A generic auditing framework for compliance verification of Internet Service Level Agreements

Author(s):
Hasan

Publication Date:
2006

Permanent Link:
https://doi.org/10.3929/ethz-a-005369625

Rights / License:
In Copyright - Non-Commercial Use Permitted
A GENERIC AUDITING FRAMEWORK FOR COMPLIANCE VERIFICATION OF INTERNET SERVICE LEVEL AGREEMENTS

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZÜRICH

for the degree of
Doctor of Sciences

presented by

HASAN

Dipl. Inf.-Ing. ETH
born March 29, 1964
citizen of Indonesia

accepted on the recommendation of

Prof. Dr. Bernhard Plattner, examiner
Prof. Dr. Burkhard Stiller, co-examiner and supervisor
Prof. Dr. Georg Carle, co-examiner

2006
Abstract

Since the early 2000’s, formal specifications of Service Level Agreements (SLA) and automated auditing of their compliances have been a research interest, motivated amongst others by the popularity of web services. In fact, specifying SLAs on IP-based networks becomes viable through network device instrumentations for Quality-of-Service (QoS) measurements, not only of transport but also of application services. However, application service SLAs pose challenges to their compliance auditing, due to the variety and the potential complexity of Service Level Objectives (SLO) that customers demand for the price they pay. An SLO defines a performance parameter and its committed value for a service contracted in an SLA. Furthermore, security support and the support of mobility in a multi-domain environment add complexity to SLO specifications.

Current approaches in SLA management and monitoring address the formal specification of a complete SLA in a specific application area, or concentrate on efficient measurement techniques for certain QoS parameters. Thereby, only certain types of SLO can be expressed or processed, and security in inter-domain auditing interactions is not considered.

Therefore, the generic auditing framework developed and termed AURIC (Auditing Framework for Internet Services) provides all necessary core functionality to conduct audit tasks which may require reliable and secure inter-domain interactions of its distributed components as well as complex operations to decide on compliances with an SLO. The high reusability of this framework’s functionality is achieved through the functional decomposition
of an audit task into a sequence of subtasks to allow for a modular specification, and through the separation of common audit functionality from SLO-specific auditing logic, as well as a formal language — termed Sapta — to specify audit tasks in full detail.

An SLA does not only oblige the provider to deliver services with the committed quality level, but also obliges the customer to pay for the services consumed. In order to justify billing and to protect a provider against a user’s false denial of having consumed a service, this thesis develops an architecture termed **NorCIS** (Non-repudiation of the Consumption of Internet Services) for providing non-repudiation of service consumption, and designs the required protocols. The architecture is designed to be used in a Mobile IP-based environment with the support of Authentication, Authorization, and Accounting (AAA) infrastructure.

Protocols for non-repudiation of service consumption aim at transferring irrefutable evidence of service consumption to interested parties. Thereby, neither proof before consumption nor proof after consumption is fair for service providers and users, since one of the parties gains advantage over the other. Hence, this thesis designs a new non-repudiation protocol intended for a use in a time-based accounting scheme, which supports fairness in the above mentioned sense. For usage in item-based and volume-based accounting schemes this thesis adapts and improves two existing non-repudiation protocols.

Moreover, this thesis proposes the use of virtual identifiers in the authentication protocol and in the evidence of service consumption to support identity privacy of a mobile user. The evaluation of the protocols shows that collected evidences are able to resolve possible disputes. In addition, possible threats are analyzed to show the support of security.

A prototype of AURIC key components has been designed completely and implemented to show the high reusability of AURIC functionality and its support of complex SLOs. Also a NorCIS* non-repudiation entity for a mobile terminal has been designed in full and implemented prototypically to study its protocol interactions.
Kurzfassung


Existierende Ansätze im SLA-Management und -Überwachung behandeln die formale Spezifikation eines kompletten SLAs in einem bestimmten Anwendungsgebiet oder konzentrieren sich auf effiziente Messverfahren von gewissen QoS-Parametern. Dabei lassen sich nur bestimmte Typen von SLOs ausdrücken oder verarbeiten. Darüberhinaus werden Sicherheitsaspekte in Inter-Domain-Auditing-Interaktionen nicht berücksichtigt.

