtained from an N-ary tree-based key management scheme with the binary tree-based method, as well as the erasure encoding method [10]. When the group size is 600, then the key computation time taken by the GC in the N-ary case is 11 ms, which is smaller than the time for a binary tree based key management scheme and slightly smaller than for the erasure encoding scheme.

Table 2: Comparison of number of multicast messages

Binary tree based key management		N-ary based key management method	
Group size	Number of messages	Group size	Number of messages
8	3	9	2
16	4	27	3
32	5	81	4
64	6	243	5

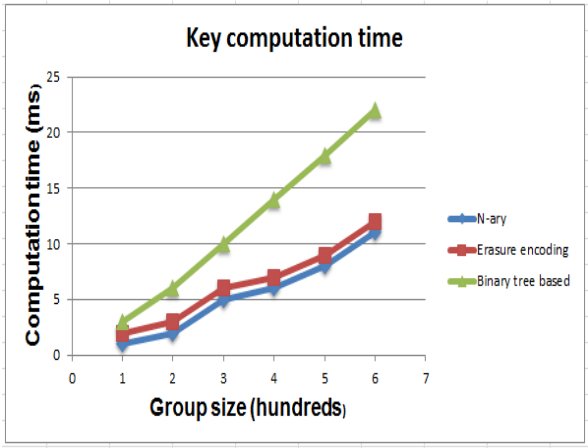

Figure 4: Key computation time of various key distribution schemes at group center

The following evaluation result shows that N-ary tree-based key management scheme needs fewer re-keying messages to recover a sub-group key and session key. When the group size is 243, and when we have a group-leave scenario, then only 5 messages need to be sent in the N-ary method for renewing the sub-group and session key while 8 messages are required in the binary tree scheme. The results in Figure 5 compare N-ary based key distribution scheme with the two approaches. From this figure we observe that when the group size is 600, the time taken in recovering a key is 4ms in the N-ary approach, which is 1 ms better than the erasure encoding scheme and 3 ms better then the binary tree based. Key recovery involves re-keying message exchanges which are explained in detail in [9].

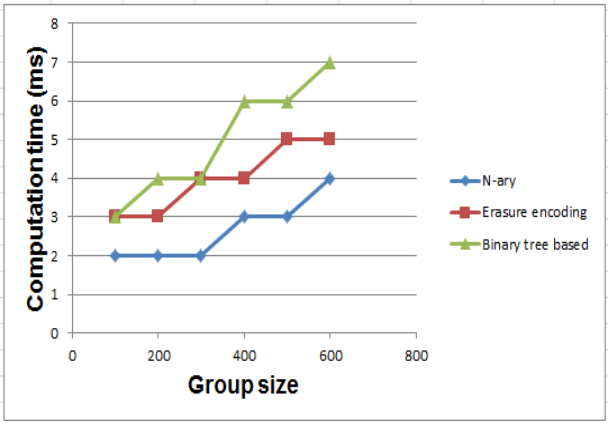

Figure 5 Key generation time of various key distribution schemes at group center

Both the analysis results presented in [9] and in this section highlight the efficacy of the N-ary tree based key management algorithm for the CloNe infrastructure. The analysis results in this paper prove that the N-ary key management scheme is both faster and has a larger key space than either the binary-tree based or the erasure encoding scheme. Therefore, the proposed N-ary key management algorithm could be successfully deployed in the CloNe security architecture, as it is more efficient and secure than binary-tree based and erasure encoding scheme.

5. RESULTS AND COMPARISON

In this paper, an N-ary tree based key distribution protocol was customized and deployed into a simple testbed mimicking the CloNe architecture. The chosen key management algorithm can be deployed both in a centralized or a distributed setting, and provides a larger key space and lower computational complexity than either binary or erasure encoding schemes. The key management algorithm allows virtual machines to efficiently distribute keys within an administrative domain (a single administrative area of an *infrastructure provider*) with few number of message exchanges as compared to the binary tree-based scheme. Moreover, the algorithm can function without an external GC in a distributed setting, thus, authentication and key distribution can occur among the virtual machines even in the absence of an *infrastructure service* component.

