Performance of End-to-End Secure Data Sharing

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Holger Kühner

aus Speyer

Tag der mündlichen Prüfung:	08. Februar 2017
Erster Gutachter:	Prof. Dr. rer.nat. Hannes Hartenstein Karlsruher Institut für Technologie (KIT)
Zweiter Gutachter:	Prof. Dr. rer.nat. Thomas Walter Universität Tübingen

Zusammenfassung

Das Teilen von Daten bildet die Grundlage für nahezu jede IT-gestützte Zusammenarbeit im geschäftlichen und privaten Kontext. Typische Realisierungen der Autorisierung und der Durchsetzung von Zugriffsrechten auf den gemeinsam genutzten Daten erfordern, dass die Benutzer, die Daten miteinander teilen, hinsichtlich der Vertraulichkeit und Integrität der Daten auf Dritte vertrauen müssen. Solche Realisierungen bergen also im speziellen das Risiko, dass Daten durch einen Insider-Angriff auf den vertrauenswürdigen Dritten gefährdet werden. Mit Hilfe clientseitig ausgeführter kryptographischer Operationen können die Autorisierung und die Durchsetzung von Zugriffsrechten für beliebige Speicherdienste in weiten Teilen von den Benutzern selbst durchgeführt werden, was in einem Ende-zu-Ende-gesicherten System zum Teilen von Daten (*End-to-End Secure Data Sharing, E2E-SDS*) resultiert. E2E-SDS-Systeme werden jedoch nur dann von potenziellen Anwendern akzeptiert, wenn die ihnen bekannten Autorisierungsprozesse weitgehend unverändert bleiben, und die Leistungseinbußen für die Autorisierung und den Datenzugriff nicht zu gravierend sind.

Das Hauptziel dieser Arbeit ist die Bewertung der Leistungseinbußen, die auf einem Benutzer-Client mit einem gegebenen E2E-SDS-Protokoll in der Realität zu erwarten sind. Für bestehende E2E-SDS-Protokolle ist das asymptotische Verhalten in Bezug auf Leistungsmetriken wie Rechenzeit oder Netzwerkverkehr in der Regel bekannt. Das asymptotische Verhalten lässt jedoch nur schwache Schlussfolgerungen auf die absolute Höhe der Leistungseinbußen zu. Neben dem E2E-SDS-Protokoll selbst hängt die reale Leistung von der eingesetzten Hardware, den Sicherheitsparametern und dem konkreten Freigabe- und Nutzungsszenario ab, also vom Freigabe- und Nutzungsverhalten der Benutzer im System. Die Bewertung der realen Leistungseinbußen bringt im wesentlichen zwei Herausforderungen mit sich: Erstens muss das zu untersuchende E2E-SDS-Protokoll unter Einbeziehung der vorgenannten Einflussfaktoren auf die Leistung modelliert werden, wobei Implementierungsdetails nach Möglichkeit im Modell einfach austauschbar sind. Zweitens müssen realistische Freigabe- und Nutzungsszenarien vorliegen, die entweder auf Beobachtungen basieren, oder mit Hilfe von Schätzungen generiert werden.

Das Ziel dieser Arbeit ist die detaillierte Bewertung der realen Leistung von E2E-SDS-Protokollen. Der Fokus der Arbeit liegt auf E2E-SDS-Protokollen, die ein gruppenbasiertes Autorisationsmodell realisieren, die es also ermöglichen, Daten mit benannten Benutzergruppen zu teilen, die von beliebigen Benutzern verwaltet werden. Diese Funktion wird von weit verbreiteten verteilten Dateisystemen wie NFSv4 oder

CIFS angeboten.

In dieser Arbeit werden Methoden zur Bewertung der realen Leistung von E2E-SDS-Protokollen vorgestellt. Aus der Beobachtung realer Speicherdienste gewonnene Freigabe- und Nutzungsszenarien werden charakterisiert und eine Methode zur Erzeugung synthetischer Freigabe- und Nutzungsszenarien eingeführt. Unter Nutzung dieses Instrumentariums wird die Leistungsfähigkeit sowohl bestehender als auch neuartiger E2E-SDS-Protokolle evaluiert und mögliche Maßnahmen zur Verbesserung der Leistung auf Seiten des Anwenders vorgeschlagen.

Um realistische Freigabe- und Nutzungsszenarien zu erhalten, wurden die Mitglieder, Aktivitäten und Berechtigungen von Benutzergruppen auf zwei produktiven Speicherdiensten beobachtet. Die daraus resultierenden Szenarien werden hinsichtlich ausgewählter Parameter charakterisiert.

Für die Leistungsbewertung von E2E-SDS-Protokollen in realistischen Szenarien wurden zwei Methoden entwickelt: Die analytische Methode liefert in vielen Fällen hinreichend genaue Ergebnisse. Die simulative Methode ist erforderlich, wenn die Leistung komplexer E2E-SDS-Protokolle detailliert analysiert werden soll. Für die simulative Methode wird ein Simulationsmodell vorgestellt, das einen Vergleich von E2E-SDS-Protokollen auf einer einheitlichen Abstraktionsebene ermöglicht.

Um die Performance von E2E-SDS-Protokollen auch dann bewerten zu können, wenn keine aus Beobachtungen resultierende Freigabe- und Nutzungsszenarien vorliegen, werden synthetische Szenarien erzeugt, die auf Schätzungen bestimmter Parameter des Szenarios basieren. Dazu wird ein Erzeugungsverfahren vorgestellt, das Abhängigkeiten zwischen den vorab spezifizierten Parametern berücksichtigt. Die NP-Schwere des zugrundeliegenden Problems der Erzeugung von Szenarien wird für bestimmte Kombinationen von vorab spezifizierten Parametern bewiesen.

Die vorgestellten Methoden zur Leistungsbewertung werden einerseits auf E2E-SDS-Protokolle angewandt, die auf traditioneller Kryptographie basieren, die also mittels symmetrischer und asymmetrischer Kryptographie Chiffrate erzeugen, die nur mit einem einzigen Schlüssel dechiffriert werden können. Andererseits werden die vorgestellten Methoden auf E2E-SDS-Protokolle angewandt, die auf Attributbasierter Verschlüsselung (*Attribute-Based Encryption, ABE*) basieren, mit deren Hilfe eine Gruppe von Benutzern mit nur einem einzigen Chiffrat adressiert werden kann.

Die Leistungsbewertung des traditionellen E2E-SDS-Protokolls zeigt, dass in den betrachteten Nutzungs- und Nutzungsszenarien für die meisten Autorisierungsoperationen nur geringe Leistungseinbußen zu erwarten sind. Beträchtliche Leistungseinbußen sind für Benutzer zu erwarten, die Gruppenmitgliedschaften in großen benannten Benutzergruppen verwalten, d.h. Benutzergruppen mit einigen tausend oder mehr Mitgliedern. Diese Leistungseinbußen können durch die Integration eines Group Key Management-Ansatzes deutlich gesenkt werden, also eines Ansatzes, der auf eine effiziente Verteilung und Erneuerung von kryptographischen Schlüsseln innerhalb von Benutzergruppen abzielt.

Ein auf ABE basierendes E2E-SDS-Protokoll wird realisiert, indem bestehende ABE-Verfahren hinsichtlich ihrer Eignung für E2E-SDS evaluiert, und das attributbasierte Autorisationsmodell eines geeigneten ABE-Verfahrens auf das gruppenbasierte Autorisationsmodell abgebildet wird. Eine Leistungsbewertung verschiedener Varianten dieser Abbildung zeigt, dass das ABE-basierte Protokoll eine etwas schlechtere Leistung als das auf traditioneller Kryptographie beruhende Protokoll bietet.

Schließlich wird ein neuartiges E2E-SDS-Protokoll vorgestellt, das auf kooperative Autorisierungsoperationen verzichtet. Diese Operationen erfordern, dass die Endgeräte der Benutzer zu jedem Zeitpunkt erreichbar und bereit für die Ausführung rechenintensiver kryptographischer Operationen sind. Diese Anforderungen sind insbesondere beim Einsatz mobiler Endgeräte nicht immer sichergestellt. Ein wesentlicher Vorteil des vorgeschlagenen Protokolls liegt darin, dass es den praktischen Einsatz von Hierarchien benannter Benutzergruppen in E2E-SDS ermöglicht. Die damit verbundenen, potenziell hohen Leistungseinbußen werden detailliert ausgewertet. Weiterhin wird gezeigt, dass die Unterstützung von Gruppenhierarchien ohne kooperative Autorisierungsoperationen grundsätzlich gewisse Einschränkungen hinsichtlich der Aktualität der Zugriffsberechtigungen impliziert, was die Grenzen der Anwendbarkeit von E2E-SDS aufzeigt.

Abstract

Data sharing forms the basis for almost any IT-based collaboration in both business and personal contexts. Typical realizations of authorization and access control enforcement on the shared data require the group of sharing users to trust third parties to protect data confidentiality and integrity. Such realizations of data sharing bear the specific risk of data being compromised by an insider attack on the trusted third party. By means of client-side cryptography, major parts of the authorization and access control enforcement for arbitrary storage services can be carried out by the sharing users themselves, resulting in an End-to-End Secure Data Sharing (E2E-SDS) system. However, such systems will only be accepted by potential users if the authorization processes they are familiar with remain largely unaltered, and if the performance penalties for authorization or data access are not too heavy.

The main objective of this thesis is to evaluate the real-world performance penalties that have to be expected on a user client with a given E2E-SDS protocol. For existing E2E-SDS protocols, the asymptotical behavior is usually known with regard to performance metrics such as computation time or network traffic volume, but this does not offer much insight into how big the performance penalties are in absolute terms. Besides the E2E-SDS protocol itself, the real-world performance depends on the employed hardware, the security parameters, and the concrete sharing and usage scenario, i.e., the sharing and usage behavior of the users in the system. Thus, the challenges of a real-world performance evaluation of E2E-SDS protocols are twofold: First, the protocol under study has to be modeled with regard to the aforementioned performance influencing factors while factoring out implementation details. Second, realistic sharing and usage scenarios have to be observed, or generated based on partial estimated scenarios.

The objective of this work is to evaluate the real-world performance of E2E-SDS protocols in depth. The focus is on E2E-SDS protocols that realize a group-based authorization model, i.e., that allow to share data with named user groups that are managed by arbitrary users. This feature is supported by the authorization model of widely deployed distributed file systems such as NFSv4 or CIFS.

In this work, methods for the evaluation of the real-world performance of E2E-SDS protocols are presented. Realistic usage scenarios taken from real-world storage services are characterized, and a method for the generation of synthetic sharing and usage scenarios is introduced. Based on these instruments, the performance of both existing and novel E2E-SDS protocols is evaluated, and possible measures to improve the performance on the user's side are proposed.

To get realistic sharing and usage scenarios, the members, activities and permissions of user groups were observed on two real-world storage services. The resulting scenarios are characterized with regard to selected parameters.

For the performance evaluation of E2E-SDS protocols in real-world scenarios, two methods were developed: The analytical method yields results that are sufficiently accurate in many cases. The simulative method is required when the performance of a certain operation is to be analyzed in more detail while studying more complex E2E-SDS protocols. For the simulative method, a simulation model is presented that enables a comparison of E2E-SDS protocols on a unified layer of abstraction.

To be able to evaluate the performance of E2E-SDS protocols when no observed sharing and usage scenarios are available, synthetic scenarios are generated that adhere to estimations of certain parameters of the scenario. For this purpose, a generation method is presented that takes dependencies between predetermined parameters into account. The NP-hardness of the scenario generation problem is proven for certain combinations of predetermined parameters.

The presented performance evaluation methods are applied to E2E-SDS protocols based on *traditional cryptography*, i.e., symmetric and asymmetric cryptography that produces ciphertexts dedicated to a single recipient only, and based on *Attribute-Based Encryption (ABE)*, which enables to address a whole group of users with just a single ciphertext.

The performance evaluation of the traditional E2E-SDS protocol shows that in the considered sharing and usage scenarios, only minor performance penalties have to be expected for most of the authorization operations. Major performance penalties hit users that manage group memberships in large named user groups, i.e., user groups with a few thousand or more members. These performance penalties can be decreased significantly by integrating a *Group Key Management* approach, which aims at an efficient distribution and renewal of cryptographic keys within user groups.

An E2E-SDS protocol that leverages ABE was realized by evaluating existing ABE schemes with regard to E2E-SDS properties, and mapping the attribute-based authorization model of a suitable ABE scheme to the group-based authorization model. A performance evaluation of different mapping variants shows that the ABE-based protocol offers a slightly worse performance than the protocol based on traditional cryptography.

Finally, a novel E2E-SDS protocol is presented that omits joint authorization operations. Such operations require user devices to be reachable and ready to carry out computationally intensive cryptography at arbitrary points in time, which might be problematic especially when mobile devices are used. A major benefit of the proposed protocol is that it enables the employment of hierarchies of named user groups in E2E-SDS. The potentially heavy performance penalties that come with these hierarchies are evaluated in detail. Furthermore, it is shown that the support of group hierarchies without joint authorization operations fundamentally implies certain limitations regarding the freshness of access permissions, which indicates the limits of the applicability of E2E-SDS.

Contents

Zusammenfassung i

Abstract v

List of Figures xi

List of Tables xiii

1 Introduction 1

1.1 Challenges and Objectives 3

1.2 Contributions 5

1.3 Thesis Outline 8

2 Fundamentals and Related Work 9

2.1 Authorization Models for Data Sharing 10

2.2 SDS Protocols Using Traditional Cryptography 15

2.2.1 Cryptographic Primitives 17

2.2.2 SDS Protocols with Focus on Confidentiality 19

2.2.3 Named User Groups and Joint Authorization Operations . . 24

2.2.4 Applications of More Enhanced Cryptography 26

2.3 SDS Protocols Using Attribute-Based Encryption 27

2.3.1 Mathematical Background 27

2.3.2 Evolution of Attribute-Based Encryption Schemes 29

2.4 Performance Evaluation of SDS Protocols 31

2.4.1 Performance Evaluation Methods 31

2.4.2 Sharing Models and Workloads 33

2.4.3 Workload Generation 35

2.5 Conclusions 37

3 SDS Workload Modeling and Generation 41

3.1 Authorization Model 42

3.2 Definition of Workloads 43

3.3 Generation of Real-World Workloads from Traces 45

3.4 Workload Parameterization 48

3.4.1 Static Workload Parameters 49

3.4.2 Dynamic Workload Parameters . . . 51
3.4.3 Workload Parameter Dependency Groups . . . 51
3.5 Characterization of Real-World Workloads . . . 53
3.6 Generation of Synthetic Workloads . . . 59
3.6.1 Workload Generation Method . . . 59
3.6.2 NP-hardness of Workload Generation . . . 61
3.7 Conclusions and Outlook . . . 69

4 Methods for SDS Performance Evaluation 71
4.1 Analytical Method . . . 72
4.1.1 Analyzing the Performance of Single Workload Events . . . 72
4.1.2 Analyzing the Performance of Complete Workloads . . . 74
4.2 Simulative Method . . . 76
4.2.1 Key Graph . . . 76
4.2.2 Simulative Performance Evaluation Method . . . 80
4.2.3 Limitations and Relation to Analytical Method . . . 81
4.3 Conclusions and Outlook . . . 83

5 E2E-SDS Using Joint Authorization Operations 85
5.1 Requirements for E2E-SDS Supporting Named User Groups . . . 86
5.1.1 System Entities and Authorization Model . . . 86
5.1.2 Expressiveness of Policies . . . 87
5.1.3 Trust and Authorization Process . . . 87
5.1.4 Revocation . . . 88
5.2 Attacks on E2E-SDS Protocols . . . 88
5.3 Performance of E2E-SDS Using Traditional Cryptography . . . 93
5.3.1 Protocols Under Study . . . 94
5.3.2 Performance Evaluation Results for Real-World Sharing Scenarios . . . 96
5.4 E2E-SDS Using Attribute-Based Encryption . . . 100
5.4.1 Evaluation of Existing ABE Schemes . . . 101
5.4.2 DAC-MACS Entities and Operations . . . 103
5.4.3 Mapping Alternatives . . . 105
5.4.4 Mapping Details . . . 107
5.5 Performance of E2E-SDS Using Attribute-Based Encryption . . . 109
5.5.1 DAC-MACS Benchmark Results . . . 110
5.5.2 Performance Evaluation Results for Real-World Sharing Scenarios . . . 110
5.5.3 Performance Comparison with Traditional Cryptography-Based E2E-SDS Protocol . . . 113
5.6 Conclusions and Outlook . . . 114

6 E2E-SDS Without Joint Authorization Operations 117
6.1 Challenges and Problem Statement 118
6.2 Protocol Overview and Design Decisions 120
6.2.1 Administrative Key Graph: Basic Idea 120
6.2.2 Application of Administrative Key Graph 122
6.2.3 Freshness of Validity State and Key Signatures 124
6.2.4 Implementation Discussion 126
6.3 Protocol Details 127
6.4 Performance Evaluation Results 135
6.4.1 Performance Evaluation Results for Real-World Sharing Scenarios 135
6.4.2 Sensitivity Analysis 138
6.5 Conclusions and Outlook 141

7 Summary, Conclusions and Outlook 143

Bibliography 149

List of Figures

2.1 Entities and their relations in the Role-Based Access Control (RBAC) model . . . 11
2.2 Exemplary sequences of g-SIS actions . . . 13
2.3 Visualization of a key graph . . . 20
2.4 Keys and lockboxes generated by the SiRiUS protocol after upload of an encrypted data file . . . 22
2.5 Keys and lockboxes generated by the SiRiUS protocol after sharing an encrypted data file . . . 23

3.1 Generating real-world workloads from traces of real-world systems 47
3.2 Frequency of named user groups with respect to the group size . . . 54
3.3 Proportion of users with respect to the number of memberships in named user groups . . . 54
3.4 Frequency of write events with respect to the size of data written . . 55
3.5 Frequency of users with respect to the number of accessible files . . 57
3.6 Frequency of workloads with respect to the size of all contained files 57
3.7 Visualization of analyzed workload generation problems . . . 64

4.1 Visualization of an exemplary key graph . . . 79
4.2 Manipulation of key graph by transaction *AddEncryptedResource* . . 80

5.1 Exemplary key graph for basic protocol . . . 94
5.2 Exemplary key graph for extended protocol . . . 95
5.3 Average computation time of a group member removal operation with basic protocol . . . 98
5.4 Average computation time of a group member removal operation with extended protocol . . . 99
5.5 Average outgoing network traffic volume of a group member removal operation with extended protocol . . . 100
5.6 System model of ABE scheme *DAC-MACS* . . . 104
5.7 Overview of alternatives for mapping *DAC-MACS* to a group-based authorization model . . . 106
5.8 Average key management computation time for different operations with ABE-based E2E-SDS protocol . . . 112
5.9 Comparison of computation time for group member removal operations with ABE-based and traditional E2E-SDS protocol . . . 113

6.1 Relation between administrative and base key graph after generation of resource and group keys 122
6.2 Key graph after group access is granted 123
6.3 Key graph after a user was removed from group g_1 124
6.4 Sensitivity of the average computation time for group member removal operations 139
6.5 Sensitivity of the average computation time for removing a group from another group 140

List of Tables

3.1 List of static workload parameters 49
3.2 List of dynamic workload parameters 50
3.3 List of workload parameter dependency groups 52
3.4 Correlation of groupware workload parameters 58
3.5 Correlation of e-learning workload parameters 58
3.6 List of analyzed workload parameter combinations 63

4.1 List of allowed key graph edge types and their semantics 78

5.1 Overview of attacks on confidentiality and integrity of shared data in E2E-SDS 89
5.2 Cryptographic operation benchmarks for different devices 97
5.3 Average key management computation time for write operations with extended protocol 97
5.4 Evaluation of ABE schemes supporting multiple Attribute Authorities 102
5.5 Benchmarks of *DAC-MACS* operation running times 111

6.1 Naming conventions for cryptographic data structures used in the E2E-SDS protocol 128
6.2 Performance of authorization operation types with constant computation times and network traffic 136

1
Introduction

Data sharing forms the basis for almost any IT-based collaboration in both business and personal contexts. It comprises personal use cases like sharing pictures with Facebook friends, but also institutional use cases like working on a document in a closed user group. In many cases, data is shared over a *group-external* storage provider, i.e., a storage provider that is not maintained by the user group itself, as this usually brings advantages in terms of usability, availability, access speed or amount of available storage. Examples for the use of an external storage provider are the user group that leverages a commercial photo gallery service to share pictures, as well as the organization department that manages documents on the organization's central groupware server or even on an outsourced groupware service.

Typical realizations of authorization and access control enforcement on the shared data require the group of sharing users to trust third parties to protect data confidentiality and integrity. Such realizations of data sharing bear the specific risk of data being compromised by an insider attack on the trusted third party, but it also renders the trusted third party a potentially valuable target for external attackers. In consequence, a group of sharing users has to balance the advantages that a group-external storage provider offers against the confidentiality and integrity of the shared data, and they may decide to avoid using a group-external storage provider if they evaluate the risk of data corruption as too high, which might be true when the data is sensitive and the trust in the group-external storage provider is low.

By means of client-side cryptography, major parts of the authorization and access control enforcement for arbitrary storage services can be carried out by the sharing users themselves, which is denoted as *End-to-End Secure Data Sharing (E2E-SDS)* in this thesis. Therefore, in an E2E-SDS system, no party outside the group of sharing users has to be trusted with regard to the confidentiality and integrity of the shared data. For this purpose, the data has to be encrypted and digitally signed by the users before it is uploaded to the storage provider. The encryption ensures that the storage provider is not able to inspect the shared data, and the use of digital signatures allows at least to detect unauthorized modifications of the data. In summary, E2E-SDS leverages client-side cryptography to protect the confidentiality and integrity of the shared data without the need for a trusted third party.

The challenges arising in the design of E2E-SDS protocols are addressed by the research community for almost two decades. Early protocols, such as Cepheus [Fu99], SiRiUS [GSMB03] or Cryptree [GMSW06] are based on "traditional" cryptographic primitives, i.e., symmetric and asymmetric encryption. Since then, the set of available E2E-SDS protocols has been continuously growing, striving for more secure protocols that support more expressive access control models. This goal was approached, e.g., by employing more recent cryptographic primitives such as Attribute-Based Encryption [SW05], which allows to dedicate a single ciphertext to multiple recipients. However, despite a broad attention in research, E2E-SDS systems still do not experience wide-spread adoption in reality: For example, a survey from 2016 showed that while about 70% of the participating large enterprises are concerned about the security of data outsourced to cloud service providers, only 24% of the enterprises encrypt the data before outsourcing[1].

One barrier that hinders the adoption of E2E-SDS systems in practice is the loss in comfort that these systems usually bring about. For example, the initial key exchange that is required by every E2E-SDS system may be cumbersome, and facilities for key exchange that are both usable and secure are still subject to research (e.g., [KW14]). As another example, losing a cryptographic key may lead to a loss of data, so the keys should be stored redundantly without compromising their confidentiality. There are plenty of aspects that are crucial to users' acceptance of E2E-SDS. This thesis focuses on two of these aspects: the authorization models the users are familiar with have to remain largely unaltered, and the performance penalties for authorization or data access should not be too heavy.

The authorization models for data sharing strongly differ, dependent on the concrete data sharing system and scenario. The diversity of authorization models does not only originate from the diverse nature and purpose of the shared data, but also from the differences in users' expertise. As diverse as the authorization models are, a feature that many of them have in common is that they offer facilities to share data not only with single users, but with groups of users. This feature is exhibited by traditional distributed file systems, such as NFSv4 or CIFS, but was also retained in more recent data sharing systems. For example, user groups are supported in some Sync&Share services, such as Dropbox, or in some online social networks, such as Facebook. The essential property of these user groups is that they are not existing only in the context of permissions on some data, but are independent entities that can be granted data access via a group handle. Thus, these user groups are denoted as *named user groups* in this thesis. This thesis focuses on E2E-SDS systems that implement an authorization model with support for named user groups.

[1]`https://www.vormetric.com/campaigns/datathreat/2016/pdf/cloud-big-data-iot-2016-vormetric-dtr-deck-v2.pdf`

The support for named groups bears some special challenges in combination with E2E-SDS. These challenges lie in the required collaboration between the *group owner*, i.e., the user managing the members of a group, and the *resource owner*, who manages permissions for her resources. For some sharing operations, these two entities have to carry out cryptographic operations in a concerted way. For this purpose, some existing E2E-SDS protocols employ *joint authorization operations*, which require that the involved users are reachable in the network and able to allocate the necessary computational resources immediately upon request. This severely impacts the usability of E2E-SDS when devices with low computational power and unstable network connections are used. Alternatively, group and resource owners can exchange information over an untrusted storage provider if the confidentiality, integrity and especially the freshness of the exchanged information are cryptographically ensured. However, this poses a fundamental challenge when no trusted third party is available. The challenge of collaboration between group and resource owners in an E2E-SDS system is addressed in this thesis.

For the performance of E2E-SDS, it is obvious that client-side cryptography implies performance penalties, e.g., in terms of computing time and network traffic volume: Besides the computation time that is needed to encrypt and decrypt the data to be shared, the system has to support several key management tasks that make up the key lifecycle, starting from the creation of data encryption keys, to their distribution to the group of legitimate accessors, to the invalidation and replacement of a key whenever a user leaves the group of legitimate accessors. The impact on the performance depends on the role a user acts in, i.e., does a user only up- and download encrypted data, or does she also manage named user groups, which involves key management tasks.

It is an open issue whether the performance overhead that is induced by the client-side cryptography employed by E2E-SDS is significant in reality. On the one hand, the bandwidth of wired and wireless networks and the computing power of mobile and desktop processors grow—many new mobile processors even support encryption in hardware[2]. On the other hand, the size of "cloud data" that can potentially be shared also grows[3], and the recommended key length—and therewith the computing time required—increase for symmetric and asymmetric ciphers[4].

1.1 Challenges and Objectives

The objective of this thesis is to evaluate the *real-world performance* of E2E-SDS protocols. Besides the concrete cryptographic primitives that are required for the different protocol operations, the real-world performance also depends on the concrete implementation of these primitives, and the hardware the primitives are carried out

[2]`https://www.apple.com/business/docs/iOS_Security_Guide.pdf`

[3]`http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html`

[4]`http://www.keylength.com/en/4/`

on. Another major influence factor for the protocol performance is the *sharing model* that underlies the E2E-SDS system. The sharing model formalizes the sharing behavior of the users, i.e., it describes the number and size of the named user groups, the permissions of these groups and so on. All of these factors have to be considered when designing a performance evaluation method that reflects the real-world performance.

Evaluating the real-world performance of E2E-SDS protocols is substantially different from the performance evaluations that are usually carried out for these protocols in literature, as these evaluations focus on the *asymptotical behavior* of the performance. Asymptotical analyses are a good starting point to identify factors that are relevant for the performance, but they do not offer much insight into how big the performance penalties of E2E-SDS are in absolute terms. This is due to the very high abstraction level that asymptotical analyses strive to achieve. In contrast, measurements of the performance of a given protocol implementation on a given hardware with a given sharing model yield precise results in absolute terms. On the other hand, measurements obviously lack generality, as they only provide results for one combination of protocol implementation, hardware and sharing model. Thus, the challenge addressed in this thesis is to find methods for performance evaluation that allow to estimate the performance in absolute terms, but also allow to make statements about the performance on a more abstract level. The transition from asymptotical analyses to methods that estimate the performance in absolute terms bears two major challenges, which are discussed in the following.

The first challenge is that an asymptotical analysis abstracts from constant factors and constant terms, which may be highly relevant for the performance in reality. With asymptotical analyses, the performance of secure data sharing operations of different types, e.g., uploading data or adding a member to a sharing group, is expressed in terms of asymptotical behavior of some performance metric in relation to some input parameter N. Especially when N is small, and a single operation is expensive, these constants might be the dominant influence factor for the performance. Thus, a performance evaluation method is required that is based on a more fine-grained representation of the E2E-SDS protocol, which also models the constant factors and terms. At the same time, as the objective of this work is to evaluate and compare the performance on an E2E-SDS protocol level, the method should abstract from implementation details. For this purpose, an abstraction level has to be identified that enables to model the relevant differences between the protocols, while yielding results that are accurate enough for practical purposes.

As second challenge, the knowledge about the sharing and usage model that underlies the E2E-SDS system is usually incomplete. Thus, it is unclear which values the parameter N typically takes in reality. While it might be possible that certain aspects of the sharing or usage model, i.e., certain *sharing model parameters*, can be approximated roughly, others may be much harder to estimate, even by a domain expert. For example, there may be some rough idea of the range of sharing group sizes or the overall size of data shared in the sharing model. However, other sharing model parameters that may be relevant for the performance are harder to determine, e.g., the average number of files accessible per sharing group. One possibility to determine realistic values for sharing model parameters is to observe real-world E2E-SDS systems. While the observation might only provide a small and non-representative set of realistic parameter values, these values can nonetheless be used to get a rough indication of the real-world performance of a given protocol. However, observation requires that the E2E-SDS system under study is already in place. If the E2E-SDS system doesn't exist, or the observed sharing or usage model is expected to change in the future, a performance evaluation based solely on observed parameter values will not be sufficient.

If the observation of realistic values for the sharing and usage model parameters is impossible, it can be tried to approximate the values of required, but unknown sharing model parameters based on estimable sharing model parameters. In this context, estimable sharing model parameters refer to parameters for which a domain expert could provide an estimation regarding the E2E-SDS system under study. However, to the best of our knowledge, the dependencies between different sharing model parameters are not discussed systematically in literature.

In summary, the challenges of evaluating the real-world performance of E2E-SDS protocols that were discussed in this section lead to the following research question, which will be addressed in this thesis:

How can the real-world performance of E2E-SDS protocols be evaluated, both based on observable and on estimated sharing models?

As stated before, the motivation for identifying or constructing such performance evaluation methods is to provide concrete and absolute numbers for the performance penalties of E2E-SDS, with a focus on E2E-SDS protocols that support named user groups. While the application of the performance evaluation methods obviously has to be limited to some chosen E2E-SDS protocols and sharing models, it can nonetheless provide a tendency whether the performance penalties hinder the adoption of E2E-SDS in practice. Therefore, the following research question will be addressed in this thesis:

Which performance penalties have to be expected in reality using E2E-SDS protocols with support for named user groups?

1.2 Contributions

The contributions of this thesis can be classified into *methodic contributions*, comprising sharing models and methods that are required for the evaluation of the real-world performance, and *applied contributions*, where the developed methods are applied to evaluate the performance of chosen E2E-SDS protocols.

The main methodic contributions of this thesis are:

– **Tracing and characterization of real-world sharing model instances:** To get realistic sharing model instances, the members, activities and permissions of user groups are tracked on real-world storage services. For this purpose, sharing operations are traced in e-learning and groupware services running in production with a user base of around 40 000 users, and these traces are transformed into a set of about 100 replayable *workloads*. The sharing model instances are characterized with regard to selected parameters. To the best of our knowledge, no such characterization of sharing model instances has been published before.

- **Synthetic sharing model generation method:** To be able to carry out performance evaluations based on an estimated sharing model instances, the sharing model parameters that influence the performance have to be derived from a set of sharing model parameters that can be estimated by a domain expert for the system under study. The derivation of sharing model parameters can be implicitly achieved by generating synthetic sharing model instances that adhere to predetermined parameter assignments. For this purpose, a generation method is presented, which takes into account that predetermining dependent parameters might lead to conflicting requirements on the resulting sharing model instance. Furthermore, it is proven that the generation of sharing model instances is an NP-hard problem when certain combinations of parameters are predetermined.

- **Simulative performance evaluation method:** For the performance evaluation of E2E-SDS systems based on realistic sharing models, two methods are presented. The analytical method yields results that are sufficiently accurate in many cases. However, the method can hardly be applied if the performance of a certain sharing operation depends on more than one sharing model parameter. To cope with this issue, a simulative performance evaluation method was developed. The simulative method specifies a uniform abstraction layer for all modeled E2E-SDS protocols. It supports all cryptographic primitives that are common in E2E-SDS protocols today, i.e., encryption with symmetric and asymmetric ciphers and digital signatures. The method was implemented by building a simulator.

The sharing models and methods just introduced are applied in this thesis in the following ways:

- **Performance evaluation of E2E-SDS protocols based on traditional cryptography:** The real-world performance of a set of E2E-SDS protocols that leverage only traditional cryptography is evaluated in depth. In the context of this thesis, the term *traditional cryptography* refers to symmetric and asymmetric cryptography that produces ciphertexts dedicated to a single recipient only. The performance evaluation is carried out based on the results of cryptography benchmarks that were performed on low-end and mid-range user client devices.

  The performance evaluation of a common E2E-SDS protocol shows that major performance penalties hit users that manage group memberships in large user groups, i.e., user groups with a few thousand or more members. Possible solutions to decrease the performance penalties of group membership management are offered by the research area of Group Key Management that aims at an efficient distribution and renewal of cryptographic keys within groups. For this reason, the common E2E-SDS protocol is enhanced with a promising Group Key Management approach. The performance evaluation shows that with this enhancement, the performance penalties for the management of group memberships in large user groups are decreased significantly.

– **Performance evaluation of E2E-SDS protocols based on Attribute-Based Encryption:** Attribute-Based Encryption (ABE) allows to dedicate a single ciphertext to multiple recipients which, at the first glance, promises to enable resource-saving SDS systems in terms of ciphertexts that have to be generated and distributed. However, many of the ABE schemes proposed so far are centralized, i.e., they rely on a central party that is able to decrypt any ciphertext in the system. To achieve E2E-SDS using ABE, an ABE approach was identified that offers end-to-end security. This approach was transformed to an E2E-SDS system that is capable of supporting named user groups, resulting in different implementation variants. The performance of these variants was evaluated in depth, again based on real-world sharing model instances. A part of the performance evaluation was the implementation and benchmarking of the ABE protocol on mid-range user client devices based on Android.

– **Design and performance evaluation of an E2E-SDS protocol without joint authorization operations:** The E2E-SDS protocols proposed in literature leverage joint authorization operations, which require that the involved users are reachable in the network and able to allocate the necessary computational resources immediately upon request. In this thesis, an E2E-SDS system is presented that realizes the support for named user groups without joint authorization operations. The performance of this protocol is simulatively evaluated based on real-world sharing models. Furthermore, the sensitivity of the performance of the most computing-intensive operations is assessed based on synthetic sharing model instances.

The contributions of this thesis have partially been presented before in the following publications:

Holger Kühner and Hannes Hartenstein. Schlüsselverwaltung für sichere Gruppeninteraktionen über beliebigen Speicheranbietern : ein Überblick. In *6. DFN Forum Kommunikationstechnologien*, pages 119–129, 2013

Holger Kuehner and Hannes Hartenstein. Spoilt for Choice: Graph-based Assessment of Key Management Protocols to Share Encrypted Data (poster paper). In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy (CODASPY '14)*, pages 147–150, 2014

Holger Kuehner and Hannes Hartenstein. On the Resource Consumption of Secure Data Sharing. In *Proceedings of the 2015 IEEE International Symposium on Recent Advances of Trust, Security and Privacy in Computing and Communications (RATSP '15)*, pages 880–889, 2015

Holger Kuehner and Hannes Hartenstein. Decentralized Secure Data Sharing with Attribute-Based Encryption: A Resource Consumption Analysis. In *4th ACM International Workshop on Security in Cloud Computing (SCC '16)*, pages 74–81, 2016

Kevin Koerner, Holger Kuehner, Julia Neudecker, Hannes Hartenstein, and Thomas Walter. Bewertungskriterien und ihre Anwendung zur Evaluation und Entwicklung sicherer Sync&Share-Dienste. In *8. DFN-Forum Kommunikationstechnologien*, pages 71–82, 2015

1.3 Thesis Outline

In Chapter 2, necessary fundamentals on authorization models, SDS protocols and performance evaluation are provided. Chapter 3 introduces *workloads* as a formalization of sharing models. Workloads constitute the basis for the analytical and simulative performance evaluation methods, which are introduced in Chapter 4. In Chapter 5, the performance of E2E-SDS protocols that demand joint authorization operations is evaluated. The evaluation comprises protocols that are based on traditional cryptography, as well as protocols based on Attribute-Based Encryption. A novel E2E-SDS protocol that works without joint authorization operations is presented in Chapter 6. The thesis is summarized and concluded in Chapter 7.

2 Fundamentals and Related Work

In this chapter, the challenges that arise in the design and performance evaluation of *End-to-End Secure Data Sharing (E2E-SDS)* protocols are discussed in detail. For this purpose, existing E2E-SDS protocols are presented, their limitations with regard to the support of certain authorization models are shown, and the challenges in evaluating their real-world performance are analyzed.

In Section 2.1, the basic entities within a sharing system are introduced, and various authorization models for data sharing are discussed. An analysis of the authorization model of some exemplary sharing systems motivates the support for named user groups, which are an essential feature of the authorization model that is predominantly used in this thesis.

Section 2.2 introduces the notion of SDS protocols, which is further refined by the notion of E2E-SDS protocols. SDS protocols are presented that are based on traditional cryptographic primitives, and these protocols are analyzed with regard to the support of named user groups. Basic mechanisms used in these protocols serve as building blocks for the protocols that are evaluated in this thesis.

In Section 2.3, the novel cryptographic primitive of Attribute-Based Encryption is introduced, which enables to build ciphertexts that are dedicated to multiple recipients instead of a single one. The evolution of SDS approaches based on Attribute-Based Encryption is sketched, and open issues are identified.

In Section 2.4, performance evaluation methods are discussed in the context of SDS protocols. A major requirement for an evaluation of the real-world performance are realistic sharing and usage scenarios that reflect the behavior of the users in the system. Different approaches to model the sharing and usage behavior are presented, especially *workloads* as a basic model that combines sharing and usage models. As realistic workloads cannot be built from traces of real-world systems in any case, the challenges of generating synthetic workloads are discussed.

2.1 Authorization Models for Data Sharing

Sharing data is always related to some concept of who should be able to access the data, and who should not. Sometimes this conception might be rather fuzzy, when data is shared publicly with some target audience like "potential customers" in mind. In other cases, however, the conception is very specific, with a clearly defined group of recipients, and the goal to protect the data from inspection by any non-recipient. In the latter cases, the access to the shared data has to be controlled.

Access control for data sharing can be considered at different abstraction levels. One approach for differentiating between these abstraction levels is provided by the *OM-AM* framework [San00]. The acronym OM-AM denotes the four layers that are incorporated by the framework: The most abstract layer is the *objective* layer, which describes recipients and non-recipients in non-technical terms as a high-level policy. *Access control models* define how the group of recipients can be specified [Ben06]. While these two upper layers specify *what* should be achieved, the lower layers architecture and mechanism state *how* the access control objectives are achieved, i.e. how access control is *enforced*. E2E-SDS protocols are located at the architecture and mechanism layer. In this section, some access control models will be discussed that can build the basis for SDS protocols.

In the recent decades, a plethora of access control models was presented that can be used for data sharing. Among the most popular and widely deployed access control models for data sharing are Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC), which will be discussed in the following. Note that the scope of these access control models is not restricted to data access, but these models can be applied to control access to resources in a broader sense, including, e.g., access to services or hardware.

Before a discussion of the different access control models, some related terms have to be clarified. An *entity* is "a person, organization, software program, machine, or other thing making a request to access a resource" [Win05]. A *digital identity* is a digital representation of a real-world entity in the context of a certain access control domain. In this thesis, the term *user* is used as a synonym for digital identity. A user is described by a set of *attributes* [DH08]. The attributes can be naturally related to a user or the entity behind, respectively, such as date of birth, but can also be assigned by an authority, such as "is student at a certain university". Typically, at least one attribute constitutes an *identifier*, i.e., it "unambiguously distinguishes one entity from another one in a given domain" [ISO11a]. Note that an entity might hold a multitude of corresponding users, even within one access control domain. The objects of the access control model are *resources*. In the context of this thesis, a resource represents the smallest chunk of data that access can be granted to independently from other resources.

Discretionary Access Control (*DAC*, cf. [Ben06]) is based on two main principles: First, any resource in the system is managed by one or more dedicated users, denoted as *resource owners*. The resource owner is responsible for granting access to the resource for other users. Second, permissions can be assigned to other users in a fine-

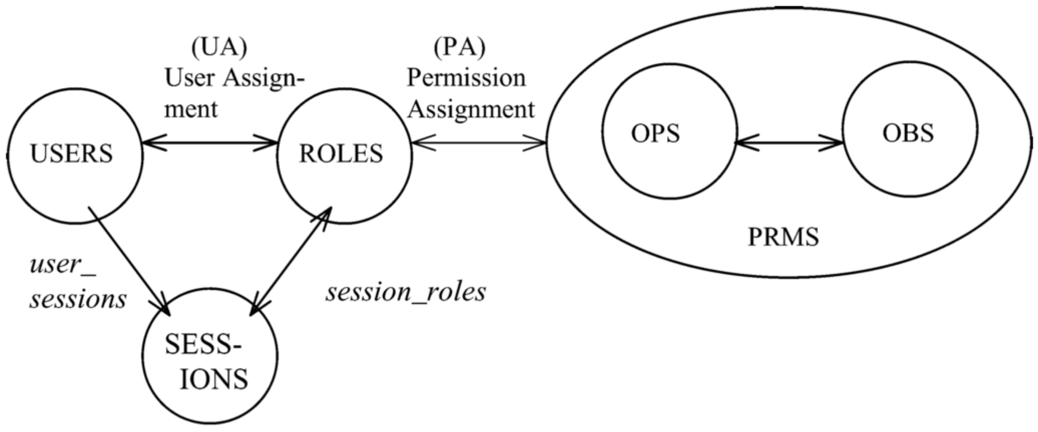


Figure 2.1: Entities and their relations in the Role-Based Access Control (RBAC) model. Roles are assigned to users and permissions. Permissions describe operations on objects. Source: [SCFY96]

grained manner by specifying direct relations between users and resources [TCS85]. These resulting permissions are typically abstracted as an *access control matrix*, where a row represents a user, a column represents a resource, and a cell contains the set of permissions that the associated user is granted on the associated resource. There are two general implementation alternatives for the access control matrix: *Access control lists* represent the subjects that are allowed to access a resource from the perspective of a resource, while *capabilities* list the accessible resources from the perspective of a subject. DAC is in widespread use today, for example, in commodity operating systems, such as Windows and Linux, but also in recent Sync&Share services.

Mandatory Access Control (MAC) [TCS85] is often thought of as counterpart of DAC, as when facilitating MAC, the access permissions are determined by the system administrator rather than by the users. For this purpose, the administrator assigns sensitivity labels to each subject and object in the access control domain. The labels are usually ordered in a hierarchy, constituting *classification levels*. One instantiation of MAC is the famous Bell-LaPadula model [BP76]. This model stipulates that a user is allowed to read a resource if her classification level is greater than or equal to the classification level of the resource. On the opposite, a user is only allowed to write a resource if the classification level of the user is less than or equal to the classification level of the resource. This principle is often summarized as "read down, write up" [Ben06], and should prevent a user from leaking sensitive information by moving the information from a sensitive resource to a less sensitive one.

Role-Based Access Control (RBAC) was proposed by Ferraiolo et al. [FBK92] and refined by Sandhu et al. [SCFY96] in the 1990s. RBAC is based on the concept of roles, which link a set of users to a set of permissions these users are granted. The objective of RBAC is to reduce the management effort that is necessary for the potentially numerous permissions that DAC might require. Further refinements of RBAC [SFK00, FSG+01] finally lead to a NIST standard [INC04] that specifies RBAC as family of access control models with *core RBAC* as the basic model (cf. Figure 2.1, which was extended by two more expressive models: *Hierarchical RBAC* enables to build a hierarchy of roles, and permissions are inherited along the hierarchy. Role hierarchies are typically used to resemble an organizational hierarchies and responsibilities, for example, the manager role might inherit the permissions of the clerk role. *Conditional RBAC* allows to express conditions on the assignment of users to roles. The most prominent applications of Conditional RBAC are *Static* and *Dynamic Separation of Duties* [AS99, AS00], which enable to define restrictions regarding the role combinations that a user is allowed to possess or activate at the same time. The deployment of RBAC bears the challenge of defining appropriate roles, which essentially can be done by inspecting the existing responsibilities within the organization, or by analyzing the existing access control matrix. Both approaches are extensively discussed in research [LY14]. RBAC is employed today on a broad basis [LO10], and is part of commodity file systems like NFSv4 or CIFS, which are presented later in this section.

Attribute-Based Access Control (*ABAC*, cf. [HFK+14]) is a more recent development. In ABAC, authorization decisions are based on the attributes of users and resources, and possibly also on attributes of the environment, such as time of day. ABAC can be considered as generalization of DAC or RBAC, since attributes can be used to represent identities or roles. ABAC allows to construct complex access policies out of attributes and Boolean operators, and is thus considered a flexible and expressive access control model, especially in distributed environments [YT05]. However, the comparatively high expressiveness may render the analysis of effective user permissions problematic [KCW10]. As of its strength with regard to distributed environments, ABAC is employed, for example, for sharing research data within scientific communities [Sch15].

In summary, the "big four" access control models form the basis for many concrete sharing systems. In many cases, sharing systems do not implement one of these access control models in its pure form, but combine, extend and adapt these models for their purposes. There are more access control models that describe or influence current sharing systems. An exemplary one is presented in the following.

Group-Centric Secure Information Sharing (g-SIS) [KSNW09, KNSW11] was introduced as a framework that specifies access control semantics based on the notion of groups. In the context of g-SIS, a group comprises users and resources. An important aspect of g-SIS is time: Users join and leave a group at certain points in time, and the same applies for resource additions and removals. Each of these actions can be carried out in a liberal or a strict variant. To decide whether a certain group user is authorized to access a certain group resource, the history of join, leave, add and remove actions of the user and the resource, as well as the respective variant, are taken into account. Figure 2.2 shows some exemplary sequences of these actions for only one user and one resource. A thick line visualizes the time frame the user is allowed to access the resource. When comparing, e.g., the second and third line of Figure 2.2, it can be seen that a user that was strictly joined is not allowed to access a resource that was added to the group before. However, if the user was joined liberally, she may access the resource if the resource was also added liberally. Some extensions for

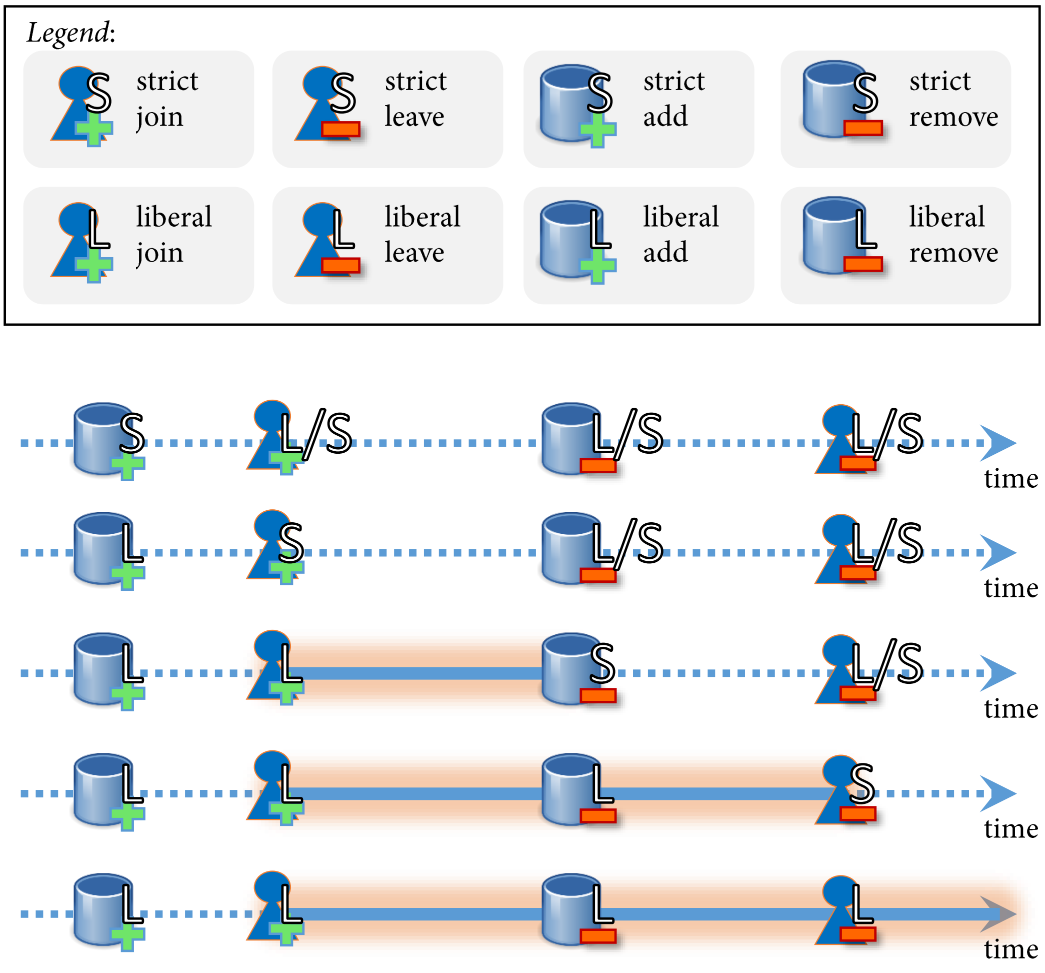


Figure 2.2: Exemplary sequences of g-SIS actions for a single user and a single resource. Time frames in which the user has access to the resource are highlighted.

g-SIS were proposed, e.g., a generalization for multiple groups and relations between these groups was discussed [SKNW10]. Furthermore, data structures necessary to save and look up the operation history [KS11] as well as enforcement mechanisms for g-SIS [KS09] were proposed. It was shown that g-SIS might not strictly be more expressive than RBAC, but be more efficient in group-centric sharing scenarios regarding, e.g., the size of the access control state [GQL14]. The advantage of g-SIS in the context of E2E-SDS is that it represents a unified framework for describing and comparing the access control models that underlie many E2E-SDS protocols.