Das im Rahmen dieser Arbeit entwickelte generische Auditing-Frame
work AURIC (Auditing Framework for Internet Services) stellt deswegen Kernfunktionen zur Verfügung, welche für die Ausführung von Auditing-
Aufgaben notwendig sind, die sowohl zuverlässige und sichere Inter-Domain-Interaktionen ihrer verteilten Komponenten als auch komplexe Operationen für die Einhaltungsüberprüfung eines SLOs benötigen. Die hohe Wiederverwendbarkeit von Framework-Funktionen wird anhand folgender Mechanismen erreicht: Der funktionalen Zerlegung einer Auditing-Aufgabe in eine Reihe von Teilaufgaben, um eine modulare Spezifikation zu ermöglichen, der Trennung allgemeiner Funktionen von SLO-spezifischer Logik, und der Verwendung einer neuen Auditing-Spezifikationssprache.


Ferner wird die Verwendung von virtuellen Benutzerkennzeichen im Authentifizierungspankolll und in Dienstbenutzungsevidenzen vorgeschlagen, um die reale Identität eines Benutzers zu verbergen. Die Evaluation hat gezeigt, dass die Evidenzen mögliche Dispute lösen können, und dass die Protokollinteraktionen gegenüber verschiedenen Attacken sicher sind.

Mit einem Entwurf und einer prototypischen Implementierung der AU-RIC-Schlüsselkomponente wurden die hohe Wiederverwendbarkeit von AU-RIC-Funktionalität und die Unterstützung komplexer SLOs gezeigt. Weiterhin wurde eine NorCIS*-Entität für ein mobiles Terminal entwickelt und prototypisch implementiert, um ihre Protokollinteraktionen zu studieren.
# Table of Contents

Abstract iii  

Kurzfassung v  

Table of Contents vii  

List of Figures xiii  

List of Tables xv  

List of Abbreviations xvii  

1 Introduction 1  
1.1 Motivation 3  
1.2 Problem Statement 5  
1.3 Thesis Goal and Contribution 6  
1.4 Thesis Outline 7  

2 Definitions and Related Work 9  
2.1 Key QoS Parameters 9  
2.1.1 Traditional QoS Parameters 10  
2.1.2 Impacts of Mobility Support 11  
2.1.3 Impacts of Security Support 12  
2.2 Definition of Auditing 14  
2.3 Auditing Related Work 17  
2.3.1 IDS Modelling and Architecture 17  
2.3.2 WSLA Framework 18  
2.3.3 Cremona 19  
2.3.4 Monitoring Schemes for SLA Violation 20  
2.3.5 TAPAS SLAng 21  
2.3.6 Moby Dick’s SLA Compliance Auditing 22  
2.3.7 Daidalos’ SLA Compliance Auditing 23  
2.4 Non-repudiation Related Work 23  
2.5 Comparison and Conclusion of Related Work 26
### 3 Generic Model and Architecture

#### 3.1 AURIC’s Generic Model

- **3.1.1 Model Description**
- **3.1.2 General Auditing Process**
- **3.1.3 Mapping the Model to Various Auditing Areas**

#### 3.2 AURIC’s Generic Architecture

- **3.2.1 Design**
- **3.2.2 Support of Multi-Administrative Domain Environment**

#### 3.3 A Multi-Domain Application Scenario

- **3.3.1 Topology and Business Relation among Parties**
- **3.3.2 Services and SLOs**
- **3.3.3 Measurement**
- **3.3.4 Auditing**