The access control policy function and the identity management function, especially its authentication and authorization functionalities, are greatly enhanced by a secure and lightweight key management algorithm [31]. Moreover, due to the low computational requirements of the N-ary algorithm, the computational load is reduced in comparison to the other two algorithms, which results in increased performance of the CloNe (security) architecture. Minimizing the load on these resources is extremely important because the underlying resources are utilized concurrently (potentially by many users), increased load would lead to unpredictable resource utilization (as resources can not be utilized unless the request to be allocated them can be fulfilled). In such a scenario, it is extremely important that the security and management modules should be lightweight and avoid adding undue load on the resource set.

6. CONCLUSION AND FUTURE WORK

In this paper, the customization and deployment of an N-ary tree based centralized key distribution protocol for the CloNe architecture has been proposed and evaluated. This key management algorithm is a collaborative key computation scheme and supports the authentication module of the identity management function of the CloNe security architecture. The results obtained show that the key computation time of the algorithm is reduced considerably when compared with similar key management algorithms that exist in the literature. Moreover, the use of this N-ary based key management algorithm reduces the rekeying cost during member join and leave operations. Finally, this key management algorithm is capable of functioning without an external GC (*infrastructure service* component) in a distributed environment. Future extensions to this work include integration of the key management algorithm and the authentication module with the access control policy function and auditing and assurance function. Additionally, we need to add code so that the underlying hypervisor can authenticate the identity of authorized virtual machines.

7. ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the European Commission for its funding through the “Scalable and Adaptive Internet Solutions”, SAIL Project (FP7-ICT-2009-5-257448). We would like to thank Sathyanarayanan Rangarajan, Rajyadeep Dhunagana, and Ingmar Schoen from Fraunhofer AISEC, Munich for their valuable comments and suggestions regarding the CloNe architecture and its supporting security functions.

8. REFERENCES

[1] Edwall, T. Scalable & Adaptive Internet Solutions (SAIL).Report, 2011.

[2] Murray, P. et al. D-5.2 (D-D.1) Cloud Network architecture Description. Report. 2011.

[3] Schoo, P., Fusenig, V., Souza, V., Melo, M., Murray, P., Debar, H., Medhioub, H., and Zeghlache, D. Challenges for cloud networking security,” in Mobile Networks and Management, ser. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Berlin Heidelberg. 2010.

[4] Fusenig V. and Sharma, A. Security architecture for cloud networking. In *Proceedings of the 2012 International Conference on Networking and Computing, ICNC 2012.IEEE Computer Society*, 2012.

[5] Chengyi Sun, Wanzhen Wang, and Gao X. Z. Scored Pareto-MEC for multi-objective optimization. In *Proceedings of the 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*. 2005. SMCia/05, 2005, pp. 105– 110.

[6] Bjurling, B., Steinert, R., and Gillblad, D. Translation of probabilistic QoS in hierarchical and decentralized settings. *APNOMS 2011*, pp. 1-8.

[7] Abou El Kalam, A., El Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miège, A., Saurel C., and Trouessin G. Organization Based Access Control. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003)*, Lake Come, Italy, June 4-6, 2003.

[9] Vijayakumar, P., Bose, S., Kannan, A., and Siva Subramanian, S. A Secure Key Distribution Protocol for Multicast Communication. *Communications in Computer and Information Science*, Springer, Vol 140, pp. 249–257, 2011.

[10] Somekh-Baruch, A. and Merhav, N. Exact Random Coding Exponents for Erasure Decoding. *Information Theory, IEEE Transactions*, vol.57, no.10, pp.6444-6454, Oct. 2011 doi: 10.1109/TIT.2011.2165826

[18] Li, M., Poovendran, R., and Berenstein, C. Design of secure multicast key management schemes with communication budget constrain. *IEEE communications Letters, 6(2002)*, pp. 108-110.

[19] Poovendran, R. and Baras, J. S. An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes. *IEEE Transactions on Information Theory*. 47(2001), pp. 2824–2834.

[20] Trappe, W., Song, J., Radhapoovendran, and Ray Liu, K. J.