An access control model can only be employed in combination with an *administrative model*. Administrative models can be considered special instances of access control models that exclusively deal with *administrative permissions*, i.e., with permissions regarding changes of the access control state. In other words, an administrative model specifies formalisms that are used to define the subjects that are allowed to change certain parts of the access control state. While DAC and MAC are bound to administrative models adhering to certain paradigms by their very definition, RBAC and ABAC are not restricted to certain administrative models. As a consequence, a bunch of administrative models was introduced especially with regard to RBAC, for example, *ARBAC97* proposed by Sandhu et al. [SBM99]. A major challenge in the design of administrative models is to "provide a mechanism for defining administrative domains" [SBM99], i.e., to define subsets of the access control state in a manageable way. ARBAC97 addresses this challenge by distinguishing between the management of user-role assignments and the management of role-permission assignments, which is reflected in the definition of two separate partial models, *URA97* for user-role assignments and *PRA97* for permission-role assignments. This distinction is also an important aspect of the authorization model used in this thesis (cf. Section 3.1). The term *authorization model* denotes the combination of access control model and administrative model.

To show some concrete instantiations of the authorization models described above, the respective models of the file systems NFSv4 and CIFS are presented in the following. *NFSv4* is a distributed file system that is widely deployed in Unix-based IT environments. It is usually implemented on top of an existing local file system. In NFSv4 [HN15], files and directories can be protected using *Access Control Lists (ACLs)*. An entry in this list (*Access Control Entry, ACE*) either allows or explicitly denies the execution of a set of operations, such as read, write, add or delete, for a certain subject. Subjects can be both users and named user groups, which will be introduced shortly. NFSv4 also supports inheritance of access permissions along the directory structure, i.e., an ACE on a directory can be flagged to be valid for all files in the directory, or recursively for all subdirectories. In addition, NFSv4 offers traditional UNIX access control, where the access control subjects are constituted of the resource owner, the resource owner's primary group, and everyone. In summary, NFSv4 directly implements DAC, and the support for user groups enables a combination of NFSv4 and RBAC.

Another popular distributed file system is *CIFS* [Mic11a], which can be considered the Windows analog to NFS. CIFS shares its access control model with its extensions SMBv1 [Mic11b], SMBv2 and v3 [Mic11c]. Similar to NFSv4, access permissions are stored within an ACL, which is part of the so-called *security descriptor* (cf. [Mic14]). The content of a CIFS ACE is very similar to that of an NFSv4 ACE, containing facilities to define a set of allowed or denied operations for a certain subject, where subjects can be users or named user groups. Inheritance of access permissions is supported as well. Opposed to NFSv4, CIFS does not offer facilities for a traditional UNIX access control. Similar to NFSv4, CIFS supports DAC, and the support for named user groups allows to combine CIFS with RBAC. When used on recent Windows versions, CIFS includes a MAC implementation, which is restricted to a predefined set of security labels though.

Both the authorization models of NFSv4 and of CIFS show strong similarities to the authorization model that is used in this thesis (cf. Section 3.1). A central feature of this authorization model is the support for *named user groups*. The essential property of these user groups is that they are not existing only in the context of permissions on a set of resources. Therefore, named user groups are not implicitly defined by the associated permissions. Instead, named user groups are independent entities that can be granted data access via a group handle. The user that manages group memberships is referred to as *group owner* in this thesis.

The concept of named user groups is closely related to that of roles in RBAC: They are similar in that they exist even without any assigned permissions, and that in general, access for either named user groups and roles can be granted to arbitrary resources. The difference between named user groups and roles is a rather semantic one: While roles are thought to comprise a set of users and a set of permissions, named user groups describe only the set of users, but not the assigned permissions [San96].

The need to control the usage of data after sharing lead to a recent shift in the research area of access control towards usage control. Usage control essentially extends access control with a continuous control of access permissions. Thus, access permissions are not only checked at the time of the access request, but are continuously checked during resource usage. The UCON [PS02, PS04] model formalizes usage control. Essential parts of UCON are *Authorizations*, *Obligations* and *Conditions*: While authorizations reflect traditional access control rules based on the attributes of subjects and objects, obligations require a subject to take some action before accessing the resource. Conditions are subject- and object-independent properties of the environment that impact the usage control decision. Each of these parts can be evaluated or enforced at access time or at usage time. A usage control model might support only a subset of these parts or evaluation options, resulting in a variety of possible usage control models. Models for usage control in distributed environments [PHB06, Pre09] were also proposed. An important application for usage control is Digital Rights Management (DRM) [PHS+08]. The enforcement of usage control is beyond the scope of this thesis.

2.2 SDS Protocols Using Traditional Cryptography

In this section, the necessary terminology regarding the context of Secure Data Sharing (SDS) is introduced, and existing SDS protocols are presented.

In existing literature, no definition of *Secure Data Sharing (SDS)* could be found that is both precise and comprehensive. Nonetheless, the prevalent meaning of SDS can roughly be paraphrased with "using cryptography to enable data sharing over a storage provider while the storage provider is prevented from inspecting the data or from tampering with the data". Thus, the central security objectives of SDS are *confidentiality*, which is defined as "Property that information is not made available or disclosed to unauthorized individuals, entities, or processes." [ISO14], or *integrity*, which is defined as follows: "Integrity of data is ensured if it is impossible to insert, manipulate or delete data without authorization and detection" [Eck12].

The object of study of this thesis are *cryptographic protocols* that enable SDS. In general, "a cryptographic protocol is defined as a series of steps and message exchanges between multiple entities in order to achieve a specific security objective" [Sch03]. Thus, in this thesis, a cryptographic protocol is considered an *SDS protocol* if it strives to protect the confidentiality or integrity of the shared data with regard to a storage provider that is not fully trusted.

In this context, a *storage provider* is considered as network-reachable service that allows to store and retrieve opaque data. From the perspective of an SDS protocol, the internal organization of the storage provider is usually transparent, i.e., the storage provider might be realized as a single server, as a cloud of virtual machine instances, or even as a distributed storage service that is maintained by a large number of peers.

An important part of each SDS protocol is the accompanying *trust model*. The trust model specifies the amount of trust that entities put into one another regarding a specific security objective. Typically, SDS protocols assume that users' trust in the storage provider is limited with regard to a security objective, such as confidentiality or integrity. An example for an entity with limited trust is an *honest but curious* entity, which is assumed to carry out each step of the cryptographic protocol honestly, but is suspected to inspect data without authorization.

The definition of *End-to-End Secure Data Sharing (E2E-SDS)* protocols further restricts the definition of SDS protocols in terms of trust in parties external to the group of sharing users: E2E-SDS protocols leverage cryptography to ensure that "no-one, other than the data owner and the members of the [sharing] group, should gain access to the data, including the Cloud Service Provider" [TCNC14] [1]. Note that this definition implies the enforcement of both confidentiality and integrity by cryptography, thus, the enforcement of just one of these two security objectives is no longer sufficient.

From a trust perspective, the definition of E2E-SDS stipulates that only the members of the sharing user group are trusted with regard to confidentiality and integrity of the shared data. As a consequence, not only the storage provider must be prevented by technical means from inspecting or altering the data, but anyone apart from the sharing user group must be prevented. This requires that the authorization is carried out solely by users who are allowed to access the data anyway, without the help of any additional entity that has to be trusted. The access control enforcement also has to be carried out solely by the users by means of cryptography, as in any other case, the additional entity that is to enforce access control could collaborate with some non-authorized user to grant access to this user.

Note that while E2E-SDS protocols exist that guarantee confidentiality and integrity in such "low-trust" environments, practical SDS systems usually strive to achieve full *data security* [Shi00], which additionally includes *availability* of the data shared over the storage provider. Availability is defined as "property of being accessible and usable upon demand by an authorized entity" [ISO14]. Availability of the data on the storage provider can hardly be guaranteed by means of cryptography. Thus, all of the E2E-SDS protocols presented and discussed in this thesis are based on the assump-

[1]As this definition refers to Secure Data Sharing in the original publication, it excludes many protocols that are referred to as Secure Data Sharing protocols by their designers. Thus, it can be considered more appropriate for *End-to-End* Secure Data Sharing, which is the case in this thesis.

tion that the users put trust in the storage provider with regard to availability.

Because of the relaxed definition of SDS compared to E2E-SDS, many of the SDS protocols presented in this section do not constitute E2E-SDS protocols in a strict sense. The presented SDS protocols can nonetheless be used as a basis that can be enhanced to an E2E-SDS protocol, or provide valuable building blocks that can be composed to E2E-SDS protocols.

While confidentiality and integrity are essential security objectives for both SDS and E2E-SDS, there are a lot more security objectives that are neither required by SDS nor by E2E-SDS, according to the definitions given above. Nonetheless, many SDS protocols put a focus on such additional security properties, which often constitutes the major differentiator to other SDS protocols. One example for an additional security properties is *privacy* as "the right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed" [Shi00]. A further example is *accountability* as "the property of a system (including all of its system resources) that ensures that the actions of a system entity may be traced uniquely to that entity, which can be held responsible for its actions" [Shi00]. Security objectives apart from confidentiality and integrity are beyond the scope of this thesis.

From an access control perspective, SDS protocols can be classified as architecture or enforcement mechanism within the OM-AM framework (cf. Section 2.1). In a broader scope, unauthorized access to shared data can be seen as an information security risk [ISO11c]. Therefore, from a risk management perspective, an SDS protocol constitutes a measure to *mitigate* a risk, i.e., to reduce the likelihood of risk occurrence.

The employment of SDS protocols for providing confidentiality or integrity of the shared data is recommended in various guidelines for a secure usage of cloud resources. For example, the Cloud Security Alliance proposes end-to-end encryption of data as preferred method to protect data that is to be moved to the cloud [All14]. The German "Bundesamt für Sicherheit in der Informationstechnik" (BSI) demands that "to store, process and transport sensitive information in a secure manner, adequate cryptographic methods and products should be employed." [BSI11]. As a final example, the US-American "National Institute of Standards and Technology" (NIST) states with regard to data protection: "A guiding principle is for employees of the organization to be in control of the central keying material and to configure the key management components for cloud-based applications" [JG11].

2.2.1 Cryptographic Primitives

Cryptographic primitives are the basic building blocks of any E2E-SDS protocol. The protocols presented in this section essentially employ well-known cryptographic primitives, such as symmetric and asymmetric encryption, digital signatures and Keyed-Hash Message Authentication Codes, which are briefly introduced in this subsection. These cryptographic primitives are referred to as *traditional cryptography* in this thesis. Protocols based on the more recent cryptographic primitive of

Attribute-Based Encryption are presented in Section 2.3.

For using *symmetric ciphers*, all parties that exchange encrypted information share a common cryptographic key. This key is employed both for encryption and decryption. A popular symmetric cipher is the *Advanced Encryption Standard (AES)* [AES01], which was standardized by NIST in 2001. AES is a block cipher, i.e., it requires that plain text data is fragmented into blocks of equal size before encryption. The block size supported by AES is 128 bits, while keys with a length of 128, 192 or 256 bits are allowed. At least up to 2015, no practical attacks that enable to decrypt AES-encrypted data without knowledge of the symmetric key were known, apart from so-called "related-key attacks" against AES with key lengths of 192 or 256 bit [BK09]. Hardware implementations of AES are broadly discussed in research [HHH07], and realized in commodity consumer devices[2].

Asymmetric ciphers employ a key pair for both the encrypting and the decrypting party [Eck12]. A key pair consists of a public key that can be published freely and is required for encryption, and a private key that has to be kept secret and is used for decryption. A widely deployed asymmetric cipher is *RSA*, which is named after its designers Rivest, Shamir, and Adleman[RSA78]. The combination of RSA and the OAEP padding was proven to yield indistinguishable ciphertexts under an adaptive chosen ciphertext attack (IND-CCA2 secure) in the random oracle model and under the RSA assumption [FOPS01], which in turn relies on the wide-spread assumption that factoring integers is a hard problem [vTJ11].

Many of the SDS protocols introduced in this section are based on *hybrid encryption*, i.e., the shared data is symmetrically encrypted, and the symmetric encryption key is asymmetrically encrypted for confidential distribution.

Digital signature schemes are used to protect the authenticity and integrity of data. These schemes employ asymmetric cryptography, where the private key is used for data signing, and the public key is required for the verification of the signature. While digital signatures schemes strive to make even the smallest changes on the data detectable in general, some existing schemes allow for a set of defined and controlled changes to the data without invalidating the signature. For example, with *sanitizable signatures* [ACdMT05], designated blocks of the signed data can be changed by certain parties without invalidating the signature, while *redactable signatures* [JMSW02] are valid even when defined parts of the signed data are removed. RSA can be used as a signature scheme as well. Some more recent signature schemes, such as the *Elliptic Curve Digital Signature Algorithm (ECDSA)* [ANS05], leverage the assumption that the discrete logarithm problem is hard to solve in elliptic curves, which are a special kind of algebraic groups. EC-DSA achieves a security level similar to RSA with shorter key lengths [BSI16], and requires less computation time to generate and validate signatures [GGCS02]. While a growing number of applications of Elliptic Curve Cryptography (ECC) can be observed, such as the German electronic passport [BSI15], the US National Security Agency does not recommend to migrate existing cryptography solutions to ECC, claiming that the advent of quantum computers renders ECC vulnerable [KM15].

[2]`https://www.apple.com/business/docs/iOS_Security_Guide.pdf`

The integrity of data can also be protected using *Keyed-Hash Message Authentication Codes (HMACs)* [KBC97]. HMACs are based on symmetric cryptography, and the symmetric key is required for both generation and verification. Opposed to a digital signature, an HMAC cannot be verified by the public.

The combination of encrypting and signing data to protect data confidentiality as well as data authenticity and integrity is a common use case. However, cryptographic primitives have to be combined with caution to form more sophisticated security mechanisms, as the security mechanism might not provide security properties that each cryptographic primitive provides in isolation (cf. [Can01]). In the case of encryption and signing, the order of applying encryption and signature, i.e., *encrypt-then-sign* or *sign-then-encrypt*, has some security implications. For example, *encrypt-then-sign* can render a cipher that is IND-CCA2 secure in isolation unsecure [ADR02]. As another example, encrypt-then-sign does not prevent an eavesdropper from replacing the original signature of the ciphertext by a self-generated one, thus, impersonating the originator of the data. On the other hand, sign-then-encrypt allows the recipient to encrypt the signed data with another key and forward the data to a third party, which might be tricked into believing that it is the recipient that was intended by the originator [Dav01]. Whether one of these scenarios is regarded a security issue, depends on the concrete application. Some public key schemes were proposed that combine encryption and signing [PSST11].

2.2.2 SDS Protocols with Focus on Confidentiality

SDS protocols that are based on traditional cryptography as introduced in the previous subsection require an initial exchange of cryptographic keys. The initial key exchange is typically not specified within the SDS protocol itself, but is considered as prerequisite for all protocol steps. As a consequence, the specification of an SDS protocol usually only includes a set of requirements on the initial key exchange, but does not define mechanisms or processes to achieve these requirements. For the initial key exchange, well-known mechanisms are widely deployed in an organizational or enterprise context: *Kerberos* [NYHR93] enables the pairwise exchange of symmetric keys, while *Public Key Infrastructures (PKIs)* (cf. [Eck12]) facilitate an authentic distribution of public keys. The design of key exchange mechanisms that might be accepted by users also in rather personal use cases is ongoing research. Recent approaches allow, for example, to exchange keys by means of QR codes in a peer-to-peer fashion [KW14].

The basic working principle of almost any SDS protocol that strives to ensure confidentiality is to encrypt the shared data with a data encryption key, and to enable each authorized user to "infer" this data encryption key by means of the keys that resulted from the initial key exchange. The process of key inference might span several "intermediate" keys, resulting in key cascades that start at an initially exchanged key and finish at a data encryption key.

As these key cascades are interconnected, the totality of key cascades can be conceptually considered as directed graph, where the keys constitute the graph nodes,

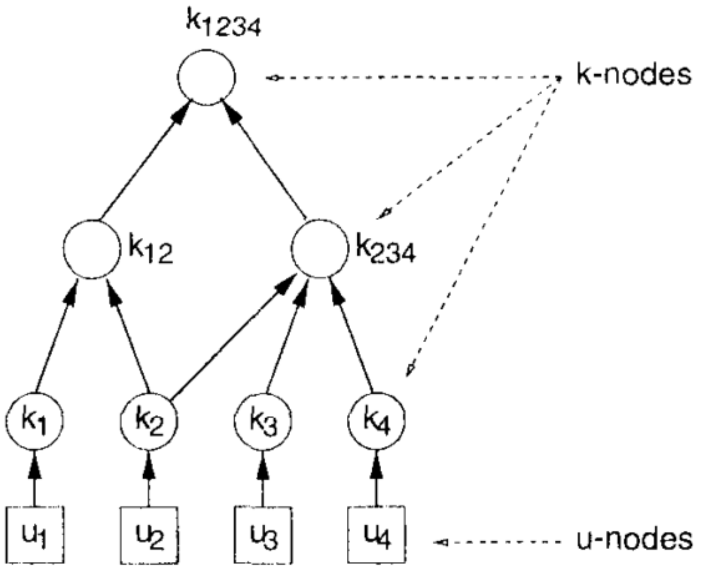


Figure 2.3: Visualization of a key graph. k-nodes correspond to cryptographic keys, u-nodes to users. Source: [WGL98]

and an edge connects one key to another iff the key at the end of the edge can be inferred from the key at the edge start. Users can be brought in as further node type, where a user node is connected to a key node iff the user knows the key. This type of graph was introduced as *key graph* in [WGL98]. An exemplary instance of a key graph is depicted in Figure 2.3. From a key graph perspective, the technical objective of an SDS protocols is to build and maintain the key graph in a way that the target authorization state is preserved at every point in time.

There are different ways for SDS protocols to implement key inference based on cryptographic primitives. Basically, there are mechanisms that enable one key to be inferred from another one without the need for any additional information, e.g., by applying a hash function to the key [HBB07]. In other cases, additional information is required to infer one key from another, e.g., the public key of the resource owner [KRS+03, FKK06] or a public "broadcast" [CC89, ZDB08, SNPB10]. Many of these key inference mechanism use cryptographic primitives that go beyond traditional cryptography as presented in the previous subsection. However, key inference can be implemented based on traditional cryptography by leveraging so-called lockboxes.

Lockboxes enable key inference by encrypting one key with another one. Thus, the encrypted key can be retrieved with the knowledge of both the encrypting key and the lockbox. A lockbox can be generated by a symmetric encryption of either a symmetric key or an asymmetric key pair, and also by an asymmetric encryption of a symmetric key. The asymmetric encryption of an asymmetric key pair is not possible in any case: With padded RSA, for example, a plaintext of a given size cannot be encrypted with a key of the same size. From a key graph perspective, a lockbox can be considered as a realization of a key graph edge.

A comparatively simple and introductory example of a lockbox-based SDS protocol is the *Encrypted File System (EFS)* [Cro02]. This protocol demands the resource owner to encrypt the resource with a symmetric key, and then generate a lockbox by encrypting this symmetric key with her personal public key. Thereafter, both the encrypted resource and the lockbox are uploaded to the storage provider. The resource owner grants access to the resource for another user by generating another lockbox that encrypts the symmetric encryption key with the public key of this user. One shortcoming of EFS is that it does not protect the integrity of lockboxes. This protection is necessary to prevent an attacker from replacing genuine lockboxes with forged ones that provide attacker-chosen keys to potential writers of the resource. This attack will be discussed in more detail in Section 5.2. In the following, it is described how this vulnerability and further issues are addressed by more sophisticated SDS protocols.

One of the earliest lockbox-based SDS protocols is *SiRiUS* [GSMB03]. The protocol protects both the confidentiality and integrity of the shared data. SiRiUS will be presented in detail, as many building blocks of SiRiUS are incorporated into other protocols.

Using SiRiUS, each resource owner manages a dedicated resource tree, i.e., the resources are organized in a hierarchical structure, and access permissions are propagated along the hierarchy (cf. NFSv4 and CIFS as described in Section 2.1). The resource owner is the only user that grants or revokes access to her resources, i.e., SiRiUS implements a discretionary access control model.

SiRiUS requires an authentic and integer distribution of asymmetric public keys, which could be carried out, e.g., by leveraging a PKI. This requires that each user generates a personal asymmetric key pair up front. The basic working mechanism of SiRiUS is that the resources itself are symmetrically encrypted and digitally signed, and the encrypted resource and the signature are uploaded to the storage provider. Both the symmetric encryption key and the asymmetric private signing key are encrypted with the public key of each permitted user, and these lockboxes are also uploaded to the storage provider. In the following, the required steps will be presented in more detail.

Figure 2.4 visualizes the status of the storage provider (left) and the client of the resource owner (right), who is denoted as *User A*, after the initial upload of a resource. The encrypted resource is stored on the storage provider as *data file*, the digital signature is appended. Each data file is accompanied by a *metadata file*, which forms the container for all lockboxes that are related to the encrypted resource, and for the public signing key. As SiRiUS is implemented as an overlay for an existing file system, from the storage provider's view, both data file and metadata file constitute regular files.

On the user client of the resource owner, the personal key pair *UA*, the file encryption key *FEK*, and the file signing key pair *FSK* are stored. The metadata file contains two lockboxes: Both the file encryption key and the private part of the file signing key pair are encrypted with the public part of the personal key pair of the

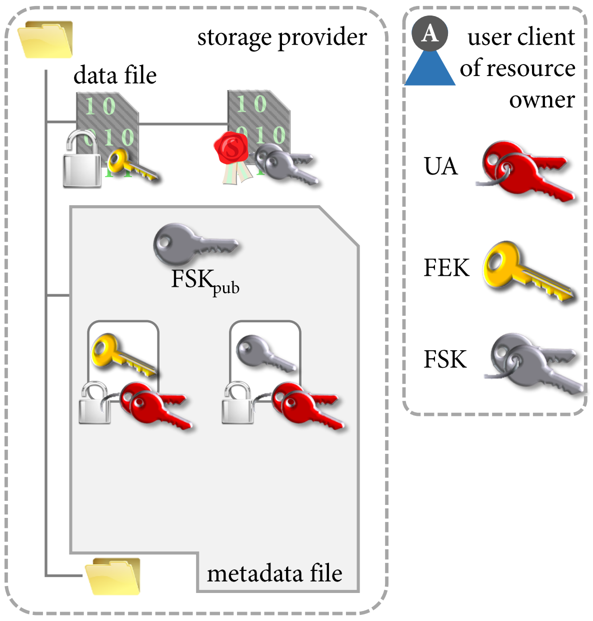


Figure 2.4: Visualization of keys and lockboxes generated by the SiRiUS protocol after an encrypted data file was uploaded.

resource owner. This allows the resource owner to retrieve the respective keys when the resource should be read or written at a later point time, thus omitting the local caching of these keys. In addition, the public part of the file signing key pair is contained within the metadata file, as this key is required by any reader of the resource to validate the signature.

The status of the storage provider after read and write permissions are granted to *User B* is shown in Figure 2.5. For granting read access, the file encryption key is encrypted with the personal public key of *User B*, and the resulting lockbox is added to the metadata file. Additional write access is granted by encrypting the private part of the file signing key pair with the personal public key of *User B*.

To revoke the access rights of a user, all resource-specific keys that are known to this user have to be renewed, i.e., the keys have to be replaced by newly generated ones. The consequence of renewing the file encryption key is that each lockbox that encrypts this key must be recreated. The same applies for the file signing key pair. Moreover, to immediately revoke read access, the encrypted data itself must also be re-encrypted with the new file encryption key. This means that the data file must be downloaded from the storage provider, decrypted with the old file encryption key, then encrypted with the new file encryption key, and finally uploaded to the storage provider. This cumbersome mechanism for revocation enforcement is denoted as *eager revocation* in this thesis. A more resource-saving, yet less strict revocation enforcement mechanism is known as *lazy revocation* and will be introduced in a moment.

SiRiUS protects not only the integrity of the data file, but also the integrity of the metadata file, by means of digital signatures. Although this prevents an attacker from

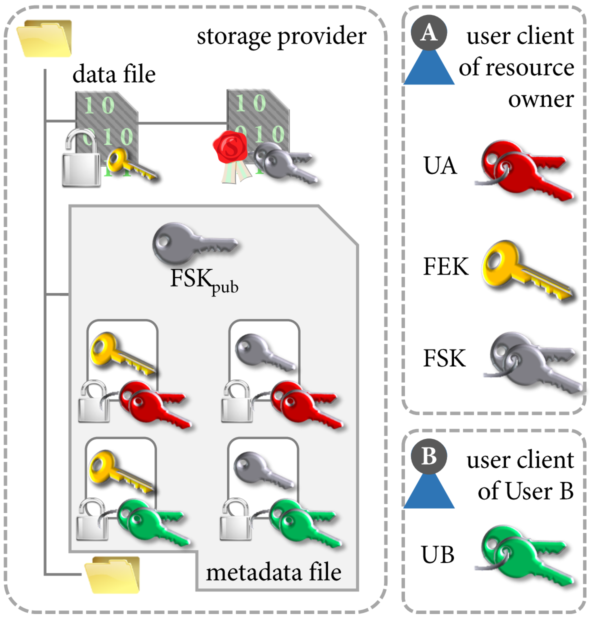


Figure 2.5: Visualization of keys and lockboxes generated by the SiRiUS protocol after an encrypted data file was shared with another user.

simply replacing genuine lockboxes by forged ones, it still allows an attacker to "roll back" the metadata file to a previous version, which is known as *rollback attack*. A rollback attack could be carried out, e.g., by a user that formerly had write access, with the objective to restore the write access. Rollback attacks can be fought by incorporating an expiration date into the signatures, and periodically refreshing the signature before expiration. This way, a rollback attack is only possible as long as the rolled back signature is not yet expired. For reasons of efficiency, SiRiUS creates a hierarchy of signatures along the directory hierarchy by leveraging a Merkle Hash Tree [Mer80]. With this construction, a signature related to a directory also guarantees the integrity of each file in this directory, and recursively of each subdirectory. In summary, SiRiUS fulfills all the requirements on an E2E-SDS protocol.

From a performance perspective, SiRiUS is not intended for use in large-scale scenarios [GSMB03], as the number of necessary lockboxes grows linearly with both the number of files, and the number of users that are allowed to access these files. Given that n users are allowed to read m files, a total number of $n \times m$ lockboxes have to be generated by the resource owner. In addition, when a user is revoked read access to all of these files, the resource owner has to recreate $(n-1) \times m$ lockboxes. More recent SDS protocols strive to lower the number of required lockboxes by introducing auxiliary keys for resource groups. This allows to grant a user access to the whole resource group by generating just one single lockbox. Two examples for such protocols are presented in the following.

Plutus [KRS+03] exhibits a feature set that is very similar to SiRiUS: Plutus supports a discretionary authorization model, offers to grant read-only access, and protects the integrity of data and metadata. For performance reasons, Plutus enables to assemble files with identical permissions to *filegroups*. Each user with read permission on the filegroup is provided with a *file-lockbox key*, which enables to get the encryption keys for each file in the filegroup.

To further improve performance, Plutus leverages a mechanism denoted as *lazy revocation*. When lazy revocation is used, a resource owner only renews the resource encryption key after user revocation. The re-encryption of the resource is delayed until the next time the resource is written. This way, a potentially resource-consuming re-encryption of a resource is omitted. The drawback of lazy revocation is that from a cryptographic point of view, a revoked user can continue reading the resource as long as its contents have not been changed.

The Cryptree [GMSW06] protocol is also based on the idea of grouping files with identical permissions. Cryptree relies on the assumption that files within one directory tend to be accessible by the same set of users. For this reason, each directory is related to a set of keys, which allow to derive the encryption keys for all the files in the directory. Moreover, a directory key can be used to derive the directory keys of each subdirectory, which makes it possible to recursively retrieve the encryption keys for all the files in all the subdirectories.

2.2.3 Named User Groups and Joint Authorization Operations

The SDS protocols presented in the previous subsection—EFS, SiRiUS, Plutus and Cryptree—implement a discretionary access control model, i.e., neither of these protocols supports named user groups. In this section, some SDS protocols are presented that offer to grant access permissions to named user groups.

The support for named user groups bears a challenge. To support named user groups, authorization operations are required to manage group memberships, i.e., to add a user to and remove a member from a group. These authorization operations are initiated by the respective group owner. However, the operations might require the renewal of keys that are managed by a resource owners, which means that these keys are clearly related to one or more resources instead of a user or a named user group. As an example, when a member is removed from a group, the encryption keys of all resources the group is allowed to access have to be renewed. This brings up the question who should be responsible for renewing resource-related keys after a group member removal.

In existing SDS protocols, the support for named user groups was implemented essentially in one of two ways: Either the renewal of resource-related keys after a group member removal was simply skipped. This obviously weakens security, as the removed user is cryptographically able to access the data until the resource-related key is renewed for another reason.

The alternative way to implement support for named user groups are *joint authorization operation*. In this thesis, the term indicates that a single authorization operation is carried out by at least two different entities in a collaborative manner. An operation is only considered as joint operation if every entity that is involved carries out some cryptographic operations that lead to changes in the key graph. This restriction prevents that, e.g., the generation and upload of a lockbox is considered as joint authorization operation, although it involves two parties: the lockbox creator and the storage provider.

Joint authorization operations require that all involved entities are reachable in the network, and are ready to carry out possibly computation-intensive cryptographic operations. Both aspects can be problematic especially when mobile devices are involved. The reachability of a mobile device can be limited in many situations, e.g., if the device is located inside a train or an airplane. Potentially computation-intensive cryptographic operations might influence the responsibility of the device. Therefore, in this thesis, only protocols are considered that strive for minimizing the employment of joint authorization operations.

Cepheus [Fu99] is an SDS protocol that uses joint authorization operations to support named user groups. Cepheus is based on the traditional Unix owner-group-world access control model (cf. Section 2.1), i.e., Cepheus does not enable the resource owner to share data with arbitrary groups. To remove a member from a group, Cepheus requires the group owner to renew the group-related keys first. Thereafter, all resource owners who granted access to the group have to be informed about the revocation. Each of these resource owners has to renew the resource-related keys of any resource that the group is allowed to access.

Other SDS protocols also support named user groups, but skip the renewal of resource-related keys altogether. For example, the *Secure File System (SFS)* [HF01] supports data sharing with an arbitrary number of named user groups. SFS requires a trustworthy group server, which stores all of the file-related keys, and provides them upon authenticated request. However, a renewal of file-related keys is not discussed at all. As another example, *SNAD* [MLFR01] also allows for sharing data with an arbitrary number of groups, but does not offer means for a group member removal at all.

A further challenge that the support for named user groups brings along is the performance of group member removal operations. A named user group is typically assigned a group key that has to be renewed when a member is removed from the group. Using a trivial implementation, the group key is encrypted with the public key of each group member, i.e., the number of necessary lockboxes grows linearly with the size of the group. In consequence, renewing the group key requires to replace each of those lockboxes, except the one that was related to the removed group member.

To improve the performance of renewing the group key of large groups, [GLHW12] proposed to leverage key management approaches from domain of secure multicast, which are known as *Group Key Management (GKM)* approaches. A plethora of GKM approaches was proposed in the last decades. An overview of GKM approaches and an evaluation of such approaches with regard to their asymptotic performance is provided in [KH13].

2.2.4 Applications of More Enhanced Cryptography

To conclude this section, some exemplary applications of cryptographic primitives beyond those discussed as "traditional cryptography" are introduced.

A first application of enhanced cryptographic primitives are SDS protocols that focus on integrity. The major challenge that is addressed by many of these protocols is data freshness, i.e., preventing an attacker from replacing the most recent state of the data by an older state. As long as only a single user is allowed to alter the data, freshness can be achieved by a periodic refreshment of the data signatures. As already described, this mechanism is leveraged by SiRiUS to protect the integrity of metadata.

For periodic refreshment, the refreshing user has to store a fingerprint of the most recent state of the data on her local user client. However, when multiple users are allowed to alter the data, a periodic refreshment of the data signatures cannot be leveraged. This is due to the fact that none of the writing users can be sure that her local copy of the fingerprint is still up-to-date.

In fact, it is a fundamental problem to prevent any kind of rollback attack when multiple writing users share data without at least one trusted storage provider. It was shown in [LKMS03] that under these conditions, the best consistency guarantee that can be given is *fork consistency*, which is intuitively described as follows: "A file system with fork consistency might conceal users' actions from each other, but if it does, users get divided into groups and the members of one group can no longer see any of another group's file system operations" [MS02].

Approaches were proposed that leverage enhanced cryptographic primitives to ensure fork consistency. However, these approaches exhibit major limitations. For example, the *SUNDR* protocol [LKMS03] causes a potentially huge amount of network traffic that quadratically grows in size with the number of clients that share a certain resource. While the protocols introduced in [CSS07] only cause the network traffic size to grow linearly with the number of clients, they require that all users are honest. A further limitation of these protocols is that every user has to store some information on her local client in order to check the validity of the data on the storage provider. This may especially by an issue if a user employs multiple client devices for data sharing, as the validation information has to be synchronized between these clients. Other protocols employ client-to-client communication to periodically synchronize the most recent fingerprint of the data [SCC+10], or rely on a trusted entity that is always reachable in the network [KWM15].

A plethora of more recent SDS protocols employ novel cryptographic primitives to outsource some of the required cryptographic effort to the storage provider. For this purpose, lots of protocols use *proxy re-encryption* [BBS98]. Given a storage provider that stores a ciphertext decryptable with private key K_A, this primitive enables the storage provider to re-encode the ciphertext to be decryptable with another private key K_B. Protocols such as [AFGH06] use proxy re-encryption, e.g., to have the storage provider distribute an encrypted resource encryption key to authorized users without knowing the key itself. These protocols exhibit the property that the access control enforcement is partially delegated to the storage provider, which clearly contradicts the definition of an E2E-SDS protocol. For this reason, this class of SDS protocols is not further considered in this thesis.

2.3 SDS Protocols Using Attribute-Based Encryption

Many SDS approaches proposed in the last few years are based on *Attribute-Based Encryption (ABE)* for confidentiality protection. One important reason for this is that ABE allows to dedicate a single ciphertext to multiple recipients which, at the first glance, promises to enable resource-saving SDS systems in terms of ciphertexts that have to be generated and distributed. For this reason, ABE-based SDS protocols are an object of study in this thesis.

In ABE approaches in general, both ciphertexts and secret keys are bound to attributes. A set of attribute matching rules determine whether a key can be used to decrypt a ciphertext. The expressiveness of possible matching rules depends on the concrete ABE scheme, and may allow for simple threshold rules where k out of n attributes have to match, as well as for arbitrary Boolean functions.

Most ABE schemes come in one of two flavors: *Ciphertext-Policy ABE (CP-ABE)* or *Key-Policy ABE (KP-ABE)*. With CP-ABE, the ciphertext encodes an access policy, which therefore has to be defined by the encrypting party. An access policy is typically formalized as Boolean term over attributes. An example for an access policy is "`IsCIO` $\lor$ (`IsManager` $\land$ `ITDepartment`)". The secret key corresponds to a plain set of attributes, which is not further structured in any way. The following procedure is used to evaluate whether the secret key satisfies the access policy of the ciphertext: Assign *true* to an attribute in the access policy exactly if the attribute is part of the secret key attribute set. If the access policy evaluates to *true*, decryption is possible; in any other case, decryption will fail.

Key-Policy ABE can be considered the opposite of CP-ABE: Keys encode access policies, and ciphertexts correspond to plain sets of attributes. With KP-ABE, the access policy is defined by the party that issues secret keys. Note that no matter whether CP-ABE or KP-ABE is used, the access control decision is based on both an access policy and a plain set of attributes. For this reason, the authorization process cannot be clearly assigned to a single party, but has to be considered as distributed and collaborative process.

The binding of users to attributes are managed by an *attribute authority (AA)*, which issues secret keys encoding a user's attributes upon authenticated request. When using CP-ABE, the attributes within one secret key are often "glued together" with a user-dependent value to prevent different users from pooling their attributes, a property known as *collusion resistance*. This user-dependent value is often generated by a central fully-trusted entity.

In this section, first a short introduction on the mathematical background of ABE is given. Thereafter, the evolution of ABE schemes is sketched.

2.3.1 Mathematical Background

ABE schemes are typically based on *bilinear pairings*, which are defined as follows:

Definition 1 (Bilinear pairing [ZSNL03]). Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and $G2$ be a cyclic multiplicative group of the same order q. A bilinear pairing is a map $e : G_1 \times G_1 \rightarrow G_2$ with the following properties:
P1 *Bilinear*: $e(aP, bQ) = e(P, Q)^{ab}$;
P2 *Non-degenerate*: There exists $P, Q \in G_1$ such that $e(P, Q) \neq 1$;
P3 *Computable*: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Bilinear pairings can help to solve the *discrete logarithm (DLOG) problem* in a cyclic group. The DLOG problem is defined as given g and h as elements of a cyclic group G, find a such that $h = g^a$ [CS11]. The existence of a bilinear pairing $G_1 \times G_1 \rightarrow G_2$ implies that solving the DLOG problem in G_1 is no harder than solving the DLOG problem in G_2: From the bilinearity property, it follows that $e(aP, P) = e(P, P)^a$. Therefore, to get the DLOG of aP in G_1, it is sufficient to calculate the DLOG of $e(P, P)^a$ to the basis $e(P, P)$ in G_2.

For this reason, bilinear pairings were originally used in cryptanalysis. The security of some cryptography schemes, such as the *Digital Signature Algorithm* [DSS13], relies on the assumption that it is not feasible for an attacker to calculate the DLOG within the cyclic group G_1 that is employed in the concrete implementation. However, even if there are no efficient algorithms known to calculate the DLOG within G_1 directly, it might still be possible to construct a bilinear pairing from G_1 to a cyclic group G_2, in which the DLOG can be calculated more efficiently. This attack can be leveraged in practice to reduce the DLOG problem within elliptic curves, which is assumed to be intractable, to the DLOG problem within finite fields, for which subexponential algorithms are known [MOV93].

The security of ABE schemes is usually based on the Decisional Bilinear Diffie-Hellman (BDH) assumption, which is defined as:

Definition 2 (Decisional Bilinear Diffie-Hellman (BDH) Assumption [SW05]). Suppose a challenger chooses $a, b, c, z \in Z_p$ at random. The Decisional BDH assumption is that no polynomial-time adversary is to be able to distinguish the tuple ($A = g^a$,$B = g^b$,$C = g^c$,$Z = e(g, g)^{abc}$) from the tuple ($A = g^a$,$B = g^b$,$C = g^c$,$Z = e(g, g)^z$) with more than a negligible advantage.

In the context of this definition, g is a generator of a multiplicative cyclic group, which was denoted as G_1 in the previous paragraphs. The BDH assumption obviously relies on the assumption that the DLOG problem is intractable in G_1, as in any other case, the BDH challenge could be solved easily by calculating a, b, c, and finally $e(g, g)^{abc}$. As already mentioned, the calculation of the DLOG is assumed to be infeasible in elliptic curves. Therefore, in existing ABE schemes, G_1 is often proposed to be an elliptic curve.

A basic example for an application of bilinear pairings is the three-party key exchange proposed by Joux [Jou00]. This protocol is an extension of the well-known Diffie-Hellman key exchange. The protocol has party A choose a random group element a, and send g^a to parties B and C. B and C calculate and distribute g^b and g^c in the same way. The common secret key $e(g, g)^{abc}$ can consequently be calculated by each user. For example, party A calculates the common secret key as $e(g^b, g^c)^a$.

2.3.2 Evolution of Attribute-Based Encryption Schemes

The foundation for Attribute-Based Encryption was laid in [SW05], which introduces ABE as a special application of *Fuzzy Identity-Based Encryption*. *Identity-Based Encryption (IBE)* allows to encrypt data for a certain recipient by using the recipients' identifier as public key [JN09]. This identifier might be, for example, an e-mail address. Thus, by using IBE, a PKI can be omitted, as there is no longer a need to retrieve a public key that is bound to the identifier in an authentic way. A fully functional IBE scheme was proposed by Boneh and Franklin in 2001 [BF01].

The main intention behind Fuzzy Identity-Based Encryption is to enable a decryption to users even if their identifier does not exactly match the identifier that was targeted in the encryption process. For this purpose, the identifier is decomposed into a set of binary-valued attributes. In consequence, both the user key and the ciphertext are related to a set of attributes. With Fuzzy IBE, a key can be used to decrypt a ciphertext if they have at least d attributes in common, where d is a system-wide security parameter.

The challenge that was addressed by Fuzzy IBE is to prevent users from collusion. This means that users should not be able to combine their keys in a way that the combined key encodes a comprehensive set of attributes, which neither of the colluding users holds alone. To achieve this, each secret key consists of components that encode an attribute each, and the components are "personalized" for the secret key holder by binding them to a user-dependent part.

Fuzzy IBE involves four operations: In the *Setup* operation, the trusted key center generates and distributes public system parameters. The *Key Generation* operation is carried out by the key center for each user. The operation yields a secret key that is bound to the user's attributes, which are managed by the key center.

For the *Encryption* operation, the encrypting user first has to specify a set of attributes, from which at least d have to be encoded by a secret key to enable decryption. This attribute set has to be larger than the system-wide threshold d. The public parameters that correspond to the attribute set have to be retrieved from the key center, and are part of the input of the encryption process.

For the *Decryption* operation, the decrypting user identifies d attributes that both her secret key and the ciphertext have in common. The secret key components that correspond to these attributes, together with the ciphertext, form the input for the decryption operation.

On the one hand, Fuzzy IBE shows some major limitations: a limited access control model, the necessity of a central key center that has to be fully trusted, the lack of revocation capabilities, and performance issues. On the other hand, one goal of this thesis is to identify an ABE scheme that offers E2E-SDS and supports named user groups, and to evaluate the real-world performance of this scheme. Thus, each of the limitations of Fuzzy ABE might prevent an ABE scheme from being used for E2E-SDS. For this reason, more recent ABE schemes that address these limitations are presented in the following.

The first limitation of Fuzzy IBE is the rather limited access control model, as Fuzzy IBE only offers a matching of plain attribute sets with a system-wide threshold. Possible extensions are CP-ABE [BSW07] and KP-ABE [LCLS08], which support access policies as Boolean terms over attributes. Many of CP-ABE or KP-ABE approaches support only *monotonic access structures*. In the case of CP-ABE, this means that given a secret key that comprises a superset of the attributes of another secret key, the former can be used to decrypt any ciphertext that is decryptable by the latter. In consequence, adding an attribute to a secret key always extends access permissions, and never limits access permissions. However, the limitation of monotonic access structure was overcome by some ABE approaches [OSW07].

ABE schemes generally implement Attribute-Based Access Control, whereas named user groups are a feature closely related to Role-Based Access Control. Thus, to implement SDS with named user groups based on an ABE scheme, a mapping between ABAC and RBAC is required. A few papers elaborate on this mapping, e.g., Zhu et al. [ZMHH13, ZHHW14] propose the specification of attribute hierarchies to be able to map role hierarchies to ABAC in a natural fashion.

Another limitation of Fuzzy IBE is that it relies on a central key center that has to be fully trusted. This limitation was addressed by many ABE schemes by distributing the key center functionality on multiple authorities, a property denoted as *multi-authority*. For example, an early approach proposed by Chase [Cha07] in 2007 enables to partition the attribute space, such that each partition is managed by another *attribute authority (AA)*. However, a central key center still generates the private key of every user and AA in the system. The rationale behind this architecture is that the key center generates the aforementioned user-dependent value that is required for collusion resistance. While, e.g., in [MKE08], a scheme is presented that supports more flexible access policies than [Cha07], a central key center is still required. In [LW11], a multi-authority ABE scheme was proposed that does no longer rely on a central key center. The main challenge addressed in this work is collusion prevention without a central party that generates the necessary user dependent values.

Furthermore, Fuzzy IBE does not offer any revocation capabilities. More recent ABE schemes propose different approaches and mechanisms to support revocation. Some schemes leverage an additional trusted party that holds a part of the necessary decryption key [XM12, IPN+09, YWRL10b, LLYY14]. This approach allows for immediate attribute revocation by adapting the access control list on the additional trusted party. Other schemes attach an expiration date to secret keys [BSW07], which consequently have to be renewed periodically. The revocation capabilities of ABE schemes further differ in granularity: While the schemes described so far support attribute revocation, i.e., a user is revoked from holding an attribute, some schemes only provide user revocation, where a user is excluded from the system completely [OSW07, LLLS11].

A final limitation of Fuzzy IBE are performance issues with regard to the computation time required for the typically expensive bilinear pairing operations, and the length of secret keys and ciphertexts. All of these performance metrics linearly depend on the number of attributes involved. For this reason, ABE schemes were proposed that work with constant computation costs and ciphertexts of constant size [DJ14, ZZC+15]. The performance of an ABE scheme can also be increased by outsourcing cryptographic operations to the storage provider[LDGW13, QDLM15].

For each major limitation of Fuzzy ABE, a multitude of ABE schemes was proposed that addresses this limitation. However, early schemes tended to address only one or two of the discussed limitations, while retaining all other limitations. More feature-rich ABE schemes that tried to combine an expressive access control model, multi-authority and revocation capabilities with acceptable performance only came up in recent years. Nonetheless, it is hard to identify ABE schemes that fulfill the whole set of requirements on E2E-SDS. In Section 5.4.1, a detailed evaluation of multi-authority ABE schemes with regard to their applicability for E2E-SDS is presented.

2.4 Performance Evaluation of SDS Protocols

In this section, existing work on the performance evaluation of SDS protocols will be discussed. An important influence factor for the performance is the sharing and usage behavior of the users, which have to be operationalized as sharing and usage model. Therefore, different types of sharing and usage models are presented. One of the presented models are workloads, which constitute a combined sharing and usage model that can serve as a basis for performance evaluation. Finally, the challenges that arise in the generation of synthetic workloads are discussed.

2.4.1 Performance Evaluation Methods

The performance of a technical system is always defined in terms of a *performance metric*. According to [Le 15], "a performance metric is a measurable quantity that precisely captures what we want to measure—it can take many forms. There is no general definition of a performance metric: it is system dependent, and its definition requires understanding the system and its users well." Performance metrics typically quantify system aspects such as the time behavior, the resource utilization, or the capacity of the system [ISO11b].

The process of quantifying the service delivered by a computer or communication system, i.e., to obtain certain performance metrics that describe the system, is referred to as *performance evaluation* (cf. [Le 15]). The purpose of a performance evaluation is typically "to compare design alternatives when building new systems, to tune parameter values of existing systems, and to assess capacity requirements when setting up systems for production use" [Fei14].

Basic methods to carry out performance evaluations are measurements, simulation, and analytical approaches. While measurements yield realistic and comparatively accurate results, they require that the sharing system is already in place, and that the client hardware under study is available. Furthermore, measurements do not provide any hints on the performance of the sharing system if the SDS protocol, the users' sharing behavior, or the hardware changes, thus, measurements lack generality. For these reasons, measurements are only used to estimate the performance of cryptographic primitives on a small set of client devices, but a measurement of the performance of the whole system is beyond the scope of this thesis.

Discrete Event Simulations (DES) are the most wide-spread method for performance evaluation [Le 15]. A DES "concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time" [Law06]. A simulation of a system usually requires less effort than performance measurements. The drawback of a simulation model is that major influence factors on the system performance usually cannot be derived from the model easily, but have to be identified in a systematic way, e.g., by means of a sensitivity analysis. A sensitivity analysis is carried out to assess "how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input" [STCR04]. In other words, a sensitivity analysis assesses how changes in the input of a model influence the output of the model.

Analytical performance evaluation methods are based on "a mathematical model of the system [that] is analyzed numerically" [Le 15]. These methods can be categorized into *empirical models* that are derived from analyzing the correspondence between system inputs and outputs in a black-box fashion, and *mechanistic models* that are inferred from at least a basic understanding of the system as a white box [Eec10]. Analytical approaches are typically much faster than simulations. Furthermore, an analytical performance evaluation model might make fundamental influence parameters of the system performance emergent, i.e., it might provide more fundamental insights. The drawback of analytical approaches is that the usual tendency for simplification comes with a loss in accuracy.

Analytical methods for performance evaluation were proposed that leverage established modeling formalisms, such as petri nets (cf. [Mur89]) or queuing networks [Kel76] etc. (cf. [BKR09]). The main reason for using such formalisms is that theorems exist that make one property of the model deductible from another property. However, using such a formalism requires to abstract the system under study to make it "fit" the formalism. Since this thesis mainly focuses on the real-world performance of SDS protocols, such formalisms were avoided in favor of more flexible and accurate analytical performance evaluation methods.

Performance evaluations for existing SDS protocols are usually related to the performance metrics *computation time*, *network traffic volume*, and *storage*. Existing evaluations were carried out almost exclusively based on two methods: either by describing the asymptotic behavior of some performance metrics with regard to a growing input parameter N, or by means of micro-benchmarks based on measurements of prototypical implementations.

