Spoilt for Choice: Graph-based Assessment of Key Management Protocols to Share Encrypted Data

Holger Kuehner
Karlsruhe Institute of Technology (KIT)
Steinbuch Centre for Computing (SCC)
Karlsruhe, Germany
holger.kuehner@kit.edu

Hannes Hartenstein
Karlsruhe Institute of Technology (KIT)
Steinbuch Centre for Computing (SCC)
Karlsruhe, Germany
hartenstein@kit.edu

ABSTRACT

Sharing data with client-side encryption requires key management. Selecting an appropriate key management protocol for a given scenario is hard, since the interdependency between scenario parameters and the resource consumption of a protocol is often only known for artificial, simplified scenarios. In this paper, we explore the resource consumption of systems that offer sharing of encrypted data within real-world scenarios, which are typically complex and determined by many parameters. For this purpose, we first collect empirical data that represents real-world scenarios by monitoring large-scale services within our organization. We then use this data to parameterize a resource consumption model that is based on the key graph generated by each key management protocol. The preliminary simulation runs we did so far indicate that this key-graph based model can be used to estimate the resource consumption of real-world systems for sharing encrypted data.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Data sharing*; E.1 [**Data Structures**]: [Graphs and networks]

General Terms

Performance, Security

Keywords

key management protocols, workloads

1. INTRODUCTION

A lot of services nowadays offer data sharing between users as core functionality, e.g., photo sharing communities like Flickr or Picasa, but also sync-and-share services like Dropbox. Typically, the service provider itself is able to get insight into the data that is shared. However, there may be a need for hiding shared data from the service provider, driven by the sensitivity of the data on the one hand, and the compromisability of the service provider on the other hand.

The most obvious approach to hide shared data from the service provider is to encrypt the data before uploading it and to provide each intended reader of the data with the cryptographic keys necessary for decryption. The latter is achieved by key management protocols. Lots of key management protocols have been proposed in the last decades [1], and finding an appropriate protocol for a given data sharing service is hard: The protocols differ in the access control model they support, the level of security they ensure, and implementation details, and all these factors affect the resources a protocol requires in terms of computing power, storage and network traffic. The resource consumption of a protocol further depends on the scenario it is applied to, i.e., the uploading and sharing behavior of the users, and has a major impact on the scalability of the whole system. So far, the interdependency between scenario and resource consumption is only known for simplified scenarios, often comprising only one set of data, where each user is either allowed to read the whole data set or no data at all. However, real-world scenarios are usually far more complex: different users may have access to different data sets, and for ease of access management, users and data may be grouped, and these groups may be hierarchically ordered, allowing to employ inheritance of permissions along these hierarchies. In consequence, a real-world scenario is determined by many parameters, and it is unclear which of these parameters have a major impact on the resource consumption of the key management protocol and, therefore, the system as a whole.

We propose to identify scenario parameters that are relevant for the resource consumption of key management protocols by a simulation approach: we simulate systems for sharing encrypted data and then infer relevant scenario parameters from the simulation results. This methodology, however, brings up some challenges:
i) A model parametrization has to be found which is representative for real-world scenarios. Data describing such scenarios is, to the best of our knowledge, not published. Furthermore, collecting this data is hard since it is individual-related and, therefore, typically protected.
ii) A model that represents systems for sharing encrypted data has to be built that is at the same time general enough to comprise a large set of protocols and detailed enough to yield useful estimations of the resource consumption.

CODASPY'14, March 3–5, 2014, San Antonio, Texas, USA.
ACM 978-1-4503-2278-2/14/03.
http://dx.doi.org/10.1145/2557547.2557583 .

iii) From the large set of parameters that describes a real-world scenario, the parameters have to be determined that are relevant for the resource consumption of the system.

Our contributions in this poster paper are as follows:

1. We monitor large-scale services within our organization to obtain data that represents real-world scenarios (see Section 2). While we use the collected data to parameterize our simulation model, we claim that this data is useful in a far broader context, as it builds a basis for performance assessments of access management strategies in general.

2. We present a simulation model for resource consumption estimation that supports a broad range of key management protocols. Our model facilitates the concept of a "key graph" that was already introduced in literature, but not used for resource consumption estimations before (see Section 3).

In Section 4, we give an outlook on how our simulation model can be leveraged to get a generalized resource consumption model for systems that offer sharing of encrypted data.