Key management and distribution for secure multimedia multicast. *IEEE transactions on Multimedia*. 5(2003), pp .544-557

[21] Blaum, M., Bruck, J., and Vardy, A. MDS array codes with independent parity symbol. *IEEE Transactions on Information Theory*, 42(1996), pp. 529-542.

[22] Xu, L. and Huang, C. Computation-efficient multicast key distribution. *IEEE Transactions on Parallel and Distributed Systems*.19 (2008), pp .1-10.

[23] Trappe, W., Song, J., Poovendran, R., and Liu, K.J.R. Key distribution for secure multimedia multicasts via data embedding. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Maryland University, College Park, MD ,2001.

[24] Wong, C., Gouda, M., and Lam, S. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*. 8(2000), pp.16-30.

[25] McGrew, D. A., and Sherman, A. T. Key establishment in large dynamic groups using one-way function trees. *Cryptographic Technologies Group, TIS Labs at Network Associates*, (1998), pp.6-12, [Online]. Available: draft-balenson-groupkeymgmt-oft-00.txt.

[26] Shi. H. and He, M. A communication-efficient key agreement Protocol in Ad hoc Networks. *IEEE International Conference on Wireless Networks, Communications and Mobile Computing*,China, 2005.

[27] Seba, H., Tigrine, F., and Kheddouci, H. A tree-based group key agreement scheme for secure multicast increasing efficiency of rekeying in leave operation. *IEEE Symposium on Computers and Communications*, Bourg-en-Bresse, France, 2009.

[28] Ramkumar, M. The subset keys and identity tickets (SKIT) key distribution scheme. *IEEE Transactions on Information Forensics And Security*. (2010), pp.39-51

[29] Bennani, N., Damiani, E., and Cimato, S. Toward Cloud-Based Key Management for Outsourced Databases. In *Proceedings of Computer Software and Applications Conference Workshops (COMPSACW)*, 2010 IEEE 34th Annual, 2010, pp. 232–236.

[30] Lei, S., Zishan, D., and Jindi, G. Research on Key Management Infrastructure in Cloud Computing Environment. In *9th International Conference on Grid and Cooperative Computing (GCC)*, 2010, pp. 404–407.

[31] Kambourakis, G., Konstantinou, E., and Gritzalis, S. Binary tree based public-key management for Mobile Ad Hoc Networks. In *Proceedings of IEEE International Symposium on Wireless Communication Systems*. 2008. ISWCS '08, 2008, pp. 687–692.

[32] Striki, M. and Baras, J.S. Towards integrating key distribution with entity authentication for efficient, scalable and secure group communication in MANETs. *Communications, 2004 IEEE International Conference on* , vol.7, pp. 4377- 4381, 20-24 June 2004 doi: 10.1109/ICC.2004.1313374
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1313374&isnumber=29132

[33] Simmons, G. J. Symmetric and Asymmetric Encryption. *ACM Comput. Survev*. 11, 4 (December 1979), 305-330 doi: 10.1145/356789.356793
URL:http://doi.acm.org/10.1145/356789.356793

[34] Li, D., Li, Y., Wu, J., Su, S., and Yu, J. ESM: Efficient and Scalable Data Center Multicast Routing. *Networking, IEEE/ACM Transactions on* , no.99, pp.1, 0 doi:10.1109/TNET.2011.2169985
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6045301&isnumber=4359146

[35] Li, D., Xu, M., Zhao, M. C., Guo, C., Zhang, Y., and Wu, M.Y. RDCM: Reliable data center multicast. *INFOCOM, 2011 Proceedings IEEE* , pp.56-60, 10-15 April 2011
doi: 10.1109/INFCOM.2011.5935228
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5935228&isnumber=5934870

[36] Li, D., Cui H., Hu, Y., Xia, Y., and Wang, X. Scalable data center multicast using multi-class Bloom Filter. *19th IEEE International Conference on Network Protocols (ICNP), 2011*, pp.266-275, 17-20 Oct. 2011 doi:10.1109/ICNP.2011.6089061
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6089061&isnumber=6089029