An evaluation of the asymptotical behavior of some performance metrics is part of many publications introducing an SDS protocol (e.g., [HN11], [YWRL10a], [AHL+12], [LXZZ13], [DJ14]) . As already discussed, while this performance evaluation method indicates whether the protocol scales with a growing parameter N, it fails to provide hints on the real-world performance of the protocol in absolute terms. The reason for this is the very high abstraction level of asymptotic analyses, which by definition omit constant terms and factors in the calculation of the performance. A major challenge addressed in this thesis is to find an abstraction level for the performance evaluation that takes constant terms and factors into account, while abstracting from implementation issues or hardware.

Another restriction of the evaluation of the asymptotical performance is that the values which the parameter N takes in reality are usually unclear. The concrete parameter that is encoded as N depends on the context, but the parameter value is typically determined by the sharing and usage behavior of the users in the SDS system. A discussion of approaches to model and operationalize the sharing and usage behavior with regard to performance evaluation follows in the next subsection.

Micro-benchmarks were also leveraged to evaluate the performance of many SDS protocols (e.g., [BSW07], [HN11], [YJ14], [ZVH13], [KRS+03]). The purpose of micro-benchmarks is to evaluate the performance of a single sharing operation in isolation, where input parameters of the operation are systematically varied. An example is to carry out a micro-benchmark for a removing a member from a named group with varying group sizes. The results of micro-benchmarks are usually coupled to a single implementation and a single hardware configuration, thus, the results can hardly be generalized to other implementations and hardware. Furthermore, the results can only be interpreted if a realistic range for varying the input parameter can be estimated, which again depends on the sharing behavior of the users. Micro-benchmarks are a building block of the analytical performance evaluation method presented in Section 4.1.

This thesis focuses on the same performance metrics that are targeted in existing performance evaluations of SDS protocols, especially computation time and network traffic volume. However, performance metrics can be defined for system aspects in a much broader sense. For example, even the security of an organization can be quantified in terms of *security metrics* [CSS+08]. Within the IT security community, it is broad consent that security and performance are often conflicting goals, and solutions that are deployed in the real world have to balance these goals (cf. [Köh15]). In the context of SDS protocols, this trade-off becomes emergent in the choice of the length of cryptographic keys, where longer keys are generally more secure, but usually have a negative impact on the time required for the cryptographic operation in question. The objective of this thesis is to estimate the real-world performance of SDS protocols, which serves as an input for the process of balancing performance and security. However, the actual balancing process is beyond the scope of this thesis.

2.4.2 Sharing Models and Workloads

As illustrated in the previous sections on SDS protocols, the performance of an SDS protocol is heavily influenced by the users' sharing and usage behavior. For example, the number of users who are added to a named user group is a major impact factor for the performance of a subsequent group member removal, as the performance of this operation usually depends on the current size of the group. The usage behavior, i.e., the read and write operations on the shared data, should also be considered when evaluating the performance of an SDS protocol: First, the performance of a read operation is directly perceived by a user if the shared data should be accessed or consumed immediately after it was read. Second, the usage behavior also has an influence on the key graph, as a write operation typically involves the generation of keys or signatures. These changes on the key graph impact the performance of subsequent operations.

To enable an analytical or simulative performance evaluation, the sharing and the usage behavior have to be operationalized. For this purpose, the sharing behavior is formalized as a sharing model and the usage behavior as a usage model, respectively. In the following, first the sharing model and then the usage model will be discussed.

A *sharing model* is based on an authorization model. In the context of this thesis, the sharing model specifies the way the access control state of a system changes. For this purpose, the sharing model might involve a notion of time and bind each authorization operation to a certain point in time, or it might at least require a sequential ordering of operations. In existing literature, the sharing and the authorization model are often combined into a common "sharing model" (e.g., [TCN+14], [MA11]). However, for this thesis, the distinction between authorization and sharing model is important, as multiple sharing models corresponding to a single authorization model will be presented. A concrete instance of a sharing model will be referred to as *sharing scenario* in this thesis.

A very basic sharing model can be formalized as a sequence of authorization operations. More sharing models can be built by abstracting from this concrete sequence. The resulting sharing models can be roughly categorized into constructive and descriptive models: *Constructive models* focus on the behavior of single elements of the modeled system, and the resulting behavior of the system has to be derived by analysis or simulation. On the opposite, *descriptive models* directly describe properties of the system as a whole, without trying to reveal the factors that led to these properties [SF03]. In the context of sharing models, a constructive model rather describes the behavior of each single user, which in summary results in a certain behavior of the whole system. A descriptive model rather describes the properties of the authorization state, and the changes of the authorization state over time. Constructive models will be inspected in more detail in the next subsection.

As already discussed, a major challenge of the evaluation of the real-world performance is that realistic values for different sharing parameters are required. These values can be provided by descriptive sharing models. However, the descriptive sharing models presented in literature are comparatively abstract and high-level. For example, an analysis of the friendship relations within social networks (e.g., [LTH11]), together with the access control semantics of a friendship relation, can be considered as a sharing model. However, such a sharing model is relatively implicit, leaving open many degrees of freedom, such as the size of the shared data, the frequency of authorization operations and so on. More concrete sharing scenarios can be found, e.g., in [GMSW06], which states the fraction of each sharing operation type on the whole set of sharing operations. However, the sharing model that is implemented by those sharing scenarios is not specified explicitly, and it remains unclear whether these sharing scenarios are realistic.

The notion of a *usage model* widely varies in literature, depending on the system under study. For example, a usage model can describe the way graphical user interfaces are used [EWZC11], or the way hardware products are employed [Sim05], or the way users interact with component-based software in general [BKR07]. In the context of data sharing, a usage model specifies the way shared data is read and written over time. Similar to a sharing model, a usage model requires some basic model that defines the set of possible "usage operations", which could be as coarse-grained as *read* and *write*, or more fine-grained, such as *copy* or *append*. Again, a basic usage model is constituted by a sequence of read and write operations. A discussion of constructive usage models is deferred to the next subsection.

The descriptive usage models that were proposed so far with regard to data sharing cover different abstraction levels. For example, rather abstract usage models describe the usage behavior regarding popular online content on a large scale [SH10, CKR+07]. Comparatively concrete usage models were presented to characterize the access patterns for file systems [LPGM08, HH95, And00, HH96]. In [LPGM08], for example, traces of large-scale file systems running in production were characterized with regard to the data sizes that are read and written, the frequency of different file operations and so on. Moreover, the SPEC benchmark SFS2014 [SPE14] provides standardized sequence of file access operations that can be employed to compare the performance of different file systems.

In this work, we carry out performance evaluations based on combined sharing and usage models. The basic combined model is represented by a sequence of sharing and usage operations, which is denoted as *workload* in this thesis. Note that this is a relatively narrow and concrete definition. In literature, often more general definitions can be found, such as "The [work]load characterizes the quantity and the nature of requests submitted to the system." [Le 15].

2.4.3 Workload Generation

Realistic workloads as a basis for the performance evaluation of an SDS protocol can be obtained in two ways: Either the workloads are created from traces of the sharing system under study, or they are synthetically generated. It can be argued that while trace-based workloads exactly represent at least one real-world sharing or usage scenario, they may cover only a small sample of the space of interesting scenarios, and their internal representation is comparatively verbose [Fei14]. Besides these advantages and disadvantages, the generation of workload constitutes the only option to obtain workloads if the system under study does not exist yet, or if the performance of the system should be evaluated with regard to a changed sharing and usage behavior.

Synthetic workloads can be generated by means of constructive sharing and usage models. Constructive models strive to model the process that leads to a certain workload, rather than properties of the workload itself [Fei14]. Building such a model usually involves the analysis of existing real-world traces, and fitting assumed probability distributions to the traced behavior of the system elements.

In literature, no constructive sharing models could be found that are sufficiently concrete to serve as a basis for workloads generation. Some work exists that is driven by an economical view on users, where sharing decisions result in a monetary gain or loss (e.g., [PM01], [LvFS14]). However, these sharing models are again very abstract, leaving open many degrees of freedom, and therefore cannot be used for workload generation without further refinement.

In contrast to constructive sharing models, concrete constructive usage models were proposed in the context of data access. As an example, [And00] presents a method to generate workloads for NFS (cf. Section 2.1). In this method, the workload generation process is separated into two phases: In the initial *creation phase*, a directory tree is generated, according to a set of probability distributions that determines, e.g., the number of files and directories within a directory, the maximum directory depth within the directory tree, and the popularity and the size of the files. In the subsequent *measurement phase*, NFS operations are generated according to probability distributions for operation types and read and write sizes, and a load level that specifies the overall operation rate. Some limitations of this method are that bursts of operations cannot be modeled, as the inter-arrival time of operations is fixed to be uniformly distributed, and that dependencies between operations cannot be modeled either.

The generation of workloads that combine sharing and usage behavior would require both a concrete constructive sharing and a concrete constructive usage model. It was just shown that while constructive usage models were proposed in literature, no concrete constructive sharing models could be found. However, the mere identification or definition of a constructive sharing model would fall short of fulfilling an important requirement addressed in this thesis: The resulting workload should adhere to some parameters that are estimable, i.e., a domain expert should be able to provide an estimation for these parameters regarding the considered sharing system. However, it can be assumed that parameters describing some properties of the whole system tend to be estimated more easily than parameters describing the behavior of single users. This assumption is supported by the observation that descriptive parameters can be extracted, e.g., from a series of snapshots of the authorization state. On the other hand, the extraction of constructive parameters tends to require a whole "stream" of user sharing operations. Besides the technical effort that is necessary to capture and persist this stream of operations, privacy issues can arise with this kind of user behavior tracking.

In summary, a requirement on a workload generation method that can be employed for the purpose of this thesis—evaluating the performance of an existing or envisioned sharing system—is that the resulting workload adheres to a descriptive sharing model. As already mentioned, it is not trivial to build constructive sharing or usage models in a way that the resulting workload adheres to some descriptive sharing or usage model. For this reason, one challenge addressed in this thesis is to design a non-constructive workload generation method that leads to predefined instances of a descriptive sharing model.

In this thesis, a method is presented that allows to generate workloads that adhere to a set of predefined descriptive sharing parameters, such as the distribution of the size of named user groups. The method is discussed in detail in Section 3.6. Similar to [And00], the workload generation involves two phases: In the first phase, an initial authorization state is built, which is incrementally altered within the second phase. As the first phase boils down to graph generation, a short discussion of graph generation methods related to our setting concludes this section.

According to [CZF04], graph generation methods can be categorized as either degree-based or procedural. Degree-based methods strive to generate a graph in a way that the degree distribution of the resulting graph matches some target distribution. On the opposite, procedural methods focus on simple per-node or per-edge generation rules. An example for the latter is the Barabasi-Albert method [AB00]. This method incrementally inserts edges in an existing set of nodes, where the probability that the edge is appended to a given node increases with the number of edges that are already appended to this node. Procedural graph generation cannot be applied for the workload generation as required in this work. The reasons for this were already explained above in the context of constructive models.

A famous descriptive graph generation method is the *configuration model* [Bol80]. This method involves two steps: In the first step, the set of nodes is created, and each node is assigned a target node degree, according to a target probability distribution. In the second step, edges are inserted by randomly selecting two nodes, and checking whether the edge can be appended to these nodes without violating the target degree of each node.

The configuration model cannot be applied directly in the context of workload generation for data sharing, because there are certain requirements on the graph that has to be generated: Both nodes and edges are labeled, and no edges are allowed between nodes with the same label. With some minor simplifications, the generated graph can be considered a tripartite graph, i.e., a graph comprising three classes of nodes, where nodes within one class are never connected by an edge. While generating graphs with a given distribution of node degrees was broadly covered in research (cf. [MKI+04]), no graph generation method could be found in literature that focuses on the generation of tripartite graphs.

2.5 Conclusions

A variety of authorization models exists that can be employed for data sharing. This comprises well-known models, such as Discretionary, Mandatory, Role-Based or Attribute-Based Access Control, but also more recent models, such as Group-Centric Secure Information Sharing. Many common data sharing systems offer the possibility to grant access rights to named user groups, which do not exist only in the context of permissions on a set of resources, but are independent entities that can be granted data access via a group handle. This applies, e.g., to the widely deployed distributed file systems NFSv4 and CIFS.

A plethora of SDS protocols were proposed in the last decades, targeting different security objectives like confidentiality, integrity, privacy or accountability. This thesis focuses on E2E-SDS protocols, which are defined as a special class of SDS protocols with essentially two properties: E2E-SDS protocols ensure confidentiality and integrity of the shared data by means of client-side cryptography, and they require that authorization and access control for read accesses is carried out solely by the users who are allowed to access the data. Therefore, an E2E-SDS protocol does not allow for a “trusted entity” that takes part in authorization or access control of read accesses.

As only few SDS protocols comply with this strict definition of E2E-SDS, the combination of E2E and support for named user groups is rarely offered by existing SDS protocols. Cepheus was identified as only E2E-SDS protocol that supports named user groups. For this purpose, Cepheus leverages joint authorization operations, which means that a single authorization operation is carried out by at least two different users in a collaborative manner. The drawback of joint authorization operations is that the moment the operation is initiated, all involved users have to be reachable within the network, and be ready to carry out the required cryptographic operations immediately. Both aspects can be problematic especially with regard to mobile devices, as the reachability can be limited in many situations, and the necessary cryptographic computations might have an impact on the responsibility of the device. An E2E-SDS protocol that supports named user groups while omitting joint authorization operations will be presented in Chapter 6.

In recent years, lots of SDS protocols were proposed that leverage Attribute-Based Encryption (ABE). ABE-based protocols have the advantage that a single ciphertext is sufficient to target a potentially large number of recipients, which might be advantageous in terms of performance. However, it is challenging to identify ABE schemes that allow for E2E-SDS, and at the same time provide an authorization model expressive enough to be adaptable for a support of named user groups.

The performance of SDS protocols is usually evaluated by means of asymptotical analysis, i.e., the behavior of the performance is described with regard to a growing parameter N. This performance evaluation method intentionally abstracts from constant factors and terms. However, these terms can be highly relevant for the real-world performance of the protocol in absolute terms. The challenge in constructing a more concrete performance evaluation model lies in finding a unified abstraction layer that allows to model performance-relevant differences between protocols while abstracting from implementation details. This challenge is addressed by the performance evaluation models presented in Chapter 4.

Furthermore, the asymptotical performance analyses typically provide no hints on the values that parameter N might take in reality. The concrete values depend on the sharing and usage behavior of the users, which have to be operationalized as sharing and usage models. If the sharing system under study exists, instances of sharing and usage models can be built based on traces of the sharing and usage operations. For this thesis, this was done with regard to a special sharing and usage model that is referred to as *workload*. In this context, a workload represents a plain sequence of sharing and usage operations.

In other cases, the sharing system under study might not exist yet, or the impact of future changes in the sharing or usage behavior of the users should be assessed. In these cases, synthetic model instances must be generated. These synthetic instances can be built based on an expert’s estimations for a certain subset of the sharing and usage parameters. The challenge addressed in this thesis is that the estimable parameters are typically related to a *descriptive* sharing or usage model. On the other hand, the instance generation methods proposed in literature rely on *constructive* sharing or usage models in most cases. Constructive models can hardly be designed in a way to yield a given descriptive sharing or usage model instance. For this reason, a generation method that targets a descriptive model was designed for this thesis, and is introduced in Chapter 3.

3 SDS Workload Modeling and Generation

As discussed in the previous chapter, the sharing and usage model is an important factor for the performance of SDS protocols. Thus, to evaluate the performance of an SDS protocol, both the sharing and the usage model have to be incorporated into the performance evaluation method. In this chapter, we describe *workloads* as the combined sharing and usage model that is used in this thesis.

Workloads can informally be seen as a sequence of sharing and usage operations. The set of possible sharing operations is determined by the authorization model implemented by the SDS protocol. Section 3.1 provides a definition of the authorization model that is considered in this thesis. Based on the authorization model, a definition of workloads is given in Section 3.2.

The performance evaluations of concrete SDS protocols that are described in this thesis are mainly based on real-world workloads. To generate those workloads, two sharing services running in production at Karlsruhe Institute of Technology (KIT) were monitored for changes in the permission structures. While the resulting real-world workloads are considered exemplary, they give an indication of the performance of an SDS protocol in reality. In Section 3.3, it is shown how these workloads were generated from traces.

In the last three sections of the chapter, the abstraction of workloads to a set of workload parameters is discussed. This abstraction becomes necessary, as fully specified workloads require very detailed knowledge about the sharing and usage scenarios in the system under study, which will be available only rarely in practice. In Section 3.4, a set of possible workload parameters is given. This set was designed with the contradicting goals of being comprehensive enough to specify a workload to a sufficient degree, while containing only parameters that are known or at least predictable by an SDS provider. The real-world workloads are characterized with regard to this set of workload parameters in Section 3.5.

In Section 3.6, a method for the generation of workloads based on predefined workload parameters is presented. Workload generation is necessary if the performance evaluation method requires a fully specified workload, but the workload cannot be retrieved from the system under study. Some instance of the related optimization problem are NP-hard, which is shown in the second part of the section.

Parts of this chapter have already been published before in [KH15].

3.1 Authorization Model

As stated before, in this work, we consider SDS protocols that implement an authorization model that is similar to those of NFSv4 or CIFS (cf. Chapter 2). The subjects of this authorization model are referred to as *users*. These can be natural persons, as well as organizations or processes on machines. The set of all users in the system is described by U.

The objects of the authorization model are referred to as *resources*. These are the smallest data chunk that users can share independently from other data chunks. Depending on the application context, resources may represent, e.g., files or database entries. The set of all resources that access can be granted to is referred to as R. Each resource is assigned to exactly one user, referred to as *resource owner*, that grants and revokes access permissions on this resource. The permissions on a resource either allow read access only, or allow read access as well as write access. A resource owner can manage an arbitrary number of resources.

The resource owner may grant resource access to another user explicitly—referred to as *direct permissions*—or to a *named user group*. A named user group—or simply *group* for the remainder of this thesis—is a set of users that is managed by exactly one user, referred to as *group owner*, and each resource owner might grant access to her resources for arbitrary groups. A group owner can manage an arbitrary number of groups.

The authorization model supports resource and group hierarchies. The resource hierarchy feature allows to arrange resources within a set of resource trees that models inheritance of permissions, i.e., any user that has access to a resource has implicitly access to all descendants of this resource in the respective resource tree. Group hierarchies allow one group to become a member of another group, with the result that all members of the former group are implicitly also members of the latter group.

The authorization model combines aspects of Discretionary Access Control, in which the access permissions of each user or each resource are listed explicitly, and of Role-Based Access Control, where access rights are given to users implicitly by via roles, which are considered equal to named groups in the context of this work (cf. Section 2.1).

An instance of the authorization model is formalized as authorization state, which is defined as follows:

Definition 3. An *authorization state* is completely described by the set of users U, the set of resources R, the set of groups G, the set of permissions $P = \{Read, ReadWrite\}$, and the following relations on these sets:

- The membership relation $\mathit{MemberUser}(U, G)$, which indicates that a user $u \in U$ is a member of a group $g \in G$.
- The group hierarchy relation $\mathit{MemberGroup}(G, G)$, which indicates that a group $g_1 \in G$ is a member of another group $g_2 \in G$. The group hierarchy relation is asymmetric.
- The access relation $\mathit{HasAccess}(U \cup G, R, P)$, which indicates that a user $u \in U$ or a group $g \in G$ has either read access or read and write access on resource $r \in R$. Read and write access implies read access: $\mathit{HasAccess}(u, r, \mathit{ReadWrite}) \Rightarrow \mathit{HasAccess}(u, r, \mathit{Read})$
- The child relation $\mathit{ChildResource}(R, R)$, which marks a resource $r_{child} \in R$ as child of another resource $r_{parent} \in R$. The child relation is a forest.

This definition implies that a user is allowed to read a resource if the user himself or a group the user is member of has read access on the resource itself or on an ancestor of the resource in the resource hierarchy. The same applies for write access.

The authorization state can be represented as *authorization graph*, which is constructed from the authorization state users, resources and groups as nodes, and the authorization state relations between these entities as edges:

Definition 4. An *authorization graph* is a directed graph (V,E) with V = $U \cup R \cup G$. An edge between nodes V_1 and V_2 exists iff (V_1, V_2) is an element of $\mathit{MemberUser}(U, G)$, $\mathit{MemberGroup}(G, G)$, $\mathit{HasAccess}(U \cup G, R, P)$ or $\mathit{ChildResource}(R, R)$.

The authorization model shows many similarities with those of NFSv4 and CIFS (cf. Section 2.1). A central feature of the authorization model is the support for named user groups. For this reason, the authorization model is referred to as *group-based authorization model* in this thesis. In the following, the group-based authorization model will be used as a basis for the definition of a workload.

3.2 Definition of Workloads

Workloads are the combined sharing and usage model that is used in this thesis. A workload is essentially a sequence of changes in the authorization state, e.g., the owner of a resource grants read access on one of her resources to a certain user. These changes in the authorization state constitute the input of a secure data sharing protocol, together with commands to upload resources to or download them from the storage provider.

We refer to changes of the authorization state and to upload and download commands as *workload events*. Each event has one or more parameters, with a mandatory parameter *actor* that states the user that actually carries out the cryptographic operations required to enforce the authorization state change. Thus, a workload is defined as follows:

Definition 5. A *workload* is a finite sequence of changes to the authorization state, together with upload and download commands, which are referred to as workload events. A workload event has one of the following types:

- *AddUser*(u_{actor}, u): adds a user u to the set of users U.
- *RemoveUser*(u_{actor}, u): removes user u from the set of users U.
- *AddGroup*(u_{actor}, g): adds a group g to the set of groups G.
- *RemoveGroup*(u_{actor}, g): removes group g from the set of groups G.
- *AddResource*(u_{actor}, r, r_{parent}): adds a resource r to the set of resources R and adds (r, r_{parent}) to relation $Child$.
- *RemoveResource*(u_{actor}, r): removes resource r from the set of resources R.
- *AddUserToGroup*(u_{actor}, u, g): adds user u to group g by adding (u, g) to relation *MemberUser*.
- *AddGroupToGroup*(u_{actor}, g_1, g_2): adds group g_1 to group g_2 by adding $(g1, g_2)$ to relation *MemberGroup*.
- *RemoveUserFromGroup*(u_{actor}, u, g): removes user u from group g by removing (u, g) from *MemberUser*.
- *RemoveGroupFromGroup*(u_{actor}, g_1, g_2): removes group g_1 from group g_2 by removing $(g1, g_2)$ from *MemberGroup*.
- *GrantUserReadAccess*(u_{actor}, u, r): grant read access for user u on resource r (and all child resources of r) by adding $(u, r, Read)$ to *HasAccess*.
- *GrantGroupReadAccess*(u_{actor}, g, r): grant read access for group g on resource r (and all child resources of r) by adding $(g, r, Read)$ to *HasAccess*.
- *GrantReadWriteAccess*(u_{actor}, u, r): grant read and write access for user u on resource r (and all child resources of r) by adding $(u, r, ReadWrite)$ to *HasAccess*.
- *GrantGroupReadWriteAccess*(u_{actor}, g, r): grant read and write access for group g on resource r (and all child resources of r) by adding $(g, r, ReadWrite)$ to *HasAccess*.
- *RevokeUserReadAccess*(u_{actor}, u, r): revokes read access for user u on resource r (and all child resources of r) by removing $(u, r, Read)$ from *HasAccess*.
- *RevokeGroupReadAccess*(u_{actor}, u, r): revokes read access for group g on resource r (and all child resources of r) by removing $(g, r, Read)$ from *HasAccess*.
- *RevokeUserReadWriteAccess*(u_{actor}, u, r): revokes read and write access for user u on resource r (and all child resources of r) by removing $(u, r, ReadWrite)$ from *HasAccess*.

- *RevokeGroupReadWriteAccess(u_{actor}, u, r)*: revokes read and write access for group g on resource r (and all child resources of r) by removing $(g, r, \mathit{ReadWrite})$ from *HasAccess*.
- *WriteData($u_{actor}, r, dataSize$)*: data for resource r is uploaded, the size of the data is given by *dataSize*.
- *ReadData($u_{actor}, r, dataSize$)*: data for resource r is downloaded, the size of the data is given by *dataSize*.

The workload events that change the authorization state, i.e., all workload events except for *WriteData* and *ReadData*, are referred to as *authorization operations*.

Note that a workload is not bound to some notion of time, in a sense that a workload event carries a timestamp, or that a time difference between consecutive events could be specified.

The fact that the definition of a workload is based on the authorization state sets and relations implies that a workload has to adhere to some consistency rules. For example, a user can only be removed from the set of users if she was added before, i.e., the event RemoveUser(u_{actor}, u) may only occur if the event AddUser(u_{actor}, u) has occurred before.

Some further consistency rules are imposed that ensure that read and write events are always successful, i.e., all read and write events are allowed from an access control point of view. This means that unsuccessful or even malicious read or write events and their costs cannot be modeled as part of a workload as defined above.

Note that there is no distinct workload event type to „downgrade“ a user from read and write access to read-only access. This event has to be modeled by a *RevokeUserReadWriteAccess* or *RevokeGroupReadWriteAccess* event, and a subsequent *GrantUserReadAccess* or *GrantGroupReadAccess* event.

There are two ways to get workloads that adhere to the definition given above: Either the workloads are retrieved from real world systems that enable data sharing in some way, or the workloads are generated, based on some predefined parameters. For this thesis, both methods have been implemented. In the remainder of this chapter, first the generation of workloads from traces of real-world systems and the characteristics of these workloads will be discussed. Thereafter, a generation method for synthetic workloads will be presented.

3.3 Generation of Real-World Workloads from Traces

Workloads retrieved from real-world data sharing systems allow to get an indication whether the performance of certain SDS protocols allows to employ them in reality. For this thesis, workloads from two systems running in production at KIT were retrieved, namely from an e-learning platform based on ILIAS[1] and from a groupware platform based on Microsoft SharePoint Server 2013[2]. These workload are referred

[1]http://www.ilias.de/

[2]https://products.office.com/en-us/SharePoint/collaboration

to as *real-world workloads* for the rest of the thesis. In this section, the process of generating workloads from traces of real-world systems is described. An overview of the generation process is shown in Figure 3.1. A characterization of the retrieved workloads is given in Section 3.5.

To generate real-world workloads, daily snapshots of the authorization structure of both of these platforms were taken. As the groupware allows to grant permissions to groups defined in Microsoft's directory service implementation named Active Directory (AD)[3], daily snapshots were also taken of the KIT AD installation. Essentially, taking a snapshot provided the current authorization state of the three system, and by comparing a snapshot to the snapshot taken the day before, a set of workload events could be calculated as "diff" between the two snapshots. The snapshot diffs were collected from the e-learning platform for 118 days, and from the groupware platform and the directory service for 91 days in summer 2014.

The initial snapshot was compared with an "empty" snapshot, so an *AddUser* event was created for each initial user, an *AddResource* event was created for each initial resource and so on. As this approach only yields daily granularity, the events occurring within a day have to be ordered, and this is done based on the event type: First, "constructive events" such as *AddUser*, *AddUserToGroup* etc. are added, then *WriteData* events, and finally "destructive" events. This ensures workload consistency, as, e.g., for user u and resource r, first executing *GrantUserReadAccess*(u_{actor}, u, r) and then *WriteData*$(u_{actor}, r, dataSize)$ ensures that at the time of write access, the user has write permissions.

A limitation of this workload generation approach is that write events can only be detected by comparing the "last modified" timestamp of each resource, and assuming a write event whenever this timestamp changes between two consecutive snapshots. This way, if multiple write events occur between two consecutive snapshots, only the last one can be detected and added to the workload. The same limitation applies for, e.g., the repeated addition of a certain user to a certain group between two consecutive snapshots, but it is assumed that this case will only occur very rarely in practice.

The generation of workloads from traces might bear some technical challenges in practice. For example, the ILIAS-based e-learning platform supports resource hierarchies with inheritance of permissions; however, the permissions granted on a resource are propagated to all descendant resources at the time of the permission change. This way, the new permission is listed with each resource in the whole resource subtree that is rooted in the resource the permission was granted on. Thus, a processing step was introduced that "pushes the permissions up the resource tree" as far as possible, based on the assumption that the authorizing user issued only one permission grant for the whole resource subtree. More technically, this means that the whole resource tree is traversed from the leaves to the root. Whenever a whole set of sibling resources has an identical permission, this permission is moved to the parent resource instead.

As another example, to generate workloads from traces of the SharePoint-based groupware platform, a first, single-threaded version of the generation tool needed

[3]`https://technet.microsoft.com/en-us/library/dd448614.aspx`

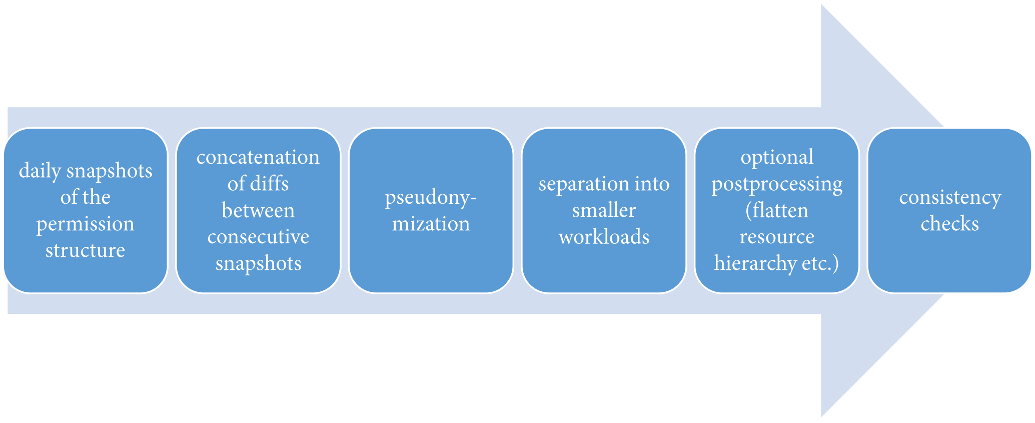

daily snapshots of the permission structure
concatenation of diffs between consecutive snapshots
pseudony-mization
separation into smaller workloads
optional postprocessing (flatten resource hierarchy etc.)
consistency checks

Figure 3.1: Overview of the process to generate real-world workloads from traces of real-world systems.

more than one day to generate one snapshot, and therefore would have decreased the system performance during working hours. A multi-instance version of the tool solved this issue, but technically all instances had to concurrently access the same user and group database, requiring a carefully designed database transaction management.

In the course of the workload generation process, all user, group and resource identifiers are pseudonymized by replacing them with a salted hash of the original identifier.

The result of these workload generation steps is one huge "raw workload" per platform. This workload had to be broken down into smaller workloads that represent a certain logically connected part of the system. For this purpose, the raw workload was partitioned along the lines of projects in the case of the groupware and of faculties in the case of the e-learning platform. For all resources belonging to a certain part of the system, all permitted users and groups were retrieved, and workload events regarding one of these entities were compiled to a new workload. For the remainder of this work, the term *real-world workloads* refers to these pre-processed, smaller workloads.

Further processing steps included, e.g., removing cycles in the group hierarchy from the directory service raw workload, and combining the cleaned workload with the respective groupware workloads. In addition, some SDS protocols are not able to deal with the full workload feature set as stated in the definition in Section 3.2. For example, some SDS protocols do not support resource hierarchies, and some do not support group hierarchies. For this reason, there are optional processing steps that, e.g., "flatten" a resource hierarchy to only contain the leaf resources, while preserving the effective permissions on these resources. In a similar way, another optional processing step allows to replace group hierarchies by effective group memberships, i.e., the transitive closure of all group membership relations is calculated, and all entries in the *MemberUser* relation are retained, while all entries in the *MemberGroup* relation are discarded.

As final processing step, some consistency checks are performed on the resulting workloads, e.g., it is checked whether a resource is added before access is granted on the resource and so on (cf. Section 3.2).

3.4 Workload Parameterization

The definition of workload as given in Section 3.2 requires a very detailed knowledge about the sharing and usage scenario in the system under study. In practice, such detailed knowledge will be available only very rarely: The system under study might not even exist yet, and if it exists, the tracing of the systems requires some effort, as shown in the previous section. Furthermore, the performance prediction with consideration of an assumed future sharing or usage scenario is also an important use case for a performance evaluation.

While in practice, a fully specified workload might be unavailable, there might still exist some knowledge about certain characteristics of the workload. These workload characteristics are referred to as *workload parameters* for the rest of this thesis. An example for a workload parameter is the average size of a user group. The set of parameters that can be derived from a workload is huge. Workload parameters may describe an authorization state at a given time, as well as changes or change rates over time. The parameters may be sampled each time an authorization event occurs, or only when an event of a certain type occurs. They may refer to recursive structures in an authorization state, such as the child relation $Child(R, R)$. They may be frequency distributions, e.g., the number of groups of a given size, or they may be related statistical measures of frequency distributions, such as average or variance.

The performance evaluation methods presented in this thesis are based on the assumption that the provider of an SDS system knows some parameters of the expected workload, or is at least able to give a range that the parameter values probably lie within. It is obvious that the usability of a performance evaluation model depends on whether the input parameters are known or can be predicted within a tight range. Thus, the goal of both the workload characterization in Section 3.5 and the workload generation method presented in Section 3.6 is to focus on parameters that are likely to be known or predictable. Unfortunately, it is hard to determine in general whether a parameter is likely to be known, or even if one parameter is more likely to be known than another. As a rough guideline, in this work, workload parameters are considered more likely to be known if they can be derived by processing only a subset of the workload events. For example, the frequency distribution of the group sizes can be calculated by only considering *AddUserToGroup* and *RemoveUserFromGroup* events.

In this section, a set of workload parameters is described that builds the basis for both workload characterization and generation (cf. Section 3.5 and Section 3.6). Especially for workload generation, the set of considered workload parameters has to be comprehensive enough to specify the workload in sufficient detail. Thus, the set of workload parameters is considered a compromise between the chance that the parameters in the set are known, and the comprehensibility of the workload description.

parameter name	parameter type	transitive?
number of users	scalar	no
number of resources	scalar	no
number of groups	scalar	no
groups per user	frequency distribution	no
users per group	frequency distribution	no
group members per group	frequency distribution	no
group memberships per group	frequency distribution	no
resources per group	frequency distribution	no
groups per resource	frequency distribution	no
direct resources per user	frequency distribution	no
direct users per resource	frequency distribution	no
group resources per user	frequency distribution	yes
group users per resource	frequency distribution	yes

Table 3.1: List of static workload parameters

The parameter set is divided into two parts: static and dynamic parameters. The intuition behind this is that a workload can be described by a combination of an initial state, or more exact, and initial authorization graph, and iterative changes to this initial authorization graph. The static parameters reflect the properties of the initial graph, while the dynamic parameters reflect the graph changes. The workload parameterization and the parameter dependency groups described in this section were assembled as part of a Bachelor's thesis [Lei16].

3.4.1 Static Workload Parameters

Static workload parameters describe the number and the degree of the nodes in the authorization graph. While the number of nodes is described by scalar parameters, the degrees of the nodes are described by frequency distributions. The static workload parameters are listed in Table 3.1.

The static parameters can be further divided into direct and transitive parameters. This distinction will be important for the feasibility of the workload generation method introduced in Section 3.6. Direct parameters reflect the node degrees of the different authorization graph node and edge types. For example, the direct parameter *resources per group* reflects the number of outgoing *HasAccess* edges of the group nodes in the authorization graph. Transitive parameters, however, are related to transitive edges that are not directly part of the definition of an authorization graph instance. Instead, transitive edges are redundant "overlay" edges that are created by connecting each user node to each resource the user has access to because of a group membership. Thus, the transitive outgoing edges of a user represent the effective permissions of a user that are determined by the user's group memberships, and the permissions of these groups.

	parameter name	**transitive?**
users	users added	no
	users removed	no
groups	groups added	no
	groups removed	no
resources	resources added	no
	resources removed	no
users and groups	groups per user added	no
	groups per user removed	no
	users per group added	no
	users per group removed	no
users and direct resources	direct resources per user added	no
	direct resources per user removed	no
	direct users per resource added	no
	direct users per resource removed	no
groups and resources	resources per group added	no
	resources per group removed	no
	groups per resource added	no
	groups per resource removed	no
users and group resources	group resources per user added	yes
	group resources per user removed	yes
	group users per resource added	yes
	group users per resource removed	yes
group hierarchy	group members per group added	no
	group members per group removed	no
	group memberships per group added	no
	group memberships per group removed	no
read and write events	write actions per user	no
	written data size per write action	no
	read actions per user	no

Table 3.2: List of dynamic workload parameters

3.4.2 Dynamic Workload Parameters

Dynamic workload parameters describe the probability that certain changes occur from one workload revision to another. For each static parameter, there exist two related dynamic parameters: one for the addition of nodes or increase of the node degree, and one for the removal of nodes or decrease of the node degree. All dynamic parameters are expressed by frequency distributions. The dynamic workload parameters are listed in Table 3.2.

3.4.3 Workload Parameter Dependency Groups

Workload parameters depend on each other, i.e., the value of one parameter may be completely determined or at least bound by a combination of other parameters. This means that an assignment of values for dependent parameters that does not comply with these dependencies leads to conflicting requirements on the resulting workload. In this case, no workload can be generated that adheres to all predefined parameter assignments. For this reason, it is important to take account of workload parameter dependencies as part of the workload generation process.

The sets of dependent workloads can be formalized as workload parameter dependency groups:

Definition 6. A workload parameter set is a *workload parameter dependency group*, or *dependency group* for short, if there is at least one parameter assignment that can be fulfilled in each strict subset of the parameter set, but that cannot be fulfilled in the whole parameter set.

Note that this definition contains a minimality condition by requiring that the "unfulfillable" parameter assignment can be fulfilled in each strict subset. This minimality condition prevents that each parameter set that contains a dependency group is a dependency group itself. To prove this minimality condition for a given parameter set with size n, it is enough to prove that the assignment can be fulfilled in each subset with size $n-1$, as this implies that the assignment can be fulfilled in any smaller subset.

In this work, dependency groups between static parameters are introduced and described. For this purpose, three types of dependency groups were identified: upper bounds, bipartite-graph induced dependency groups, and transitive dependency groups. A complete overview of the identified dependency groups can be found in Table 3.3.

Upper bounds are the simplest and most obvious type of dependency groups. An upper bound simply states that a scalar parameter works as an upper bound for the maximum of a frequency distribution parameter. An example for this is the frequency distribution *groups per resource*, whose maximum value is bound by the total number of group nodes in the graph.

Bipartite graph induced dependency groups can be identified wherever a part of the authorization graph, i.e., an authorization graph comprising only a subset of node and edge types, forms a bipartite graph. For example, the set of user and group nodes,

	parameter set	**type**
1	number of groups, groups per user	upper bound
2	number of groups, groups per resource	upper bound
3	number of users, users per group	upper bound
4	number of users, direct users per resource	upper bound
5	number of resources, resources per group	upper bound
6	number of resources, direct resources per user	upper bound
7	number of resources, group resources per user	upper bound
8	number of users, group users per resource	upper bound
9	number of groups, number of users, users per group, groups per user	bipartite graph
10	number of users, number of resources, direct resources per user, direct users per resource	bipartite graph
11	number of groups, number of resources, groups per resource, resources per group	bipartite graph
12	number of users, number of resources, group users per resource, group resources per user	bipartite graph
13	groups per user, resources per group, group resources per user, groups per resource	transitive
14	groups per resource, users per group, group users per resource, groups per user	transitive

Table 3.3: List of workload parameter dependency groups

together with all edges related to the *MemberUser* relation, can be considered as bipartite graph. This induces dependency group 9 (cf. Table 3.3) that consists of the parameters *number of groups*, *number of users*, *users per group* and *groups per user*. In this case, the dependency group property can be proven by the unfulfillable assignment: *number of groups* = 2, *number of users* = 2, *users per group* = 2 for each group, and *groups per user* equals 2 for one group and 1 for the other group. As the sum of *users per group* for all group nodes differs from the sum of *groups per user* for all user nodes, there is no graph that can fulfill this assignment. However, it can be fulfilled if one of the parameters is omitted.

Transitive dependency groups reflect the dependency between one transitive and three non transitive parameters. As an example, the transitive dependency group 13 consists of the parameters *groups per user*, *resources per group*, *group resources per user* and *groups per resource*. The intuition behind this dependency is that the transitive parameter *group resources* of a certain user can be calculated by iterating over this user's groups, and summing up the number of accessible resources of each of these groups. However, this calculation only yields exact results if *groups per resource* = 1 for every resource in the graph. If *groups per resource* > 1 for a resource, e.g., *groups per resource* = 2, this means that this resource is accessible by two groups. If a user is member of both of these groups, the calculation of her group resources would count this resource twice, and therefore yield a wrong result. For this reason, *groups per resource* also must be a part of the dependency group.

3.5 Characterization of Real-World Workloads

In this section, the real-world workloads that were retrieved according to the process described in Section 3.3 are characterized. The characterization focuses on the workload parameter set introduced in Section 3.4, but also considers some other parameters.

The real-world workloads that are characterized in this section are exemplary. Thus, it is not assumed that these workloads are representative for any kind of data sharing application. The first reason to present the workload characteristics anyhow is that it can provide some guidance if workload parameters are missing. An IT operator who has to estimate the performance of an SDS system may know some of the workload parameters, but not all that are necessary for the performance evaluation method. It is also possible the IT operator can estimate some of the parameters in average, but cannot estimate the shape of the underlying probability distributions. To strengthen the guidance character of this section, correlations between different (scalar) workload parameters are analyzed, as this may allow for a better estimation of missing parameters. Again, the correlations are only presented with regard to the exemplary real-world parameters. The second reason for presenting a characterization of the real-world workloads is that these workloads form the input for the performance evaluation of concrete SDS protocols, which is described in Chapter 5 and Chapter 6.

The workload characterization in this section consists of two parts. In the first part, all of the e-learning workloads and all of the groupware workloads are considered as one large workload each, and the parameters of these large workloads are analyzed and presented. This part starts with an analysis of group sizes and group permissions, and their evolution over time. Thereafter, the frequency and size of file uploads is characterized. The part is concluded with a presentation of resource permissions and the frequency of permission denials. In the second part of this section, correlations between scalar workload parameters are analyzed.

The characterized workload data base consists of 11 workloads, each reflects a faculty of our university on an e-learning platform, and of 76 workloads that reflect a certain team or project on a groupware platform. In the following, when distinguishing between e-learning and groupware workloads, the value for groupware workloads appears in parentheses.

The workloads comprise 10 920 (751) groups in total, reaching from zero-member-groups to groups with 37 018 users. A group has 36 members on average. The distribution of group sizes in buckets of ten is shown in Figure 3.2.

To make statements about magnitudes of group sizes, each group is categorized into one of four classes, based on group size: *small* groups comprise up to 20 members, *medium* groups more than 20 and up to 200 members, *large* groups more than 200 and up to 2000 members, and *extra large* groups more than 2000 members.

Considering all groups in all workloads, a vast majority of 85.6% small groups is

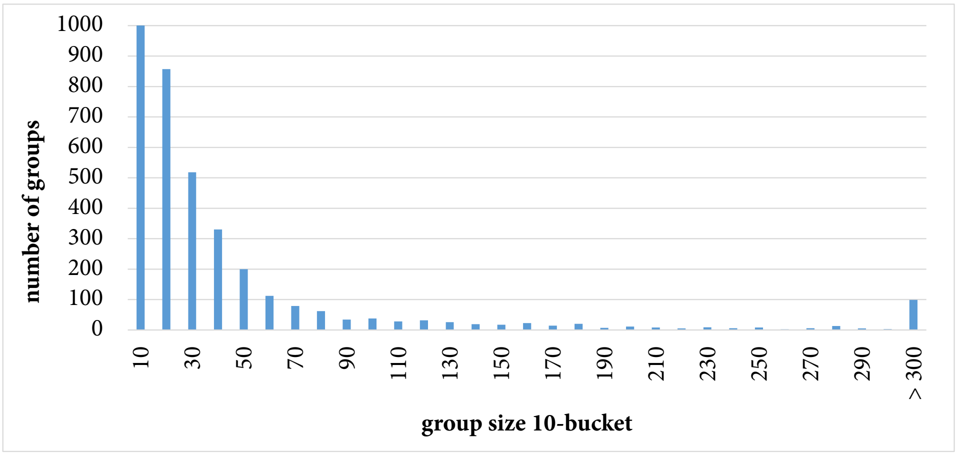


Figure 3.2: Frequency of named user groups with respect to the group size.

found, followed by 12.9% medium, 1.3% large and only 0.1% extra large groups.

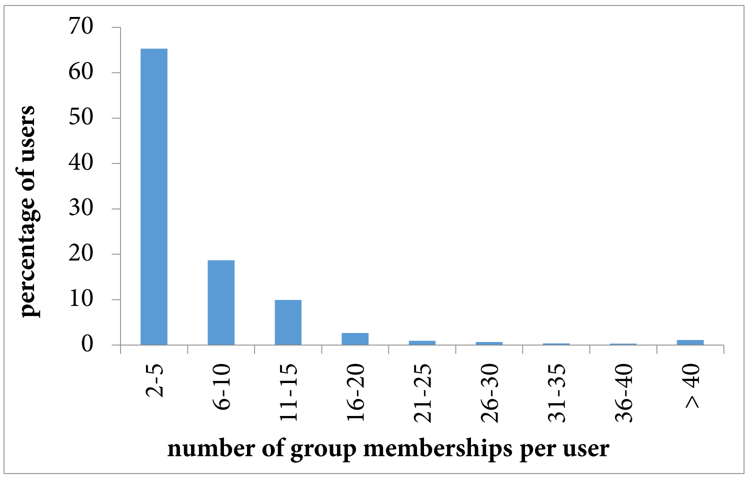


Figure 3.3: Frequency of users with respect to the number of memberships in named user groups. Only users with more than one group membership are considered.

Regarding the frequency of group member additions, to 68.9% all of groups, no member was added within the monitoring period. Not much surprisingly, the frequency of group member additions correlates with the group sizes: While no user was added at all to most of the small groups, a user was added to medium groups once in about 6 (2) days. About 1.25 (4) users are added to a large and about 3.7 users to an extra large group per day. The group with the most user additions per day is one of the large groups with about 5 user additions per day.

Group member removals are much rarer events. From about 85.0% of all groups, no user was removed at all during the monitoring period. On average, one removal occurs for a given group once in 133 (268) days, with a standard deviation of 20 (39) days. The group with most removals had 286 removals in 118 days, which equals about 2.5 removals per day. Apart from the fact that the maximum group size is an

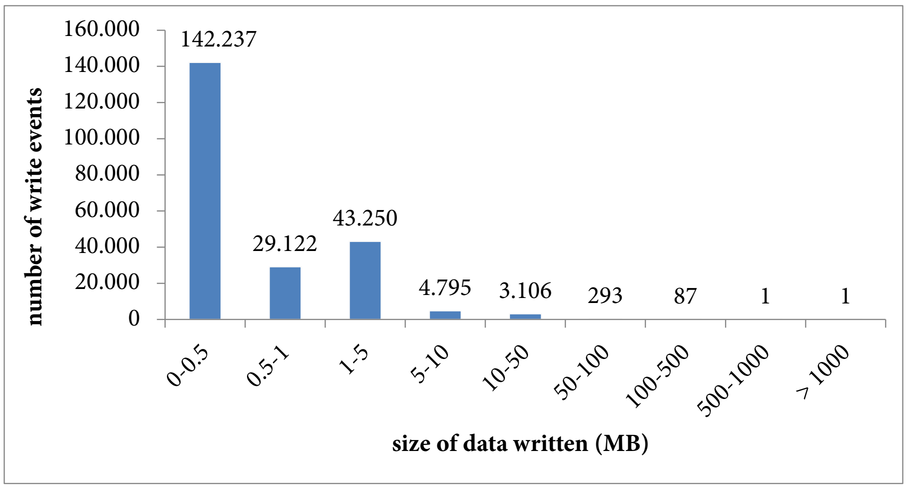


Figure 3.4: Frequency of write events with respect to the size of data written.

upper bound for the number of removals, no correlation was found between the size of a group and the frequency of removals.

When analyzing group membership relations from the perspective of a user, it is found that a user is member of 1.6 groups in average and of 852 groups in maximum. Only 0.3% of all users are member of no group at all, the vast majority of 89.2% of all users is member of exactly one group. The frequency of users with more than one group membership is shown in Figure 3.3.

In the following, hierarchy between groups are analyzed. The analysis was only carried out with regard to workloads that contain at least one group-group membership. This applies to 13 workloads from the groupware workloads. The e-learning platform workloads are not considered, as this platform does not support group hierarchies at all.

In these 13 workloads, 87% of all groups are involved in a group hierarchy. When only these groups are considered, a group is member of 1 group in average, and thus has 1 member that is a group in average.

An important factor for the performance of the protocol presented in Chapter 6 is the number of *descendant groups* of a group *g*, i.e., all groups that a member of *g* is member of. In the considered workloads, a group has 5 descendant groups on average. Another important influence factor for the performance of the mentioned protocol is the sum of group members of all descendant groups regarding a certain group. In the considered workloads, a group has about 6 descendant group members on average, with 358 descendant group members in maximum.

As the 13 workloads contain only a limited number of group-group memberships, the raw KIT-AD workload was also analyzed in a similar manner. Note that this workload will not be used as an input for a performance evaluation, as it only comprises users and groups and relation between those entities, but it does not contain any resources or permissions on resources. In the raw workload, a group is member of 1 group in average. Again, a group has about 6 descendants groups on average, with a maximum of 1 026 descendant groups. Furthermore, in the raw KIT-

AD workload, a group has about 182 descendant group members on average, with a maximum of 38 961.