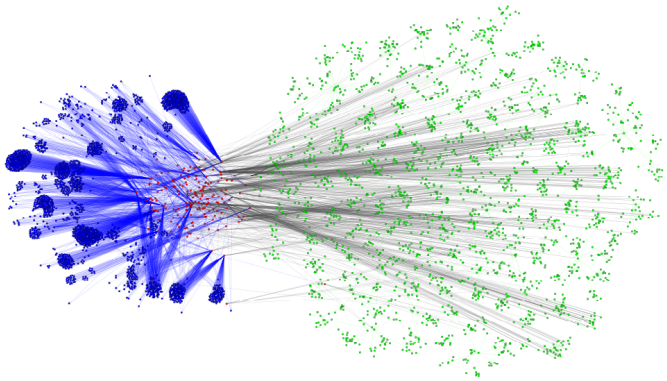

Figure 1: An exemplary scenario representing a part of our e-learning platform. The blue nodes (left) represent users, the red nodes (middle) groups and the green nodes (right) resources. Edges from users to groups indicate group membership, edges from groups to resources indicate access permissions. Users with the same group memberships are clustered. Resources are organized as a tree, with the central node as root.

2. REAL-WORLD SERVICES AND THEIR SHARING DYNAMICS

To be able to realistically parameterize our simulation model, we need information about the aspects of real-world user data sharing behavior that influence the resource consumption of key management protocols. Essentially, each change in the sharing permissions and each data upload operation may cause the key management protocol to create and publish new keys, thereby consuming resources like computing power, storage and network traffic. Therefore, we need information about changes in the permission structure and about upload operations.

It is challenging to collect data concerning the permission structure or upload operations from real-world services: Since this data allows conclusions to be drawn about user behavior and interests, many services do not make it publically available. Therefore, we leverage large-scale collaboration services available within our organization to retrieve pseudonymized usage information. As a starting point, we collect this information from our group collaboration portal based on Microsoft Sharepoint and from our e-learning platform based on ILIAS. We chose these services as both allow the data owner to grant and deny data access, opposed to a system-wide administrator. Furthermore, we wanted to cover services with different expected sharing behaviors, ranging from sharing in a publisher-subscriber fashion within our e-learning platform to a rather homogenous sharing between peers within the group collaboration portal.

By monitoring these services, we get what we call *scenario*. A scenario consists of users and resources, which can be linked either directly or by group memberships. These links may change over time, and the sequence of link changes makes up a *workload*. An access control model based on "Group-Centric Secure Information Sharing" (g-SIS) [3] allows to infer a user's current access permissions from the history of link changes. g-SIS can be parameterized in some aspects, for example, whether a user joining a group is allowed to access resources that were added to the group before. Additionally, our access control model allows to order resources hierarchically to support permission inheritance, i.e., a user allowed to access the parent resource can access all child resources. Finally, a workload specifies which resources were read or written by which user.

The scenarios we collected so far exhibit a complexity and heterogeneity that go far beyond simplified access-all-or-nothing structures: For example, a snapshot of our e-learning platform that implements RBAC includes 35 593 users, 35 666 groups and 143 936 resources. A user is a member of 13 groups on average. Groups are permitted to read two resources and all their child resources on average, ranging from only one up to 477 resources that a group is allowed to read. Figure 1 shows a graph visualization of a part of this snapshot. The complexity and heterogeneity we found make it essential to parameterize our simulation with real-world scenarios.

3. PREDICTION OF RESOURCE CONSUMPTION

To build a simulation model for calculating the resource consumption of key management protocols is challenging as available descriptions of these protocols differ in their abstraction level, ranging from mere descriptions of approaches to completely and formally specified protocols. A model based on the key graph each protocol generates can be applied to a broad range of key management protocols at the one hand, but yields more exact results than existing high-level models on the other hand.

Key graphs have already been introduced in literature [4]. Basically, a key graph is a directed acyclic graph made up of user, key and resource nodes. Edges from users to keys or resources indicate that a user knows or possesses a key or a resource, whereas key-resource edges represent a resource encrypted with a key. Key management protocols typically facilitate a multitude of keys, where a user can use a key she already knows to derive a key she needs. Edges between keys indicate that the edge target key can be derived by everyone knowing the edge source key.