#### 3.4 A Use Case: Network Throughput Commitment

- **3.4.1 Executing Unit**
- **3.4.2 Controlling Unit and Compliance Specifications**
- **3.4.3 Accounting Unit and Facts**
- **3.4.4 Auditing Unit: Automated Auditors**
- **3.4.5 Report Handling Unit and Audit Reports**
- **3.4.6 Policy Definition Unit and Policies**

#### 3.5 Chapter Summary

### 4 Audit Task Decomposition and Specification

#### 4.1 Audit Function

- **4.1.1 Compliance Specifications**
- **4.1.2 Facts and Audit Reports**

#### 4.2 Audit Subtasks

#### 4.3 Auditability and Subtask Functions

#### 4.4 Audit Specifications

- **4.4.1 Facts Filtering Function Specification**
- **4.4.2 Facts Grouping Function Specification**
- **4.4.3 Property Function Specification**
- **4.4.4 Property-Based Compliance Function Specification**
- **4.4.5 Compliance Calculation Specification**
- **4.4.6 Report’s Attribute Function Specification**
- **4.4.7 Report Composition Specification**
- **4.4.8 Examples**

#### 4.5 Required Language Constructs for Audit Specifications

#### 4.6 Sapta: an Audit Specification Language

- **4.6.1 Data Types and Constants**
- **4.6.2 Expressions**
4.6.3 Statements .............................................. 95
4.6.4 FilterFunction Specification ....................... 96
4.6.5 GroupingFunction Specification .................... 97
4.6.6 PropertyFunction Specification .................... 102
4.6.7 ComplianceFunction Specification .................. 103
4.6.8 ComplianceCalculation Specification ............... 103
4.6.9 AttributeFunction Specification .................... 105
4.6.10 ReportComposition Specification .................. 105
4.6.11 Pre-defined Functions .............................. 107
4.6.12 Examples ............................................ 109
4.7 Chapter Summary ........................................ 116

5 SLA Compliance Auditing ................................. 117
  5.1 Design .................................................. 118
      5.1.1 Metering and Accounting ......................... 119
      5.1.2 Auditing ........................................ 120
  5.2 Interfaces ............................................. 121
      5.2.1 AM Interface .................................... 122
      5.2.2 ST Interface .................................... 126
      5.2.3 FT Interface .................................... 126
      5.2.4 RT Interface .................................... 127
  5.3 Security Consideration ................................ 127
  5.4 Implementation ....................................... 129
      5.4.1 Implementation of an SLA Compliance Auditor .... 129
      5.4.2 Implementation of a Fact/Report Server ........ 139
  5.5 Chapter Summary ....................................... 141

6 Non-repudiation of Service Consumption ................. 143
  6.1 Roaming User Scenario ................................ 145
  6.2 Requirements ......................................... 146
  6.3 The NorCIS* Model ................................... 148
      6.3.1 Service Delivery and Consumption ............. 150
      6.3.2 Generation and Verification of Statements on Service Consumption ............. 150
      6.3.3 Generation and Verification of Evidences ........ 152
  6.4 The Design of NorCIS* Non-repudiation Protocols .... 153
      6.4.1 Transfer of an HP Evidence .................... 154
      6.4.2 Transfer of User and TTP Evidences ............ 154
  6.5 The NorCIS* Architecture ............................. 163
      6.5.1 Architectural Components ....................... 163
      6.5.2 User Identifiers ................................ 165
### 7 Evaluation 185

#### 7.1 Technical Feasibility 186
- 7.1.1 Metering 186
- 7.1.2 Accounting 194
- 7.1.3 Auditing 199

#### 7.2 Economic Feasibility 200
- 7.2.1 Economic Efforts 200
- 7.2.2 Economic Gain 204

#### 7.3 AURIC 207
- 7.3.1 Scalability 207
- 7.3.2 Reusability 209
- 7.3.3 Processing Time 214
- 7.3.4 Memory Requirements 220
- 7.3.5 Reliability 221