Next, *write events* are analyzed. In this case, only a lower bound can be provided for the frequency of write events, as the workload tracing and generation method described in Section 3.3 only allows to capture one write event per resource per day. Under this assumption, about 95 write events per day in the e-learning and about 1 516 write events per day in the groupware platform can be detected. In the average write event, 1.19 MB of data is uploaded to the storage provider. The biggest upload in all of the workloads comprises 1 056 MB. The distribution of data size per write action is shown in Figure 3.4. As the plot shows, more than 63% of all written files are smaller than 0.5 MB, another 13% are sized between 0.5 MB and 1 MB, and only about a quarter of all write actions are larger than 1 MB.

Finally, resource permissions and the frequency of permission denials are analyzed, as these may require potentially resource-intensive re-encryption operations if eager revocation (cf. Section 2.2) is used. In the following, the effective file permission are characterized, i.e., folder permissions are recursively applied to subfolders and finally to files. On average, 97 (55) users are permitted to read and 4 (32) users are permitted to read and write a file.

On the e-learning platform, permissions cannot be assigned to users directly but only to groups; on average, 2.5 groups have read and 1.7 groups have read and write access to a file. On the groupware platform, permissions can also be assigned to users directly. In this case, 3.8 groups and 4.8 users are permitted to read a file, while 2.3 groups and 3.7 users are additionally permitted to write it.

From the perspective of a user, she is permitted to read 10.7 (421) files, and both read and write 0.5 (254) files in average. Figure 3.5 shows the distribution of the number of read and write permissions per user. In the groupware workloads, 94% of all the read and 93% of the read and write permissions are granted by group membership. A group has read access on 21 and additional write access on 13 files in average, ranging from 1 up to 15 499 accessible files. No correlation could be found between the group size and the number of accessible files.

A denial of read or write permissions is a rare event: While read-only access is never denied in the analyzed workloads, write access on a given file is denied once in about 2 375 (1 576) days.

In the remainder of this section, correlations between different scalar parameters of single workloads are analyzed. While the results of this correlation analysis also does not claim to yield results that are representative, they can give hints on how to configure the workload generator introduced in Section 3.6. The analysis focuses on the relationship of the overall "size" of a workload and several properties of the groups the workloads comprises, represented by the average of the respective frequency distribution.

The analysis considers two parameters representing the workload "size": on the one hand the total number of files, on the other hand the sum of all file sizes, referred to as "workload data size". Our workloads comprise between 10 and 15484 files, with an average of 1220 files.

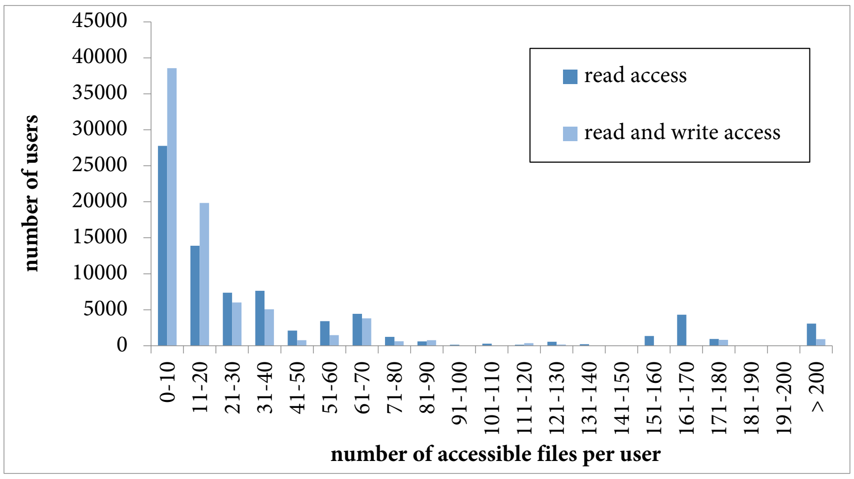


Figure 3.5: Frequency of users with respect to the number of accessible files.

The total data size of the workloads ranges from 0.9 MB up to 29.8 GB, with an average of 1 932 MB. The distribution of workload data size is shown in Figure 3.6.

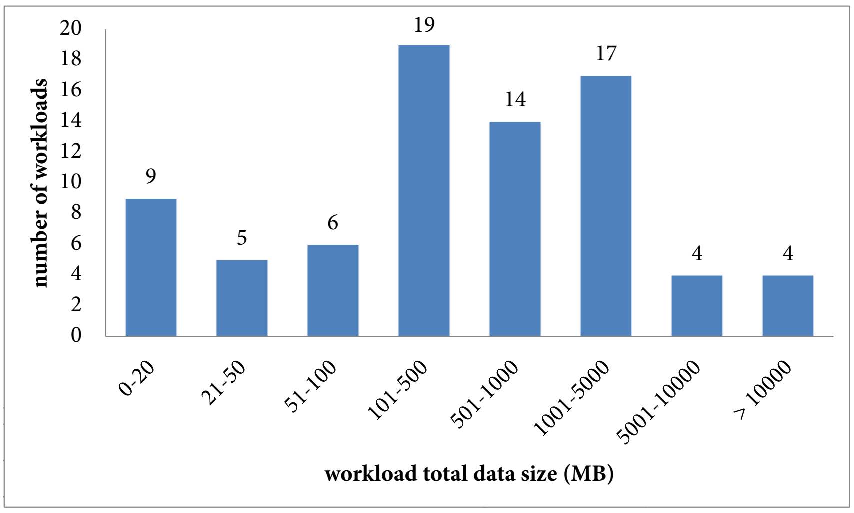


Figure 3.6: Frequency of workloads with respect to the size of all contained files.

Regarding the groups in the workload, the analysis considers the average number of files a group has read access to, and the sum of the sizes of these files. Finally, the analysis comprises the number of write actions that were carried out “by the group”, i.e., by a user leveraging the write permissions she gets by her group membership, and the size of data written “by the group”. About 96% of all workloads contain small groups, about 50% contain medium ones, about 12% contain large groups and about 6% contain extra large groups.

The correlation between these scalar workload parameters is shown in Table 3.4 for the groupware and in Table 3.5 for the ILIAS workloads. The first value in every table cell represent Spearman’s correlation coefficient, which was used rather than Pearson’s correlation coefficient because the involved parameters are not normally

Table 3.4: Spearman's correlation coefficient and p-value of groupware workload parameters. Cells shaded in gray indicate that the correlation is statistically significant, assuming a significance level $\alpha = 0.05$.

	1	2	3	4	5
1) workload number of files	1.00				
2) workload data size	0.88 (1,07E-22)	1.00			
3) average number of accessible resources per group	0.85 (9,29E-20)	0.77 (2,62E-14)	1.00		
4) average accessible data size per group	0.76 (8,80E-14)	0.89 (7,46E-24)	0.85 (9,29E-20)	1.00	
5) average number of write actions per group	0.78 (7,33E-15)	0.66 (1,24E-09)	0.87 (1,23E-21)	0.72 (6,58E-12)	1.00
6) average data size written per group	0.69 (1,06E-10)	0.80 (4,64E-16)	0.78 (7,33E-15)	0.92 (3,86E-28)	0.82 (2,12E-17)

Table 3.5: Spearman's correlation coefficient and p-value of e-learning workload parameters. Cells shaded in gray indicate that the correlation is statistically significant, assuming a significance level $\alpha = 0.05$.

	1	2	3	4	5
1) workload number of files	1.00				
2) workload data size	0.69 (1,88E-02)	1.00			
3) average number of accessible resources per group	0.58 (6,14E-02)	0.19 (5,76E-01)	1.00		
4) average accessible data size per group	-0.10 (7,70E-01)	0.55 (2,26E+00)	-0.04 (9,07E-01)	1.00	
5) average number of write actions per group	0.00 (1,00E+00)	0.03 (9,30E-01)	0.15 (6,60E-01)	0.00 (1,00E+00)	1.00
6) average data size written per group	-0.11 (7,47E-01)	0.40 (2,23E-01)	-0.26 (4,40E-01)	0.64 (3,39E-02)	0.46 (1,55E-01)

distributed. The second value is the p-value that indicates the statistical significance of the correlation found. When a correlation has a significance level higher than 0.05, the respective table cell is shaded gray.

It is easy to see that in regarding the ILIAS workloads, only two significant correlations could be identified: The number of files and the total file size of a workload correlate, as well as the average aggregated size of all files accessible to the group, and the total size of data written by using group permissions. Note that the latter correlation is quite obvious if only one group has access to a set of files. However, if more than one group is allowed to access a set of files, such a correlation is not mandatory any more.

Regarding the groupware workloads, significant correlations could be identified between any pair of analyzed workload parameters. Strong correlations could be found between the same parameter pairs as in the case of ILIAS workloads. In addition, strong correlations could be identified between, e.g., the total file size of a workload and the average aggregated size of all files accessible to a group, and between the average number of files accessible to a group and the number of write actions by using group permissions.

3.6 Generation of Synthetic Workloads

Workload generation means that based on a set of workload parameters, a workload is generated that adheres to the definition given in Section 3.2. The generation of a workload is necessary if the performance evaluation method requires a fully specified workload, but the workload cannot be retrieved from real-world systems. This may be the case if the real-world system under study does not exist yet, or if a possible future change to the sharing or usage scenario of such a system should be analyzed.

The workload generation method introduced in this section is based on the workload parameterization presented in Section 3.4. The method leverages the separation between static and dynamic workload parameters by first generating an initial authorization graph according to the static parameters, which is afterwards incrementally altered according to the dynamic parameters. Finally, it is proven that certain instance of the workload generation problem are NP-hard. Thus, the generation of large workloads is infeasible if certain parameter combinations are predefined.

3.6.1 Workload Generation Method

The workload generation method described in this section was constructed as part of a Bachelor's thesis [Lei16]. The method takes a set of workload parameter assignments as input, and outputs a workload according to the definition given in Section 3.2. The workload generation process comprises three steps: First, the set of predefined workload parameters—and partially also the assigned values itself—are checked. If this check is successful, an instance of an authorization graph is generated based on the predefined static parameters. Finally, according to the predefined dynamic parameters, incremental changes are applied to the authorization graph, and

ReadData and *WriteData* events are inserted. In the following, each of these steps will be explained in more detail.

The set of predefined workload parameters is restricted by both some minimality criteria that prevent that a workload depends too much on the generation method itself, as well as some maximality constraints that are derived from the dependency groups. The minimality constraints require that from each dependency group that reflects the properties of a bipartite graph, at least three out of four parameters are predefined.

To account for dependency groups, the set of predefined workload parameters is checked against each dependency group. In case of upper bounds, it is simply checked whether the assignments of the respective workload parameters comply with the upper bound. Regarding bipartite graph induced or transitive dependency groups, it is checked whether the set of predefined parameters contains one of these dependency groups completely, i.e., whether all of the parameters of any dependency group are specified at the same time. In this case, the workload generation is aborted. This restriction is overly strict in a sense that it prevents the processing of a workload parameter assignment that may be valid. However, the only alternative would be to calculate valid ranges for one workload parameter based on all other parameters in a dependency group. It is unclear whether this is possible at all, especially for transitive dependency groups.

In the next step, the initial authorization graph is generated, based on the static predefined workload parameters. The generation method leverages two algorithms to build the graph: The first algorithm is an extension of the configuration model, a well-known algorithm for the generation of random graphs (cf. Section 2.4.3). This algorithm first generates all graph nodes, according to the predefined parameter assignments. Thereafter, node degrees are drawn from the predefined frequency distribution parameters. For example, if the parameter *groups per user* is predefined, a target number of groups is drawn for each user node. Finally, edges are randomly added with consideration of the target degree of each node. The running time of this algorithm grows linearly with the number of edges in the authorization graph, and the resulting graph adheres to the predefined workload parameters very closely. However, this algorithm can only be applied if the set of predefined parameters does not contain any transitive parameter.

If transitive parameters are predefined, the method has to switch to another algorithm. In this case, the workload generation problem showed to be NP-hard with regard to the number of groups. The NP-hardness proofs are given later in this section. Consequently, it is not feasible to generate large workloads that adhere to the predefined workload parameters exactly. Instead, the method employs a heuristic algorithm, whose running time grows linearly with the number of transitive edges in the graph. The heuristic first generates user and resource nodes, and inserts temporary edges between these node sets, according to the predefined transitive parameters. This again can be done by using the configuration model. Further node degrees are drawn from the other predefined frequency distribution parameters. Thereafter, the algorithm orders the temporary edges by the user nodes they are starting from, and iterates over the edges. In the iteration process, the start user and end resource nodes of consecutive temporary edges are added to a newly created group node, as long as the target node degrees of the user and resource nodes already connected to the group node are not exceeded. In this case, a new group node is created.

The final workload generation step is the alteration of the initially generated authorization graph according to the predefined dynamic workload parameters. This step is independent of the graph generation algorithm used in the preceding step. The authorization graph is first altered by adding or removing nodes; the number of added or removed nodes of a certain type is drawn from the frequency distribution of the respective predefined parameter. Thereafter, edges in the authorization graph are added or removed. For each node, first the number of added and then the number of removed edges is drawn from the respective frequency distribution.

One limitation of the generation method is, as already mentioned, the overly strict parameter checking: As it is hard to check whether a given parameter assignment is valid especially with regard to transitive dependency groups, the check simply prevents the predefinition of all parameters of a dependency group. This obviously might also rule out valid parameter assignments, and therefore limits the space of workloads that can be generated.

Another limitation of the generation method is that it is impossible to define correlations between parameters. For example, there is no way to predefine that the number of accessible resources of a group should grow linearly with the group size. Furthermore, the generation method does not support "locality in space", i.e. there is no possibility to enforce clusters of users, groups and resources, with the probability that a user becomes member of a group within the cluster is higher than for a group in another cluster. The generation method does also not support "locality in time", e.g., the predefinition of bursts of a certain workload event type to imitate bulk operations. One more limitation is that resource hierarchies cannot be generated.

The limitations regarding correlations, locality or resource hierarchies could only be overcome by extending the set of supported workload parameters. This would require both a further analysis of dependency groups and an enhancement of the workload generation method. While this might be feasible from a conceptual point of view, for applying the tool practically, this would require the user to know or predict even more workload parameters.

3.6.2 NP-hardness of Workload Generation

In this subsection, the NP-hardness of a set of workload generation problems is proven. Note that the considered problems require a solution that exactly adheres to the predefined parameters. In many applications, it might be sufficient when the predefined parameters are approximately met by the generated workload. However, approximate solutions are beyond the scope of this section.

A generic formalization of a workload generation problem would be: Given a set of predefined static workload parameters (cf. Section 3.4), generate an authorization graph that adheres to the predefined parameters as closely as possible for some optimization criterion. This formalization leaves open, on the one hand, the concrete set of predefined workload parameters, and on the other hand, the concrete optimization criterion. Thus, the specification of the predefined workload parameters, together with the optimization criterion, constitutes a workload generation problem. In the remainder of this section, the predefined parameters are referred to as *input parameters*.

Restricting the Set of Analyzed Workload Problems

The set of static parameters given in Table 3.1 comprises 13 parameters, and therefore allows for 2^{13} parameter combinations. As an NP-hardness analysis of each parameter combination would go far beyond the scope of this thesis, the space of considered parameter sets has to be restricted.

As a first restriction step, a more basic version of the authorization graph is considered that does not allow for a group hierarchy, i.e., the relation *MemberGroup* is empty. This way, only 11 parameters have to be analyzed.

As a next restriction step, only those parameter combinations are considered that do not completely specify any of the dependency groups as listed in Section 3.4. In other words, only parameter combinations are considered that are a valid input for the workload generation method introduced in the previous subsection. By testing each parameter combination for conflicts with any dependency group, the number of non-conflicting parameter combinations is calculated as 1 493.

In a further restriction step, only those parameter combinations are considered that contain at least one transitive parameter. The reason for this is that the workload generation problems without transitive parameter can be solved with an extension of the configuration model in linear time, as described in the previous subsection. With this restriction, the set of considered parameter combinations comprises 1 066 elements.

Furthermore, only parameter combinations are considered that adhere to the following requirement: If the parameter combination comprises the number of nodes of a certain type, it also comprises at least one a frequency distribution parameter that is related to this node type, and vice versa. For example, if the combination contains the parameter *users per group*, it also has to contain *number of groups*. Only 54 parameter combinations comply with this limitation.

In addition, the direct parameters *direct resources per user* and *direct users per resource* are also excluded, as in the workload generation method, these parameter are treated independent of the transitive parameters. They can easily be considered by the extended configuration model. Thus, the parameter combinations of interest are decreased to 18.

In the final restriction steps, it is taken advantage of the fact that the set of static workload parameters can be seen as symmetric, i.e., each parameter combination has a "mirrored" parameter combination. If an algorithm is known to generate an authorization graph according to a parameter combination, this algorithm can also be applied to the mirrored parameter combination. Therefore, from each pair of parameter combinations, only one representative can be chosen arbitrarily. For this reason, the NP-hardness analysis only considers parameter combinations that contain the parameter *GroupResourcesPerUser*, and omits combinations that contain the mirrored parameter *GroupUsersPerResource* as only transitive parameter. As a result, the NP-hardness analysis deals with 9 workload parameter combinations, which are listed in Table 3.6.

workload generation problem	**input parameters**
1	number of users, group resources per user
2	number of users, group resources per user, number of resources, groups per resource
3	number of users, group resources per user, number of groups, users per group
4	number of users, group resources per user, number of groups, resources per group
5	number of users, group resources per user, groups per user
6	number of users, group resources per user, number of groups, users per group, number of resources, groups per resource
7	number of users, group resources per user, number of groups, users per group, resources per group
8	number of users, group resources per user, number of groups, resources per group, groups per user
9	number of users, group resources per user, number of resources, groups per resource, groups per user

Table 3.6: List of analyzed workload parameter combinations

NP-hardness Analysis

A workload generation problem Workload-Gen-X requires, given one of the workload parameter combinations P_x from those listed in Table 3.6, to generate an authorization graph that adheres to all the input parameters in $P_x \setminus \{GroupResourcesPerUser\}$ exactly, and minimizes the sum of differences between the input and the actual frequency distribution of *GroupResourcesPerUser*, i.e., minimizes
$\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.

The Workload-Gen-X problems corresponding to the workload parameter combinations are visualized in Figure 3.7. For each problem, the diagram visualizes the input parameters. The user nodes are represented as blue squares on the left side of each problem visualization, the group nodes as yellow squares in the middle and the resource nodes as green squares on the right side. The circles indicate the degree of each node; while the yellow circles represent direct parameters, the red circles at the top of the user nodes represent the transitive parameter *GroupResourcesPerUser*.

In Figure 3.7, the problems are ordered in three layers and connected by blue ar-

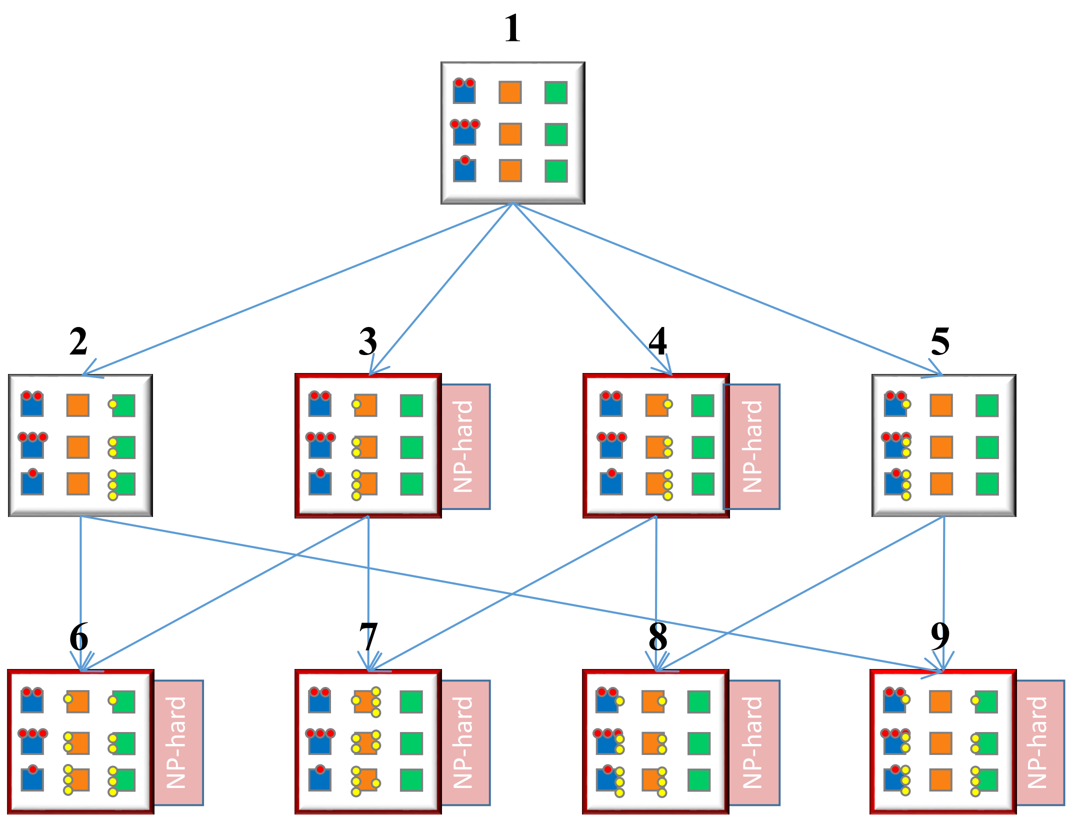


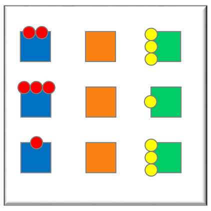
Figure 3.7: Visualization of analyzed workload generation problems. The arrows connecting different problems do not indicate a specialization or generalization relation, but superset/subset relations between the respective sets of input parameters.

rows: An arrow indicates that the parameter set of the target problem equals the union of the parameter sets of the starting problems. It is important to stress that this visualized relation between different problems is no specialization or generalization relationship, i.e., an instance of one of those problems cannot be expressed as an instance of another problem just by applying some restrictions.

For the sake of brevity, the node degree that was drawn for a certain node, i.e. the target node degree, is referred to in an object-oriented language like manner, such as, e.g., *user.targetGroups* for the drawn number of groups a certain user node is connected to. The number of groups the user node is connected to after the algorithm has finished is referred to as *user.groups*. In the following, it is shown that problems 1, 2 and 5 can be solved in polynomial time by describing a linear-time algorithm.

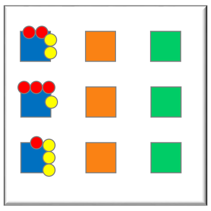
Definition 7 (Workload-Gen-2)**.** Given frequency distributions for the parameters *number of users*, *group resources per user*, *number of resources* and *groups per resource*, generate an authorization graph adhering to *number of users*, *number of resources* and *groups per resource* exactly, and that minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.

Workload-Gen-2 can be solved in linear time with regard to the number of users and resources: For each resource node, resource.targetGroups group nodes are created and connected to the resource node. Each user node is connected to user.targetResources group nodes, with the limitation that these group nodes have to be connected to pairwise different resources.



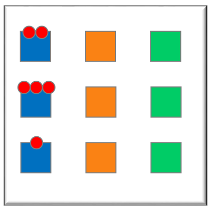
Definition 8 (Workload-Gen-5)**.** Given the *number of users* and frequency distributions for *group resources per user* and *groups per user*, generate an authorization graph that adheres to *number of users* and *groups per user* exactly, and minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.

Workload-Gen-5 can also be solved in linear time with regard to the number of users: For each user node, user.targetGroups group nodes are created and connected to the user node. Furthermore, user.targetResources resource nodes are created and connected to one of the newly created group nodes; the remaining group nodes are not connected to any resource node.



Definition 9 (Workload-Gen-1)**.** Given the *number of users* and a frequency distribution for *group resources per user*, generate an authorization graph that adheres to *number of users* exactly, and minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.

Workload-Gen-1 can also be solved in linear time with regard to the number of users by enhancing it to either Workload-Gen-2 or Workload-Gen-5. For example, Workload-Gen-1 can be enhanced to Workload-Gen-5 by arbitrarily defining user.targetGroups for each user node.



The workload generation problems 3, 4, 6, 7, 8 and 9 are NP-hard problems, which will be proven as follows: One of the well-known NP-hard problems Subset-Sum or Partition, which are both defined in the next paragraph, is reduced to Workload-Gen-X. Note that Subset-Sum or Partition are both decisional problems, where the solution to the problem is either yes or no, while Workload-Gen-X is an optimization problem. Therefore, the basic idea is to construct reduction proofs in a way that the decisional problem evaluates to yes iff a perfect solution for the optimization problem can be found, i.e., a solution where user.resources equals user.targetResources for each user node. In each proof, it is shown that the existence of a perfect solution to Workload-Gen-X is equivalent to a positive evaluation of Subset-Sum or Partition.

The Subset-Sum problem is defined as follows:

Definition 10 (Subset-Sum problem [CLO91])**.** Given n integers $a_1,\ldots,a_n$ and an integer s, is there a subset I of $\{1,\ldots,n\}$ such that $\sum_{i\in I} a_i = s$?

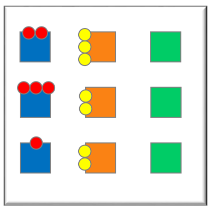
The Partition problem is defined as follows:

Definition 11 (Partition problem [BRV13])**.** Given n integers $a_1,\ldots,a_n$, is there a subset I of $\{1,\ldots,n\}$ such that $s := \sum_{i\in I} a_i = \sum_{i\notin I} a_i$?

Note that while efficient algorithms exist that provide approximate solutions to Subset-Sum and Partition [KX15], this does not imply that efficient approximation algorithms also exist for the workload generation problem. To show that algorithms for well-known problems are also applicable to a workload generation problem, the workload generation problem must be reduced to the well-known problem, with certain constraints on the complexity of the reduction. This is the opposite of what is done in this section.

In the remainder of this section, each of the NP-hard workload generation problems is defined, and a proof of the NP-hardness is given.

Definition 12 (Workload-Gen-3)**.** Given the *number of users* and the *number of groups*, and frequency distributions for *group resources per user* and *users per group*, generate an authorization graph that adheres to *number of users*, *number of groups* and *users per group* exactly, and minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.

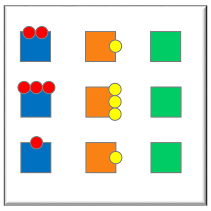


The NP-hardness of Workload-Gen-3 with regard to the number of group nodes is proven by reducing the Partition problem. To transform the Partition problem instance to Workload-Gen-3, first insert s user nodes with user.targetResources = 1, and another s user nodes with user.targetResources = 2. For each integer a_i, insert one group node with group.targetUsers = a_i. This transformation can be done in linear time.

After graph generation, each user is assigned to exactly one group: As for each user, user.targetResources > 0, each user is assigned to at least one group. But as there are $2s$ user nodes and the sum of group.targetUsers for all groups equals $2s$, each user is connected to at most one group.

If a perfect solution was found, each group node is connected to either one or two resource nodes, depending on the value of user.targetResources of the group members. Thus, the group nodes are partitioned into "one-resource groups" and "two-resource groups", where each one-resource group is only connected to user nodes with user.targetResources = 1. Therefore, the sum of group.targetUsers equals s for all one-resource groups. This works exactly when s can be represented as the sum of a subset of the integers $a_1,\ldots,a_n$.

Definition 13 (Workload-Gen-4)**.** Given the *number of users* and the *number of groups*, and frequency distributions for *group resources per user* and *resources per group*, generate an authorization graph that adheres to *number of users*, *number of groups* and *resources per group* exactly, and minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.

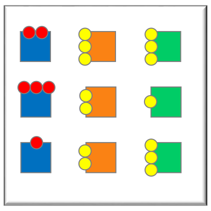


The NP-hardness of Workload-Gen-4 with regard to the number of group nodes is proven by a reduction of Subset-Sum. The Subset-Sum instance is transformed to a Workload-Gen-4 instance by creating one user node with user.targetResources = s, and another user node with user.targetResources = $\sum_{i=1}^{n} a_i$. For each integer a_i, insert one group node with group.targetResources = a_i. This transformation requires a number of steps that is linear in the Subset-Sum problem size n.

In the resulting graph, since the sum of group.resources over all groups equals $\sum_{i=1}^{n} a_i$, this is an upper bound for the number of resources that is connected to a group at all.

If a perfect solution was found, the generated graph contains exactly $\sum_{i=1}^{n} a_i$ resource nodes that are connected to a group at all: There cannot be fewer resources connected to a group at all, since user.targetResources of the second user node equals $\sum_{i=1}^{n} a_i$. This observation also implies that each resource is connected to no more than one group node, i.e., it is impossible that there exists more than one path from a certain user to a certain resource. The sum of group.resources of all groups the first user node is connected to equals user.targetResources, which was defined as s. This is equivalent to the statement that s can be represented as the sum of a subset of the integers $a_1,\ldots,a_n$.

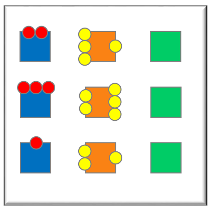
Definition 14 (Workload-Gen-6)**.** Given the *number of users*, the *number of groups* and the *number of resources*, and frequency distributions for *group resources per user*, *users per group* and *groups per resource*, generate an authorization graph that adheres to *number of users*, *number of groups*, *users per group*, *number of resources* and *groups per resource* exactly, and minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.



Workload-Gen-6 can be proven NP-hard with regard to the number of group nodes by a reduction of Subset-Sum. The reduction proof requires that the size of the subset that sums up to s—denoted as k—is defined up front. The original Subset-Sum problem can be solved by iteratively increasing k from 1 to $n/2$, which is obviously linear in n. Thus, this constrained version of Subset-Sum is nonetheless NP-hard.

The instance transformation requires to add s user nodes with user.targetResources = 1, and of k resource nodes with resource.targetGroups = 1. For each integer a_i, one group node is created with group.targetUsers = a_i. This transformation is linear in the Subset-Sum problem size n. Assuming a perfect solution, the generated graph has exactly k group nodes connected to one resource each, and the remaining group nodes are not connected to any resource. No group node can be connected to more than one resource node, as this would violate the restriction of user.targetResources = 1. For the same reason, a user node can only be assigned to exactly one group node that is connected to a resource. Thus, each of the s user nodes is connected to exactly one of the k group nodes that are connected to a resource at all. This is only possible if the sum of group.targetUsers for these k group nodes equals s, which is equivalent to the statement that a k-sized subset of $a_1,\ldots,a_n$ exists that sums up to s.

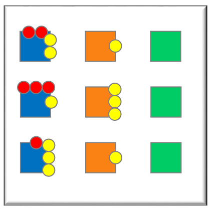
Definition 15 (Workload-Gen-7)**.** Given the *number of users* and the *number of groups*, and frequency distributions for *group resources per user*, *users per group* and *resources per group*, generate an authorization graph that adheres to *number of users*, *number of groups*, *users per group* and *resources per group* exactly, and minimizes the target function $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.



The proof of the NP-hardness of Workload-Gen-7 with regard to the number of groups is done by a reduction of Partition. The linear-time transformation from the Partition instance to an instance of Workload-Gen-7 requires to create two user nodes with user.targetResources = s each. A third user node with user.targetResources = $2s$ is created. For each integer a_i, one group node is inserted with group.targetResources = a_i and group.targetUsers = 2.

A perfect solution for the generated graph comprises exactly $2s$ resources that are connected to a group at all, and each of these resources is connected to only one group. The reason for this was already described in the proof of NP-hardness of Workload-Gen-4. The third user node is connected to all groups, as only this way, the requirement user.targetResources = $2s$ can be met. In consequence, each group is additionally connected to either the first or the second user node, i.e., the set of groups is partitioned into "first-user groups" and "second-user groups". As user.resources = s for the first and the second user node, the sum of group.resources over the groups of each partition equals s. The existence of this partitioning is equivalent to the statement that Partition evaluates to yes for $a_1,\ldots,a_n$.

Definition 16 (Workload-Gen-8)**.** Given the *number of users* and the *number of groups*, and frequency distributions for *group resources per user*, *resources per group* and *groups per user*, generate an authorization graph that adheres to *number of users*, *number of groups*, *resources per group* and *groups per user* exactly, and minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.



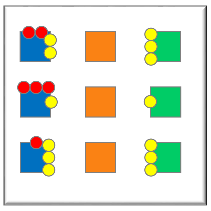
The NP-hardness of Workload-Gen-8 with regard to the number of group nodes can be proven by a reduction of Subset-Sum. Similar to the NP-hardness proof of Workload-Gen-6, the size of the subset whose elements sum up to s is predefined as k, thus, the algorithm has to be repeated linear times.

The transformation of the Subset-Sum instance involves the creation of one user node with user.targetResources = s and user.targetGroups = k. Another user node is inserted with user.targetResources = $\sum_{i=1}^{n} a_i$ and user.targetGroups = n. Again, for each integer a_i, one group node is created with group.targetResources = a_i, which can be done in a number of steps that grows linearly with the problem size n.

In the resulting graph, the second user node will be connected to every group node, which is caused by user.targetGroups set to n.

Assuming a perfect solution was found by the graph generation algorithm, there will be exactly $\sum_{i=1}^{n} a_i$ resource nodes that are connected to a group node at all, and each of these resource nodes will be connected to exactly one group. The reason for this was already discussed in the NP-hardness proof of Workload-Gen-4. The first user node is assigned to k groups, and the sum of group.resources for each of these groups must equal user.targetGroups, which is s. This is equivalent to the statement that s can be represented as the sum of k integers out of the integer set $a_1,\ldots,an$.

Definition 17 (Workload-Gen-9)**.** Given the *number of users* and *number of resources*, and frequency distributions for *group resources per user*, *groups per resource* and *groups per user*, generate an authorization graph that adheres to *number of users*, *number of resources*, *groups per resource* and *groups per user* exactly, and minimizes $\sum_{i=1}^{n} |u_i.targetResources - u_i.resources|$.



Finally, Workload-Gen-9 can be proven NP-hard with regard to the number of users by reduction of Subset-Sum. As in the NP-hardness proofs of Workload-Gen-6 and Workload-Gen-8, the size of the subset whose elements sum up to s is predefined as k.

The transformation of the Subset-Sum instance requires the creation of one user node for each integer a_i with user.targetResources = a_i and user.targetGroups = 1. An additional user node is inserted with user.targetResources = s and user.targetGroups = k. Furthermore, $\sum_{i=1}^{n} a_i$ resource nodes are created, each with resource.targetGroups = 1.

If the graph generation algorithm yields a perfect solution, every user node except of the last one will be connected to exactly one group node, which in turn is connected to a_i resource nodes. The last user node is assigned to k group nodes, and the sum of group.resources for these groups equals user.targetGroups, which is s. This is possible exactly if s can be represented as the sum of a k-sized subset of $a_1,\ldots,a_n$.

The fact that some specific workload generation problems are NP-hard indicates that workload generation, and performance evaluation methods that rely on workloads, might hardly be feasible for large workloads.

3.7 Conclusions and Outlook

The sharing and usage scenario has a major influence on the performance of SDS protocols, and therefore has to be a part of performance evaluation methods. In this chapter, workloads were introduced as a combined sharing and usage model. It was shown how real-world workloads can be retrieved from sharing services running in production. However, it was argued that fully specified workloads will be available in practice only rarely. This can be due to several reasons: The system under study might not even exist yet, the tracing of these systems might require too much effort, or it might be necessary to assess the performance of a given SDS system under varying sharing or usage scenarios, e.g., to predict the changes in performance caused by assumed future changes in the sharing behavior. It was argued that even if fully specified workloads are unavailable, at least a set of workload parameters might be known or can be estimated. Thus, a performance evaluation method that is usable in practice should accept a set of workload parameters as input.

The identification of a set of workload parameters that constitutes a sufficient input for a performance evaluation method is challenging, as this set should fulfill two conflicting goals at the same time: On the one hand, the set should only contain parameters that are likely to be known by the provider of the sharing system, who is assumed to carry out the performance evaluation. On the other hand, the set should be comprehensive enough to specify the workload in sufficient detail. In this chapter, a set of workload parameters was presented that can serve as a basis for workload generation, which again can be considered an initial step of a performance evaluation. This set was constructed as a trade-off between these goals: Some degrees of freedom regarding the workloads that comply to the parameter set were accepted, e.g., correlations between different parameters cannot be predefined. However, this deliberate limitation allows to avoid workload parameters that may only be known or estimated after a very detailed inspection of the sharing and usage scenario of the SDS system under study.

Based on the presented workload parameter set, real-world workloads were characterized. These workload were retrieved from an e-learning and a groupware platform running at KIT in production. While it cannot be claimed that these workloads are also representative for different data sharing applications, they can be considered as a guideline if workload parameters that are required for workload generation are unknown.

Finally, the workload generation method was presented. It could be shown that the hardness of the graph generation problem that has to be solved for workload generation depends on the set of predefined workload parameters: As long as the set does not contain a transitive parameter, i.e., a parameter that describes the indirect relation of users and resources via group memberships and permissions, the related problem instances can be solved in linear time. If a transitive parameter is predefined, it was proven that many of the related problem instances are NP-hard. Future work could try to identify approximate solution approaches that are feasible in practice.

The problem instances considered in the NP-hardness proofs require that possible solutions exactly adhere to all non-transitive parameters. However, for many applications, an approximate solution might be sufficient. Therefore, a future task could be to analyze the hardness of workload generation problems that also accept a solution that fits the predefined parameters "good enough".

A possible enhancement of the workload definition would be a media type label for resources, which indicates whether the resource is a text document, a picture, a movie etc. As will be discussed in Chapter 4, this would allow to estimate the time it requires the user client to process the resource after the download. As this processing time is required no matter if the data sharing is secure or "insecure", it could serve as a baseline to assess the overhead of security in data sharing.

4 Methods for SDS Performance Evaluation

In existing literature, the performance of SDS protocol when processing different workload event types, e.g., uploading data or adding a member to a sharing group, is usually expressed in terms of asymptotical behavior of some performance metric in relation to some workload parameter N. While asymptotical analyses are a good starting point to identify factors that are relevant for the performance, they are not sufficient to estimate the concrete performance of secure data sharing for two reasons. First, the values that the workload parameter N typically takes, i.e., the underlying sharing model of the system, is often unclear: While we assume that a storage provider may have some rough idea of the range of sharing group sizes or the overall size of data shared, many possibly relevant parameters are harder to determine, e.g., the average number of files accessible per sharing group. This issue was already addressed by the workload characterization presented in Section 3.5. Second, the asymptotical analysis abstracts from constant factors, e.g., the properties of the client hardware, which may be highly relevant for the performance in reality. Thus, a more fine-grained method for evaluating the performance of SDS protocols is necessary.

In this chapter, the methods used in this thesis for performance evaluation of SDS protocols are presented: an analytical method in Section 4.1, and a simulative method in Section 4.2. As the performance depends not only on the SDS protocol, but also on the workload and the client hardware, these factors are incorporated into both methods. The methods especially differ in that the simulative method requires fully specified workloads, while the analytical method might operate on more abstract workload parameters, which renders the latter more usable in real-world applications. However, it will be shown that there are cases in which the analytical method requires mathematical techniques that are hard to handle in practice. In those cases, the simulative method might be preferred.

The goal of both the analytical and the simulative performance evaluation method is the estimation of the computation time and incoming and outgoing network traffic required on a user's client device. The first dimension, the computation time, represents the time it takes the client device to process a workload event completely. The use of SDS requires computation time especially for cryptographic operations, e.g., for the generation of keys, for encryption and decryption, or for the generation of digital signatures. This dimension is relevant for the deployability of the SDS system, as it has a major influence on the delay the user experiences, e.g., when she downloads and opens an encrypted document. In this thesis, it is assumed that a workload event is processed sequentially, i.e., the concurrent processing is not considered.

As second dimension, the network traffic that is induced by SDS is considered. SDS generates network traffic in addition to the network traffic required by "insecure" data sharing, as it requires the exchange of cryptographic material between the users, e.g., encrypted data, encrypted keys, or digital signatures. This dimension also has a major impact on the deployability of the SDS system, as especially mobile devices may be connected to wireless networks with small bandwidths. The performance evaluation methods consider incoming and outgoing traffic separately. In the following, the term *costs of an operation* is used as a placeholder for either the computation time or the network traffic induced by an operation.

The basic assumption behind both the analytical and the simulative method is that the performance is largely determined by the cryptographic operations. Thus, only the computation time and the network traffic generated by cryptographic operations are considered by the methods.

Parts of this chapter have already been published before in [KH15] and [KH16].

4.1 Analytical Method

In this section, the analytical performance evaluation method is presented. The modeling process for this method usually starts with the formalization of the performance of a single workload event, which will be introduced in the first part of this section. In the second part, it is shown how the single-event model can be generalized to evaluate the performance of an SDS protocol under a complete workload.

4.1.1 Analyzing the Performance of Single Workload Events

With the analytical method, the performance is estimated by using a set of mathematical expressions. These expressions have to be evaluated by inserting the concrete costs of cryptographic operations on a certain device, and concrete workload parameters (cf. Chapter 3). The expressions are built in a manual process by analyzing the SDS protocol under study. For each workload event type, the SDS protocol defines a sequence of operations that have to be carried out. From this definition, which is usually given in pseudocode, the number and type of cryptographic operations have to be extracted. The result of the protocol analysis is one expression per workload event type for each of the dimensions computation time, incoming network traffic, and outgoing network traffic.

Whereas the goal of the method is to evaluate the performance regarding a whole workload, the SDS protocol analysis usually only yields the performance for the execution of a single workload event. There are different challenges for the generalization of the resulting per-event, or *raw*, expressions to per-workload expressions. These challenges will be discussed in the next subsection.

A simple example for a raw expression allows to calculate the computation time for processing a *RemoveUserFromGroup* event of a fictitious SDS protocol as

$$\mathit{ctRemoveUserFromGroup} = \mathit{ctCreateSymKey} + \mathit{ctAsymEncryptKey} \times \mathit{groupSize}, \quad (4.1)$$

where *ctCreateSymKey* represents the time required to create a symmetric key, *ctAsymEncryptKey* represents the time required for the encryption of a symmetric key with an asymmetric cipher, and *groupSize* is the current size of the user group. From a syntactical point of view, the expression is an algebraic expression, and therefore comprises variables that are linked by algebraic operations. The variables in a raw expression are either costs of cryptographic operations, such as *ctCreateSymKey* and *ctAsymEncryptKey* in the example above, or entity parameters such as *groupSize*. An entity parameter is a parameter that is related to a specific entity within the workload, i.e., a user, a group, or a resource. Similar to workload parameters, the set of thinkable entity parameters is not limited (cf. Section 3.4). An entity parameter might be as simple as "the number of groups the user is member of" or "the data size of a resource", while it might as well be as complex as "the number of users that have access to more than 50% of the resources the user has access to". The last example shows that to evaluate an entity parameter, it may be necessary to not only consider the entity in question, but also other entities that are related in some way. An entity parameter can be generalized to a workload parameter in many ways, e.g., by defining a workload parameter that reflects the average of an entity parameter over all entities of this type.

The validity of the analytical method is based on some assumptions. The first assumption is that both the computation time as well as the network traffic required for processing a workload event can be estimated with sufficient precision by only considering cryptographic operations. As the traffic size of the actual shared (and encrypted) data is part of the workload, it is also taken into account in the performance evaluations carried out for this thesis, and can also serve as a baseline to compare SDS with "insecure" data sharing. On the other hand, the computation times required to process data by the client-side application, such as rendering a document or a movie, are not taken into account. This is discussed as future work at the end of this chapter.

The second assumption is that the costs of the cryptographic operations carried out to process one workload event are independent from each other. This means that the costs of a repetition of a cryptographic operation scale linearly with the number of repetitions. This assumption may break if a cryptographic operation consists of an initialization phase that may take a comparatively long time, and an arbitrary number of comparatively fast repetitions. In fact, some micro-benchmarking shows that this assumption breaks, e.g., when using certain implementations of AES: The total time required to encrypt a large chunk of data cannot be formulated as multiple of the time required to encrypt a small chunk of data. However, the micro-benchmarking also shows that the error of a linear approximation is within the range of microseconds, which can be considered acceptable for the purpose of this thesis.

The third assumption regarding the validity of the analytical method is that the costs for processing the workload events can be considered isolated from each other. This especially means that after processing a workload event, the client-side state is completely cleared, i.e., all required and generated cryptographic material is locally deleted, apart from personal keys of the user. Thus, if the costs for processing a workload event is dependent on previous workload events in any way, then this dependency has to be modeled as an entity parameter.

As already mentioned, an expression can be built by using an arbitrary combination of algebraic operations. In course of the performance evaluation of different SDS protocols carried out in this thesis, various types of expressions were actually met. For example, sometimes the costs can be expressed independent from any entity parameters by simply scaling the costs of a single cryptographic operation:

$$ctAddUser = 2 \times ctCreateAsymKeyPair. \tag{4.2}$$

In other cases, the costs are expressed as linear combination of different entity parameters, similar to the following example:

$$\begin{aligned} ctRevokeUserReadWriteAccess = & resourceReaders \times ctAsymEncryptKey + \\ & 2 \times resourceWriters \times ctAsymEncryptKey, \end{aligned} \tag{4.3}$$

where a key has to be asymmetrically encrypted for each user that is granted read access on the resource, and two keys have to be asymmetrically encrypted for each user that is granted read and write access. Finally, some expressions requires to sum up one parameter of multiple entities, where the number of sum elements is determined by another entity parameter:

$$ctRemoveUserFromGroup = \sum_{i=1}^{userGroups} groupSize \times ctAsymEncryptKey \tag{4.4}$$

In this example, the computation time of a group member removal event is determined by the sum of the group sizes of the removed user's groups.

4.1.2 Analyzing the Performance of Complete Workloads

To generalize the raw expressions that are related to a single workload event to expressions that are related to the workload as a whole, the entity parameters have to be replaced by workload parameters. For example, to generalize the raw expression given in Equation 4.1 to yield the average computation time for the group member removal event over the whole workload, the entity parameter *groupSize* has to be replaced by an average over the whole workload. More generally, the workload parameter must reflect a set of samples of the entity parameter taken in the course of the workload. As each expression is related to a certain workload event type, the entity parameter should only be sampled when an event of exactly this type occurs. For example, the entity parameter *groupSize* in the raw expression Equation 4.1 should be replaced by a workload parameter that reflects the group size whenever a group member is removed. Depending on the purpose of the performance evaluation and the required accuracy, it might be valid to insert a more general sample set of the entity parameter: In the example above, the entity parameter *groupSize* might also be replaced by a workload parameter that represents the group size whenever an arbitrary workload event occurs.

The types of workload parameters considered in this thesis are either scalar values or frequency distributions (cf. Section 3.4). Therefore, an entity parameter can be replaced by either the frequency distribution of a set of samples of this parameter, or by a statistical measure of this frequency distribution. However, this replacement is related to some assumption and challenges, which are discussed in the following.

The first challenge is that replacing an entity parameter by a frequency distribution or a statistical measure of such a distribution is only possible if the expression comprises no more than one workload parameter. The reason for this is that algebraic operations involving multiple frequency distributions—or their statistical measures—are not meaningful. The only exception is the average of the frequency distributions. There are two ways to deal with this issue: Either the entity parameters are not replaced by the respective frequency distributions one by one, but they are replaced by a joint frequency distribution as a whole. For example, the raw expression Equation 4.3 can be generalized by retrieving the joint frequency distribution of the entity parameters *resourceReaders* and *resourceWriters*, and replacing the sum of the entity parameters with the frequency distribution of *resourceReaders* + 2 × *resourceWriters*.

However, from a methodical point of view, replacing entity parameters by joint frequency distributions can be considered as a step towards a simulative performance evaluation method. In fact, the simulative method discussed in Section 4.2 implicitly takes these joint frequency distributions into account. A way to deal with this limitation while retaining the analytical method is to switch from descriptive frequency distributions to probability distributions, which is discussed in the remainder of this section.

While it is possible to apply algebraic operations on the probability distributions underlying the frequency distributions of the workload parameters, there are still some assumptions required. For example, without knowing the joint distribution of the workload parameters, the expression can only be evaluated if the contained workload parameters are statistically independent. This breaks, e.g., as soon as there are multiple occurrences of one and the same workload parameter within an expression [Wil89]. In case of independent workload parameters, there may be ways to, e.g., calculate a linear combination of the parameter probability distributions by convolutions of these distributions. While convolutions are well-known for some concrete families of probability distributions, cumbersome mathematical techniques such as numerical methods are required to calculate convolutions of other distribution families: "This is because the resulting formulae are sometimes so complex (e.g. infinite series of transcendental functions) that a computer program is needed to calculate the specific values and to determine the behaviour of the distribution." [Wil89]

The property of statistical independence of the workload parameters is also important when carrying out a sensitivity analysis, i.e., when the impact of one or more workload parameters on the performance should be analyzed. A common way to carry out a sensitivity analysis is to vary one parameter while fixing all other parameters, which is known as *One-Factor-at-a-Time experiment* [Dan73]. This requires that the parameters can be varied independently from each other, which is obviously possible with statistically independent parameters. If the parameters depend on each other, it is only possible to vary the joint distribution of these parameters, which defeats the goal of quantifying the impact of a distinct workload parameter on the performance.