Systems for sharing encrypted data implicitly generate and manipulate key graphs. Changes in the key graph reflect the major resource-consuming actions, e.g., a new edge from

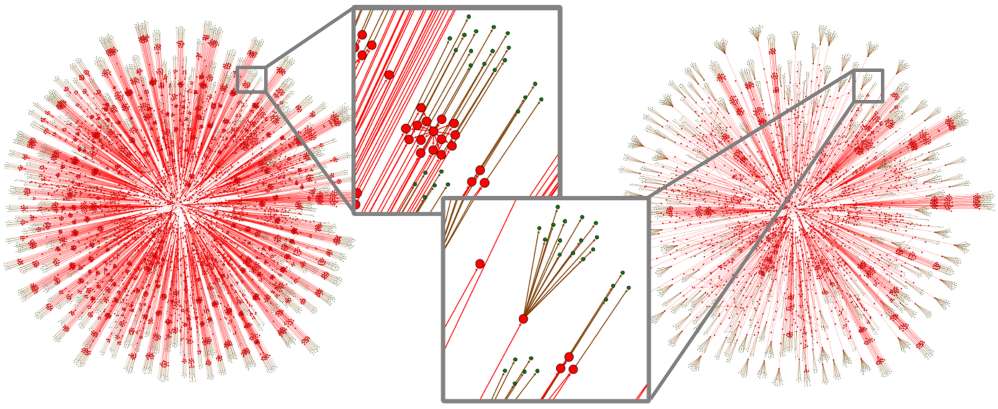

Figure 2: Key graph generated by unmodified (left) and modified (right) Cryptree protocol. Green nodes represent resources and red nodes keys. For the sake of clarity, user nodes are omitted. In Cryptree, the key structure is based on the resource tree (central node as root).

	Key Manager		
	comp (ms)	stor (KB)	net (KB)
unmodified Cryptree	14803	6319	13540
modified Cryptree	5181	2755	4630

Table 1: Resource consumption of key manager as result of simulating unmodified and modified Cryptree. Computing power abbreviated by *comp*, storage by *stor*, and network traffic by *net*.

a user to a resource indicates that the user has downloaded and decrypted the resource, thereby requiring network traffic for the download, computing power for the decryption, and memory for the storage of the resource. We model a system for encrypted sharing as a set of rules that explicitly describe the key graph manipulations carried out in reaction to a certain event. For example, a *UserUploadsResource* event may cause a protocol to insert into the key graph a resource node, an edge from the user to the resource and an edge from the encryption key to the resource. Furthermore, we assign certain network, computing and storage costs to each type of key graph manipulation. The cost are based on preliminary measurements, e.g., adding a key node - which indicates that a key was generated - incurs 200 μs of computing time. By "replaying" a scenario as a sequence of events and tracking the costs caused by the key graph manipulations, we get an estimation for the resource consumption of the system.

To evaluate this approach, we have already modeled the Cryptree protocol described in [2] and ran simulations based on the partial e-learning scenario described in Section 2, which comprises 26 734 read access grants and 2 053 upload events. Cryptree builds on a hierarchically organized file system; a dedicated key manager assigns a set of keys to each file and folder. We also implemented a modification to Cryptree, which generates a key set for a file only if there are dedicated access permissions assigned to this file. Since in the e-learning scenario, access permissions are rather granted on folders than on files, we expect our modification to decrease the number of required keys and, therefore, the resource consumption for the key manager. The simulation of both unmodified and modified Cryptree yields results that met our expectations: The visualization of the key graphs (cf. Fig. 2) shows that unmodified Cryptree generates more file keys, visible as red key nodes on the border of the visualized graph. Consequently, the simulation shows considerable differences in the resource consumption (cf. Table 1) for the key manager, while the differences in the resource consumption for the users and the storage provider are negligible. Note that our test scenario does not include access revocations; if, e.g., all access permissions of 10% of the users would be revoked, about 41 GB of data had to be re-encrypted to make sure these users lose access immediately after revocation. In general, we expect the resource consumption to be significantly higher with key management protocols that re-encrypt data immediately after an access revocation.

4. NEXT STEPS

We will continue collecting empirical data that represents real-world scenarios. Since we see a great potential in this data as a basis for assessing access management strategies in general, we encourage the community to collect, anonymize and contribute such data. Furthermore, we will model and simulate more key management protocols, based on the different scenarios we collected so far. In addition, we will validate our simulation model against measurements taken from a system that implements sharing of encrypted data. Thereafter, we plan to analyze real-world scenario parameters and simulation results to derive a generalized resource consumption model for systems that offer sharing of encrypted data.

5. REFERENCES

[1] Y. Challal. Group Key Management Protocols: A Novel Taxonomy. *International Journal of Information Technology*, 2(2):105–118, 2005.

[2] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer. Cryptree: A Folder Tree Structure for Cryptographic File Systems. In *25th IEEE Symposium on Reliable Distributed Systems - SRDS '06*, pages 189–198, 2006.

[3] R. Krishnan, R. Sandhu, J. Niu, and W. H. Winsborough. Foundations For Group-Centric Secure Information Sharing Models. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies - SACMAT '09*, page 115, 2009.

[4] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. *ACM SIGCOMM Computer Communication Review*, 28(4):68–79, Oct. 1998.