#### 7.4 NorCIS 222
- 7.4.1 Dispute Resolution and Fairness 222
- 7.4.2 Identity Privacy 228
- 7.4.3 TTP Involvement 228
- 7.4.4 Non-repudiation Protocol Overhead 229
- 7.4.5 Evidence Size 230
- 7.4.6 Costs of Unsuccessful Transactions 231
- 7.4.7 Reliability 234
- 7.4.8 Scalability 235
- 7.4.9 Possible Attacks 236

#### 7.5 Chapter Summary 239

### 8 Summary and Conclusions 241

#### 8.1 Summary 241
#### 8.2 Discussion of Results 244
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3 Conclusions</td>
<td>247</td>
</tr>
<tr>
<td>8.4 Future Work</td>
<td>249</td>
</tr>
<tr>
<td>Appendix A Mathematical Notation</td>
<td>251</td>
</tr>
<tr>
<td>Appendix B Sapta Grammar</td>
<td>255</td>
</tr>
<tr>
<td>B.1 Audit Subspecification</td>
<td>255</td>
</tr>
<tr>
<td>B.1.1 Filter Function Specification</td>
<td>256</td>
</tr>
<tr>
<td>B.1.2 Grouping Function Specification</td>
<td>256</td>
</tr>
<tr>
<td>B.1.3 Property Function Specification</td>
<td>257</td>
</tr>
<tr>
<td>B.1.4 Compliance Function Specification</td>
<td>257</td>
</tr>
<tr>
<td>B.1.5 Compliance Calculation Specification</td>
<td>257</td>
</tr>
<tr>
<td>B.1.6 Attribute Function Specification</td>
<td>258</td>
</tr>
<tr>
<td>B.1.7 Report Composition Specification</td>
<td>258</td>
</tr>
<tr>
<td>B.2 Statements</td>
<td>258</td>
</tr>
<tr>
<td>B.3 Expressions</td>
<td>259</td>
</tr>
<tr>
<td>B.4 Constants</td>
<td>261</td>
</tr>
<tr>
<td>Appendix C AURIC Processing Time</td>
<td>263</td>
</tr>
<tr>
<td>C.1 Average Processing Time</td>
<td>263</td>
</tr>
<tr>
<td>C.2 Average Processing Time per Fact</td>
<td>264</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>265</td>
</tr>
<tr>
<td>References</td>
<td>267</td>
</tr>
<tr>
<td>Curriculum Vitae</td>
<td>275</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1.1: A scenario of cooperating providers ........................................ 3
Figure 1.2: Role of auditing and non-repudiation in business interactions .................. 5
Figure 3.1: Generic auditing model .............................................................. 32
Figure 3.2: Generic model applied to SLA compliance auditing ...................... 35
Figure 3.3: Generic auditing architecture .................................................... 37
Figure 3.4: Multi-administrative domain architecture .................................. 39
Figure 3.5: A multi-domain scenario’s topology ........................................... 42
Figure 3.6: Auditing of network QoS commitment ....................................... 50
Figure 3.7: Different traffic directions to be metered .................................... 51
Figure 4.1: The audit function $\text{audit}_0$ ..................................................... 60
Figure 4.2: The general audit function $\text{audit}_g$ ........................................... 61
Figure 4.3: Illustrating Facts filtering for Service Breakdown SLO ................... 68
Figure 4.4: Illustrating Facts grouping for Service Breakdown SLO ................ 68
Figure 4.5: The flowchart of audit subtasks ................................................. 70
Figure 4.6: The use of audit specifications in audit subtasks ............................ 76
Figure 4.7: Inputs and output of the general audit function $\text{audit}_g$ .................. 77
Figure 4.8: Completeness criteria for SMI “Success of Service Request” ............ 112
Figure 5.1: AURIC’s architecture for SLA compliance auditing ..................... 118
Figure 5.2: Typical MSCs in audit management sessions ............................... 125
Figure 5.3: Access control to accounting data ............................................. 128
Figure 5.4: Implementation architecture of an SLA Compliance Auditor .......... 130