In conclusion, the analytical method is a straight-forward method that is based on a manual analysis of the SDS protocol under study. The resulting set of expressions is a comparatively intuitive performance evaluation model. The analytical method does not require fully specified workloads as input, but workload parameters are sufficient, which enhances the employability of this method in practice. In many cases, it can be used to do a sensitivity analysis with low effort, especially when only the average performance is of interest. However, if the performance of an SDS protocol is determined by dependent workload parameters, the input of the method must either be based on joint frequency distributions, which is a step towards the simulative method presented in the next section. Alternatively, the deterministic frequency distributions are replaced by probability distributions. However, the mathematical techniques required to calculate with certain families of probability distributions are cumbersome. Thus, in case of dependent workload parameters, the simulative method might be more practical.

4.2 Simulative Method

In this section, a simulative performance evaluation method is presented. The main idea of this method is that SDS protocols respond to workload events by manipulating the key graph that represents the system state. These key graph manipulations directly reflect the costs of the operation. In the first part of this section, a definition of the key graph is introduced that serves as a basis for the simulative performance evaluation method. In the second part, the simulative performance evaluation method is presented.

4.2.1 Key Graph

As shown in Section 2.2, an SDS protocol leverages cryptographic data structures, such as keys, lockboxes, or digital signatures to achieve confidential, integer and authentic data sharing. As a simple example, when a resource should be written, an SDS protocol may first create a symmetric key to encrypt the resource, and then create a lockbox by encrypting this symmetric key with the public key of the writing user. The main idea of the simulative method is based on two observations: First, the generation of these cryptographic data structures requires computation time, and the up- and download of the data structures generates network traffic. And second, lockboxes and signatures are about linking multiple keys, or linking a key to a resource: both lockboxes and digital signatures are parameterized by a key—the encrypting or signing key, respectively—and they are related to some encrypted or signed data, which may be a key by itself, or a resource. Therefore, the cryptographic data structures and their relations form a graph—the key graph —, and the performance can be estimated from the changes to the key graph. Thus, for performance evaluation purposes, SDS protocols can be modeled in terms of the manipulations they carry out on the key graph.

In the following, a definition is given for the key graph that is used as the basis for simulative performance evaluation. This definition of a key graph is an extension of the definition given in Section 2.2. For the rest of this thesis, the term *key graph* refers to the definition given below, if not stated otherwise. In the following, the possible types of nodes and edges are specified.

Definition 18. A *key graph node* has one of the following types: *user*, *resource*, *key*, *lockbox*, *signature*. The set of user nodes is denoted by U, the set of keys as K, the set of resources as R, the set of lockboxes as L, and the set of signatures as S. The set of key graph nodes is denoted as V_{kg}.

The edges between node types carry different semantics, as, e.g., an edge from a user node to a key node indicates that the user has access to this key in clear text. However, there are combinations of edge source and edge target node types cannot be assigned meaningful semantics for SDS, such as an edge from signature node to a user node. Therefore, the set of allowed key graph edges has to be restricted.

Definition 19. The edge source node s and edge target note t of a *key graph edge* fulfill the following conditions:

1. $s \in U \Rightarrow t \in K \cup R \cup L \cup S$
2. $s \in K \Rightarrow t \in L \cup S$
3. $s \in L \Rightarrow t \in K \cup R$
4. $s \in S \Rightarrow t \in L$
5. $s \notin R$

The set of key graph edges is denoted as E_{kg}.

An overview of the allowed edge types and the semantics of those is given in Table 4.1.

Based on these definitions, the key graph can easily be defined as:

Definition 20. A *key graph* is a directed acyclic graph (V, E) with $V = V_{kg}$ and $E = E_{kg}$.

source node type	**target node type**	**semantics**
user	key	the user stores the key in plain text on her client
user	resource	the user stores the resource in plain text on her client
user	lockbox	the user just created or retrieved the lockbox
user	signature	the user just created or retrieved the signature
key	lockbox	the lockbox was created using the key
key	signature	the signature was created using the key
lockbox	key	the lockbox encrypts the key
lockbox	resource	the lockbox encrypts the resource
signature	lockbox	the signature can be used to check the integrity of the lockbox

Table 4.1: List of allowed key graph edge types and their semantics

An exemplary key graph is visualized in Figure 4.1.

Both symmetric keys and asymmetric key pairs are modeled as key nodes. An edge from an asymmetric key node to a lockbox node means that the lockbox was created using the public part of the key pair and has to be decrypted using the private part. An edge from a lockbox node to an asymmetric key node means that the lockbox contains the private part of the asymmetric key pair. Lockbox nodes additionally store the size of the ciphertext they encrypt.

As the semantics of the node and edge types prohibit certain anomalies such as a lockbox node without an outgoing edge, the following requirements are specified for a key graph to be *consistent*:

- Each key node must have at least one incoming edge, as otherwise, no user could ever retrieve this key in plain text.
- Each resource node must have at least one incoming edge, as otherwise, the resource would not be stored anywhere.
- Each lockbox node must have at least one[1] incoming edge and exactly one outgoing edge.
- Each signature node must have exactly one incoming edge—starting at the key used for signing—and at least one outgoing edge, as a signature can sign multiple lockboxes.

To ensure that the key graph is always in a consistent state, *atomic manipulations* on the key graph, i.e., adding or removing a node or edge of a certain type, are encapsulated in *transactions*. An exemplary transaction is $AddEncryptedResource(r, size, k, u)$,

[1]The use of so-called “broadcast secret schemes” can lead to more than one incoming edge for a lockbox.

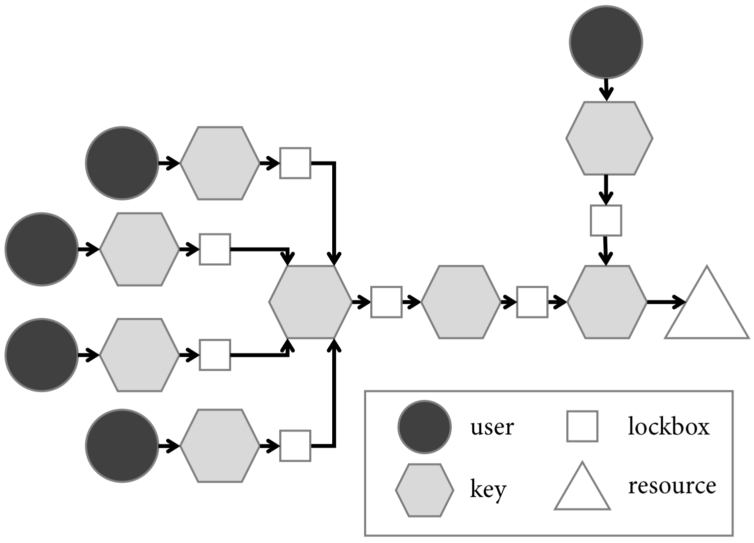


Figure 4.1: Visualization of an exemplary key graph.

which indicates that user *u* encrypted resource *r* of size *size* with key *k* and uploaded the encrypted resource to the storage provider. The manipulations made by this transaction are listed in Listing 4.1 , and visualized in Fig. 4.2.

```
AddResourceNode(newResourceId);
AddLockboxNode(newLockboxId, encryptedResourceSize);
AddLockboxResourceEdge(newLockboxId, newResourceId);
AddKeyLockboxEdge(encryptionKeyId, newLockboxId);
AddUserLockboxEdge(creatorId, newLockboxId);
```

Listing 4.1: Sequence of atomic key graph operations in transaction *AddEncryptedResource*

An SDS protocol uses the key graph transactions to manipulate the key graph when triggered by a workload event. From this perspective, an SDS protocol can be considered as a set of procedures, with one procedure per workload event type. Each of these procedures takes the workload event parameters and the current key graph as input, and carries out a sequence of key graph transactions. The control flow of the procedure may be defined by using arbitrary control flow constructs, such as branches or loops.

Note that this definition of a key graph cannot be used to evaluate whether a user can or cannot access a resource, i.e., the definition cannot be used for a reachability analysis in an access control sense. The reason for this is that the definition allows to delete lockbox and signature nodes, which simply reflects that the corresponding cryptographic data structures are deleted. Opposed to that, a model that evaluates the reachability would probably assume that cryptographic data structures are never

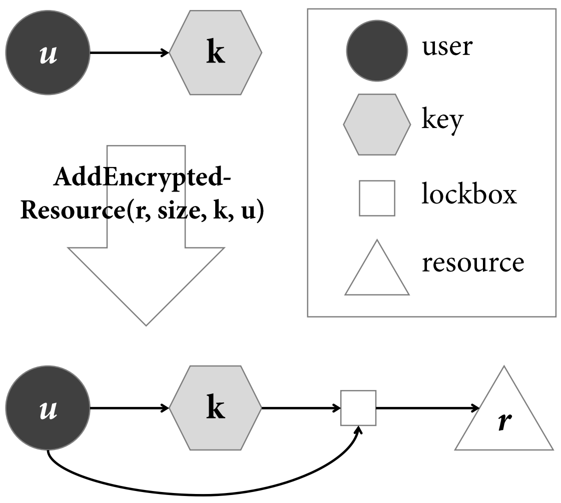


Figure 4.2: Manipulation of key graph by transaction *AddEncryptedResource*.

deleted. However, a "reachability evaluation" key graph can easily be constructed from a "performance evaluation" key graph as defined above by simply skipping all node or edge removals.

4.2.2 Simulative Performance Evaluation Method

The simulative performance evaluation method is based on a state machine. The input symbols of this machine are the workload events defined in Section 3.2. The states are key graph states, i.e., instances of the key graph as specified in the previous subsection. The state transition function is determined by the SDS protocol. Finally, the output of each transition is the estimated incoming and outgoing network traffic and computation time that the acting user needed to carry out the respective key graph manipulation.

Formally, we define the simulative performance evaluation method as follows:

Definition 21. The *simulative performance evaluation method* is a state machine $(\mathcal{E}, \mathcal{R}, \mathcal{K}, kg_0, \delta, \omega)$ with:

- input symbols from $\mathcal{E}$: the set of workload events as specified in Section 3.2;
- output symbols from $\mathcal{R}$: the set of tuples $(u, t_{comp}, n_{in}, n_{out})$ where u is a workload user (cf. Section 3.2), and t_{comp}, n_{in} and n_{out} are the required computation time, incoming and outgoing network traffic volume, respectively;
- states from $\mathcal{K}$: the set of consistent key graph states;
- the starting state kg_0: the key graph state with $V = \varnothing$ and $E = \varnothing$;
- the state transition function δ, which is the key graph transformation function: $\delta\colon \mathcal{K} \times \mathcal{E} \rightarrow \mathcal{K}$;

– the output function ω, which is the performance function: $\omega\colon \mathcal{K} \times \mathcal{E} \rightarrow \mathcal{R}$

In the following, the different components of this definition are discussed.

The set of states in our state machine is specified as the set of consistent key graph states $\mathcal{K}$. In the former subsection, the consistency requirements for a key graph were described, and it was argued that an SDS protocol manipulates the key graph by means of key graph transactions. For the simulative method, this implies that the SDS protocol determines the transitions between key graph states when triggered by a workload event $e \in \mathcal{E}$. Thus, to model an SDS protocol, it must be specified how the key graph transformation function δ is calculated.

This leads to the following definition:

Definition 22. An *SDS protocol model* is a function $\delta\colon \mathcal{K} \times \mathcal{E} \rightarrow \mathcal{K}$ that, given a consistent key graph instance k_0 and a workload event e, transforms k_0 to a new key graph instance k_1 by applying an arbitrary number of key graph transactions.

As shown in the previous subsection, the state transition function can be implemented as a set of procedures, where each procedure is triggered by a certain workload event type.

The output of the method are the costs required for a key graph state transition. More specifically, the output states the required computation time and incoming and outgoing network traffic volume for the user that carried out the key graph manipulation. The performance function ω maps a key graph state transition to concrete values for the costs, i.e., it assigns costs to each manipulation made on the key graph. The complete implementation of the performance function is omitted here, as most of the cost assignments are quite obvious. As an example, the *AddEncryptedResource* transaction that is shown in Fig. 4.2 incurs computation time to user u for the encryption of resource r, and it incurs outgoing network traffic volume to upload the encrypted resource r to the storage provider.

4.2.3 Limitations and Relation to Analytical Method

The simulative method is based on some assumptions that are necessary due to the abstraction by the key graph. These assumptions generally require that generation and transfer of cryptographic data structures are always coupled. For example, it has to be assumed that a signature that is uploaded to the storage provider was generated just before the upload, or that a downloaded lockbox is decrypted right after the download. Thus, the upload of already existing signatures or lockboxes, or the cloning of keys without actually generating them cannot be modeled.

In the remainder of this subsection, the differences between the analytical and the simulative method are discussed. The most obvious difference is that for the simulative method, fully specified workloads complying with the definition given in Section 3.2 are required. Opposed to the analytical method, it is not enough to provide a set of workload parameters, which is challenge for using this method in practice. However, depending on the set of known workload parameters, the workload generation method shown in Section 3.6 might be sufficient to prepare a workload as input for the simulative method.

Furthermore, the analytical method tends to make fundamental influence parameters for the performance more emergent: While the different procedures that form the protocol model also have to be built by a manual analysis, the simulative protocol model is closer to a given pseudocode representation than in the analytical case. This can be considered a drawback of the simulative method, as relations between the workload parameters and the performance can hardly be understood by inspection of the protocol model. At the same time, the process of protocol modeling is usually simpler than for the analytical method.

A further difference between the analytical and the simulative method is the set of SDS protocols that the respective method can be applied to. The simulative method can be applied to only a subset of SDS protocols that can be evaluated with the analytical method. The reason for this is that an SDS protocol can only be simulated if it retains to the cryptographic structures of keys, lockboxes and signatures. On the other hand, as long as this requirement is fulfilled, SDS protocols using arbitrarily complex cascades of keys, lockboxes and signatures can be simulated.

There are methods that "lie between" the analytical and the simulative method in some sense. For example, the simulative method could be altered by omitting the key graph, and just summing up the costs of the cryptographic operations instead. This would not influence the accuracy of the method, and would even enhance the set of SDS protocols the method can be applied to. However, the key graph-based method has two advantages: First, the method specifies a fixed abstraction level, i.e., it specifies the operations that have to be included in the model. This promotes a fair comparison between different SDS protocols. Second, from a simulation management perspective, the key graph holds the complete state of the authorization model, which is usually required by the SDS protocol. In some cases, the information stored by the authorization model is not sufficient for the SDS protocol. For example, a protocol may require information about the number of group member additions and removals that occurred in a given user group in the past. If this information is required, it can typically also be inferred from the key graph. Therefore, the key graph manages the major part of the simulation state, and supersedes an additional implementation of the state management.

Another possibility to alter the simulative method is to change the output function in a way that is does no longer yield fixed costs, but one or more expressions according to the analytical model (cf. Section 4.1). These expressions already include an evaluation of the workload parameters, but the costs of cryptographic operations are still expressed as variables. If the total operation costs depend on only one workload parameter, a single expression is sufficient. If the expression costs depend on more than one workload parameter, the output function yields a vector of expressions, with one vector element for each workload event of the given operation type. These vector elements can be considered as raw expressions in the terminology of the analytical method. The resulting expressions are evaluated by inserting concrete costs for cryptographic operations. The advantage of this adapted simulative method is the decoupling of the simulation and the simulated hardware, as the output is hardware independent. Thus, the SDS performance of different hardware could be evaluated with only one simulation run per workload and SDS protocol. While this adaptation potentially reduces the number of necessary simulation runs, it does neither influence the accuracy of the method, nor does it enhance the set of SDS protocols the method can be applied to. Therefore, the adaptation can be regarded as future work.

4.3 Conclusions and Outlook

In this chapter, it was argued that a performance evaluation of SDS protocols regarding their asymptotical behavior is not sufficient to assess the real-world deployability. Two performance evaluation methods were presented that both yield concrete costs for an SDS protocol employed under a given workload. The analytical method requires only workload parameters instead of fully specified workloads. However, the analytical method has limitations when the performance of a given operation is influenced by more than one workload parameter, and statistical measures other than the average should be analyzed. The simulative method overcomes this drawback, but requires fully specified workloads, which might be unavailable in many cases. In some of these cases, the required workloads can be generated according to the generation method presented in Chapter 3. In the remaining chapters of this thesis, both methods are used to carry out performance evaluations of different SDS protocols.

While the performance evaluation methods presented in this chapter can be used to estimate the expected performance, they do not give any advice on whether the performance is acceptable for the end users in practice. One approach to assess this question is to compare the expected performance of secure data sharing with that of "insecure" data sharing, assuming the same workload. Therefore, the overhead of security could be evaluated as "the price to pay". For the required network traffic, the analytical performance evaluation methods allow to separate the transfer size of the shared data from the size of the cryptographic data structures. The simulative method can easily be enhanced to do so. Thus, both methods can be used to assess the overhead of secure data sharing in terms of network traffic. On the other hand, the computation times for all "administrative" workload events, such as *GrantUserReadAccess* or *AddUserToGroup*, can be assumed as zero in the insecure case, making the overhead assessment trivial. An overhead assessment of the computation time for *ReadData* and *WriteData* workload events is challenging and not supported by the performance evaluation methods presented in this chapter. The reason for this is that the computation time is highly dependent on both the media type of the shared document, such as a spreadsheet or a movie, and the concrete client application used for processing the document. However, the media type is not part of the workload definition given in Section 3.2. It is considered future work to incorporate the media type into both the workload definition and the performance evaluation methods.

Another future research direction might be the consideration of further performance dimensions. For example, the network latency caused by an SDS protocol would be a relevant metric, since it has a strong impact on the waiting time experienced by the end user. The latency is heavily influenced by the number of roundtrips between end user and storage provider that are required for a certain operation. The challenge of evaluating the number of roundtrips is that it depends not only on the SDS protocol itself, but also on the implementation of the protocol. This is especially the case if the exact cryptographic data structures that have to be retrieved from the storage provider are not known to the client at the beginning of an operation, and have to be determined dynamically. Without modeling implementation details, it is impossible to tell whether the client fetches the required cryptographic data structures one by one, or fetches all potentially required data structures in one operation, which trades a low number of roundtrips for a higher volume of transferred data. As further possibility, the storage provider might be able to deliver exactly the required data structures. However, in this case, the storage provider has to know the SDS protocol that is used, which again is an implementation issue. Thus, considering the latency requires a different model of an SDS protocol that also takes more implementation details into account.

5
E2E-SDS Using Joint Authorization Operations

In this chapter, the performance evaluation methods presented in the previous chapter are applied to different E2E-SDS protocols. The focus of this chapter is on E2E-SDS protocols that support named user groups, i.e., user groups that are managed by a dedicated group owner. This feature is part of popular distributed file systems, such as NFSv4 and CIFS. The details of the group-based authorization model that is realized by the protocols are described in Section 3.1.

The protocols analyzed in this chapter employ joint authorization operations, i.e., operations that are carried out by at least two different users in a collaborative manner. These operations require that the involved users are reachable in the network and able to allocate the necessary computational resources immediately upon request. On the one hand, this limits the practical applicability of the protocol especially in conjunction with mobile user devices. On the other hand, joint authorization operations constrain the attack vector for certain attacks on E2E-SDS protocols, which will be detailed in this chapter. A novel E2E-SDS protocol that omits joint authorization operations is introduced in Chapter 6.

The chapter starts with a specification of the requirements that arise from the combination of E2E-SDS and the support for named user groups in Section 5.1. In Section 5.2, attacks on E2E-SDS protocols are discussed with regard to necessary attacker capabilities, prerequisites, and countermeasures.

The main contributions of this chapter are performance evaluations of concrete E2E-SDS protocols. The computation time and network traffic volume required by the different authorization operations are evaluated by means of the two methods introduced in Chapter 4. The performance evaluations are based on the real-world workloads that were characterized in Section 3.5.

In Section 5.3, the performance of an E2E-SDS protocol is evaluated that is similar to protocols that are in productive use today. This protocol is based on traditional cryptographic primitives, i.e., symmetric and asymmetric cryptography that produces ciphertexts dedicated to a single recipient only. To improve the performance of the protocol for large user groups, it is extended with an existing Group Key Management approach.

Moreover, the introduced performance evaluation methods were applied to an E2E-SDS protocol based on Attribute-Based Encryption (ABE). On the one hand, ABE promises to enable resource-saving SDS systems in terms of ciphertexts that have to be generated and distributed. On the other hand, some ABE schemes were shown to exhibit a performance worse than traditional cryptography, at least in small and artificial sharing scenarios. In Section 5.4, first an ABE scheme is identified that facilitates E2E-SDS, which conflicts with the need for a central trusted entity that many ABE schemes bring along. Second, the attribute-based authorization model of a suitable ABE scheme is mapped to the group-based authorization model. Three different alternatives for the mapping were identified, and the performance of each mapping alternative is evaluated in Section 5.5. Finally, the performance of the E2E-SDS protocols based on traditional cryptography and the ABE-based protocol are compared.

Conclusions and future work are presented in Section 5.6.

The content presented in this chapter has partially been published before in [KH13], [KH14], [KH15] and [KH16].

5.1 Requirements for E2E-SDS Supporting Named User Groups

One objective of this thesis is to evaluate the performance of E2E-SDS protocols that support named user groups. In this section, the requirement on E2E-SDS protocols with support for named user groups are stated explicitly. The requirements are derived from the definition of E2E-SDS as given in Section 2.2, and from the group-based authorization model introduced in Section 3.1. This section strives to compile and summarize the requirements, and to discuss requirements that arise from the combination of E2E-SDS and the support of named user groups.

After a recap of the entities in the system and the authorization model, the required expressiveness of authorization policies is discussed. Thereafter, requirements regarding the trust between the system entities are stated. Finally, requirements on the revocation of access permissions are introduced. The numbering of a part of the requirements serves as a reference for the evaluation of ABE schemes presented in Section 5.4.1.

5.1.1 System Entities and Authorization Model

The system entities that strive to share data are represented by users. The shared data is stored at a storage provider as a set of resources, where a resource constitutes a data chunk that can be shared independently from other data chunks. Each resource has a resource owner that grants and denies read or read and write access to other users—referred to as direct permissions—or to named user groups. A named user group—or simply "group" for the remainder of this thesis—is a set of users that is managed by a group owner, and exists independently from any permissions on resources. Each resource owner might grant access on her resources to arbitrary groups. The described group-based authorization model was detailed in Section 3.1.

5.1.2 Expressiveness of Policies

To be able to implement the group-based authorization model sketched in the previous section, fine-grained access control policies are required that can be applied on a per-resource basis. These policies have to be expressive enough to grant read and additionally write access to arbitrary users and groups (Req.1). Especially, the policies must allow the owner of a resource to grant access to groups owned by different group owners (Req.2). For example, a resource owner could specify that her resource r can be accessed by all members of g_1 owned by user u_a, and by all members of g_2 and g_3, which u_b owns. On the other hand, a resource owner should not have to be forced to share data with at least one group from each group owner (Req.3).

5.1.3 Trust and Authorization Process

The essential property of E2E-SDS is that the authorization and access control enforcement on the shared data is carried out solely by the users that are allowed to access the data. Thus, no entity outside the set of users that is allowed to access a resource has to be trusted to protect the confidentiality or integrity of the resource. This also applies to the storage provider, which is not trusted to protect the confidentiality or the integrity of the stored resources in any way. Note that the storage provider is trusted to protect the availability of the shared data by enforcing write access control. The avoidance of a trusted third party especially implies that there cannot be a central authority that is technically able to access all of the resources (Req.4).

Regarding the trust between users, users that have access to a common shared resource trust each other not to disseminate the data outside the sharing group. Furthermore, there are no system-wide "administrative" or "privileged" users. Therefore, every user may act as resource or group owner for an arbitrary number of resource or groups, respectively (Req.5).

Regarding the authorization process, an independence of user actions is required in a sense that one user might not—deliberately or by intention—stop another user from carrying out an authorization operation. A strict formulation of this requirement would be that each authorization operation involves only the user that initiated the operation, and does not require the cooperation of another user, i.e., joint authorization operations are omitted altogether. However, this strict formulation is not fulfilled by any E2E-SDS protocol found in literature so far. Therefore, the requirement is relaxed to the avoidance of joint authorization operations, except for the case of group member removal operations (Req.6).

Finally, the order of authorization operations should not be constraint, apart from trivial constraints like that a user has to be granted a permission before the permission can be revoked (Req.7).

5.1.4 Revocation

E2E-SDS requires that it is possible to revoke user access permissions at any point in time. This applies to both the revocation of direct user permissions, and to group memberships. The permissions must be revocable independent from each other, i.e., it must be possible that a user loses access to resource r_1, but keeps access to resource r_2; the same applies for group memberships (Req.8).

The requirements stated so far have implications on the way revocation could be implemented: The revocation cannot be implemented by use of a proxy that holds some cryptographic information that is needed for decryption, and controls access to this information using logical access control (Req.9). The reason for that is that in this case, the proxy could easily share the cryptographic information with a revoked user, and has to be trusted to protect the confidentiality of the shared data. Furthermore, revocation cannot be implemented by leveraging multiparty computation of all users who keep their permission (R.10), because this would conflict with the independence of authorization operations as stated in Req.6. Finally, revocation cannot be implemented by enforcing an expiration date for access permissions or group memberships (R.11). This would also conflict with Req.6: The resource or group owner would stop the permitted users to read or write the resource if she omits to renew the required cryptographic material in time.

For revocation of read access, *lazy revocation* is considered sufficient, as many SDS protocols incorporate this concept (cf. Section 2.2). Lazy revocation allows the revoked user to continue reading the resource as long as its contents have not been changed.

5.2 Attacks on E2E-SDS Protocols

The goal of E2E-SDS is to protect the confidentiality and integrity of shared data that is stored at the storage provider. However, the question whether these security objectives are achieved by a certain E2E-SDS protocol can only be answered with regard to the capabilities of the attacker. In this section, possible attacks on the confidentiality and integrity of shared data are discussed with regard to different attacker models.

An overview of different attacks is provided in Table 5.1. The table is not meant provide a complete overview of all possible attacks, but is restricted to attacks commonly found in literature. Attacks on the cryptographic primitives that are used by the E2E-SDS protocol, and attacks on the devices of the users are out of scope. In the following, the different attacks are discussed.

The first two attacks listed in Table 5.1 were already discussed in Section 2.2: An attacker with read-only access on the data stored at the storage provider might obviously endanger the confidentiality of the shared data, which can be prevented by em-

attacker capabilities	**attack**	**security objective attacked**	**further prerequisites for attack**	**countermeasure**
read access on storage provider	read data	confidentiality of an attacker-chosen resource	none	encrypt resources
read and write access on storage provider	manipulate encrypted data	integrity of an attacker-chosen resource	data cipher is vulnerable to tampering	digitally sign resources
	key graph forgery	confidentiality or integrity of an attacker-chosen resource	protocol specifics	digitally sign key-to-function bindings
	rollback of key graph and resources	freshness of an attacker-chosen resource	none	periodic refreshment of resource signature (single resource writer), or SUNDR or similar (multiple resource writers), or local caching of resources
	forking of key graph and resources	freshness of an attacker-chosen resource wrt. to a certain group of readers	multiple resource writers	impossible within E2E-SDS trust model
read and write access on storage provider, collaboration with revoked user	rollback of key graph and resources	integrity of a resource the revoked user had access to	revoked user had write access to resource before	periodic refreshment of resource signature (single resource writer), or SUNDR or similar (multiple resource writers)
	forking of key graph and resources	integrity of a resource the revoked user had access to wrt. to a certain group of readers	revoked user had write access to resource before, multiple resource writers	impossible within E2E-SDS trust model

Table 5.1: Overview of attacks on confidentiality and integrity of shared data in E2E-SDS. The countermeasures described in one row are to be applied in addition to all countermeasures described in the rows above.

ploying client-side encryption. A stronger attacker might have both read and write access on the storage provider. This obviously applies to the storage provider itself, but could also be achieved by an attacker leveraging some vulnerabilities in the software used by the storage provider. Such an attacker can try to tamper with the encrypted data. Whether it is possible to tamper with encrypted data in a controlled way depends on the cipher that is used for encryption. Unauthorized manipulations of the shared data can be prevented altogether by applying digital signatures in addition to the encryption.

The next attack is referred to as *key graph forgery attack* in this thesis. A key graph comprises cryptographic data structures that are stored at the storage provider, such as lockboxes or signatures, and was defined in Section 4.2.1. The authorization state of an SDS system at a given moment is determined by the state of the key graph. If the integrity of the lockboxes is not protected, the key graph, and thus, the authorization state of the SDS system might be manipulated by an attacker that has read and write access to the storage provider. Such an attack might be carried out by using a public key that is part of the legitimate key graph to encrypt a forged key that is known by the attacker, which can then be the root of a part of the key graph that is completely under the control of the attacker. A more detailed description of the key graph forgery attack, its necessary preconditions and the common countermeasure are given in the following.

The final goal of the key graph forgery attack is to breach the confidentiality or integrity of an attacker-chosen resource. For this purpose, the attacker places one or more forged keys in the key graph, which are subsequently used by legitimate users to encrypt further keys or resources. To be able to carry out this attack, the attacker is required to know the specifications of the E2E-SDS protocol that is employed. For the attack to work, the attacker does not have to be a legitimate user in the E2E-SDS system, i.e., a user that knows a part of the key graph, nor does the attacker have to collaborate with a legitimate user.

The feasibility of the key graph forgery attack strongly depends on the concrete E2E-SDS protocol that is employed. The E2E-SDS protocol is vulnerable to the attack if at any instance in time, the protocol yields a key graph with certain properties. To describe those properties conveniently, some terms are specified for the context of this section. A *subgraph* of the key graph is defined by a subset of the key graph nodes, and is required to contain any key graph edge that connects two nodes within this subset. A *descendant-complete subgraph* is a subgraph that, if it contains a certain node, it also contains all *descendant nodes* of this certain node, i.e., all nodes that can be reached from this certain node by traversing the directed key graph edges. A *source node* of a subgraph is a node that belongs to the subgraph, but does not have any incoming edges that are part of the subgraph. Similar to that, a *sink node* is a subgraph node without any outgoing edge within the subgraph.

An E2E-SDS protocol is vulnerable to the key graph forgery attack regarding resource r if at any instance in time, the key graph instance representing the current authorization state contains a descendant-complete subgraph with the following properties:

- The target of the attack, resource r, is part of the subgraph.
- The key that will be used to encrypt the resource r in the next write event is part of the subgraph.
- Every source node of the subgraph is a public key.
- The plausibility of the content of any resource in the subgraph is not checked by the writer, i.e., the resources are not downloaded and checked before writing.

When these conditions are fulfilled, the attacker might carry out the attack by forging the descendant-complete subgraph, except for the source nodes. For this purpose, the attacker replaces every non-source key in the subgraph with an arbitrary key by overwriting all lockboxes that belong to the subgraph at the storage provider. The resources in the subgraph are replaced by arbitrary ciphertexts. To connect the forged subgraph to the genuine parts of the key graph, the attacker uses the public keys that are related to the source nodes of the subgraph to generate the missing lockboxes. As soon as a legitimate user wants to write resource r, this user downloads the forged encryption key and encrypts the resource with this key. In consequence, the encrypted resource can also be decrypted by the attacker.

The common countermeasure against the key graph forgery attack is to sign the binding between each key and its function in the E2E-SDS protocol. This mechanism is employed by existing E2E-SDS protocols, such as SiRiUS [GSMB03] (cf. Section 2.2). This way, each user that retrieves a key can check whether this key was created by a legitimate user for the intended function. The set of key functions that an E2E-SDS protocol requires to work strongly depends on the protocol. The E2E-SDS protocols that are considered in this thesis usually require key functions that on the one hand refer to a user, a named user group, or a resource, and on the other hand describe the purpose of the key, such as encryption or signing. Thus, a typical example of a key function is “signing key for resource r”.

Signing the binding of keys and their functions bears a challenge with regard to the freshness of the signatures: While the signatures guarantee that a key was bound to a function at some past instance in time, they cannot guarantee that the key is still bound to the function at the moment of validation. This shortcoming enables a *rollback attack*, which allows an attacker to replace the current key bound to a function with an older one, and also replace the regarding resources with older ones.

Note that an attacker cannot roll back arbitrary keys in isolation. The granularity of rollback attacks is limited by the fact that the E2E-SDS protocols analyzed in this thesis always update descendant-complete subgraphs as a whole. The reason for this is that the protocols update keys only when access of a user to a certain key should be revoked. In this case, all keys that can be inferred from this key are renewed as well, which are exactly the keys in the descendant-complete subgraph. As a consequence, only descendant-complete subgraphs as a whole can be rolled back, i.e., if a certain key node is rolled back, all descendant nodes of this key node have to be rolled back as well.

A possible countermeasure against the rollback attack is a periodic refreshment of the signature of the key-function binding. For this purpose, an expiration date is incorporated into the signatures, and the signature is periodically refreshed before expiration. This way, a rollback attack is only possible as long as the rolled back signature is not yet expired. The limitation of this solution is that it can only be applied when only a single user generates the respective signatures: Periodic refreshing requires to check the validity of the signature before refreshment by comparing a fingerprint of the signed data on the storage provider with a local copy of the fingerprint. When multiple users are allowed to generate a signature, the data on the storage provider and the fingerprint may have been changed by another legitimate user in the meantime.

When a periodic refreshment of the key-function binding signature is applied to a certain key in the key graph, it protects every descendant-complete subgraph that contains this node from being rolled back as a whole. Although this protects the freshness of a descendant-complete subgraph as whole, it constitutes only a limited protection for each "child" subgraph of the descendant-complete subgraph. When the child subgraph does not contain the key that was protected by a periodic signature refreshment, the child subgraph can still be rolled back within a certain time range: In this case, the child subgraph can be rolled back to the time of the latest change on the "parent" subgraph.

Although all of the E2E-SDS protocols analyzed in this and the next chapter leverage signatures of key-function bindings with periodic refreshment, they differ in the size of the descendant-complete subgraphs that cannot be protected by the periodic refreshment. The protocols presented in this chapter protect the whole key graph except for the resources. On the other hand, with the protocol presented in the next chapter, certain parts of the key graph cannot be protected by periodic signature refreshment, which is the result of a trade-off between security and practical applicability (cf. Section 6.2).

Countermeasures against a rollback attack in case of multiple users that are allowed to write a resource or generate a key are provided, e.g., by SUNDR [LKMS03] or Cachin et al. [CSS07]. However, these approaches have several drawbacks, e.g., in terms of required network traffic or the underlying trust model. Furthermore, the approaches require the user devices to hold a local state, thus, switching the device requires to synchronize the local state in an integrity-preserving way.

A further countermeasure against rollback attacks constitutes the local caching of resources, as it is realized by Sync&Share services. If both the local and the remote copy of a resource are signed with a timestamp, a rollback of the resource can be detected by comparing the timestamps of the signatures before updating the local copy. In fact, local caching of resources provides a comparatively strong protection against rollback attacks on the key graph: A descendant-complete subgraph contains at least one resource node in almost any case. This prevents the whole subgraph from being rolled back.

The last attack discussed in this section is the *forking attack*. The goal of this attack is not to replace a part of the key graph by an older version. Instead, a forking attack aims to partition the set of users that is allowed to access a part of the key graph, and a user in one partition can no longer see the changes made by a user in any other partition [MS02]. This way, forking allows to hide changes made to a resource from other users, and thus endangers the freshness of the resource. However, a forking attack has the limitation that different forks of a state can never be joined again without detection.

A countermeasure against forking is provided by signing the key-function binding or the resource, and periodically refreshing the signature. Again, this only works in case of a single user that is allowed to update the key or the resource. In case of multiple users updating the key or the resource, it was shown that forking cannot be prevented without a trusted third party [LKMS03]. When an E2E-SDS protocol protects the freshness of the encryption keys by means of periodic refreshment, a resource can only be forked as long as the encryption key is not renewed, plus the expiration of the signature of the old encryption key. As this applies to the protocols introduced in this chapter, the attack vector for forking is rather limited.

Finally, an attacker is considered that has not only read and write access to the storage provider, but also collaborates with a revoked user, and targets resources that the revoked user was allowed to write before the revocation. Under this assumption, a rollback attack might not only endanger the freshness of a resource, but also its integrity: A rollback attack now allows an attacker to replace the current key bound to a function with an older one that is known to the revoked user. This attack can be prevented by a periodic refreshment of signatures in case of a single writer, and SUNDR and similar approaches in the case of multiple writers. However, the local caching of resources cannot prevent this kind of attack: If the attacker and the collaborating user regain write access to a resource through the rollback attack, they actually do not have to roll back the resource, but can write the resource immediately after the rollback of the related keys. From the perspective of a user reading the resource afterwards, this looks like a legitimate update of the resource.

With regard to an attacker that collaborates with a revoked user, forking also endangers the integrity of the resource: Forking allows the attacker to create an additional view on the key graph, where the revoked user still has access permissions. Again, signing the key-function binding or the resource, and periodically refreshing the signature prevents forking of a key. In consequence, an attacker cannot gain any additional advantage from the collaboration with a revoked user if the protocols presented in this chapter are employed.

5.3 Performance of E2E-SDS Using Traditional Cryptography

The E2E-SDS protocols studied in this section are based on traditional cryptography, i.e., symmetric and asymmetric cryptography that produces ciphertexts dedicated to a single recipient only. Thus, the protocols use lockboxes, which are encrypted keys that serve the purpose to make the encrypted key available to everyone who knows

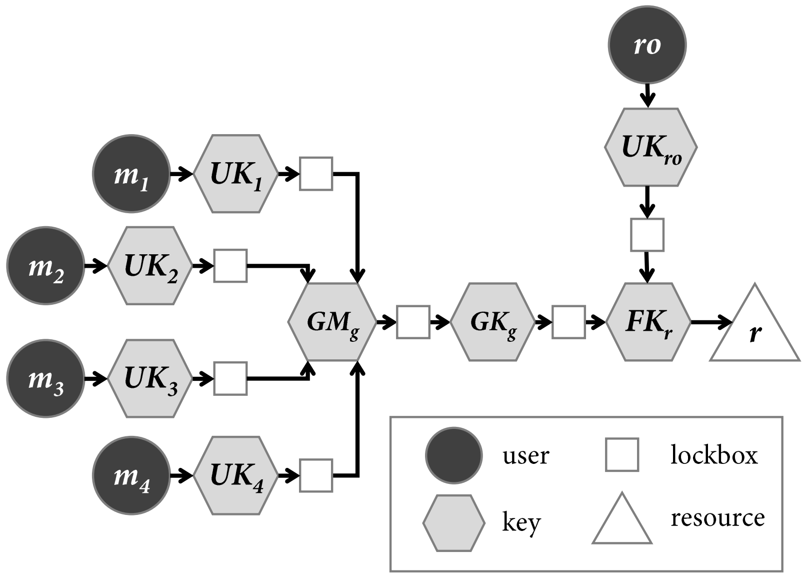


Figure 5.1: Exemplary key graph for basic protocol.

the encrypting key, and digital signatures (cf. Section 2.2). The protocols fulfill all of the requirements on E2E-SDS and a support for named user groups stated in Section 5.1. Thus, they ensure confidentiality and integrity of shared data against an attacker that has full access to all the data stored at the storage provider and fully controls the access control enforcement of the storage provider, no matter if attackers are internal or external to the storage provider.

A prerequisite for the protocols are non-compromisable facilities that allow each user to retrieve the public keys used in the protocols with guaranteed authenticity and integrity.

5.3.1 Protocols Under Study

As no existing E2E-SDS protocol could be found in literate that fulfills all of the requirements stated in Section 5.1, existing protocols were combined for this purpose. Therefore, the protocols presented in this section combine the basic workings of SiRiUS [GSMB03], including signatures of the key graph and periodic refreshment of these signatures, with named user groups as realized by Cepheus [Fu99] (cf. Section 2.2). The protocols also exhibit similarities to the protocol employed by Boxcryptor 2.0[1], which is an existing service that supports encrypted data sharing.

In this chapter, two protocols based on traditional cryptography are presented: the *basic* and the *extended* protocol. In the following, the cryptographic operations required to realize each authorization operation will be described.

The key graph that the basic protocol generates to grant group g read access on resource r is depicted in Figure 5.1. The two protocols have in common that each user u and each group g is assigned an asymmetric user key pair UK_u or group key

[1] `https://www.boxcryptor.com/en`

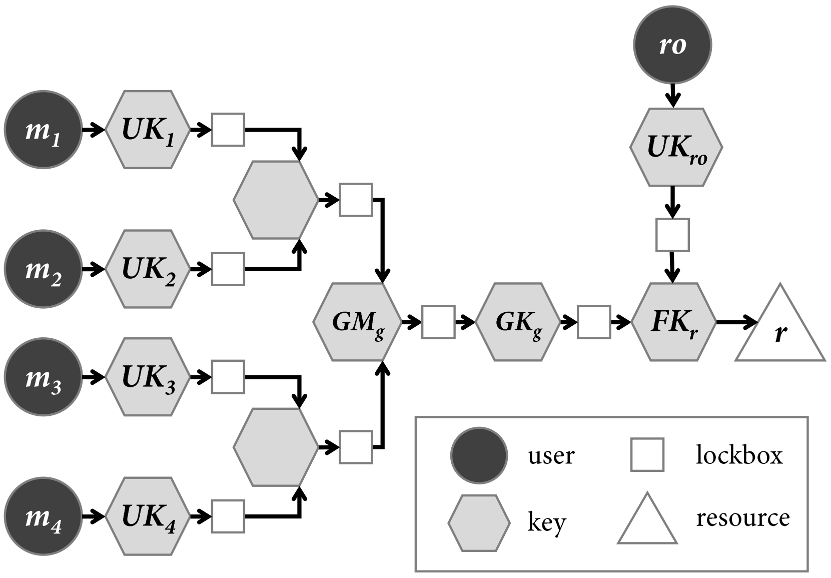


Figure 5.2: Exemplary key graph for extended protocol.

pair GK_g, respectively. Each resource is encrypted with a symmetric resource key RK_r. To grant read access on resource r to user u or group g, RK_r is encrypted with UK_u^{pub} or GK_g^{pub}. Furthermore, the resource is signed with the private part of an asymmetric resource signing key pair RSK_r^{priv} (omitted in Figure 5.1), which can be encrypted with UK_u^{pub} or GK_g^{pub} to grant write access to user u or group g.

The two protocols differ in the way GK_g is distributed to the group members. In the *basic protocol*, GK_g is distributed to the group members in a simple way by creating a symmetric master key GM_g to encrypt GK_g^{priv} and encrypting the master key with the public key UK_m^{pub} of each group member m. When a user leaves the group, GK_g and GM_g are renewed and again encrypted with the public key of each remaining group member.

In the *extended protocol*, a well-known Group Key Management approach called *Logical Key Hierarchy (LKH)* [WHA99] was integrated to avoid revocation costs that are linear in the group size. The decision for LKH was based on an evaluation of Group Key Management approaches with regard to their asymptotical performance, which was published in [KH13]. With LKH, GM_g is the root of a tree of keys, where each key is encrypted with each of its child keys (cf. Figure 5.2). The leaf keys are encrypted with the public keys of the group members. The basic idea is that when a group member is removed, only the keys on the path from the removed user's key pair to the root of the tree and all lockboxes that encrypt one of these keys must be renewed. Thus, the number of renewed keys and lockboxes grows only logarithmically with the group size.

Both protocols can be configured to use *eager* or *lazy revocation*, which applies when a user or a named user group loses access to a resource. With eager revocation, the resource owner renews the resource encryption key and re-encrypts the resource with the new key, which requires her to upload the resource again. With lazy revocation, the resource owner only renews the resource encryption key RK_r, and the re-encryption of the resource is delayed until the next time the resource is written.

The protocols leverage joint authorization operations when a user is removed from a group. After the owner of the group has renewed the group key, she collects the set of resources that can be accessed by the group, i.e., the set of *group-accessible resources*. The group owner informs every resource owner that owns at least one group-accessible resource about the revocation. The resource owner reacts by renewing the resource encryption key RK_r, and if the group has write access to the resource, also the resource signing key pair RSK_r. If eager revocation is employed, the resource owner also has to re-encrypt her group-accessible resources after a group member removal.

To prevent the attacks discussed in the previous section, the protocols digitally sign the binding of keys to their functions, and digitally sign the resources. Moreover, the signatures of all key-function bindings have to be periodically refreshed: For resource-related keys like RK_r or RSK_r^{priv}, the refreshment is done by the resource owner, while for group-related keys like GK_g^{pub}, the refreshment is done by the group owner. This prevents key graph forgery attacks altogether, and limits the attack vector of rollback and forking attacks to resources.

5.3.2 Performance Evaluation Results for Real-World Sharing Scenarios

In this section, the results of the performance evaluation of the basic and the extended protocol are presented. The performance evaluation was carried out by means of the simulative method introduced in Section 4.2, and yields the computation time and the network traffic volume required for each authorization operation. All evaluation results are related to the real-world workloads that are characterized in Section 3.5.

The evaluation is based on the assumption that AES with 256 bit keys is used for symmetric encryption and RSA with 2048 bit keys is used for asymmetric encryption and digital signatures, as these ciphers are also used by current services that allow for secure data sharing, such as the aforementioned Boxcryptor 2.0 service.

The computation times of the respective cryptographic operations were benchmarked on three devices. To include popular user devices that are prone to performance problems, two smartphones were benchmarked: an iPhone 3G produced in 2008, and a Samsung Galaxy S5 Mini produced in 2014. Additionally, a Dell Latitude E4310 notebook produced in 2011 was benchmarked. For this purpose, the OpenSSL speed benchmark[2] was used. The measured computation times for RSA and AES are shown in Table 5.2.

In short, the comparison shows that for RSA operations, the Galaxy outperforms the old iPhone 3G by a factor of about 7, while the notebook outperforms the Galaxy by a factor of about 6. When AES is considered, the differences between the devices

[2] `https://www.openssl.org/docs/man1.0.2/apps/speed.html`

	iPhone 3G	Galaxy S5 Mini	Latitude E4310
RSA 1024 sign/s	18.7	132.5	817.0
RSA 1024 verify/s	363.8	2 799.0	15 211.0
RSA 2048 sign/s	3.2	23.0	124.0
RSA 2048 verify/s	114.2	874.7	4 225.0
AES 256 KB/s	4 311	12 651	48 170

Table 5.2: Cryptographic operation benchmarks for different devices

are smaller: The Galaxy outperforms the iPhone by a factor of 3, and the notebook performs AES about 3 times faster than the Galaxy.

The focus of this section is to give realistic estimations of the performance of potentially resource-intensive authorization operations: The removal of a member from a large group, the upload of a big file, and the re-encryption of resources when a user loses access on the resource. The latter is required either when a user is removed from a group that has access to the resource, or when a directly assigned access to the resource is denied. The performance is analyzed separately for group owners, resource owners, and users that just up- and download data.

First, the performance of *write operations* is analyzed. The required computation time for a write operation depends on the size of the data written, and measurements show a strong linear correlation between the data size and the AES encryption time when data size exceeds 100 KB in the case of the Dell notebook. The encryption time for the largest file in the considered workloads, which comprises 1 056 MB, is 243 sec on the iPhone, 85 sec on the Galaxy and 19 sec on the notebook. However, for an average file with about 1.19 MB, the computing time is only about 20 ms, 92 ms, and 261 ms, respectively.

Besides the time needed for symmetric encryption and decryption of the shared data, a write operation usually requires additional computation time for further cryptographic operations, e.g., to decrypt lockboxes or similar. This additional computation time is referred to as *key management computation time* in this section. When the basic protocol is used, the key management computation time is approximately constant for each device: 955 ms for the iPhone, 134 ms for the Galaxy and 110 ms for the notebook. However, when the extended protocol is used and the write access is granted by a group membership, the computation time depends on the size of the

	iPhone 3G	Galaxy S5 mini	Latitude E4310
small groups	1.278	0.179	0.147
medium groups	1.280	0.180	0.147
large groups	1.296	0.182	0.148

Table 5.3: Average key management computation time (in seconds) for write operations when the extended protocol is used.

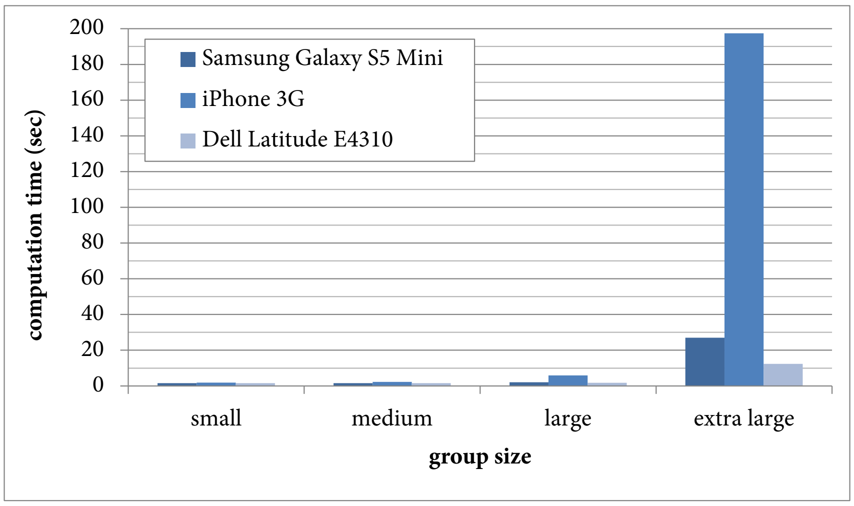


Figure 5.3: Average computation time of a group member removal operation for different group sizes when the basic protocol is used.

group. The reason for this is that a potential writer has to retrieve the current group key by traversing a path from a leaf of the underlying key tree to the root, and as the number of key tree leaves has to grow proportional with the group size, the depth of the tree also depends on the group size. The average key management computation time per device and group size bucket for the extended protocol is shown in Table 5.3. When the key management computation time is compared with the time required for the symmetric encryption of the data by looking at the proportion of both values, it is found that the proportion heavily depends on the device, and more specifically, on relation of the speed of symmetric versus asymmetric cryptography. In about 70% of all write actions, the data encryption time makes up less than 10% of the overall computation time and is therefore dominated by the key management computation time. This shows that despite the key management computation time accounts for a significant part of the total computation time for a write operation, the higher key management computation times induced by the extended protocol are still below 1.3 sec.

Next, the performance of *group member removal* operations is analyzed. These operations have to be processed by the owner of the group. Figure 5.3 shows the computation time required by the different devices using the basic protocol and illustrates that while the computation time is below 3 sec for small and medium groups, it can be significant for large or extra large groups. The maximum computation time for an extra large group was about 324 sec on the iPhone, 44 sec on the Galaxy and 19 sec on the notebook. Figure 5.4 shows the average computation time for the extended protocol. It shows that using the extended protocol increases the computation time for small, medium and large groups, but reduces the performance significantly for extra large groups, e.g., from about 197 sec to 14 sec when using an iPhone. As can be seen in Figure 5.5, the outgoing network traffic volume required to remove a member from a group is also significantly decreased when using the extended instead of the basic protocol, e.g., from 7 494 KB to 25 KB for extra large groups.

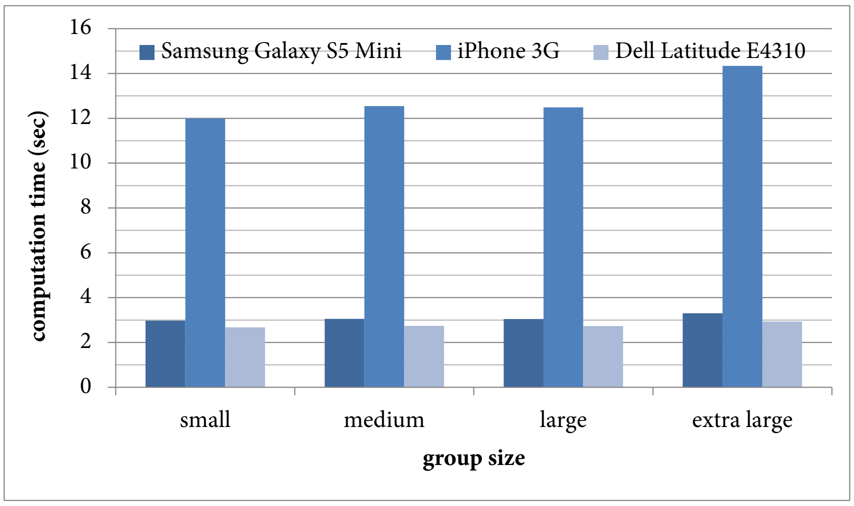


Figure 5.4: Average computation time of a group member removal operation for different group sizes when the extended protocol is used.

Finally, the performance of *re-encryption operations* is analyzed, which are carried out by the owner of the resource. Similar to write operations, re-encryption operations require computation time for key management tasks and, if eager revocation is used, for the decryption and encryption of shared data. As stated before, the data encryption and decryption time grows linearly with the size of the re-encrypted data. For the key management computation time, an average of 2543 ms was found on the iPhone, 1658 ms on the Galaxy S5 Mini and about 1627 ms on the notebook. Here again, the proportion of computation time that is required by key management operations and by the symmetric encryption of the data depends on the relation of the speed of symmetric versus asymmetric cryptography of the device. The results also show that the data encryption time exceeds the key management time when more than 5.8 MB of data are re-encrypted on the iPhone, more than 10.7 MB on the Galaxy S5 Mini and more than 35.5 MB on the notebook. In about 85% of all rekeying operations, the data encryption time makes up less than 10% of the overall computation time. Thus, the key management computation time dominates the encryption and decryption time also for re-encryption operations, which indicates that the use of eager revocation has typically not a huge impact on the required computation time. Furthermore, re-encryption operations require the upload of new lockboxes for each user or group that is still allowed to access the resource. The necessary network traffic volumes is rather small, as it ranges from 6.2 KB to 356 KB in our workloads, with 7.9 KB on average.

In summary, with regard to the considered workloads, only minor performance penalties have to be expected for most of the authorization operations, as the computation time for these operations is mostly up to some seconds, even on an iPhone 3G manufactured in 2008. However, significant performance penalties have to be expected for the management of group memberships in large user groups, i.e., user groups with a few thousand or more members. In this case, the computation times

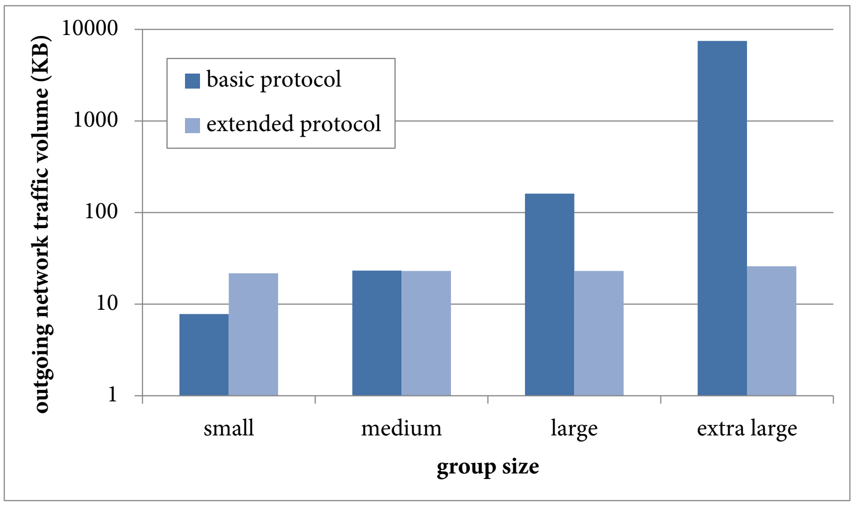


Figure 5.5: Average outgoing network traffic volume of a group member removal operation for different group sizes.

rise up to 324 seconds on the iPhone 3G. Furthermore, the group member removal operations require the group owner to upload several MB of cryptographic key material to the storage provider. With the extended protocol, these performance penalties are decreased significantly, as a group member removal operation can be carried out within few seconds, at the price of slightly raised computation times for smaller groups.

5.4 E2E-SDS Using Attribute-Based Encryption

Many SDS approaches proposed in the last few years are based on Attribute-Based Encryption (ABE) for confidentiality protection. One important reason for this is that ABE allows to dedicate a single ciphertext to multiple recipients. On the one hand, this promises to enable resource-saving SDS systems in terms of ciphertexts that have to be generated and distributed. On the other hand, some ABE schemes were shown to exhibit a performance worse than traditional cryptography in small and artificial sharing scenarios [WZSI14]. One objective of this thesis is to evaluate the real-world performance of an ABE-based E2E-SDS protocol in realistic sharing scenarios.

A first step towards this objective is to identify an ABE scheme that fulfills all of the following requirements: first, it satisfies the requirements on E2E-SDS as specified in Section 5.1, and second, it supports an authorization model that is sufficiently expressive to be adapted to support named user groups. More specifically, the ABE scheme should be adapted to support the authorization model specified in Section 3.1, which is referred to as *group-based authorization model* in this section.

These requirements are not fulfilled by the original proposal for ABE, which was introduced as *Fuzzy IBE* in the landmark paper of Sahai and Waters [SW05]. Fuzzy IBE relies on a central party that is able to decrypt any ciphertext in the system, thus, it does not adhere to the definition of E2E-SDS as stated above. While lots of work has been done to overcome the centralization issue in ABE, other issues regarding, e.g., policy expressiveness or revocation abilities render many ABE approaches inappropriate for E2E-SDS (cf. Section 2.3). Therefore, it is necessary to identify an ABE scheme which addresses all of these issues at the same time.

To identify an ABE scheme that facilitates E2E-SDS while offering an authorization model expressive enough to enhance it with support for named user groups, an evaluation of ABE schemes with regard to these requirements is presented in Section 5.4.1. The evaluation yields that only one approach complies with these requirements, which is DAC-MACS [YJR+13]. In Section 5.4.2, an overview of DAC-MACS is provided.

Due to the gap between the attribute-based authorization model of DAC-MACS and the aforementioned group-based authorization model, group-based authorization operations have to be mapped to ABE authorization operations. In Section 5.4.3, it is shown how the authorization operations of DAC-MACS can be applied or combined to implement a group-based authorization model. Different alternatives for this mapping exist, which are first compared in an overview, and subsequently explained in detail with regard to the necessary cryptographic operations. These mapping alternatives build the basis for the performance evaluation presented in Section 5.5.

5.4.1 Evaluation of Existing ABE Schemes

Within the last decade, a plethora of ABE approaches has been proposed, which differ, e.g., in policy expressiveness, revocation functionality, and trust assumptions. Some ABE schemes were already discussed on a high level in Section 2.3. In this section, the focus is on a subset of ABE schemes referred to as *multi attribute authority* or *multi-AA* approaches, i.e., different AAs manage disjoint parts of the whole attribute space. The reason for this focus is that single-AA approaches cannot be considered capable for E2E-SDS as stated in Section 5.1. The results of the analysis are shown in Table 5.4. The table columns refer to the requirements as introduced in Section 5.1. The numbering of the requirements is kept throughout this section.

One reason why some multi-AA approaches cannot be employed for E2E-SDS is that they still rely on one central party that is technically able to decrypt all the ciphertexts in the system, which contradicts Req.4. An example for this is an early approach proposed by Chase [Cha07] in 2007, where the central authority generates the private key of every user and AA in the system. The rationale behind this architecture is that the CA generates a user-dependent value that is required to "personalize" secret keys to achieve collusion resistance (cf. Section 2.3). Furthermore, the approach only supports rather inflexible access policies: the policy stipulates that the encryptor chooses a set of attributes from each AA in the system, and a user is only able to decrypt the ciphertext if from each of these attribute sets, she holds at least a fixed number of attributes. Thus, the user must be known to each AA in the system. Other ABE approaches dependent on a central trusted party are [HCL13] and [WLW11], which

Table 5.4: Evaluation of ABE schemes supporting multiple Attribute Authorities.

	AC policies			**trust and independence**				**revocation**			
	access for more than one user or group (Req.1)	**multiple group owners (Req.2)**	**no constraints on permitted groups (Req.3)**	**decentral authorization root (Req.4)**	**each user can be group owner (Req.5)**	**independent authorization operations (Req.6)**	**no constraints on order of authorization operations (Req.7)**	**attribute revocation (Req.8)**	**without proxy and logical AC (Req.9)**	**without involvement of all non-revoked users (Req.10)**	**without attribute expiration date (Req.11)**
Chase et al. [Cha07]	-	✓	-	-	✓	✓	-	-	○	○	○
Huang et al. [HCL13]	✓	✓	✓	-	✓	-	-	-	○	○	○
Wang et al. [WLW11]	✓	✓	✓	-	✓	-	-	✓	-	✓	✓
Chase et al. [CC09]	✓	✓	-	✓	✓	-	-	-	○	○	○
Lewko et al. [LW11]	✓	✓	✓	✓	✓	✓	✓	-	○	○	○
Ruj et al. [RNS11]	✓	✓	✓	✓	✓	✓	✓	-	○	○	○
Yang et al. [YJR+14]	✓	✓	✓	✓	✓	✓	✓	-	○	○	○
Liu et al. [LLYY14]	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓
Li et al. [LYZ+13]	-	✓	-	✓	✓	✓	-	✓	✓	✓	✓
Yang et al. [YJ14]	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
Yang et al. [YJR+13]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✓ : fulfills requirement; -: does not fulfill requirement; ○: not applicable, as approach does not support attribute revocation employ a hierarchy of private key generators that is rooted in a fully trusted party.

Many ABE approaches support multiple AAs without the need for a central trusted party, but lack the possibility to deassign a user from an attribute, a feature usually referred to as *attribute revocation*. For example, an improved version of the Chase approach [CC09] manages to avoid the central trusted party by having the user-dependent value be calculated by the AAs in a distributed manner. However, the aforementioned inflexible policies remain, and revocation is not supported. Lewko et al. [LW11] offers more flexible policies than [CC09], but also does not support revocation. While Ruj et al. [RNS11] extends [LW11] with revocation capabilities, the extension only allows to deassign a user from all of her attributes completely and, thus, cannot offer the granularity demanded in Section 5.1.4. Yang et al. [YJR+14] supports updating an access policy, but also does not allow to deassign a user from an attribute.

There are multi-AA approaches that do not rely on a central trusted party and offer attribute revocation, but there are still issues that render them unusable for E2E-SDS. For example, Liu et al. [LLYY14] proposed an approach where the revocation leverages a trusted *Third Party Auditor (TPA)*. To decrypt a ciphertext, the decryptor has to send her secret keys to the TPA, which issues the requested decryption token after checking that the secret keys are up-to-date. This contradicts the requirements stated in Section 5.1.4 that revocation should not be enforced by a proxy based on logical access control decisions. Other approaches are too inflexible regarding the access policies that can be defined, e.g., the approach proposed by Li et al. [LYZ+13] requires the user to hold exactly one attribute from each AA in the system. The approach of Yang et al. [YJ14] fulfills most of the requirements for E2E-SDS, but does not allow a user to act as AA, which does not comply with the requirements stated in Section 5.1.3.

The only approach that fulfills all of the requirements for E2E-SDS and provides an authorization model expressive enough to be adapted to the group-based authorization model is DAC-MACS [YJR+13]. The DAC-MACS schemes is presented in Section 5.4.2. Note that [HXL15] reports an issue in DAC-MACS regarding the backward security of ciphertext updates. However, this feature that is not used by the mappings alternatives presented in the following.

5.4.2 DAC-MACS Entities and Operations

This thesis refers to a version of DAC-MACS called "Extensive DAC-MACS", which was described in [YJR+13]. As DAC-MACS is an instance of ciphertext-policy ABE (cf. Section 2.3), data *owners* define an attribute-based access policy as initial step of encryption. Only *users* that hold a set of secret keys whose attributes match the access policy are able to decrypt the data. More specific, an access policy is a Boolean term over attributes, for example, "`IsCIO` $\vee$ (`IsManager` $\wedge$ `ITDepartment`)". To be able to decrypt data, a user holds two types of keys: one or more secret keys, where each secret key is bound to some attributes, and a global user key pair that is attribute-independent. As the secret keys are customized to the user, they are only useful in combination with a global user private key. This prevents users from collusion.

The global user key pair is issued by a central *Certificate Authority (CA)*. For this

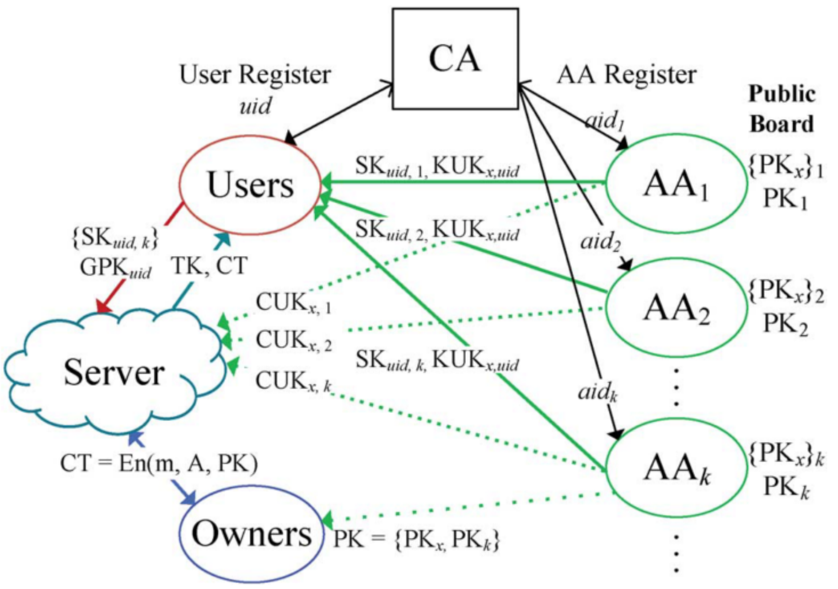


Figure 5.6: System model of ABE scheme *DAC-MACS*. Source: [YJR+13]

purpose, the CA creates and manages an identifier for each user, and also certifies that a global user public key belongs to a user with a certain identifier. Therefore, the CA has to be trusted by all other parties in the system. Note that the CA is not able to decrypt arbitrary ciphertexts, as decryption requires secret keys.

Secret keys are issued by *Attribute Authorities (AAs)*. DAC-MACS allows an arbitrary number of AAs in the system, and a user can have secret keys from an arbitrary set of AAs. Each AA manages a part of the whole attribute space, and is responsible for assigning these attributes to the appropriate users. Thus, it has to manage user-attribute bindings for a subset of the users in the system, and data owners have to trust that these bindings meet their expectations.

All encrypted data is stored on the *server*. To lower computation time on the user clients, DAC-MACS also outsources some cryptographic tasks to the server, which will be discussed in more detail below. Nonetheless, the server does not have to be fully trusted, but can be assumed as honest-but-curious, i.e., it is interested in reading the data, but it will carry out DAC-MACS operations as specified. The system model of DAC-MACS is shown in Figure 5.6.

DAC-MACS comprises the following phases: In the *System Initialization* phase, the CA first generates and publishes some global public parameters. Moreover, the CA issues a global user key pair to each user. For initialization, each AA generates an attribute-independent AA key pair, and additionally one attribute key pair for every attribute the AA manages.

After initialization, in the *Secret Key Generation* phase, the AAs issue secret keys to users upon authenticated request. Each secret key comprises one component for each attribute that the key is bound to.

In the *Data Encryption* phase, data is encrypted by owners according to some access policy. For this purpose, the data owner first has to define the access policy as a Boolean term, and then has to convert this to a matrix representation, where the matrix contains one row for each attribute that is part of the access policy. The actual encryption then requires this matrix, the plaintext, the attribute public key of all the attributes that are involved in the access policy, and the public keys of the managing AAs. The resulting ciphertext again contains one component for each attribute involved in the access policy, and is uploaded to the server. DAC-MACS employs hybrid encryption, i.e., the actual data is symmetrically encrypted, and the symmetric data key is encrypted with ABE.

For performance reasons, the computation-intensive operations in the *Data Decryption* phase are carried out by the server. For this purpose, the decrypting user presents her secret keys to the server, which the server uses to issue a user-specific decryption token for the requested ciphertext. Using the decryption token and her global user private key, the recipient is able to decrypt the ciphertext.

Finally, the *Attribute Revocation* phase is triggered whenever a user is deassigned an attribute. In this case, the responsible AA generates some key upgrade and ciphertext update information, and distributes these to all non-revoked users and the server, respectively. This way, each user can update the secret key that is bound to the attribute in question, and the server can update each ciphertext that has an access policy which involves the attribute.

The security of DAC-MACS is proven in the random oracle model under the assumption that a special variant of the Decisional Bilinear Diffie-Hellman (DBDH) problem is intractable (cf. Section 2.3). The security is proven with regard to a security game where the attacker initially has to specify a set of corrupted AAs, whose secret keys are made available to the attacker. In a first query phase, the attacker queries an oracle for secret keys encoding attributes that are managed by non-corrupted AAs. In the challenge phase, the attacker sends two equal-length messages to the challenger, together with an access policy that adheres to certain restrictions. The challenger randomly encrypts one of the messages under the access policy. A second query phase is appended. Finally, the attacker wins the game if he can guess which message was encrypted with a non-negligible success probability.

The security of DAC-MACS and RSA cannot be compared directly: First, the security of RSA is based on the RSA assumption instead of the DBDH assumption. And second, the security of RSA with the OAEP padding is proven with regard to, e.g., IND-CCA2 [FOPS01], which differs from the security game just described.

5.4.3 Mapping Alternatives

In this subsection, it is shown how the entities and operations of the group-based authorization model as introduced in Section 3.1 can be mapped to DAC-MACS entities and operations.

For the entity mapping, group-based users, resource owners and group owners are mapped to DAC-MACS user, owners and attribute authorities, respectively. The DAC-MACS CA will still be required. Note that this does not contradict the requirements stated in Section 5.1.3, as in DAC-MACS, the CA cannot issue any attributes,

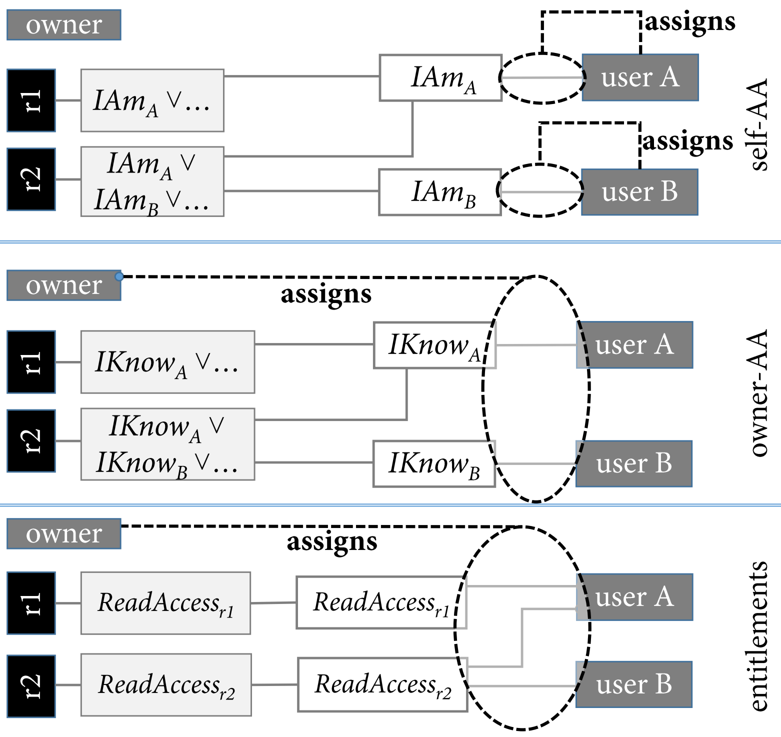


Figure 5.7: Overview of alternatives for mapping *DAC-MACS* to a group-based authorization model. Resource access policies are shown on the left, and user-attribute assignments on the right.

so it is no central trusted party. The entity mapping directly implies the mapping for most of the group-based authorization operations. Creating a named user group maps to creating an attribute in DAC-MACS, and a group owner adding a user to a named user group means that a secret key for an attribute is issued to the user by the AA. Removing a user from a named user group means that the attribute revocation operation is triggered in DAC-MACS.

The group-based authorization model also supports direct user permissions, i.e., explicit enumeration of users that are permitted to access a resource without the involvement of user groups. The necessary authorization operation—grant or deny access—have no obvious counterpart in DAC-MACS, but it is possible to "emulate" these operations using DAC-MACS operations. Three alternatives for this mapping can be considered, which are referred to as *mapping alternatives*. An overview of the mapping alternatives is provided in Figure 5.7.

The first mapping alternative is called *self-AA*: In this case, every user plays the role of an AA, and issues a secret key for the attribute $\texttt{IAm}_u$ for himself. This means that for every user, there is an additional attribute in the system, and each of these attributes is held by only one user, which is comparable to a known user identity value in Identity-Based Encryption (cf. Section 2.3). This attribute can be a part of an access policy in the encryption step to grant the user access to the respective ciphertext.

A variant of this is the alternative *owner-AA*, where the resource owner issues a secret key for any user she wants to give access to at least one of her resources. The secret key is bound to the attribute $\texttt{IKnow}_u$, which can be part of an access policy similar to the self-AA mapping alternative.

Finally, in the *entitlements* mapping alternative, two additional attributes are defined for each resource: $\texttt{ReadAccess}_r$ and $\texttt{ReadWriteAccess}_r$. The resource owner can grant access to the resource by issuing a secret key to the user that is bound to one of these attributes. In this case, denying access to the resource is done by revoking the respective attribute from the user.

As described in the previous section, the decryption and the ciphertext update steps of DAC-MACS stipulate that some cryptographic operations are carried out by the server. This contradicts the E2E-SDS requirements on the storage provider as stated in Section 5.1.3. To cope with this issue, ciphertext updates are skipped completely. This results in *lazy revocation*, which complies to the requirements stated in Section 3.1. For decryption, the generation of decryption tokens is simply moved from the server to the client of the decrypting user, which does not affect the security of the protocol.

5.4.4 Mapping Details

In this section, the steps for the different group-based authorization operations and the different mapping alternatives are described in detail. A hybrid cryptographic approach is employed, where DAC-MACS operations are used wherever possible, and traditional symmetric and asymmetric cryptography in any other case, e.g., to digitally sign encrypted data.

With each mapping alternative, for *initialization*, every user registers with the CA and retrieves a global user key pair. In addition, each user u acts as AA for an attribute $\texttt{IAm}_u$, which is only held by user u and enables to encrypt data or keys for her only. For this purpose, user u generates an AA key pair and an attribute key pair for $\texttt{IAm}_u$.

To *create a group g*, the group owner simply creates an attribute key pair for the attribute $\texttt{IsMember}_g$. To *add the user u to the group g*, the owner issues a secret key $SK_u[\texttt{IsMember}_g]$ for attribute $\texttt{IsMember}_g$ to user u, encrypts it for attribute IAm_u and uploads it. To *remove user u from the group*, the owner generates a key upgrade key KUK_x for every member x except u, and uploads it to the SP. As explained in Section 5.4.3, the ciphertext re-encryption feature of DAC-MACS is omitted.

To *add a resource r*, the resource owner creates a symmetric resource key RK_r, an asymmetric resource signing key pair RSK_r, and a symmetric resource signing master key $RSMK_r$. RSK_r^{priv} is encrypted using $RSMK_r$. RK_r is encrypted either for $\texttt{ReadAccess}_r \vee \texttt{ReadWriteAccess}_r$ in case of entitlements mapping, or for $\texttt{IAm}_{owner}$ in case of self-AA or owner-AA mapping. A similar procedure applies to the encrypted resource signing key pair $Enc_{RSMK_r}(RSK_r^{priv})$ and the attribute $\texttt{ReadWriteAccess}_r$.

The remaining operations are mainly described in pseudocode. The owner-AA alternative is omitted, as all steps can be derived from the self-AA alternative by replacing $\texttt{IAm}_u$ with $\texttt{IKnow}_u$. The only addition of owner-AA is that the resource owner creates a secret key for the attribute $\texttt{IKnow}_u$ when u is granted access to one of her resources for the first time. In the notation used in this section, $\mathit{DisjunctAppend}(pol_a, \texttt{b})$ disjunctively appends attribute $\texttt{b}$ to access policy pol_a, and $CT(x)$ denotes an encrypted key downloaded from the storage provider.

To *grant user u read and write access to resource r*, the resource owner carries out the following operations:

```
if (SELF-AA)
  $Dec(CT(RK_r), SK_u[\texttt{IAm}_{owner}])$;
  $Dec(CT(RSMK_r), SK_u[\texttt{IAm}_{owner}])$;
  $pol_{new} = DisjunctAppend(pol_{current}, \texttt{IAm}_u)$
  $Enc(RK_r)[pol_{new}]$;
  $Enc(RSMK_r)[pol_{new}]$;

if (ENTITL)
  $SecretKeyGen(u)[\texttt{ReadWriteAccess}_r]$
```

Listing 5.1: AddResource operation

To *grant group g read and write access to resource r*, the necessary actions are independent of the mapping alternative:

```
$Dec(CT(RK_r), SK_u[\texttt{IAM}_{owner}])$;
$Dec(CT(RSMK_r), SK_u[\texttt{IAM}_{owner}])$;
$pol_{new} = DisjunctAppend(pol_{current}, \texttt{IsMember}_g)$
$Enc(RK_r)[pol_{new}]$;
$Enc(RSMK_r)[pol_{new}]$;
```

Listing 5.2: Grant read and write access operation

Denying a user or a group read access to resource r requires the following steps:

```
$SymKeyGen(RK_r)$;
$pol_{new} = \texttt{IAM}_{owner}$
if (SELF-AA)
  foreach(user u with read access)
      $pol_{new} = DisjunctAppend(pol_{new}, \texttt{IAM}_u)$
  foreach(group g with read access)
      $pol_{new} = DisjunctAppend(pol_{new}, \texttt{IsMember}_g)$

if (ENTITL)
  $pol_{new} = DisjunctAppend(pol_{new}, \texttt{ReadAccess}_r)$
  $pol_{new} = DisjunctAppend(pol_{new}, \texttt{ReadWriteAccess}_r)$
  foreach(user u with read access) $UpdateKeyGen[\texttt{ReadAccess}_r]$
  foreach(group g with read access)
      $pol_{new} = DisjunctAppend(pol_{new}, \texttt{IsMember}_g)$
```

```
$Enc(RK_r)[pol_{new}]$;
```

Listing 5.3: Deny read access operation

To additionally deny write access, a new asymmetric resource signing key pair $RSK_r^{'}$, and a symmetric resource signing master key $RSMK_r^{'}$ have to be created, and all operations described above have to be repeated regarding the attribute $\texttt{ReadWriteAccess}_r$.

To *read* the resource r, the user u takes the following actions:

```
if ($hasReadAccess(r, g)$)
  $Dec(CT(RK_r), SK_u[\texttt{IsMember}_g])$;
else
  if (SELF-AA) $Dec(CT(RK_r), SK_u[\texttt{IAM}_u])$;
  if (ENTITL) $Dec(CT(RK_r), SK_u[\texttt{ReadAccess}_r])$;
$Dec(r, RK_r)$;
$Verify(Sig(r), RSK_r^{pub})$;
```

Listing 5.4: Read resource operation

For a *write* access, the writer retrieves the most recent version of RK_r, $RSMK_r^{'}$ and $Enc_{RSMK_r}(RSK_r^{priv})$, and decrypts them with the same alternative-dependent key selection logic applied for read operations. The writer encrypts the resource with RK_r, digitally signs it with RSK_r^{priv}, and uploads both ciphertext and signature.

5.5 Performance of E2E-SDS Using Attribute-Based Encryption

In this section, the performance of an SDS protocol based on the ABE scheme *DAC-MACS* [YJR+13] is evaluated. In the previous section, it was discussed how the group-based authorization model that is formalized and instantiated in this thesis (cf. Section 3.1) can be mapped to the attribute-based authorization model of DAC-MACS. Three alternatives for this mapping were identified and described in detail. The performance of these mapping alternatives is evaluated in the following. The objective of the performance evaluation is to approach the question whether the aforementioned advantage of ABE—one ciphertext for multiple recipients—makes ABE usable for E2E-SDS from a performance point of view.

The performance evaluation yields the average computation time and outgoing network traffic that are required on a user client device to carry out the different authorization operations such as granting or revoking access to data. For this purpose, the analytical performance evaluation (cf. Section 4.1) method was used, which can easily be applied if only averaged results are needed. The method requires to model each mapping alternative by a set of algebraic expressions, where one expression is required per mapping alternative, per performance metric, and per authorization operation. Each algebraic expression combines workload parameters with measurements taken from concrete devices.

The performance evaluation targets real-world sharing scenarios, thus, the workload parameters were extracted from the real-world workloads characterized in Section 3.5. These are the same workloads that were the basis for the performance evaluation of SDS protocols based on traditional cryptography (cf. Section 5.3). To get average values for all necessary workload parameters, samples were taken separately for each authorization operation type, e.g., one workload parameter is "average size of a group when a user is removed", which might differ from "average size of a group when a user is added".

To get exemplary computation times for the cryptographic operations, device benchmarks were implemented and carried out on three devices. Thereby, a focus was put on up-to-date mobile devices, as these devices are used for data sharing by many users, but, in general, suffer from slower computational performance compared to stationary devices.

To conclude this section, the performance evaluation results are compared with those of the SDS system based on traditional cryptography that was presented earlier in this chapter.

5.5.1 DAC-MACS Benchmark Results

To get exemplary computation times necessary for the different DAC-MACS operations, benchmarks were created, and ran on three different devices. As an example for a mobile user device with moderate computing power, a Samsung Galaxy S5 mini (SM-G800F) from 2014 running Android 4.4.2 was chosen. The class of mobile devices with comparably high computing power is represented by a Samsung Galaxy S6 (SM-G920F) from 2015 running Android 5.1.1. Finally, benchmarks were run on a Dell Latitude E4310 notebook from 2011 as a reference for an x86 processor. The benchmarks were implemented using the "Java Pairing-Based Cryptography"[3] library. For increased performance, all pairing operations were outsourced to the C-based PBC[4] library.

The benchmarks were conducted with a pairing based on a 1,024 bit base field and a 224 bit order elliptic curve subgroup with an embedding degree of 2. In this configuration, NIST considers the DLOG problem that must be solved to attack the ABE scheme equally hard to the DLOG problem in a field with 2,048 order[5]. The benchmark results are shown in Table 5.5.

5.5.2 Performance Evaluation Results for Real-World Sharing Scenarios

As already mentioned, the performance evaluation described in this section was carried out on the basis of the real-world workloads that were characterized in Sec-

[3]http://gas.dia.unisa.it/projects/jpbc

[4]https://crypto.stanford.edu/pbc/

[5]http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf

DAC-MACS operation	Galaxy S5 mini	Galaxy S6	Dell Latitude E4310
user key pair generation	86	30	15
AA key pair generation	162	60	27
attribute key pair generation	124	44	20
encryption			
basis	51	19	21
per-attribute overhead	184	71	30
decryption token generation			
basis	155	48	17
per-attribute overhead	217	82	23
decryption	10	2	1
user secret key generation			
basis	290	110	55
per-attribute overhead	107	51	20
update key generation			
basis	52	20	7
per-user overhead	72	32	13
user secret key update	1	1	1
ciphertext update	79	29	13

Table 5.5: Benchmarks of *DAC-MACS* operation running times (in milliseconds) with 1024 bit base field and 224 bit elliptic curve group order.

tion 3.5. As it was technically not feasible to trace read operations, these had to be synthetically added to the existing workloads. For this purpose, a ratio of read to write operations of 3 was taken, since this ratio was empirically determined in [LPGM08]. Thereby, for each write action, 3 read actions were inserted for a random user and a random resource this user is permitted to read.

In the following, the average computation time and outgoing network traffic that are required by a client to carry out workload operations are analyzed. The focus lies on operations that are either frequent or expensive. In the following, the computation times for the Galaxy S5 mini are stated, while the values for Galaxy S6 and Dell Latitude are added in parentheses. Figure 5.8 provides an overview of the computation times for the all operations whose costs differ for the different mapping alternatives. The plot shows that the choice of the mapping alternative has only minor influence on the resource consumption, as, e.g., the computation times of an operation differs by at most 1 s for different mapping alternatives.

Initially generating the necessary keys takes a user 4.1 s (2.0 s, 1.7 s) in every mapping alternative. Note that this operation is only required once per user.

The most expensive operation that was observed with regard to the given workloads is removing a user from a group. The computation time of this operation is also independent of the mapping alternative: Where the operation took 0.091 s (0.037, 0.014) in groups up to 20 users and 1.8 s (0.795 s, 0.322 s) in groups from 21 to 200 users, the required computation time increases to 31 s (14 s, 6 s) in groups from 201 to 2,000

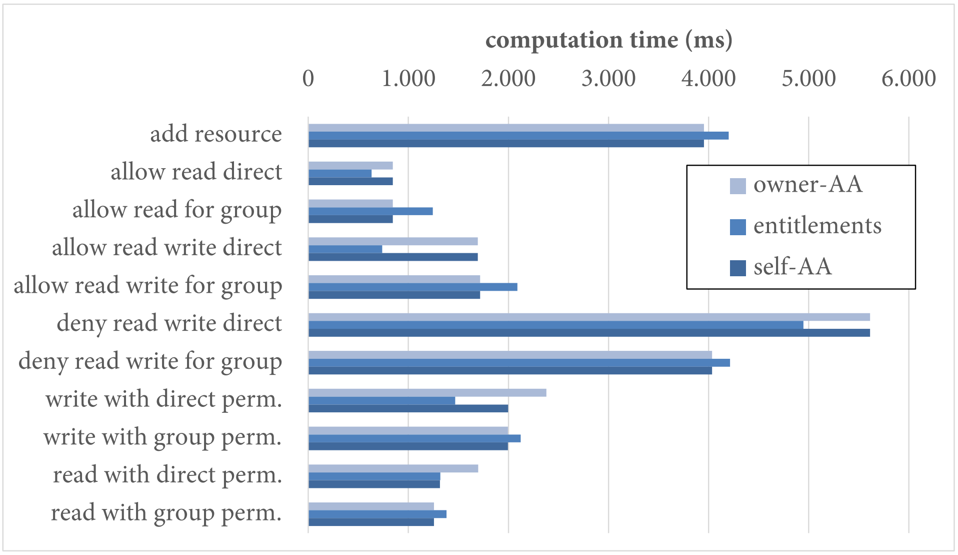


Figure 5.8: Average key management computation time (in seconds) for different operations when the ABE-based E2E-SDS protocol is used (for Samsung Galaxy S5 mini).

users, and finally it takes about 1,605 s (713 s, 290 s) in groups larger than 2,000 users.

The group member removal operation is also the most expensive with regard to outgoing network traffic: When dealing with groups up to 20 users, less than 1 KB of outgoing traffic is needed, and about 28 KB for groups up to 200 users. For groups comprising between 201 and 2,000 users, about 488 KB are required, and for groups larger than 2,000 users, almost 26 MB of key material has to be uploaded for group member removal.

Regarding the denial of access to resources, the workloads contain no operation to deny read-only access, but contain some denials of read and write access. For all mapping alternatives, the denial of access permissions for a user is more computation-intensive than for a group. With the user-AA and self-AA mapping alternatives, it takes 5.6 s (2.6 s, 1.9 s) to deny access permissions for a user and 4.0 s (2.0 s, 1.7 s) to deny access permissions for a group. With the entitlements mapping alternative, the same operations take 4.9 s (2.4 s, 1.8 s) and 4.2 s (2.0 s, 1.7 s), respectively.

The computation times for writing resources comprise the time required for the encryption of the actual data using a symmetric cipher. The computation times depend on whether the writer has direct write permissions, or gets the permissions implicitly by group membership. In the former case, the computation times considerably differ between the mapping alternatives: Where in the case of owner-AA, 2.4 s (0.9 s, 0.3 s) are required, it only takes about 1.5 s (0.5 s, 0.2 s) with the entitlement and 2.0 s (0.7 s, 0.2 s) with the self-AA mapping. In the latter case, the differences are more subtle: Write operations take 2.0 s (0.7 s, 0.2 s) with the owner-AA and self-AA mapping alternatives, and 2.1 s (0.8 s, 0.2 s) with the entitlement mapping alternative.

Finally, the computation times are analyzed for reading resources, which is the most frequent operation in our workloads, due to the aforementioned read-write ra-

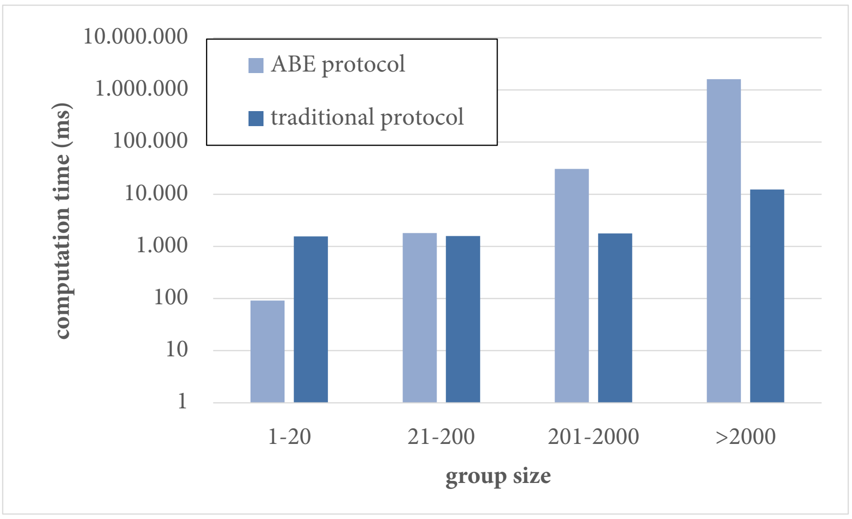


Figure 5.9: Computation time for group member removal operations. Comparison of E2E-SDS protocols based on ABE, and based on traditional cryptography.

tio of 3. Again, these computation times depend on whether the reader gets the read permission directly or by group membership. In the former case, with the owner-AA mapping alternative, the operation takes 1.7 s (0.6 s, 0.2 s), compared to 1.3 s (0.5 s, 0.1 s) with the other mapping alternatives. In the latter case, reading a resource requires about 1.3 s (0.5 s, 0.1 s) with the owner-AA and self-AA mapping alternatives and 1.4 s (0.5 s, 0.1 s) using the entitlement mapping alternative.

5.5.3 Performance Comparison with Traditional Cryptography-Based E2E-SDS Protocol

In this section, the performance of the ABE-based protocols described in the previous section is compared with the resource consumption of a E2E-SDS protocol solely based on traditional cryptography, i.e., symmetric and asymmetric cryptography that produces ciphertexts dedicated to a single recipient only. The latter was described in Section 5.3.1. For the remainder of this section, these protocols are referred to as *ABE protocol* and *traditional protocol*, respectively.

The estimated performance of the traditional protocol is based on the same workloads described in the previous subsection, and on RSA and AES benchmarks taken from the Samsung Galaxy S5 mini and the Dell Latitude E4310. As the ABE-based encryption of symmetric keys in the ABE protocol roughly corresponds to the asymmetric encryption of symmetric keys in the traditional protocol, a comparison of the computation times required for these operations gives a first indication of the protocol performance: Using RSA with 2 048 bit key length, about 875 encryption operations can be performed on the S5 mini and about 4 225 on the Dell Latitude per second, whereas using ABE encryption as described in Section 5.5.1, within one second, a ciphertext can only be generated for about 5 attributes on the S5 mini and 43 attributes on the Dell Latitude (cf. Table 5.5).

A closer look on the estimation results shows that the traditional protocol requires less computation time and outgoing network traffic than the ABE-based one for almost any operation and mapping alternative. Regarding the average computation time, the traditional protocol outperforms the ABE one by a factor between 1.1 and 18. Considering the outgoing network traffic, the traditional protocol outperforms the ABE protocol by a factor of 2 for creating a new resource, up to a factor of 30 for granting a user direct read access to a resource. The only operation that is always less resource-consuming using the ABE protocol is the creation of a new group.

Group member removal operations are analyzed in more depth, as the comparison results depend on the group size. Figure 5.9 shows the computation times for different group size ranges. The results are exemplary for the Samsung Galaxy S5 mini, but comparable results are seen for the Dell Latitude. The plot shows that the ABE-based protocol is faster for small groups with up to 20 members. However, for larger groups, the traditional crypto protocol outperforms the ABE one: For groups larger than 2000 members, the traditional crypto protocol only requires 27 s on average, whereas the ABE protocol needs about 1605 s, resulting in computation times two orders of magnitude higher.

A comparison of the outgoing network traffic required to remove a group member yields similar results: While for small groups, the ABE protocol only needs to send 77 KB as opposed to about 106 KB with the traditional one, this ratio changes in favor of the traditional protocol with larger groups. For groups with more than 2000 members, the ABE protocol requires that almost 26 MB of key material has to be sent on average, while using the traditional protocol, only about 8 MB are needed.

5.6 Conclusions and Outlook

The focus of this chapter is on E2E-SDS protocols that support named user groups, that is, user groups that are managed by a dedicated group owner, and that exist independently from any permissions on resources. The combination of E2E-SDS and the support for named user groups implies a set of requirements on the protocol, which comprise requirements regarding the expressiveness of access control policies, the trust model, the authorization process, and the revocation of access permissions.

An E2E-SDS protocol protects the confidentiality and integrity of the shared data only with respect to certain attacker models. Therefore, in this chapter, attacks on E2E-SDS protocols were structured with regard to the capabilities of the attacker, and prerequisites and countermeasures were discussed. In conclusion, for many attacks, countermeasures exist, but these countermeasures require users to hold a local state, thus, switching the device requires to synchronize the local state. For a forking attack, it was shown in literature that it is fundamentally impossible to construct a countermeasure when multiple user are allowed to change the key or resource in question, and a trusted third party is omitted. In this case, the design of an E2E-SDS protocol can only strive try to minimize the attack vector for a forking attack, i.e., to minimize the parts of the key graph that can be forked.

In the remainder of the chapter, the performance of E2E-SDS protocols that are based on joint authorization operations was evaluated. These operations are carried out by at least two different users in a collaborative manner, which requires that the involved users are reachable in the network and able to allocate the necessary computational resources immediately upon request. This is might be a problem especially when mobile devices are involved, as these devices may be temporarily located in areas with bad network coverage, or it may be undesirable for the device owner that computing-intensive cryptographic operations are carried out immediately upon request, maybe because the device becomes less responsive. The E2E-SDS protocols analyzed in this chapter use joint authorization operations for group member removal operations. This massively hinders an employment of hierarchies of named user groups in practice.

The performance evaluations presented in this chapter are carried out by means of the methods introduced in Chapter 4. The evaluations are based on the real-world workloads characterized in Section 3.5, and are carried out for a range of mobile devices manufactured between 2008 and 2015.

As a starting point, the performance of an E2E-SDS protocol was evaluated that is similar to protocols that are in productive use today. The protocol is solely based on traditional cryptographic primitives. The performance evaluation of this protocol shows that with regard to the considered workloads, only minor performance penalties have to be expected for most of the authorization operations, as the computation time for these operations is mostly up to some seconds, even on an iPhone 3G manufactured in 2008. However, major performance penalties hit users that manage group memberships in large user groups, i.e., user groups with a few thousand or more members. In this case, the computation times rise to several minutes when removing a user from a group—in the considered workloads, up to 324 seconds on the iPhone 3G. Furthermore, the group member removal operations require the group owner to upload several MB of cryptographic key material to the storage provider.

A possible solution to decrease the performance penalties of group membership management is offered by Group Key Management approaches, which were originally developed to facilitate an efficient key distribution for secure multicast in a public network. For this reason, the aforementioned E2E-SDS protocol was extended with a Group Key Management approach referred to as *Logical Key Hierarchy*. With the extended protocol, the performance penalties for the management of group memberships in large user groups are decreased significantly, as a group member removal operation can be carried out within few seconds, at the price of slightly raised computation times for smaller groups. For this reason, a future task is to construct a "hybrid" E2E-SDS protocol that applies Group Key Management approaches only to user groups beyond a certain size.

Moreover, the introduced performance evaluation methods were applied to an E2E-SDS protocol based on Attribute-Based Encryption (ABE). In a first step, an ABE scheme was identified that facilitates E2E-SDS while offering an authorization model expressive enough to enhance it with support for named user groups. For this purpose, different ABE schemes were evaluated with regard to the requirements stated at the beginning of the chapter. Only a single ABE scheme could be identified that complies with all of the requirements, which is called *DAC-MACS*. In a second step, the attribute-based authorization model of DAC-MACS was mapped to the group-based authorization model. Due to some degrees of freedom in the mapping, this resulted in three different mapping alternatives.

The performance of the mapping alternatives was evaluated by means of the analytical method, as ABE is not supported by the simulative model without any enhancements. The performance evaluation showed that the ABE-based E2E-SDS protocol exhibits a slightly worse performance than the traditional E2E-SDS protocol. Nonetheless, the ABE-based E2E-SDS protocol only requires a few seconds of computation time for any authorization operation, except for the group member removal operation. In future work, it should be investigated whether an analog of Group Key Management approaches could be constructed with ABE schemes to improve the performance of group member removal operations. This could be achieved if the underlying ABE scheme supports a hierarchy on the attributes.

6 E2E-SDS Without Joint Authorization Operations

In this chapter, an E2E-SDS protocol is presented that does not require any joint authorization operations. Joint authorization operations refer to authorization operations that are carried out by at least two different entities in a collaborative manner. This kind of operation can only be carried out successfully if the involved users are reachable in the network and able to allocate the required computational resources immediately upon request. Depending on the concrete data sharing application, this is a strong assumption: If mobile devices are involved, these may be temporarily located in areas with bad network coverage, for example, like in many airplanes or trains, or the devices might simply be switched off. Even if the devices are reachable, it may be undesirable for the device owner that computing-intensive cryptographic operations are carried out at this point in time, maybe because the device becomes less responsive. These drawbacks of joint authorization operations motivated the design of the E2E-SDS protocol introduced in this chapter.

In Section 6.1, the challenges in designing an E2E-SDS protocol without joint authorization operations are discussed. To prevent an attacker from replacing genuine keys with forged ones, the keys have to be signed digitally. It is argued that without joint authorization operations, the set of users that is authorized to sign a certain key at the storage provider is no longer static, but might change over time. In consequence, the set of public keys that are valid for a certain key signature changes over time as well. However, the set of valid public keys has to be known to validate a signature. The totality of valid public keys for every key signature in the system will be denoted as *validity state* in this chapter. To achieve E2E-SDS, the validity state has to be published over an entity that is only semi-trusted (cf. Section 5.1), i.e., that is not trusted with regard to the confidentiality and integrity of any stored data, but only with regard to availability. For reasons of simplicity, the validity state might also be published over the same storage provider that stores the shared data.