Figure 5.5: Implementation architecture of a Fact and Report Client ....... ....... 131
Figure 5.6: Implementation architecture of a Compliance Evaluator ............. 135
Figure 5.7: Implementation of a subtask module .......................................... 136
Figure 5.8: Implementation of a Fact/Report Server ..................................... 140
Figure 6.1: Roaming User Scenario ............................................................. 145
Figure 6.2: Two alternatives for evidence delivery ....................................... 147
Figure 6.3: A TTP’s involvement in various non-repudiation phases ............. 148
Figure 6.4: The NorCIS* model ................................................................. 149
Figure 6.5: The NorCIS* Architecture .......................................................... 163
Figure 6.6: Illustrating a signed UID mapping ............................................. 166
Figure 6.7: Non-repudiation interaction for proof after consumption ............ 169
Figure 6.8: Non-repudiation interaction for proof before consumption ........................................... 170
Figure 6.9: NR interaction with offline TTP for item-based accounting ............................................. 171
Figure 6.10: Starting a service consumption with Improved HS05 .................................................. 172
Figure 6.11: Service consumption termination in Improved HS05 ................................................... 173
Figure 6.12: The benefit of limiting the lifetime of an evidence ....................................................... 175
Figure 6.13: NR protocol sessions run in parallel .............................................................................. 178
Figure 6.14: Implementation architecture of the NR Client .............................................................. 180
Figure 6.15: Implementation architecture of the NR Server .............................................................. 181
Figure 6.16: Implementation architecture of the Cryptography Module ........................................... 182
Figure 7.1: Deployment of performance and usage measurements .................................................... 187
Figure 7.2: Illustrating inter-packet time interval and test cycle length ............................................. 188
Figure 7.3: Determining ping-based measurement configuration ..................................................... 189
Figure 7.4: Nmax in network performance measurements ................................................................. 197
Figure 7.5: Number of records generated in perceived QoS measurements ...................................... 198
Figure 7.6: Auditing of aggregated measurements across domains .................................................. 200
Figure 7.7: Measurement points for modules’ processing time. ....................................................... 215
Figure 7.8: Average processing time without application logic ....................................................... 217
Figure 7.9: Average processing time per Fact without application logic .......................................... 218
Figure 7.10: Average processing time per Fact in various use cases .................................................. 219
Figure 7.11: Heap memory usage of an auditor ............................................................................... 221
Figure 7.12: Possible evidence generation using a compromised key ............................................... 227
Figure 7.13: Adapted ZG97’s time costs of unsuccessful transactions ............................................. 233
Figure 7.14: Improved HS05’s time costs of unsuccessful transactions ........................................... 233
List of Tables