The publication of the validity state over a semi-trusted entity rises two challenges: First, access control on updates of the validity state has to be enforced by cryptographic means. Second, a rollback of the validity state or of signed keys has to be prevented, even with regard to the fact that multiple users are authorized to generate key signatures. In Section 5.2, it was shown that freshness guarantees are fundamentally hard to achieve in E2E-SDS with multiple writers.

Section 6.2 provides an overview of the proposed E2E-SDS protocol. The proposed protocol provides a cryptographic enforcement of updates on the validity state, and guarantees the freshness of the validity state. However, the freshness of the key signatures is not guaranteed by the protocol, as applicable approaches for preserving freshness exhibit many drawbacks in practice. The implications of weakening the freshness guarantees for key signatures are discussed.

To achieve the aforementioned features, the protocol basically encodes the validity state in a new subgraph of the key graph, a so-called *administrative key graph*. The administrative key graph extends the *base key graph*, which is already known from the E2E-SDS protocols introduced in the previous chapter. The keys in the administrative key graph are used to sign the keys in the base key graph, and to validate the signatures. The edges between the public keys in the administrative key graph determine whether a public key is authorized for signing a certain key in the base key graph. Hierarchies of named user groups are supported by the administrative key graph in a natural way.

In Section 6.3, the different operations of the proposed protocol are described in detail.

The chapter is concluded by a performance evaluation of the proposed E2E-SDS protocol in Section 6.4. The performance evaluation leverages the simulative method introduced in Section 4.2. The first part of the performance evaluation is carried out based on real-world workloads. The results show that in most cases, the performance of the protocol is worse than the performance of a similar protocol that leverages joint authorization operations as presented in the previous chapter. However, in most cases, the difference is small in absolute terms. Similar to the E2E-SDS protocols introduced in the previous chapter, the authorization operation types that require the most computation time and outgoing network traffic are group member removals.

The second part of the performance evaluation is a sensitivity analysis of the performance of group member removals, especially in sharing scenarios based on group hierarchies. This analysis is based on synthetic workloads. The results indicate that the number of memberships of a certain group in other groups is an important factor for the performance of group member removals.

6.1 Challenges and Problem Statement

In this section, the challenges that arise when designing an E2E-SDS protocol without joint authorization operations are discussed. The implication of avoiding joint authorization operations obviously is that only one single user, referred to as *actor*, can be involved in an authorization operation. In the following, it is argued that the restriction on a single actor per authorization operation has implications on the integrity of cryptographic data structures that are stored at the storage provider.

In Section 5.2, it was shown that especially the integrity of the stored keys is an essential requirement to prevent malicious changes in the SDS system's authorization state: Without protecting the integrity of the keys, an attacker might replace genuine keys with attacker-chosen ones, and trick potential resource writers into using the forged key for resource encryption. According to the E2E-SDS trust model introduced in Section 5.1, the storage provider is not trusted to guarantee the integrity of any stored cryptographic data structures. Therefore, the integrity of the cryptographic data structures, and especially the integrity of the keys, has to be enforced by cryptographic means. A common way to protect the integrity of the keys are digital signatures.

If an SDS protocol supports the group-based authorization model introduced in Section 3.1, in traditional protocols, this will generally cause that different types of authorization operations require the generation of the same key: For example, adding a resource requires the resource owner to generate some kind of resource encryption key. At the same time, removing a user from a group that has read access on the resource also requires to re-generate the resource encryption key. Note that in the case of adding a resource, the resource owner initiates that authorization operation, while a group member removal is initiated by the group owner. With joint authorization operations, the group owner is able to outsource the generation of the resource encryption key to the respective resource owner. Without joint authorization operations, this is no longer possible, thus, the group owner itself must generate and sign the resource encryption key.

A similar argumentation applies in case of hierarchies of named user groups. For the explanations in the following, if a group is member of another group, the former will be referred to as *member group*, while the latter is denoted as *host group*. With a naive cryptographic implementation of group hierarchies, the group key related to a certain group is also known to each user in each member group. Therefore, after removing a user from a member group, the group key of any host group must be renewed. Without joint authorization operations, the renewing and signing of all of these group keys has to be carried out by the group owner that initiated the group member removal.

In consequence, supporting named user groups while omitting joint authorization operations leads to situations where more than one user is allowed to generate and sign a certain resource or group key. The set of allowed users is dynamic, and changes whenever a group is granted or denied access on a resource, or when a group is added or removed as member of another group. This also applies to the set of key pairs that is valid to generate and validate the signature of a certain key, i.e., the validity state. Therefore, to enable the validation of the key signatures, the validity state has to be published, and its integrity has to be protected.

In summary, the validity state has to be published under the following circumstances:

1. The E2E-SDS protocol protects the integrity of keys stored at the storage provider by cryptographic means.
2. The E2E-SDS protocol omits joint authorization operations.
3. The E2E-SDS protocol supports named user groups or even hierarchies of named user groups.

Due to the requirements of E2E-SDS, the validity state has to be stored at an entity that is only assumed to be semi-trusted. For the sake of simplicity, is it assumed that the validity state is stored on the same storage provider that holds the shared data itself. This brings up two challenges: First, the validity state must be updated by resource and group owners over time. As the storage provider is not trusted to protect the integrity of the validity state, a cryptographic mechanism has to be built that enforces access control on the validity state.

Second, it should be prevented that the validity state or the key signatures are "rolled back" to an older version. This is especially hard with regard to the fact that multiple users are authorized to generate signatures for a certain key. As already discussed in Section 2.2, preserving the freshness of data that can be written by multiple users is a fundamental problem if no trusted entity is present.

In summary, the problem that is addressed in this chapter can be stated as follows:

> "Build an E2E-SDS protocol that realizes a group-based authorization model and cryptographically enforces the integrity of the keys stored at the storage provider without employing joint authorization operations."

6.2 Protocol Overview and Design Decisions

In this section, an E2E-SDS protocol is introduced that supports the group-based authorization model described in Section 3.1 while avoiding joint authorization operations. First, an overview of the protocol is provided, where it is shown how the validity state can be encoded and published as administrative key graph. Thereafter, the freshness of the validity state and the key signatures is discussed as a major challenge in protocol design. At the end to this section, some issues are discussed that arise when the protocol is to be deployed in the real world.

6.2.1 Administrative Key Graph: Basic Idea

In the previous section, it was argued that the first challenge in the protocol design is to provide the validity state, i.e., the set of public keys that are valid for signing a certain key, only by using a semi-trusted storage provider. The integrity of the validity state must be guaranteed, and cryptographic processes must be established to update the state. The validity state including the cryptographic access control enforcement should be constructed in a way that supports hierarchies of named user groups, i.e., a user group can be member of another user group.

The protocol addresses this challenge by encoding the validity state as a cascade of keys and signatures. This mechanism is inspired by the cascades of public keys employed by Public Key Infrastructures (PKIs, cf. Section 2.2). The cascade of keys and signatures built by the protocol matches the definition of a key graph as given in Section 4.2.1. For this reason, the cascade will be referred to as *administrative key graph*, and the keys within the administrative key graph as *administrative keys* for the remainder of this chapter.

The administrative keys are used to sign the keys in the *base key graph*, i.e., the integrity of the base key graph is enforced by means of the administrative key graph. The base key graph is similar to the key graph used by the E2E-SDS protocols that were introduced in the previous chapter: It comprises all the keys and lockboxes that are required for the decryption of resources.

Note that the administrative and the base key graph are denoted as distinct key graphs for the sake of brevity and clearness of the explanations. In fact, they simply form two parts of the overall key graph of the protocol.

More specifically, an administrative key is used to sign the binding of a key in the base key graph, i.e., a *base key*, to its function within the protocol. An example for a function within the protocol is "resource encryption for resource *r*". Every base key and every administrative key in the protocol is related to a function within the protocol: While the base key *fulfills* the function in the protocol, the administrative key *attests* that the base key is intended to fulfill the function. The base keys assigned to a certain function change over time, as, for example, a resource encryption key might have to be renewed. On the other hand, the administrative key related to a certain function is fixed.

Just as the binding of a base key has to be signed to prevent key forgery, the binding of an administrative key to its function has to be signed as well. The latter binding must be signed by a user that is trusted to generate keys for the respective protocol function. For example, the resource owner of resource *r* attests that a certain administrative key is related to the function "resource encryption for resource *r*". The establishment of trust in a certain user to sign a certain administrative key is not part of the protocol, but has to be done up front. This is similar to the establishment of trust in a root certificate authority when a PKI is used.

The administrative key can be used to sign the binding of a base key to its function. However, the permission to sign this binding can also be "delegated" to other keys by means of the administrative key graph. For this purpose, administrative keys can be connected by signatures. These signatures carry the following semantics: Assume that administrative key AK_1 was used to generate a signature of administrative key AK_2. This signature causes that AK_2 "inherits the permissions" of AK_1: If AK_1 can be used to sign a certain base key, then AK_2 can also be used to sign this base key.

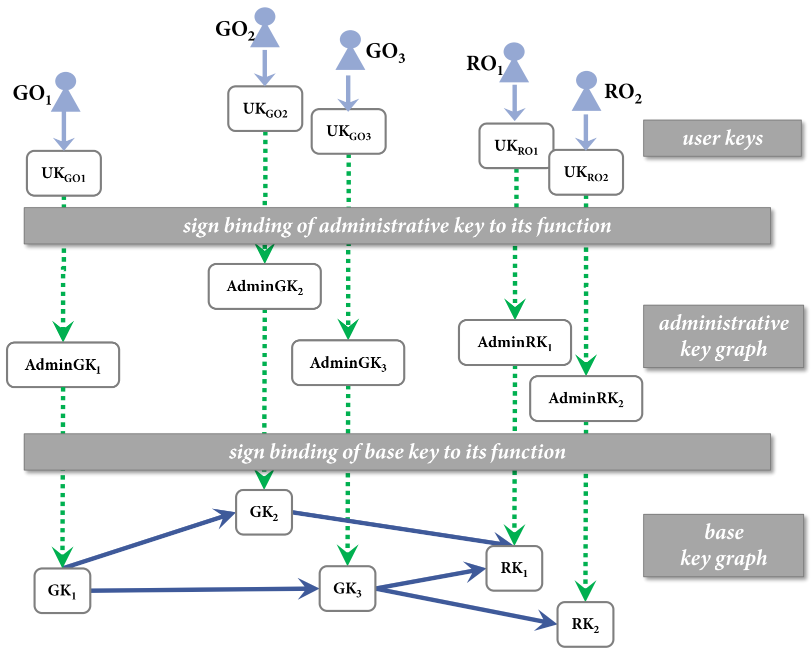


Figure 6.1: Sketch of relation between administrative and base key graph after generation of resource and group keys. Blue solid edges indicate the encryption of the end key with the starting key, while green stroked edges indicate a signature of the end key with the starting key.

6.2.2 Application of Administrative Key Graph

In this section, it is shown how the administrative key graph can be leveraged to build an E2E-SDS protocol that supports named user groups.

For the sake of simplicity, the further explanations are simplified to two functions within the protocol: a *resource encryption key RK* is used to encrypt and decrypt a certain resource, while a *group key GK* is distributed to every member of a named user group. In consequence, the administrative key graph comprises one *administrative resource encryption key AdminRK* for every resource, and one *administrative group key AdminGK* for every group.

The explanations in this section are based on a running example, which comprises resources r_1 and r_2 with resource owners RO_1 and RO_2, and groups g_1, g_2 and g_3 with group owners GO_1, GO_2 and GO_3, respectively.

The state of the key graph after the initial generation of resource and group keys is visualized in Figure 6.1. Each resource encryption key and each group key is bound to its function by a signature with the respective administrative key. Each administrative key, in turn, is bound to its function by a signature with the user key pair. In the example, each administrative group key is signed by the owner of the respective group, and each administrative resource encryption key is signed by the respective resource owner.

A signature of an administrative key graph with another administrative key is gen-

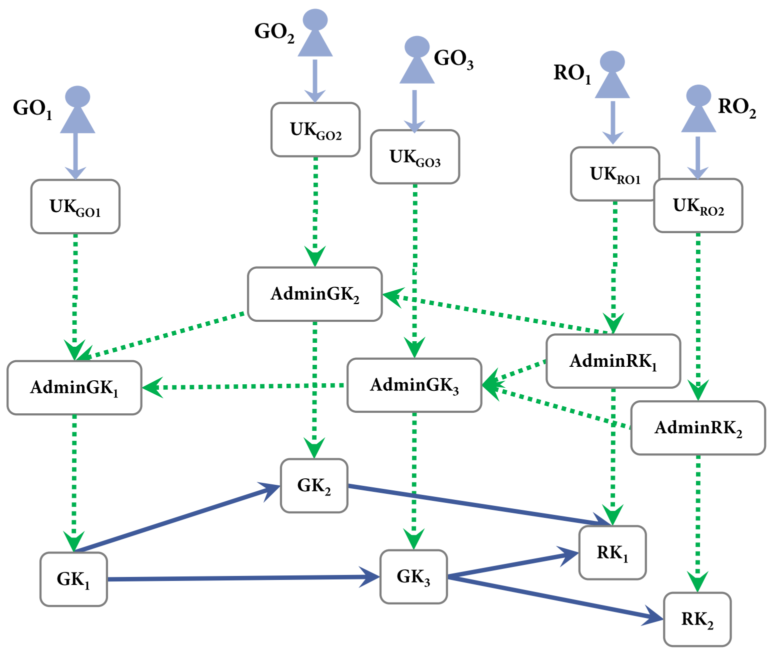


Figure 6.2: Administrative and base key graph after group access is granted.

erated in two cases: first, when a group is granted access to a resource, the administrative resource encryption key is used to sign the administrative group key. This allow the group owner to renew the resource encryption key upon group member removal. Second, when a group is added as member of a host group, the administrative group key of the host group is used to sign the administrative group key of the member group. This way, the owner of the member group can renew the group key of the host group upon group member removal.

In the running example, group g_3 is granted read access on both resources, and group g_2 is granted read access on resource r_1. Furthermore, group g_1 is added as a member of group g_2 and of group g_3. The state of the key graph after these authorization operations is depicted in Figure 6.2.

Let's assume that GO_1 removes a user from group g_1. As discussed in Section 2.2, removing a group member requires to renew all keys that are inferable by the group in the key graph, as the removed group member might have cached some or all of these keys. In the example, a member of group g_1 can infer the keys of the groups g_2 and g_3.

Furthermore, a member of g_1 can infer the resource encryption keys for resources r_1 and r_2. Thus, GO_1 renews the group keys of g_1, g_2 and g_3, and the resource encryption keys for r_1 and r_2. After the key renewal, GO_1 signs all of the keys with the administrative group key of group g_1. In Figure 6.3, the state of the key graph after the group member removal is shown.

In general, removing a user from a group g requires to renew the group keys of groups that g is member of, either directly or transitively. These groups are referred to as *descendants* of group g in the group hierarchy. Furthermore, removing a user from a group g requires to renew the resource encryption keys of all resources that

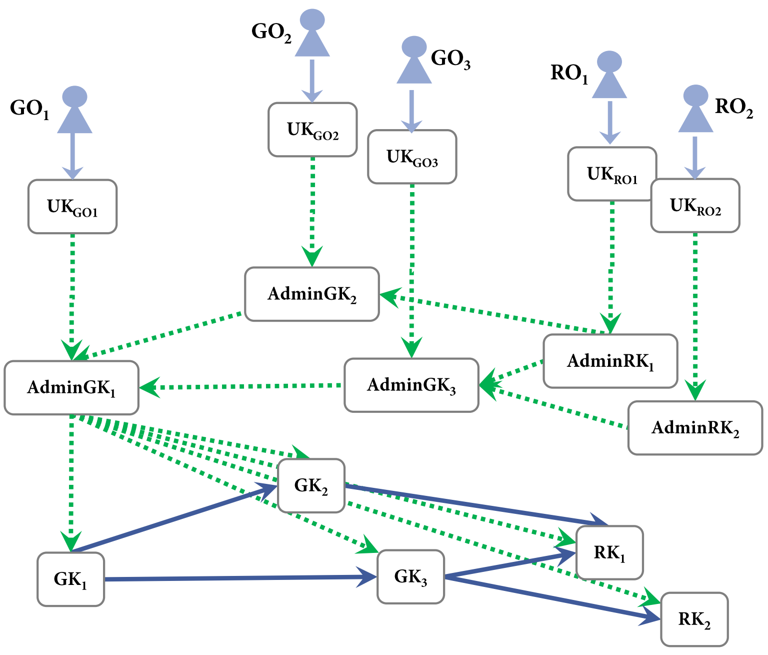


Figure 6.3: Administrative and base key graph after a user was removed from group g_1.

are accessible to at least one of the descendants of g. Depending on the workload, this might result in heavy computational costs and significant network traffic, which will be evaluated later in this chapter.

The administrative key graph can now be used to validate a key signature. If, for example, the signature of the resource encryption key of resource r_2 should be validated, the validator checks the administrative key graph for a path from the administrative resource encryption key of resource r_2 to the administrative group key of group g_1. In the example, this path is given with the administrative group key of group g_3 as intermediate node.

6.2.3 Freshness of Validity State and Key Signatures

As already discussed, the validity state, which is encoded in the administrative key graph by the proposed protocol, changes over time. This requires that signatures related to the administrative key graph have to be expired. For this purpose, it is not sufficient to delete the signatures from the storage provider, as the signatures could simply be restored by everyone who has write access to the storage provider. This way, revoked permissions can be restored, which was discussed as *rollback attack* in Section 5.2. This attack should be prevented, i.e., the freshness of the signatures should be preserved.

In this section, it will be discussed how the freshness of each the different signatures related to the administrative key graph can be protected. For this purpose, three types of signatures have to be distinguished: First, signatures of the binding of an administrative key to its function, which are generated with the user key pair of the owner of the key. Second, signatures of one administrative key with another administrative key. And third, signatures of the binding of a base key to its function, generated with an administrative key.

The first signature type, which binds administrative keys to its function, does not have to be renewed at all. The reason for this is that administrative keys are never changed in the proposed protocol, i.e., the administrative key assigned to a certain function is never renewed.

For the second type of signatures—the signature of an administrative keys with another one—a periodic refreshment of signatures can be applied. These signatures have to be expired whenever the delegation permission represented by the signature is revoked. For this purpose, an expiration date is incorporated into each signature, and the refreshing user stores a fingerprint of the last signed state on her user device. Before the signature expires, the refreshing user checks whether the local fingerprint equals the fingerprint of the state on the storage provider. If this is the case, the signature is refreshed. The periodic refreshment has to be carried out by the user that generated the signature in the first place. For example, when a resource owner grants access to a group, she signs the respective administrative group key with the administrative resource key, as shown in the previous section. From there on, the resource owner is responsible for refreshing the signature periodically.

The third type of signatures, which are generated with an administrative key to sign the binding of a base key to its function, has to be expired whenever the base key is renewed. This way, the old base key that was signed with the administrative key is invalidated. The main difference to the other types of signatures is that possibly multiple users are authorized to generate these signatures. This results from the support for named user groups while omitting joint authorization operations, which was already discussed in detail. Regarding the example in the previous section, the resource encryption key of resource r_1 can be signed by group owners GO_1, GO_2 and GO_3, and resource owner RO_1.

Since multiple users are allowed to generate signatures of the third type, periodic resigning cannot be employed. The reason for this is that the validity of a signature cannot be checked by comparing the signed value on the storage provider with the signed value stored locally, as the value on the storage provider may have been changed by another legitimate user in the meantime.

In fact, it is a fundamental problem to prevent any kind of rollback attack when multiple users share data without at least one trusted entity. This problem was already discussed in Section 5.2. The best freshness guarantee that can be given under these circumstances is *fork consistency*, which is intuitively described as follows: "A file system with fork consistency might conceal users' actions from each other, but if it does, users get divided into groups and the members of one group can no longer see any of another group's file system operations." [MS02] In the context of the protocol, this means that an attacker with read and write access to the storage provider is always able to provide different views of the key graph or the resources to different users. However, after forking the views, the attacker could never rollback any modification of the key graph or the resources, and she could never join these views again without detection.

While approaches were proposed that ensure fork consistency, these approaches exhibit major limitations. For example, they might cause a potentially huge amount of network traffic that quadratically grows in size with the number of clients that share a certain resource, or they require that all users are honest, which contradicts the trust model described in Section 5.1. A further limitation of all of these protocols is that every user has to store some information on his client in order to check the validity of the data on the storage provider. On the opposite, for the periodic signature renewal described above, only resource and group owner have to store validation information locally. This may especially by an issue if a user employs multiple client devices for data sharing, as the validation information has to be synced between these clients.

Because of the major limitations of the approaches that ensure fork consistency, the proposed protocol omits securing signatures from administrative keys to base keys against rollback attacks altogether. Obviously, this tradeoff has a negative impact on the security of the protocol: An attacker with read and write access to the storage provider cannot only fork, but additionally roll back the lockboxes or resources for a user who was already offered or has even downloaded a more recent version of the lockboxes or resources. For example, it is possible that a resource owner that has revoked write permissions for a user can also be provided with data that the revoked user has written, although she was not allowed to do so.

While this can be a serious vulnerability of the protocol in some scenarios, rollback attacks are much harder if some of the users store a copy of the resources locally, e.g., because SDS is implemented as a Sync&Share service. The impact of local caching on the feasibility of rollback attacks was discussed in Section 5.2. In summary, a rollback attack on the key graph also requires a rollback of resources in most cases, which can be detected by a Sync&Share client. Nonetheless, a rollback attack can be successful if the attacker collaborates with a user that was revoked write access to a certain resource. In this case, the rollback attack can be targeted at restoring the write permissions of the malicious user on the resource. If the resource is updated by the malicious user immediately after the rollback, this cannot be detected even when Sync&Share is employed.

6.2.4 Implementation Discussion

In this section, some issues are discussed that have to be considered before the protocol can be employed in practice.

First, the question arises whether the keys should be signed before or after they are encrypted, i.e., whether keys or lockboxes should be signed. These two alternatives are known as *sign-then-encrypt* and *encrypt-then-sign*, respectively. The security implications of the alternatives are discussed in Section 2.2. From a performance perspective, it is better to sign the keys before encryption: Keys often are encrypted multiple times with different other keys, i.e., multiple lockboxes that encrypt the same key exist. Thus, if the keys would be signed after encryption, a signature would have to be generated for each of these lockboxes, instead of just a single signature with sign-then-encrypt.

To make a signature available to the users, it has to be put on the storage provider, together with some auxiliary information. This information includes at least the function of the signed key within the protocol, and the public key that can be used to validate the signature. The signature and the auxiliary information can be compiled within one data structure, and this data structure can be addressed by the function within the protocol that was signed, e.g., "signature of encryption key of resource *r*".

An important question is how users can be enabled to traverse the group hierarchy. The protocol requires this to be done, e.g., when a group member must retrieve the group key of some descendant group to read or write a resource, as the descendant group was granted access to the resource, and the group member leverages her indirect membership of the descendant group. As another example, a group owner has to be able to identify all descendants of her group to renew all of their group keys after a group member removal. Traversing the group hierarchy is also necessary to validate signatures: The validation of base key graph signatures requires to traverse the administrative key graph, in this key graph closely reflects the group hierarchy.

One solution for this issue is to have the storage provider offer facilities for group hierarchy traversal. From a security point of view, this does not provide any attack vectors: There is no point in pretending an edge in the group hierarchy that does not exist, as this edge is not reflected in the key graph. On the other hand, an attacker could pretend that an edge in the group hierarchy does not exist; however, this would only prevent some users from reading or writing resources, but has no implications for the confidentiality or integrity of the resources.

Just as for SDS protocols that leverage joint authorization operations, the performance of processing group member removal events can be increased by extending the protocol with Group Key Management approaches (cf. Section 2.2). For the performance evaluation presented in the next section, the protocol was extended with a *Logical Key Hierarchy* for each group. Note that employing a Group Key Management approach with auxiliary keys requires that the group owner is able to retrieve every auxiliary key. At the same time, the other group members are only allowed to retrieve a subset of the auxiliary keys. In combination with the support for group hierarchies, this poses a new challenge: A group owner who wants to remove a member from her group must be enabled to retrieve every auxiliary key of every descendant group. For this purpose, the key graph is extended by a further part, which will not be described in detail here.

6.3 Protocol Details

In this section, the cryptographic operations necessary to process the different authorization operation types are listed as pseudocode. For the sake of simplicity, some steps are omitted in the listings:

abbreviation	description
UK_u	asymmetric user key pair of user u
GK_g	asymmetric group key pair of group g
PK_p	asymmetric key pair of principal p, which is either a user or a group
RK_r	symmetric resource encryption key for resource r
RSK_r	asymmetric resource signing key pair for resource r
$AdminRK_r$	asymmetric administrative key pair used to sign the *RK* for resource r
$AdminRSK_r$	asymmetric administrative key pair used to sign the *RSK* for resource r
$AdminGK_g$	asymmetric administrative key pair used to sign the *GK* for group g

Table 6.1: Naming conventions for cryptographic data structures used in the E2E-SDS protocol.

– As an RSA public key of a given length cannot be used to encrypt an RSA private key of the same length (cf. Section 2.2), symmetric keys are used as "intermediate keys", i.e., a public key is used to encrypt a symmetric key, which in turn encrypts a private key. These intermediate symmetric keys are omitted.

– After downloading an encrypted key, the signature of this key must be validated, which may require to traverse the administrative key graph. Signature validation is omitted in the listings.

– Group Key Management using a Logical Key Hierarchy is omitted, as well as the group rekeying key graph introduced in the previous section.

However, each of these steps is implemented in the simulation model that was used for the performance evaluation.

The abbreviations used in the listing are shown in Table 6.1. The symbol ↑ appended to the generation of some cryptographic data structure means that the structure is uploaded on the storage provider after the generation. The symbol ↓ prepended to a decryption means that the cryptographic data structure has to be downloaded from the storage provider before decryption.

```
// add user u

// generate user key pair
UK_u = GenAsymKeyPair(PP)
```

Listing 6.1: AddUser operation

Adding a user simply requires to generate the asymmetric user key pair.

```
// add resource $r$

// generate resource encryption key and signing key pair
$RK_r = GenSymKey(PP)$
$RSK_r = GenAsymKeyPair(PP)$
// generate and upload lockboxes for resource encryption
   key and signing key pair
$Locked(RK_r, u) = EncAsym(RK_r, UK_u^{pub}) \uparrow$
$Locked(RSK_r, u) = EncAsym(RSK_r, UK_u^{pub}) \uparrow$
// generate administrative keys
$AdminRK_r = GenAsymKeyPair(PP)$
$AdminRSK_r = GenAsymKeyPair(PP)$
// generate and upload lockboxes for administrative keys
$Locked(AdminRK_r, u) = EncAsym(AdminRK_r, UK_u^{pub}) \uparrow$
$Locked(AdminRSK_r, u) = EncAsym(AdminRSK_r, UK_u^{pub}) \uparrow$
// sign resource encryption key and signing key pair with
   administrative keys
$Sign(Id_r\&RK_r, AdminRK_r) \uparrow$
$Sign(Id_r\&RSK_r, AdminRSK_r) \uparrow$
// sign administrative key pairs with user private key
$Sign(Id_r\&AdminRK_r, UK_u) \uparrow$
$Sign(Id_r\&AdminRSK_r, UK_u) \uparrow$
```

Listing 6.2: AddResource operation

Processing an *AddResource* authorization operation involves the generation of a symmetric encryption key and an asymmetric signing key pair. In addition, the administrative key pairs for the resource have to be generated: One administrative key pair is used to sign the binding of the encryption key to its function, while another administrative key pair is necessary to sign the binding of the resource signing key pair to its function. All of the generated keys and key pairs are encrypted with the user public key of the resource owner, and uploaded to the storage provider.

```
// allow read and write access for user or group $a$

// download and decrypt lockboxes for resource encryption
   key and signing key pair
$\downarrow RK_r = DecAsym(Locked(RK_r, u), UK_u^{priv})$
$\downarrow RSK_r = DecAsym(Locked(RSK_r, u), UK_u^{priv})$

// generate and upload lockboxes for resource encryption
   key and signing key pair
$Locked(RK_r, a) = EncAsym(RK_r, UK_a^{pub}) \uparrow$
$Locked(RSK_r, a) = EncAsym(RSK_r, UK_a^{pub}) \uparrow$

if ($IsGroup(a)$)
```

```
 // download and decrypt lockboxes for administrative
    resource keys
 ↓ $AdminRK_r = DecAsym(Locked(AdminRK_r, u), UK_u^{priv})$
 ↓ $AdminRSK_r = DecAsym(Locked(AdminRSK_r, u), UK_u^{priv})$
 // sign administrative group key with admin resource keys
 $Sign(Id_g \& GK_g, AdminRK_r)$ ↑
 $Sign(Id_g \& GK_g, AdminRSK_r)$ ↑
```

Listing 6.3: AllowReadWrite operation

To grant a user or a group read and write access to a resource, the encrypted resource signing key pair and encryption key are downloaded from the storage provider. These keys are encrypted with the public key of the user or the group that should get access. If access is granted to a group, this has to be reflected in the administrative key graph. For this purpose, the administrative group key is signed with both administrative keys of the resource.

```
// deny read and write access for user or group $a$

// generate new resource encryption key and new signing key
    pair
$RK_r = GenSymKey(PP)$
$RSK_r = GenAsymKeyPair(PP)$

// download and decrypt lockboxes for administrative
   resource keys
↓ $AdminRK_r = DecAsym(Locked(AdminRK_r, u), UK_u^{priv})$
↓ $AdminRSK_r = DecAsym(Locked(AdminRSK_r, u), UK_u^{priv})$

// sign new resource encryption key and new signing key
   pair with administrative keys
$Sign(Id_r \& RK_r, AdminRK_r)$ ↑
$Sign(Id_r \& RSK_r, AdminRSK_r)$ ↑

// for all remaining readers:
foreach($a$ in GetPrincipalsWithReadPermission($r$))
 // generate and upload lockboxes for new resource
    encryption key
 $Locked(RK_r, a) = EncAsym(RK_r, PK_a^{pub})$ ↑

// for all remaining writers:
foreach($a$ in GetPrincipalsWithReadWritePermission($r$))
 // generate and upload lockboxes for new signing key pair
 $Locked(RSK_r, a) = EncAsym(RSK_r, PK_a^{pub})$ ↑
```

Listing 6.4: DenyReadWrite operation

To deny read and write access to a user or a group, the resource encryption key and the resource signing key pair have to be renewed. The new encryption key is signed with the administrative resource encryption key, and encrypted with the public keys of all remaining reader. Similar to this, the new signing key pair is signed with the administrative resource signing key, and encrypted with the public keys of all remaining writers. If access was denied to a group, the signatures of the group administrative key generated with either of the administrative resource keys must be removed from the storage provider.

```
// add group $g$

// generate group key pair
$GK_g = GenAsymKeyPair(PP)$

// generate and upload lockbox for group key pair
$Locked(GK_g, u) = EncAsym(GK_g, UK_u^{pub}) \uparrow$

// generate administrative group key
$AdminGK_g = GenAsymKeyPair(PP)$

// generate and upload lockbox for administrative group key
    pair
$Locked(AdminGK_g, u) = EncAsym(AdminGK_g, UK_u^{pub}) \uparrow$

// sign group key pair with administrative group key
$Sign(Id_g \& GK_g, AdminGK_g) \uparrow$

// sign administrative group key pair with user private key
$Sign(Id_g \& AdminGK_g, UK_u) \uparrow$
```

Listing 6.5: AddGroup operation

The addition of a group requires to generate a group key pair, and an administrative group key pair. The group key pair is signed with the administrative group key. Both key pairs are encrypted with the public key of the group owner, and uploaded to the storage provider.

```
// add member $p$ to group $g$

// download and decrypt lockbox for group key pair
$\downarrow GK_g = DecAsym(Locked(GK_g, u), UK_u^{priv})$

// generate and upload lockbox for group key pair for
   principal $p$
$Locked(GK_g, p) = EncAsym(GK_g, PK_p^{pub}) \uparrow$

if ($IsGroup(p)$)
```

```
        // sign group key pair with administrative group
           key
        $Sign(Id_p \& GK_p, AdminGK_g) \uparrow$
```

Listing 6.6: AddMemberToGroup operation

A user or group can be added to a group simply by encrypting the group key pair with the new member's public key. If a group is added, the administrative key graph has to be extended by signing the administrative group key of the added group with the administrative group key of the host group.

```
// remove principal $p$ from group $g$

// generate new group key pair
$GK_g = GenAsymKeyPair(PP)$

// download and decrypt lockboxes for administrative group
   keys
$\downarrow AdminGK_g = DecAsym(Locked(AdminGK_g, u), UK_u^{priv})$

// sign group key pair with administrative group key
$Sign(Id_g \& GK_g, AdminGK_g) \uparrow$


// for all remaining group members:
foreach($a$ in GetGroupMembers($g$))
        // generate and upload lockbox for group key pair
        $Locked(GK_g, p) = EncAsym(GK_g, PK_p^{pub}) \uparrow$

if ($IsGroup(a)$)
        // recursively rekey descendant groups
        foreach($d$ in GetDescendantGroups($g$))
                $RekeyGroup($d,g$)$

        // rekey all resources that the group has read
           access to
        foreach($r$ in GetReadAccessibleResources($g$))
                $RekeyResourceRead($r,g$)$

        // rekey all resources that the group has read and
           write access to
        foreach($r$ in GetReadWriteAccessibleResources($g$))
                $RekeyResourceReadWrite($r,g$)$
```

Listing 6.7: RemoveMemberFromGroup operation

Removing a member from a group can be expected to be the most expensive and complex operation. For this purpose, a new group key pair has to be generated and signed with the administrative group key. This new group key pair is distributed to all remaining group members by encrypting it with the public keys of these members. If the removed member is a group, the signature of the removed group's administrative key generated with that of the hosting group must be deleted from the storage provider. As already discussed in the previous section, the group keys of all descendant groups must also be renewed, which is shown in Listing 6.8. Finally, all resources that the group is allowed to read or write must be rekeyed. The rekeying process is shown in Listing 6.9 for the case that the group has read and write access.

```
// rekey group $d$
// rekeying process initiated by group owner of group $g$

// generate new group key pair
$GK_d = GenAsymKeyPair(PP)$

// sign group key pair with administrative group key of
   group $g$
$Sign(Id_d \& GK_d, AdminGK_g) \uparrow$


// for all remaining group members:
foreach($a$ in GetGroupMembers($d$))
        // generate and upload lockbox for group key pair
        $Locked(GK_d, p) = EncAsym(GK_d, PK_p^{pub}) \uparrow$

// rekey all resources that the group has read access to
foreach($r$ in GetReadAccessibleResources($d$))
        $RekeyResourceRead$(r,g)

// rekey all resources that the group has read and write
   access to
foreach($r$ in GetReadWriteAccessibleResources($d$))
        $RekeyResourceReadWrite$(r,g)
```

Listing 6.8: RekeyGroup operation that is used as subprocedure within RemoveMemberFromGroup

Rekeying a group is similar to removing a member from the group: The group key pair is renewed, and the new key pair is distributed to all group members. However, the new group key pair is not signed with the administrative group key of the group that is to be rekeyed. Instead, it is signed with the administrative key of the group that a member was removed from, i.e., whose group owner initiated the rekeying process in the first place.

```
// rekey resource $r$
// rekeying process initiated by group owner of group $g$

// generate new resource encryption key and new signing key
    pair
$RK_r = GenSymKey(PP)$
$RSK_r = GenAsymKeyPair(PP)$

// sign new resource encryption key and new signing key
   pair with administrative key of group $g$
$Sign(Id_r \& RK_r, AdminGK_g) \uparrow$
$Sign(Id_r \& RSK_r, AdminGK_g) \uparrow$

// for all resource readers:
foreach($a$ in GetPrincipalsWithReadPermission($r$))
 // generate and upload lockboxes for new resource
    encryption key
 $Locked(RK_r, a) = EncAsym(RK_r, PK_a^{pub}) \uparrow$

// for all resource writers:
foreach($a$ in GetPrincipalsWithReadWritePermission($r$))
 // generate and upload lockboxes for new signing key pair
 $Locked(RSK_r, a) = EncAsym(RSK_r, PK_a^{pub}) \uparrow$
```

Listing 6.9: RekeyResourceReadWrite operation

To rekey a resource, almost the same cryptographic operations are necessary as when denying access to the resource: The resource encryption key and signing key pair are renewed, and distributed to all users and groups who have access to the resource. The new keys are not signed with the administrative keys of the resource, but with the administrative group key of the group owner that carries out the rekeying process.

```
// resource $r$ is written

// if user $u$ has direct read and write access to resource $r$
if ($HasDirectReadWriteAccess(u, r)$)
        // download and decrypt lockboxes for resource
           encryption key and signing key pair
        $\downarrow RK_r = DecAsym(Locked(RK_r, u), UK_u^{priv})$
        $\downarrow RSK_r = DecAsym(Locked(RSK_r, u), UK_u^{priv})$

// if user $u$ has read and write access to resource $r$ via
   group membership
else
        // iterate over path in group hierarchy;
        // start at a group that user $u$ is member of
        // end at a group with write access to resource $r$
        for($g_1$ to $g_n$ in GetWriteGroupPath($u$, $r$))
                $\downarrow GK_{g_{i+1}} = DecAsym(Locked(GK_{g_{i+1}}, g_i), GK_{g_i}^{priv})$

        // download and decrypt lockboxes for resource
           encryption key and signing key pair
        $\downarrow RK_r = DecAsym(Locked(RK_r, g_n), GK_{g_n}^{priv})$
        $\downarrow RSK_r = DecAsym(Locked(RSK_r, g_n), GK_{g_n}^{priv})$


// encrypt resource and upload
$Locked(\mathtt{r}) = EncSym(\mathtt{r}, RK_r) \uparrow$

// sign resource and upload signature
$Sign(\mathtt{r}, RSK_r) \uparrow$
```

Listing 6.10: WriteData operation

To be able to write a resource, the user first retrieves the resource encryption key and the resource signing key pair. If the user has was granted access to a resource implicitly by a group membership, she has to traverse the group hierarchy from a group she is member of to the group that was granted access. While traversing the group hierarchy, the group key pair of one group is used to decrypted the group key pair of the next group. With the last group key pair, the resource encryption key and signing key pair can be decrypted. Finally, the resource is encrypted with the encryption key, and signed with the signing key pair.

6.4 Performance Evaluation Results

In this section, the performance of the E2E-SDS protocol introduced in this chapter is evaluated. In the first part, the performance is evaluated with regard to real-world workloads. As these workloads contain only a small number of the most resource-consuming events, i.e., *RemoveUserFromGroup* and *RemoveGroupFromGroup* events, synthetic workloads are used to carry out a sensitivity analysis in the second part of this section.

6.4.1 Performance Evaluation Results for Real-World Sharing Scenarios

To get an indication of the real-world employability of the protocol, its performance is evaluated with regard to real-world workloads. The performance evaluation focuses on the extended version of the protocol that leverages a Logical Key Hierarchy for group key management (cf. Section 5.3.1). Furthermore, the performance evaluation only considers lazy revocation (cf. Section 2.2).

The workloads used for the evaluation were characterized in Section 3.5. For the performance evaluation, the same method is used as for the performance evaluation of the E2E-SDS protocols working with joint authorization operations, which was presented in Section 5.3. The device-dependent computation costs for cryptographic operations, which were measured with micro benchmarks, are reused. Thus, the evaluation is based on the assumption that RSA with 2 048 bit keys and AES with 256 bit keys is used.

The protocol described in this chapter can be applied both for workloads that contain group hierarchies, and for workloads without group hierarchies. However, with regard to workloads that do not contain any group hierarchies, the performance of the protocol is not substantially different from the performance of the extended protocol presented in Section 5.3. In fact, the computation time and network traffic required for processing an authorization operation increase by a constant value, regardless of the authorization operation type. Thus, no simulations were carried out based on workloads without group hierarchies. Unfortunately, the majority of the workloads characterized in Section 3.5 does not contain any group hierarchies. Therefore, the protocol was only simulated with regard to 13 workloads.

authorization operation type	**computation time**	**incoming network traffic**	**outgoing network traffic**
AddResource	6.24 s (6.28 s; 7.68 s)	0	21 KB
AllowRead	0.11 s (0.14 s; 0.97 s)	6.4 KB	1 KB
AllowReadWrite	0.22 s (0.27 s; 1.93 s)	21 KB	2 KB
AddGroup	5.44 s (5.65 s; 12.47 s)	0.7 KB	27 KB

Table 6.2: Performance of authorization operation types with constant computation times and network traffic. The computation times are given for a Dell Latitude E4310, a Samsung Galaxy S5 mini, and an iPhone 3G.

The performance evaluation results for authorization operation types that require a constant computation time and cause constant network traffic are shown in Table 6.2. In this section, the computation times are given for a Dell Latitude E4310, and the computation times for a Samsung Galaxy S5 mini and an iPhone 3G are appended in parentheses. In relative terms, the computation times are significantly higher than for the E2E-SDS protocols that leverage joint authorization operations. This is caused by the generation of additional RSA key pairs, which constitute the administrative key graph.

The computation times for processing *AddUserToGroup* and *AddGroupToGroup* events depends on whether the key hierarchy of the group has to be extended. On average, adding a user to a group takes 0.13 s (0.16 s; 1.12 s). To add a group to another group, 0.19 s (0.23 s; 1.64 s) are required on average.

The performance of processing a *DenyRead* or *DenyReadWrite* event is not constant, but depends on the number of users and groups that have read or write access to the resource in question. The average computation time to process a *DenyReadWrite* event is 1.66 s (1.70 s; 2.95 s). Using the extended E2E-SDS protocol introduced in the last chapter, processing a *DenyReadWrite* event only takes 1.54 s (1.56 s; 1.93 s) in average, which is obviously slightly less for the Dell Laptop and the Galaxy S5 mini, and about 35% less for the iPhone. However, in absolute terms, the required computation time is still not too large.

The most interesting authorization operation types are *RemoveUserFromGroup* and *RemoveGroupFromGroup* on the one hand, since the processing of these event types works completely different from that in the E2E-SDS protocols based on joint authorization operations, and potentially requires a lot of resources. As already shown, these event types occur only rarely in the real-world workloads. Because of the small sample size, synthetic workloads are used to assess the sensitivity of the performance with regard to certain workload parameters, which is presented in the next subsection. Nonetheless, the real-world results might give a first indication for the employability of the protocol. Removing a user from a group takes 38 s (40 s; 99 s) in average, and a maximum of 911 s (935 s; 1608 s) was observed. The removal of a group from another group takes 48 s (56 s; 320 s) on average, which is slightly more than for a user removal at the Dell laptop and the Samsung Galaxy S5 mini smartphone, and about two times more on the iPhone 3G. The higher computation times for the removal of a group can be explained by the necessary renewal of the key hierarchies of all descendant groups, which is omitted in case of a user removal. Besides lots of symmetric cryptographic operations, the key hierarchy renewal requires one asymmetric encryption per group member. These asymmetric operations takes a comparatively long time, especially on the old iPhone.

The incoming network traffic volume for processing a *RemoveUserFromGroup* event is 0.45 MB on average, with a maximum of 10.43 MB. For processing a *RemoveGroup-FromGroup* event, only 0.22 MB of incoming network traffic are required on average. The reason for this difference is that removing a user from a group requires to fetch a part of the key hierarchy of each group from the storage provider. On the other hand, to remove a group from another group, the key hierarchies of all descendant groups are deleted, and rebuilt from scratch.

The outgoing network traffic for removing a user from a group is 0.31 MB on average, the maximum is 5.58 MB. With an average of 0.47 MB, the outgoing network traffic generated when removing a group from another group is slightly higher, which is due to the key hierarchy renewal for the descendant groups.

The last event type that is considered within this performance evaluation are *Write-Data* events. The processing of these events differs from that of E2E-SDS protocols with joint authorization operations, as it requires the validation of signatures with the help of the administrative key graph. As already discussed in Section 5.3, the costs of processing a *WriteData* event can be partitioned into costs that depend on the size of the resource written, and costs that are independent of this size. The latter is referred to as *key management costs*, and only these costs are considered in this section. The average key management computation time for writing data is 0.12 s (0.15 s; 1.09 s). These computation times are only slightly higher than with the SDS protocols using joint authorization operations. However, the comparatively low computation times can be explained by the fact that in the real-world workloads, users never need to leverage indirect group memberships to get write permissions. If the users had to traverse the group hierarchy, additional asymmetric cryptographic operations had to be carried out. The incoming network traffic is 31 KB on average, but would also increase if writers had to traverse the group hierarchy.

6.4.2 Sensitivity Analysis

As the performance evaluation results for real-world workloads showed, processing *RemoveUserFromGroup* and *RemoveGroupFromGroup* events requires a lot more computation time than processing other authorization operation types. This is a consequence of a key property of the protocol introduced in this chapter: All cryptographic operations that are necessary to enforce a group member removal are carried out by the group owner initiating the removal. However, as the real-world workloads collected for this thesis only comprise a small number of *RemoveUserFromGroup* and *RemoveGroupFromGroup* events, the influence of the workload on the performance cannot be assessed comprehensively and systematically. For this reason, the sensitivity analysis presented in this section is based on synthetic workloads, which were generated according to the method introduced in Section 3.6.

For the sensitivity analysis, the workload parameters are varied that obviously influence the performance of processing *RemoveUserFromGroup* and *RemoveGroupFromGroup* events. An analysis of the protocol description in Section 6.3 shows that from the workload parameters listed in Section 3.4, the parameters *users per group*, *group members per group*, and *group memberships per group* are most likely to have an impact on the performance. Since *group members per group* and *group memberships per group* depend on each other, only one of these parameters is varied, which was arbitrarily chosen to be *group memberships per group*.

The workload generation method allows to define the workload parameters as probability distributions. The approach taken to vary the workload parameters is to approximate the frequency distribution of the real-world parameter with a well-known probability distribution by curve fitting, and then vary the parameters of this probability distribution. The distribution parameters are varied in a way that the mean of the distribution takes some predefined values.

The real-world samples taken for *group members per group* and *group memberships per group* can both be fitted to a Gamma distribution, which is characterized by two distribution parameters, k and θ. As varying either k or θ changes the mean of the Gamma distribution, one degree of freedom is left for the parametrization of the Gamma distribution to reach a given mean. In general, two different Gamma distribution parametrizations were chosen per target mean value and workload parameter. However, for some distribution parametrizations, the target mean of the distribution and the actual average of the generated workload strongly differ, so this parametrizations had to be omitted in the further analysis.

The experimental setup follows a full factorial design with regard to the means of the parameters *users per group* and *group memberships per group*. Because of large simulation running times, especially when the average number of group member-

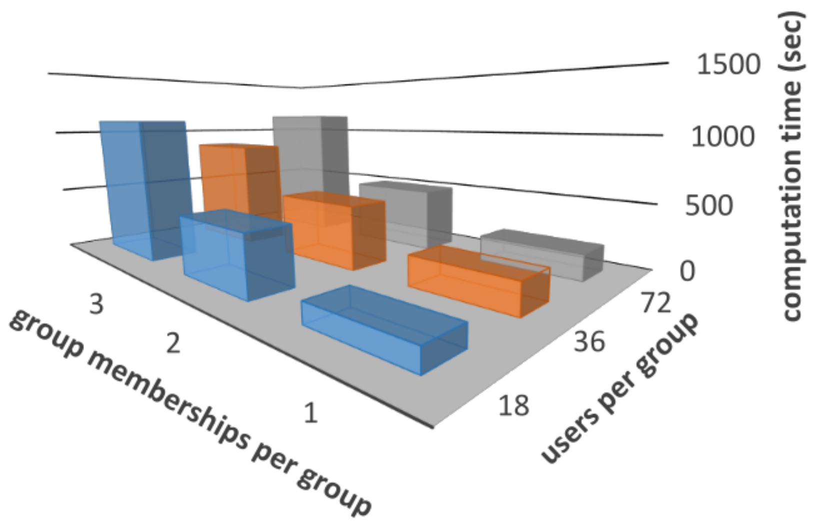


Figure 6.4: Sensitivity of the average computation time for group member removal operations. The simulations are based on synthetic workloads.

ships per group rises above 3, only a small value space could be explored. As presented in Section 3.5, a group that is part of the real-world workloads has 36 user members and 1 group membership in average. Based on these real-world averages, the target means for the parameter distributions were chosen as 18, 36 and 72 for the parameter *users per group*, and as 1, 2 and 3 for the parameter *group memberships per group*.

The influence of the workload parameter on the performance of processing the group member removal events is a priori expected as follows: The parameter *users per group* is expected to have only a minor impact on the performance of the *RemoveUserFromGroup* event, since the performance of renewing a group key depends only logarithmically on the group size. In addition, a larger group size only increases the number of necessary symmetric cryptographic operations, which are cheap in comparison to asymmetric ones. The parameter *users per group* is expected to have a larger impact on the performance of the *RemoveGroupFromGroup* event, as in this case, the key hierarchies of all descendant groups must be renewed. This requires not only a lot of symmetric cryptographic operations, but also one asymmetric encryption for each group member. Thus, the performance should linearly depend on the total number of group members of all descendant groups.

The influence of the parameter *group memberships per group* is much harder to estimate: This parameter can be expected to positively correlate with the number of descendants per group, which determines the number of groups that have to be rekeyed after a group member removal. However, this dependency is hard to quantify in general. In literature, no hints on this dependency could be found.

Figure 6.4 shows the average computation time necessary to process a *RemoveUserFromGroup* event for different means of *users per group* and *group memberships per group*. It can be observed that the parameter *group memberships per group* has a

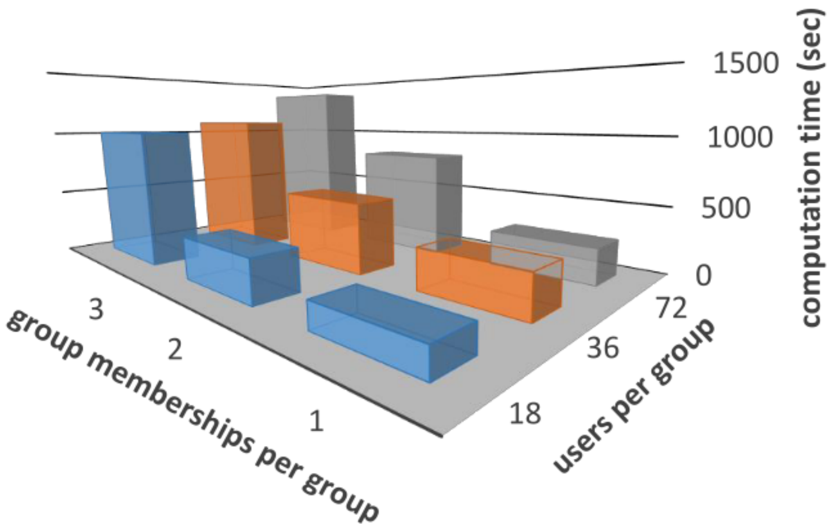


Figure 6.5: Sensitivity of the average computation time for removing a group from another group. The simulations are based on synthetic workloads.

large impact on the computation times, leading to a superlinear rise. The impact of the parameter *users per group* can hardly be assessed from the simulation results: While increasing this parameter leads to a small rise in the computation times for *group memberships per group* = 2, as it was expected, it shows a different behavior for other values of *group memberships per group*. The reason for these ambiguous results is that due to the potentially large simulation runtime, only few workloads could be simulated.

In Figure 6.5, the average computation times required to process a *RemoveGroupFromGroup* event are plotted. The results also show that higher values of *group memberships per group* yield a superlinear rise of the computation times. In addition, the plot indicates that the parameter *users per group* also has a substantial impact on the computation times. However, this impact cannot be clearly quantified from the simulation results, for the reason already mentioned above.

The trends that can be observed in the simulation results for incoming and outgoing traffic are very similar to those just presented, and do not provide any additional insights. For this reason, these results are omitted here.

In conclusion, the sensitivity analysis shows that the workload parameter *group memberships per group* has a major impact on the performance of group member removals, as increasing this parameter results in a superlinear rise of the computation times, the incoming and the outgoing network traffic. While the sensitivity analysis indicates that the parameter *users per group* also substantially influences the performance of the *RemoveGroupFromGroup* event, further simulations would be necessary to get more significant results.

6.5 Conclusions and Outlook

In this chapter, an E2E-SDS protocol was introduced that supports named user groups, but works without the need for joint authorization operations between users. For this purpose, the protocol leverages an administrative key graph as an additional part of the key graph. This way, the protocol does not require any additional cryptographic primitives, compared to the traditional E2E-SDS protocols using joint authorization operations (cf. Section 5.3.1). With the administrative key graph, the support for group hierarchies could be added to the protocol in a straight forward manner.

The major challenge in the E2E-SDS protocol design is ensuring the freshness of the signatures that are part of the administrative key graph. If only one single user is authorized to generate a certain signature, this signature can be refreshed by the user periodically. However, it was shown that the support for named user groups while omitting joint authorization operations implies that there are signatures that are potentially generated by multiple users. In this case, the best notion of freshness that can be achieved is *fork consistency*. Since existing solutions to ensure fork consistency have several drawbacks in practice, e.g., a huge network traffic overhead, the protocol does not ensure the freshness of signatures that can be generated by multiple users. First, this allows an attacker who has read and write access on the storage provider to roll back resources to an earlier version. However, this attack can be detected if the resources are cached locally, e.g., if Sync&Share is used. Second, if the attacker additionally collaborates with a user that had write access to a resource in the past, a rollback attack can enable the revoked user to regain write access to the resource. If the resource is then written immediately after the rollback, the attack cannot be detected even when caching the resources locally.

The performance of the E2E-SDS protocol was evaluated based on real-world workloads. The evaluation shows that in relative terms, the computation times are significantly higher than those of the traditional E2E-SDS protocols based on joint authorization operations. However, in absolute terms, the computation times are mostly in the range of only a few seconds. For example, although adding a resource is expensive compared to most of the other authorization operation types, it only takes between 6 and 8 seconds on average. Similar to the E2E-SDS protocols based on joint authorization operations, the most expensive authorization operation types are *RemoveUserFromGroup* and *RemoveGroupFromGroup* events, which require computation times that are a magnitude larger than for other authorization operation types, e.g., 38 s on average using the Dell laptop. While the outgoing network traffic generated by these event types is also a magnitude larger than the traffic caused by the other event types, it is still below 0.5 MB on average.

To assess the sensitivity of the performance of the expensive group member removals against changes in workload parameters, a sensitivity analysis was carried out. This analysis was based on synthetic workloads, which were generated according to the method introduced in Section 3.6. The analysis results show that the performance highly depends on the workload parameter *group memberships per group*, which has a superlinear influence on the performance for small values. In addition, the results show that the parameter *users per group* also has a significant influence on the performance, as it can be expected from analyzing the operations carried out by the protocol. However, the results do not show a clear trend. This is probably due to the comparatively small number of workloads that were simulated.

As future work, mechanisms to hinder rollback attacks could be designed and integrated into the protocol. Although, e.g., the SUNDR file system [LKMS03] offers fork consistency, i.e., the strongest notion of freshness that can be achieved in a multi-writer setting with only a semi-trusted storage provider, SUNDR does hardly scale with regard to the generated network traffic. When designing a mechanism that hinders rollback attacks, possibly a trade-off has to be made between the degree to which rollback attacks are still possible, e.g., the time frame within which can be rolled back, and the resulting performance penalties.

7 Summary, Conclusions and Outlook

The research object of this thesis are protocols that enable *End-to-End Secure Data Sharing (E2E-SDS)*. E2E-SDS uses client-side cryptography to provide confidential and integer data sharing without a trusted entity outside the group of sharing users. Thus, the major part of authorization and access control enforcement is carried out solely by the users. However, the chances that E2E-SDS is accepted by potential users depend on the loss in user comfort that these systems bring along. Besides some other aspects, this means that the authorization models the users are familiar with should be retained, and the performance penalties on the users' devices should be negligible.

Methodological Challenges and Corresponding Contributions

This thesis addresses the question how to evaluate the real-world performance of E2E-SDS protocols. In literature, the performance of such protocols was prevalently evaluated with analyses of the asymptotic behavior. Two challenges for the evaluation of the real-world performance were identified and addressed in this thesis.

The first challenge lies in the coarse-grained modeling that asymptotic analyses deliberately strive to achieve, but that abstracts from E2E-SDS protocol details that potentially have a major impact on the real-world performance. This challenge is addressed in the thesis by introducing two performance evaluation methods: an analytical and a simulative method. Both methods are based on workloads, which constitute a basic sharing and usage model, i.e., a formalization of the sharing and usage behavior of the users in the system. The methods can be used to estimate the computation time and the incoming and outgoing network traffic required on a user's client device by the protocol under study.

The analytical method can be considered a natural extension of asymptotic analyses. The method models the protocol as a set of algebraic expressions, with one expression per workload event type and per performance metric. The performance is evaluated by populating the expression with concrete costs of cryptographic operations on a certain device, and with concrete values for the workload parameters. The analytical method does not require fully specified workloads, but is based on more abstract workload parameters. Moreover, the performance evaluation model directly reflects the influence of a workload parameter on the performance.

The major limitations of the analytical method are, first, that the workload parameters involved in the algebraic expressions must be known or be estimable up front. Second, the analytical method is much harder to apply if an algebraic expression involves two or more workload parameters that cannot be assumed to be independent. In this case, either the expression has to be populated with joint frequency distributions, which are harder to estimate than univariate ones, or the evaluation has to be based on probability distributions that may be mathematically cumbersome to calculate with. The former option can be considered a step towards a simulative performance evaluation.

The simulative performance evaluation method introduced in this thesis leverages the key graph as a unified level of abstraction for E2E-SDS protocols. A key graph represents the state of a E2E-SDS systems in terms of cryptographic data structures, such as keys, lockboxes and digital signatures, and the links between these data structures. On this abstraction level, an E2E-SDS protocol is specified by the way it manipulates the key graph when processing a workload event. Although the concept of a key graph was already introduced in literature, the concept was extended in this thesis to be able to infer the protocol performance from key graph changes. Opposed to the analytical method, the simulative method requires fully specified workloads. Moreover, deriving the relations between workload parameters and the performance by inspecting the protocol model is harder than with the analytical method. Furthermore, the simulative method can only be applied to E2E-SDS protocols that retain to keys, lockboxes and signatures as cryptographic data structures.

The performance evaluations of E2E-SDS protocols that were carried out for this thesis could not have all been based on only one of the two methods, so the unique features of both methods were relevant for the thesis. The reasons for this will be discussed shortly.

The second challenge for evaluating the real-world performance is that the knowledge of the sharing and usage model of the E2E-SDS system is often incomplete, i.e., it is unclear which values a given workload parameter typically takes in a real-world setting. In this thesis, this challenge is addressed by providing a characterization of real-world workloads. These workloads were generated from traces of two real-world sharing services running at Karlsruhe Institute of Technology (KIT): an e-learning and a groupware platform. While these workloads cannot be claimed to be representative, a performance evaluation based on these workloads at least gives some rough indication on the significance of the performance penalties. The characterization might provide some guidance on the values or distributions to choose for unknown workload parameters when using the analytical performance evaluation method.

In many cases, the sharing models cannot be obtained by observing real-world sharing systems. In these cases, it is assumed that a domain expert could at least provide an estimation for some workload parameters regarding the E2E-SDS system under study. In this thesis, estimated workload parameters serve as a target that synthetic workloads should adhere to. Synthetic workloads can be directly used as input for the simulative performance evaluation method, or can be characterized with regard to workload parameters that are required as input of the analytical method.

The presented workload generation method prohibits that sets of dependent workload parameters are completely specified, and therefore avoids conflicting requirements on the generated workload. The workload generation problem includes a family of graph generation problems, whose members differ on the set of workload parameters that is specified. In this thesis, it was proven that certain instances of this graph generation problem are NP-hard problems. These NP-hard workload generation problem instances have in common that at least one transitive parameter is specified, i.e., a parameter that reflects indirect relations between users and the resources that are accessible to the users.

Contributions with Regard to the Application of Methods

The presented set of methods is employed in this thesis to evaluate the performance of both existing and novel E2E-SDS protocols. The performance evaluations are based on the real-world workloads, and are carried out for a range of mobile devices manufactured between 2008 and 2015. All of the evaluated protocols have in common that they support named user groups, that is, user groups that are managed by a dedicated group owner, and that exists independently from any permissions on resources. This feature is offered by many data sharing services that are in use today, for example, by the distributed file systems NFSv4 and CIFS.

As a starting point, the performance of an E2E-SDS protocol was evaluated that is similar to protocols that are in productive use today. The protocol is based on traditional cryptographic primitives, i.e., symmetric and asymmetric cryptography that produces ciphertexts dedicated to a single recipient only.

The performance evaluation of this protocol shows that with regard to the considered workloads, only minor performance penalties have to be expected for most of the authorization operations, as the computation time for these operations is mostly up to some seconds, even on older hardware. However, major performance penalties hit users that manage group memberships in large user groups, i.e., user groups with a few thousand or more members. In this case, the computation times rise to several minutes when removing a user from a group, and the group member removal operations require the group owner to upload several MB of cryptographic key material to the storage provider.

A possible solution to decrease the performance penalties of group membership management is offered by Group Key Management approaches, which were originally developed to facilitate an efficient key distribution for secure multicast in a public network. By extending the aforementioned E2E-SDS protocol with a suitable Group Key Management approach, the performance penalties for the management of group memberships in large user groups are decreased significantly, as a group member removal operation can be carried out within few seconds on the considered hardware.

Moreover, the introduced performance evaluation methods were applied to an E2E-SDS protocol based on Attribute-Based Encryption (ABE). On the one hand, ABE promises to enable resource-saving SDS systems in terms of ciphertexts that have to be generated and distributed. On the other hand, some ABE schemes were shown to exhibit a performance worse than traditional cryptography, at least in small and artificial sharing scenarios. In a first step, an ABE scheme was identified that facilitates E2E-SDS, which conflicts with the need for a central trusted entity that many ABE schemes bring along. In a second step, the attribute-based authorization model of a suitable ABE scheme was mapped to the group-based authorization model. The performance of different mapping variants was evaluated by means of the analytical method, as ABE is not supported by the simulative model without any enhancements. The performance evaluation showed that the ABE-based E2E-SDS protocol exhibits a slightly worse performance than the traditional E2E-SDS protocol. However, this might change in future if more efficient implementations of pairing operations are made available. Nonetheless, the ABE-based E2E-SDS protocol only requires a few seconds of computation time for any authorization operation on the considered hardware, except for the group member removal operation.

Finally, a novel E2E-SDS protocol was presented that omits joint authorization operations. These operations are carried out by at least two different users in a collaborative manner, which requires that the involved users are reachable in the network and able to allocate the necessary computational resources immediately upon request. This might be a problem especially when mobile devices are involved, as these devices may be temporarily located in areas with bad network coverage, or it may be undesirable for the device owner that computing-intensive cryptographic operations are carried out immediately upon request, maybe because the device becomes less responsive. The E2E-SDS protocols mentioned so far need joint authorization operations for group member removal operations. This massively hinders an employment of hierarchies of named user groups in practice.

The proposed E2E-SDS protocol overcomes these drawbacks. However, omitting joint authorization operations implies that parts of the access control state must be altered by multiple writers. Unfortunately, it was shown that freshness guarantees are fundamentally hard to achieve in E2E-SDS with multiple writers. The best notion of freshness that can be achieved in this case is denoted as fork consistency, and existing mechanisms that ensure fork consistency impose heavy performance penalties. Furthermore, these mechanisms require each user to hold state information locally on the client device. For this reason, the proposed protocol only realizes a more relaxed notion of freshness. This might pose a vulnerability with respect to rollback attacks on the resources, which is only relevant when local caching of the shared data is avoided. However, if the attacker additionally manages to collaborate with a user that had write access to a resource in the past, the write access to this resource can be restored by a rollback attack without being detected.

The performance evaluation of the proposed E2E-SDS protocol was carried out with the simulative method, as the performance of group member removals is determined by multiple workload parameters that cannot be assumed to be independent. The evaluation shows that in relative terms, the computation times are significantly higher than those of the traditional E2E-SDS protocols that use joint authorization operations. However, in absolute terms, the computation times are mostly in the range of only a few seconds on the considered hardware. For example, although adding a resource is expensive compared to most of the authorization operations, it only takes between 6 and 8 seconds on average. Again, the most expensive authorization operations are group member removals, especially when group hierarchies are involved. These operations require computation times that are a magnitude larger than for other authorization operations, e.g., 38 s on average using a laptop from 2011. A sensitivity analysis regarding the performance of group member removals shows that the performance highly depends on the workload parameter *group memberships per group*, which has a superlinear influence on the performance.

Conclusions and Outlook

To conclude, the performance evaluations carried out in this thesis indicate that E2E-SDS with joint authorization operations causes only moderate performance penalties on current consumer devices. However, joint authorization operations constitute a major obstacle for the adoption of E2E-SDS, especially in conjunction with mobile user devices. Therefore, the practical applicability of E2E-SDS with joint authorization operations is limited to special scenarios. For example, joint authorization operations can be used in scenarios where dedicated cryptographic devices are available that are always ready to carry out cryptographic operations with a short or no delay. This might apply to a network of sensors and actors. The performance evaluation methods introduced in this thesis could easily be applied to the case of data sharing in sensor networks. However, additional work had to be carried out for identifying realistic workloads, and benchmarking the hardware involved.

In this thesis, a novel E2E-SDS protocol was presented that omits joint authorization operations, which opens up an attack vector for a comparatively powerful attacker. While omitting joint authorization operations leads to performance penalties, these penalties are moderate in absolute terms for most authorization operations. However, major performance impacts have to be expected when the sharing scenario includes large hierarchies of named user groups. It can be assumed that large group hierarchies will be rarely met in personal sharing scenarios, but are rather limited to institutional sharing scenarios. However, this assumption could only be supported or refuted by tracing and characterizing more sharing scenarios. The performance penalties can be circumvented by using stationary devices for computing-intensive authorization operations. While this might be rather accepted in institutional sharing scenarios than in personal ones, it limits the usability of the system in any case.

The security of the novel E2E-SDS protocol could be improved by incorporating mechanisms that guarantee fork consistency. However, these mechanisms further lower the performance of the system. Furthermore, the mechanisms require the user devices to hold a local state, thus, switching the device requires to synchronize the local state. As a result, strengthening the security of the proposed protocol limits its usability.

These observations lead to the conclusion that unless new cryptographic techniques are designed, a further hardening of E2E-SDS requires a considerable price to be paid in terms of performance or usability. This raises the question whether it is sensible to further harden data sharing against attacks from or on the storage provider at all. This question gains further importance by the observation that the user device itself offers a potentially big attack surface, which is likely to be targeted by an economically aware attacker as supposed weakest link in the “security chain”.

As first implication, this observation stresses the importance of client device security as research focus. This includes hardening the hardware and operation systems against attacks, verifying that third-party apps behave benevolent, and offer convenient means for the user to grant permissions to apps or other users. A strongly secured client device is also important for the application of usage control, i.e., to control the usage of media after it was downloaded on the client device. The coupling or unification of key management mechanisms that are required for both usage control and E2E-SDS constitutes a possible starting point for future work.

As second implication, from an economic perspective, it might be reasonable to even increase the trust that has to be put in the storage provider in order to improve performance or usability. For example, lots of recent SDS protocols increase the client-side performance by outsourcing cryptographic tasks to the storage provider. As another example, both shared data and cryptographic keys can be fragmented, and distributed on multiple storage providers that are trusted not to collaborate. Lots of similar mechanisms were proposed in recent years. It would be desirable to be provided with means to combine these mechanisms, and to be able to make statements about the performance, security or robustness of the resulting system. The key-graph based method proposed in this thesis might be adaptable to support this task.

Bibliography

[AB00] Réka Albert and Albert-László Barabási. Topology of Evolving Networks: Local Events and Universality. *Physical Review Letters*, 85(24):5234–5237, 2000.

[ACdMT05] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable Signatures. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS '05)*, pages 159–177. Springer Berlin Heidelberg, 2005.

[ADR02] Jee Jea An, Yevgeniy Dodis, and Tal Rabin. On the Security of Joint Signature and Encryption. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '02)*, pages 83–107. Springer Berlin Heidelberg, 2002.

[AES01] FIPS PUB 197: Announcing the Advanced Encryption Standard (AES). Technical report, National Institute of Standards and Technology (NIST), 2001.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[AHL$^+$12] Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie de Panafieu, and Carla Ràfols. Attribute-Based Encryption Schemes with Constant-Size Ciphertexts. *Theoretical Computer Science*, 422:15–38, 2012.

[All14] Security Guidance for Critical Areas of Focus in Cloud Computing V3.0. Technical report, Cloud Security Alliance, 2014.

[And00] Darrell Anderson. Fstress : A Flexible Network File Service Benchmark. *Benchmarking*, 2000.

[ANS05] ANSI X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical report, American National Standards Institute (ANSI), 2005.

[AS99] Gail-Joon Ahn and Ravi Sandhu. The RSL99 Language for Role-Based Separation of Duty Constraints. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control (RBAC '99)*, pages 43–54, 1999.

[AS00] Gail-Joon Ahn and Ravi Sandhu. Role-Based Authorization Constraints Specification. *ACM Transactions on Information and System Security*, 3(4):207–226, 2000.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT' 98)*, pages 127–144, 1998.

[Ben06] Messaoud Benantar. *Access Control Systems: Security, Identity Management and Trust Models*. Springer US, 2006.

[BF01] Dan Boneh and Matthew Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of the 21st Annual International Cryptology Conference (CRYPTO '01)*, pages 213–229, 2001.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '09)*, pages 1–18, 2009.

[BKR07] Steffen Becker, Heiko Koziolek, and Ralf Reussner. Model-Based Performance Prediction with the Palladio Component Model. In *Proceedings of the Sixth International Workshop on Software and Performance (WOSP '07)*, pages 54–65, 2007.

[BKR09] Steffen Becker, Heiko Koziolek, and Ralf Reussner. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3–22, 2009.

[Bol80] Béla Bollobás. A Probabilistic Proof of an Asymptotic Formula for the Number of Labelled Regular Graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980.

[BP76] David Elliott Bell and Leonard J. Padula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical report, Deputy for Command and Management Systems, United States Air Force, 1976.

[BRV13] Anne Benoit, Yves Robert, and Frédéric Vivien. *A Guide to Algorithm Design: Paradigms, Methods, and Complexity Analysis*. CRC Press, first edition, 2013.

[BSI11] Sicherheitsempfehlungen für Cloud Computing Anbieter – Mindestanforderungen in der Informationssicherheit. Technical report, Bundesamt für Sicherheit in der Informationstechnik (BSI), 2011.

[BSI15] BSI TR-03110: Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token - Part 1 - eMRTDs with BAC/-PACEv2 and EACv1. Technical report, Bundesamt für Sicherheit in der Informationstechnik (BSI), 2015.

[BSI16] BSI TR-02102-1: Kryptographische Verfahren: Empfehlungen und Schlussellängen. Technical report, Bundesamt für Sicherheit in der Informationstechnik (BSI), 2016.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. *Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, 2007.

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *IEEE International Conference on Cluster Computing*, pages 136–145, 2001.

[CC89] Guang-Huei Chiou and Wen-Tsuen Chen. Secure Broadcasting Using the Secure Lock. *IEEE Transactions on Software Engineering*, 15(8):929–934, 1989.

[CC09] Melissa Chase and Sherman S.M. Chow. Improving Privacy and Security in Multi-Authority Attribute-Based Encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, page 121, 2009.

[Cha07] Melissa Chase. Multi-authority Attribute Based Encryption. In *Proceedings of the 4th Conference on Theory of Cryptography (TCC '07)*, pages 515–534, 2007.

[CKR+07] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC '07)*, pages 1–13, 2007.

[CLO91] Matthijs J. Coster, Brian A. LaMacchia, and Andrew M. Odlyzko. An Improved Low-Density Subset Sum Algorithm. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT' 91)*, pages 54–67, 1991.

[Cro02] David Cross. Encrypting File System in Windows XP and Windows Server 2003. Technical report, Microsoft Corporation, 2002.

[CS11] Sanjit Chatterjee and Palash Sarkar. *Identity-Based Encryption*. Springer US, 2011.

[CSS07] Christian Cachin, Alexander Shraer, and Abhi Shelat. Efficient Fork-Linearizable Access to Untrusted Shared Memory. *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing (PODC '07)*, pages 129–138, 2007.

[CSS+08] Elizabeth Chew, Marianne Swanson, Kevin Stine, Nadya Bartol, Anthony Brown, and Will Robinson. Performance Measurement Guide for Information Security. *NIST Special Publication*, 800(55), 2008.

[CZF04] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A Recursive Model for Graph Mining. In *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM '04)*, pages 442–446, 2004.

[Dan73] Cuthbert Daniel. One-at-a-Time Plans. *Journal of the American Statistical Association*, 68(342):353, 1973.

[Dav01] Don Davis. Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML. In *Proceedings of the General Track: 2001 USENIX Annual Technical Conference*, pages 65–78, 2001.

[DH08] Jochen Dinger and Hannes Hartenstein. *Netzwerk- und IT-Sicherheitsmanagement: eine Einführung*. Universitätsverlag Karlsruhe, 2008.

[DJ14] Nishant Doshi and Devesh Jinwala. Fully secure ciphertext policy attribute-based encryption with constant length ciphertext and faster decryption. *Security and Communication Networks*, 7(7):1988–2002, 2014.

[DSS13] FIPS PUB 186-4: Digital Signature Standard (DSS). Technical Report July, National Institute of Standards and Technology (NIST), 2013.

[Eck12] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg Verlag, München, 7th edition, 2012.

[Eec10] Lieven Eeckhout. *Computer Architecture Performance Evaluation Methods*. Morgan & Claypool Publishers, 2010.

[EWZC11] Vladimir Entin, Mathias Winder, Bo Zhang, and Stephan Christmann. Combining Model-Based and Capture-Replay Testing Techniques of Graphical User Interfaces: An Industrial Approach. In *Workshop Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST '12)*, pages 572–577, 2011.

[FBK92] David Ferraiolo, John Barkley, and Richard Kuhn. Role-Based Access Controls. *ACM Transactions on Information and System Security*, 2(1):34–64, 1992.

[Fei14] Dror G Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge University Press, 2014.

[FKK06] Kevin Fu, Seny Kamara, and Tadayoshi Kohno. Key Regression : Enabling Efficient Key Distribution for Secure Distributed Storage. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS '06)*, 2006.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is Secure under the RSA Assumption. *Journal of Cryptology*, 17(2):81–104, 2001.

[FSG$^+$01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[Fu99] Kevin Fu. *Group sharing and random access in cryptographic storage file systems*. Master thesis, Massachusetts Institute of Technology, 1999.

[GGCS02] Vipul Gupta, Sumit Gupta, Sheueling Chang, and Douglas Stebila. Performance Analysis of Elliptic Curve Cryptography for SSL. In *Proceedings of the ACM Workshop on Wireless Security (WiSE '02)*, pages 87–94, 2002.

[GLHW12] Sebastian Graf, Patrick Lang, Stefan A. Hohenadel, and Marcel Waldvogel. Versatile Key Management for Secure Cloud Storage. In *Proceedings of the 31st IEEE Symposium on Reliable Distributed Systems (SRDS '12)*, pages 469–474, 2012.

[GMSW06] Dominik Grolimund, Luzius Meisser, Stefan Schmid, and Roger Wattenhofer. Cryptree: A Folder Tree Structure for Cryptographic File Systems. In *Proceedings of the 25th IEEE Symposium on Reliable Distributed Systems (SRDS '06)*, pages 189–198, 2006.

[GQL14] William C. Garrison, Yechen Qiao, and Adam J. Lee. On the Suitability of Dissemination-centric Access Control Systems for Group-centric Sharing. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy (CODASPY '14)*, pages 1–12, 2014.

[GSMB03] EJ Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. SiRiUS: Securing Remote Untrusted Storage. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS '03)*, 2003.

[HBB07] H. Ragab Hassen, A. Bouabdallah, and H. Bettahar. A New and Efficient Key Management Scheme for Content Access Control within Tree Hierarchies. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07)*, pages 551–556, 2007.

[HCL13] Jyun-Yao Huang, Chen-Kang Chiang, and I-En Liao. An Efficient Attribute-Based Encryption and Access Control Scheme for Cloud Storage Environment. In *Proceedings of the Eighth International Conference on Grid and Pervasive Computing (GPC '13)*, pages 77–90, 2013.

[HF01] James Hughes and Christopher Feist. Architecture of the Secure File System. In *Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Technologies (MSS '01)*, pages 277–277, 2001.

[HFK+14] Vincent C Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. *NIST Special Publication*, 800:162, 2014.

[HH95] John R Heath and Stephen A R Houser. Analysis of Disk Workloads in Network File Server Environments. In *Proceedings of the 21st International Computer Measurement Group Conference (CMG '95)*, pages 313–322, 1995.

[HH96] John R Heath and Stephen A R Houser. On the Relationship of Server Disk Workloads and Client File Requests. In *Proceedings of the 22nd International Computer Measurement Group Conference (CMG '96)*, pages 205–213, 1996.

[HHH07] Panu Hämäläinen, Marko Hännikäinen, and Timo D. Hämäläinen. Review of Hardware Architectures for Advanced Encryption Standard Implementations Considering Wireless Sensor Networks. In *Proceedings of the 7th International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS '07)*, pages 443–453, 2007.

[HKDH10] Thorsten Höllrigl, Holger Kühner, Jochen Dinger, and Hannes Hartenstein. User-Controlled Automated Identity Delegation. In *Proceedings of the 6th IEEE/IFIP International Conference on Network and Service Management (CNSM '10)*, 2010.

[HKDH11] Thorsten Höllrigl, Holger Kühner, Jochen Dinger, and Hannes Hartenstein. Extension for Information Card Systems to Achieve User-Controlled Automated Identity Delegation. In *Proceedings of the First IEEE/IFIP Workshop on Managing Federations and Cooperative Management (ManFed.CoM)*, 2011.

[HN11] Junbeom Hur and Dong Kun Noh. Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221, 2011.

[HN15] Tom Haynes and David Noveck. RFC 7530: Network File System (NFS) Version 4 Protocol, 2015.

[HXL15] Jianan Hong, Kaiping Xue, and Wei Li. Comments on "DAC-MACS : Effective Data Access Control for Multiauthority Cloud Storage Systems- Security Analysis of Attribute Revocation in Multiauthority Data Access Control for Cloud Storage Systems. *IEEE Transactions on Information Forensics and Security*, 10(6):1315–1317, 2015.

[INC04] INCITS 359-2004 Information Technology – Role Based Access Control. Technical report, InterNational Committee for Information Technology Standards (INCITS), 2004.

[IPN$^+$09] Luan Ibraimi, Milan Petkovic, Svetla Nikova, Pieter Hartel, and Willem Jonker. Mediated Ciphertext-Policy Attribute-Based Encryption and Its Application. In *Information Security Applications*, pages 309–323. Springer-Verlag, 2009.

[ISO11a] ISO/IEC 24760-1:2011 Information technology – Security techniques – A framework for identity – Part 1: Terminology and concepts. Technical report, International Organization for Standardization (ISO), 2011.

[ISO11b] ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Technical report, International Organization for Standardization (ISO), 2011.

[ISO11c] ISO/IEC. ISO/IEC 27005: Information technology – Security techniques – Information security risk management. Technical report, International Organization for Standardization (ISO), 2011.

[ISO14] ISO/IEC 27000: Information technology – Security techniques – Information security management systems – Overview and vocabulary. Technical report, International Organization for Standardization (ISO), 2014.

[JG11] Wayne Jansen and Timothy Grance. Guidelines on Security and Privacy in Public Cloud Computing. *NIST Special Publication*, 800(144), 2011.

[JMSW02] Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic Signature Schemes. In *The Cryptographers' Track at the RSA Conference 2002 (CT-RSAv'02)*, pages 244–262, 2002.

[JN09] Marc Joye and Gregory Neven. *Identity-Based Cryptography*. IOS Press, 2009.

[Jou00] Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. In *Proceedings of the International Algorithmic Number Theory Symposium (ANTS 2000)*, pages 385–393, 2000.

[KBC97] H Krawczyk, M Bellare, and Ran Canetti. RFC 2104: HMAC: Keyed-Hashing for Message Authentication, 1997.

[KCW10] D. Richard Kuhn, Edward J. Coyne, and Timothy R. Weil. Adding Attributes to Role Based Access Control. *IEEE Computer*, 43(6):79–81, 2010.

[Kel76] Frank P. Kelly. Networks of Queues. *Advances in Applied Probability*, 8(2):416–432, jun 1976.

[KH13] Holger Kühner and Hannes Hartenstein. Schlüsselverwaltung für sichere Gruppeninteraktionen über beliebigen Speicheranbietern : ein Überblick. In *6. DFN Forum Kommunikationstechnologien*, pages 119–129, 2013.

[KH14] Holger Kuehner and Hannes Hartenstein. Spoilt for Choice: Graph-based Assessment of Key Management Protocols to Share Encrypted Data (poster paper). In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy (CODASPY '14)*, pages 147–150, 2014.

[KH15] Holger Kuehner and Hannes Hartenstein. On the Resource Consumption of Secure Data Sharing. In *Proceedings of the 2015 IEEE International Symposium on Recent Advances of Trust, Security and Privacy in Computing and Communications (RATSP '15)*, pages 880–889, 2015.

[KH16] Holger Kuehner and Hannes Hartenstein. Decentralized Secure Data Sharing with Attribute-Based Encryption: A Resource Consumption Analysis. In *4th ACM International Workshop on Security in Cloud Computing (SCC '16)*, pages 74–81, 2016.

[KHH11] Holger Kühner, Thorsten Höllrigl, and Hannes Hartenstein. Benutzerzentrierung und Föderation: Wie kann ein Benutzer die Kontrolle über seine Identitätsdaten bewahren? In *4. DFN-Forum Kommunikationstechnologien*, 2011.

[KKN+15] Kevin Koerner, Holger Kuehner, Julia Neudecker, Hannes Hartenstein, and Thomas Walter. Bewertungskriterien und ihre Anwendung zur Evaluation und Entwicklung sicherer Sync&Share-Dienste. In *8. DFN-Forum Kommunikationstechnologien*, pages 71–82, 2015.

[KM15] Neal Koblitz and Alfred J. Menezes. A Riddle Wrapped in an Enigma. In *Cryptology ePrint Archive, Report 2005/1018*, 2015.

[KNSW11] Ram Krishnan, Jianwei Niu, Ravi Sandhu, and William H. Winsborough. Group-Centric Secure Information-Sharing Models for Isolated Groups. *ACM Transactions on Information and System Security*, 14(3):1–29, nov 2011.

[Köh15] Jens Köhler. *Tunable security for deployable data outsourcing*. Phd thesis, Karlsruhe Institute of Technology (KIT), 2015.

[KRS+03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, 2003.

[KS09] Ram Krishnan and Ravi Sandhu. Enforcement Architecture and Implementation Model for Group-centric Information Sharing. In *Proceedings of the 1st International Workshop on Security and Communication Networks (IWSCN)*, pages 1–8, 2009.

[KS11] Ram Krishnan and Ravi Sandhu. Authorization Policy Specification and Enforcement for Group-Centric Secure Information Sharing. In *Proceedings of the 7th International Conference on Information Systems Security (ICISS '11)*, pages 102–115, 2011.

[KSNW09] Ram Krishnan, Ravi Sandhu, Jianwei Niu, and William H. Winsborough. Formal Models for Group-Centric Secure Information Sharing. Technical report, Department of Computer Science, The University of Texas, 2009.

[KW14] Kevin Körner and Thomas Walter. Sichere und benutzerfreundliche Schlüsselverteilung auf Basis von QR-Codes. In *4. Jahrestagung der Gesellschaft für Informatik (Informatik '14)*, pages 223–234, 2014.

[KWM15] Kevin Koerner, Thomas Walter, and Michael Menth. Data Freshness for Non-Trusted Environments Using Disposable Certificates. In *Proceedings of the 3rd International Workshop on Security in Cloud Computing (SCC '15)*, pages 73–80, 2015.

[KX15] Konstantinos Koiliaris and Chao Xu. A Faster Pseudopolynomial Time Algorithm for Subset Sum. *Cryptography ePrint Archive*, pages 1–16, 2015.

[Law06] Averill M. Law. *Simulation Modeling and Analysis*. McGraw-Hill Publishing Company, 4th edition, 2006.

[LCLS08] Huang Lin, Zhenfu Cao, Xiaohui Liang, and Jun Shao. Secure Threshold Multi Authority Attribute Based Encryption without a Central Authority. In *Proceedings of the 9th International Conference on Cryptology in India (INDOCRYPT '08)*, pages 426–436, 2008.

[LDGW13] Junzuo Lai, Robert Deng, Chaowen Guan, and Jian Weng. Attribute-Based Encryption With Verifiable Outsourced Decryption. *IEEE Transactions on Information Forensics and Security*, 8(8):1343–1354, 2013.

[Le 15] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*. EPFL Press, 2015.

[Lei16] Marc Leinweber. *Generierung konsistenter Workloads für das sichere Teilen von Daten*. Bachelor's thesis, Karlsruhe Institute of Technology (KIT), 2016.

[LKMS03] Jinyuan Li, Maxwell Krohn, David Mazières, and Dennis Shasha. Secure Untrusted Data Repository (SUNDR). In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI '04)*, pages 121–136, 2003.

[LLLS11] Xiaohui Liang, Rongxing Lu, Xiaodong Lin, and Xuemin Sherman Shen. Ciphertext Policy Attribute Based Encryption with Efficient Revocation. Technical report, University of Waterloo, 2011.

[LLYY14] Dongxiao Liu, Hongwei Li, Yi Yang, and Haomiao Yang. Achieving Multi-Authority Access Control with Efficient Attribute Revocation in Smart Grid. In *Proceedings of the 2014 IEEE International Conference on Communications (ICC '14)*, pages 634–639, 2014.

[LO10] Ross J. Loomis and Alan C. O'Connor. Economic Analysis of Role-Based Access Control Final Report. Technical report, National Institute of Standards and Technology (NIST), 2010.

[LPGM08] Andrew Leung, Shankar Pasupathy, Garth Goodson, and Ethan L. Miller. Measurement and Analysis of Large-Scale Network File System Workloads. In *Proceedings of the 2008 USENIX Annual Technical Conference*, pages 213–226, 2008.

[LTH11] Sebastian Labitzke, Irina Taranu, and Hannes Hartenstein. What Your Friends Tell Others About You: Low Cost Linkability of Social Network Profiles. In *Proceedings to the 5th International ACM Workshop on Social Network Mining and Analysis*, 2011.

[LvFS14] Gunn K.H. Larsen, Nicky D. van Foreest, and Jacquelien M.A. Scherpen. Power supply-demand balance in a Smart Grid: An information sharing model for a market mechanism. *Applied Mathematical Modelling*, 38(13):3350–3360, 2014.

[LW11] Allison Lewko and Brent Waters. Decentralizing Attribute-Based Encryption. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT '11)*, pages 568–588, 2011.

[LXZZ13] Qinyi Li, Hu Xiong, Fengli Zhang, and Shengke Zeng. An Expressive Decentralizing KP-ABE Scheme with Constant-Size Ciphertext. *International Journal of Network Security*, 15(3):131–140, 2013.

[LY14] Fang Liang and Guo Yunchuan. A Survey of Role Mining Methods in Role-Based Access Control System. In *Web Technologies and Applications - APWeb 2014 Workshops, SNA, NIS, and IoTS, Proceedings*, pages 291–300, 2014.

[LYZ+13] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):131–143, 2013.

[MA11] Mohamed Meky and Amjad Ali. A novel and secure data sharing model with full owner control in the cloud environment. *International Journal of Computer Science and Information Security*, 9(6), 2011.

[Mer80] Ralph C. Merkle. Protocols for Public Key Cryptosystems. In *1980 IEEE Symposium on Security and Privacy*, pages 122–122, 1980.

[Mic11a] Microsoft. MS-CIFS: Common Internet File System (CIFS) Protocol Specification. Technical report, Microsoft Corporation, 2011.

[Mic11b] Microsoft. MS-SMB: Server Message Block (SMB) Protocol Specification. Technical report, Microsoft Corporation, 2011.

[Mic11c] Microsoft. MS-SMB2: Server Message Block (SMB) Version 2 Protocol Specification. Technical report, Microsoft Corporation, 2011.

[Mic14] Microsoft. MS-DTYP: Windows Data Types. Technical report, Microsoft Corporation, 2014.

[MKE08] Sascha Müller, Stefan Katzenbeisser, and Claudia Eckert. Distributed Attribute-Based Encryption. *Proceedings of the 11th International Conference on Information Security and Cryptology (ICISC '08)*, 5461:20–36, 2008.

[MKI+04] Ron Milo, Nadav Kashtan, Shalev Itzkovitz, Mark E. J. Newman, and Uri Alon. Uniform generation of random graphs with prescribed degree sequences. *Arxiv preprint*, cond-mat/0:1–4, 2004.

[MLFR01] Ethan Miller, Darrell Long, William Freeman, and Benjamin Reed. Strong Security for Distributed File Systems. In *Proceedings of the 30th IEEE International Performance Computing and Communications Conference (IPCCC '11)*, pages 34–40, 2001.

[MOV93] Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MS02] David Mazières and Dennis Shasha. Building secure file systems out of Byzantine storage. In *Proceedings of the Twenty-First ACM Symposium on Principles of Distributed Computing (PODC '02)*, pages 108–117, 2002.

[Mur89] Tadao Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[NYHR93] Clifford Neuman, Tom Yu, Sam Hartman, and Kenneth Raeburn. RFC 1510: The Kerberos Network Authentication Service (V5), 1993.

[OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-Based Encryption with Non-Monotonic Access Structures. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS '07)*, pages 195–203, 2007.

[PHB06] Alexander Pretschner, Manuel Hilty, and David Basin. Distributed Usage Control. *Communications of the ACM*, 49(9):39–44, 2006.

[PHS+08] Alexander Pretschner, Manuel Hilty, Florian Schütz, Christian Schaefer, and Thomas Walter. Usage Control Enforcement: Present and Future. *IEEE Security & Privacy Magazine*, 6(4):44–53, 2008.

[PM01] Helen Popper and John D. Montgomery. Information sharing and central bank intervention in the foreign exchange market. *Journal of International Economics*, 55(2):295–316, 2001.

[Pre09] Alexander Pretschner. An Overview of Distributed Usage Control. In *Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques (KEPT '09)*, pages 25–33, 2009.

[PS02] Jaehong Park and Ravi Sandhu. Towards Usage Control Models: Beyond Traditional Access Control. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies (SACMAT '02)*, page 57, 2002.

[PS04] Jaehong Park and Ravi Sandhu. The UCON ABC Usage Control Model. *ACM Transactions on Information and System Security*, 7(1):128–174, 2004.

[PSST11] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the Joint Security of Encryption and Signature, Revisited. In *Proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '11)*, pages 161–178, 2011.

[QDLM15] Baodong Qin, Robert Deng, Shengli Liu, and Siqi Ma. Attribute-Based Encryption With Efficient Verifiable Outsourced Decryption. *IEEE Transactions on Information Forensics and Security*, 10(7):1384–1393, 2015.

[RNS11] Sushmita Ruj, Amiya Nayak, and Ivan Stojmenovic. DACC: Distributed Access Control in Clouds. *Proceedings of the 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '11)*, pages 91–98, 2011.

[RSA78] R L Rivest, A Shamir, and L Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[San96] Ravi Sandhu. Roles Versus Groups. In *Proceedings of the First ACM Workshop on Role-Based Access Control (RBAC '95)*, pages 25–26, 1996.

[San00] Ravi Sandhu. Engineering Authority and Trust in Cyberspace: The OM-AM and RBAC Way. In *Proceedings of the Fifth ACM Workshop on Role-Based Access Control*, pages 111–119, 2000.

[SBM99] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 2(1):105–135, 1999.

[SCC$^+$10] Alexander Shraer, Christian Cachin, Asaf Cidon, Idit Keidar, Yan Michalevsky, and Dani Shaket. Venus: Verification for Untrusted Cloud Storage. In *Proceedings of the 2nd ACM Cloud Computing Security Workshop (CCSW '10)*, page 19, 2010.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[Sch03] Günter Schäfer. *Security in Fixed and Wireless Networks: An Introduction to Securing Data Communications.* John Wiley & Sons, Ltd, 2003.

[Sch15] Steffen Schreiner. *A Security Architecture for e-Science Grid Computing.* PhD thesis, Technische Universität Darmstadt, 2015.

[SF03] Kavé Salamatian and Serge Fdida. A framework for interpreting measurement over Internet. In *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools '03)*, page 87, 2003.

[SFK00] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The NIST Model for Role-Based Access Control: Towards a Unified Standard. In *Proceedings of the Fifth ACM Workshop on Role-Based Access Control*, pages 47–63, 2000.

[SH10] Gabor Szabo and Bernardo A. Huberman. Predicting the Popularity of Online Content. *Communications of the ACM*, 53(8):80–88, 2010.

[Shi00] Rob Shirey. RFC 2828: Internet Security Glossary, 2000.

[Sim05] Erik Simmons. The Usage Model: A Structure for Richly Describing Product Usage during Design and Development. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE' 05)*, pages 403–407, 2005.

[SKNW10] Ravi Sandhu, Ram Krishnan, Jianwei Niu, and William H. Winsborough. Group-Centric Models for Secure and Agile Information Sharing. In *Proceedings of the 5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS '10)*, pages 55–69, 2010.

[SNPB10] Ning Shang, Mohamed Nabeel, Federica Paci, and Elisa Bertino. A Privacy-Preserving Approach to Policy-Based Content Dissemination. In *Proceedings of the IEEE 26th International Conference on Data Engineering (ICDE '10)*, pages 944–955, 2010.

[SPE14] SPEC Solution File Server (SFS) 2014 Benchmark. Technical report, Standard Performance Evaluation Corporation (SPEC), 2014.

[STCR04] Andrea Saltelli, Stefano Tarantola, Francesca Campolongo, and Marco Ratto. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. John Wiley & Sons, Ltd, 2004.

[SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '05)*, pages 457–473, 2005.

[TCN+14] Danan Thilakanathan, Shiping Chen, Surya Nepal, Rafael Calvo, and Leila Alem. A platform for secure monitoring and sharing of generic health data in the Cloud. *Future Generation Computer Systems*, 35:102–113, 2014.

[TCNC14] Danan Thilakanathan, Shiping Chen, Surya Nepal, and Rafael A. Calvo. Secure Data Sharing in the Cloud. In *Security, Privacy and Trust in Cloud Systems*, pages 45–72. Springer Berlin Heidelberg, 2014.

[TCS85] Trusted Computer System Evaluation Criteria. Technical report, U.S. Department of Defense, 1985.

[vTJ11] Henk C. A. van Tilborg and Sushil Jajodia, editors. *Encyclopedia of Cryptography and Security*. Springer New York Dordrecht Heidelberg London, 2011.

[WGL98] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure Group Communications Using Key Graphs. *ACM SIGCOMM Computer Communication Review*, 28(4):68–79, 1998.

[WHA99] D Wallner, Eric Harder, and R Agee. RFC 2627: Key Management for Multicast: Issues and Architectures, 1999.

[Wil89] Robert Charles Williamson. *Probabilistic Arithmetic*. PhD thesis, University of Queensland, 1989.

[Win05] Phillip J. Windley. *Digital Identity*. O'Reilly, 2005.

[WLW11] Guojun Wang, Qin Liu, and Jie Wu. Achieving fine-grained access control for secure data sharing on cloud servers. *Concurrency and Computation: Practice and Experience*, 23(12):1443–1464, 2011.

[WZSI14] Xinlei Wang, Jianqing Zhang, Eve M. Schooler, and Mihaela Ion. Performance Evaluation of Attribute-Based Encryption: Toward Data Privacy in the IoT. In *Proceedings of the IEEE International Conference on Communications (ICC '14)*, pages 725–730, 2014.

[XM12] Zhiqian Xu and Keith M. Martin. Dynamic User Revocation and Key Refreshing for Attribute-Based Encryption in Cloud Storage. In *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '12)*, pages 844–849, 2012.

[YJ14] Kan Yang and Xiaohua Jia. Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1735–1744, 2014.

[YJR+13] Kan Yang, Xiaohua Jia, Kui Ren, Bo Zhang, and Ruitao Xie. DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems. *IEEE Transactions on Information Forensics and Security*, 8(11):1790–1801, 2013.

[YJR+14] Kan Yang, Xiaohua Jia, Kui Ren, Ruitao Xie, and Liusheng Huang. Enabling Efficient Access Control with Dynamic Policy Updating for Big Data in the Cloud. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '14)*, pages 2013–2021, 2014.

[YT05] Eric Yuan and Jin Tong. Attributed Based Access Control (ABAC) for Web Services. In *Proceedings of the 2005 IEEE International Conference on Web Services (ICWS '05)*, pages 561–569, 2005.

[YWRL10a] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '10)*, pages 1–9, 2010.

[YWRL10b] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Attribute Based Data Sharing with Attribute Revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10)*, page 261, 2010.

[ZDB08] Xukai Zou, Yuan-Shun Dai, and Elisa Bertino. A Practical and Flexible Key Management Mechanism For Trusted Collaborative Computing. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '08)*, pages 538–546, 2008.

[ZHHW14] Yan Zhu, Dijiang Huang, Changjyun Hu, and Xin Wang. From RBAC to ABAC: Constructing Flexible Data Access Control for Cloud Storage Services. *IEEE Transactions on Services Computing*, 85287(c):1–1, 2014.

[ZMHH13] Yan Zhu, Di Ma, Chang-Jun Hu, and Dijang Huang. How to Use Attribute-Based Encryption to Implement Role-Based Access Control in the Cloud. In *Proceedings of the 2013 International Workshop on Security in Cloud Computing (SCC '13)*, pages 33–40, 2013.

[ZSNL03] Fangguo Zhang, Reihaneh Safavi-Naini, and Chih-Yin Lin. New Proxy Signature, Proxy Blind Signature and Proxy Ring Signature Schemes from Bilinear Pairing. *Cryptography ePrint Archive*, pages 1–11, 2003.

[ZVH13] Lan Zhou, Vijay Varadharajan, and Michael Hitchens. Achieving Secure Role-Based Access Control on Encrypted Data in Cloud Storage. *IEEE Transactions on Information Forensics and Security*, 8(12):1947–1960, 2013.

[ZZC+15] Yinghui Zhang, Dong Zheng, Xiaofeng Chen, Jin Li, and Hui Li. Efficient attribute-based data sharing in mobile clouds. *Pervasive and Mobile Computing*, 2015.