Table 2.1: Important traditional QoS parameters .............................................. 10
Table 2.2: QoS parameters for mobility ............................................................ 11
Table 2.3: Confidentiality and integrity as basis for other security properties ........ 12
Table 2.4: Definition of auditing ................................................................. 16
Table 2.5: Comparisons of Auditing Related Work ...................................... 26
Table 2.6: Comparison of Different Non-repudiation Approaches .................. 28
Table 3.1: Application of the generic model to different auditing areas ............ 35
Table 3.2: Example SLOs for network connectivity services ......................... 44
Table 3.3: Defining SLOs based on SLOs with other providers ....................... 45
Table 3.4: An example of application level SLO for a tele-lecture service ........ 46
Table 3.5: Example SLOs for perceived quality of multimedia data ................. 46
Table 3.6: Example SLOs for network QoS guarantees of a service class......... 47
Table 3.7: Example of service classes offered by provider P ......................... 53
Table 4.1: Useful examples of an SMI, its informal CS and properties ............ 62
Table 4.2: Comparison of different types of filter functions ......................... 79
Table 4.3: Comparison of different types of grouping functions .................... 81
Table 4.4: Audit specification of SMI 1 (Number of Service Breakdowns) ...... 85
Table 4.5: Audit specification of SMI 2 (Success of Service Requests) .......... 86
Table 4.6: Audit specification of SMI 3 (Downlink Throughput) ................. 88
Table 4.7: Required language constructs for audit specifications .................... 90
Table 4.8: Examples of constants in Sapta .................................................. 94
Table 4.9: Statements supported by Sapta .................................................... 95
Table 4.10: Description of the statement sequence in a FilterFunction specification .... 97
Table 4.11: Description of the statement sequence in a GroupingFunction specification .... 98
Table 4.12: Description of the statement sequence in an On_Timer construct .... 99
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.13</td>
<td>Possible notations to denote a particular Fact in a Fact-List</td>
</tr>
<tr>
<td>4.14</td>
<td>Interpretation of index &lt;n&gt; of a FactGroupExpression</td>
</tr>
<tr>
<td>4.15</td>
<td>Pre-defined functions supported by Sapta</td>
</tr>
<tr>
<td>5.1</td>
<td>Request-Answer message types for audit management</td>
</tr>
<tr>
<td>5.2</td>
<td>Notification message types for audit management</td>
</tr>
<tr>
<td>5.3</td>
<td>Various codes in ATP - AMgr communication</td>
</tr>
<tr>
<td>5.4</td>
<td>The use of the API's Process() methods</td>
</tr>
<tr>
<td>6.1</td>
<td>Different alternatives of parties generating statements</td>
</tr>
<tr>
<td>6.2</td>
<td>Frequency of statement generations</td>
</tr>
<tr>
<td>6.3</td>
<td>Parties and evidences of interest</td>
</tr>
<tr>
<td>6.4</td>
<td>Evidences to keep — as generated in the protocol Adapted ZG97</td>
</tr>
<tr>
<td>6.5</td>
<td>Evidences to keep — as generated in the protocol Improved HS05</td>
</tr>
<tr>
<td>6.6</td>
<td>Evidences to keep — as generated in the protocol Improved ZL98</td>
</tr>
<tr>
<td>6.7</td>
<td>Home Address versus CoA</td>
</tr>
<tr>
<td>6.8</td>
<td>NR Agent Interfaces</td>
</tr>
<tr>
<td>7.1</td>
<td>Economic efforts</td>
</tr>
<tr>
<td>7.2</td>
<td>Items required by a service provider for automated auditing</td>
</tr>
<tr>
<td>7.3</td>
<td>Example costs for cost calculation of automated auditing</td>
</tr>
<tr>
<td>7.4</td>
<td>The amount of credit in relation to TTR</td>
</tr>
<tr>
<td>7.5</td>
<td>Average processing time per Fact for 100000 Facts</td>
</tr>
<tr>
<td>7.6</td>
<td>Possible relations among a RegID, a VID, and an SPID</td>
</tr>
<tr>
<td>7.7</td>
<td>Required evidences to solve Dispute 2</td>
</tr>
<tr>
<td>7.8</td>
<td>Non-repudiation protocol overhead</td>
</tr>
<tr>
<td>7.9</td>
<td>Typical size of information elements</td>
</tr>
<tr>
<td>7.10</td>
<td>Size of evidences kept by FP and U per service consumption</td>
</tr>
<tr>
<td>7.11</td>
<td>The costs of a cancelled transaction</td>
</tr>
<tr>
<td>7.12</td>
<td>The costs of an invalid service request</td>
</tr>
<tr>
<td>7.13</td>
<td>Behavior of NR components after a restart</td>
</tr>
<tr>
<td>7.14</td>
<td>Counter measures against MITM attacks</td>
</tr>
<tr>
<td>7.15</td>
<td>Protection against replay attacks</td>
</tr>
<tr>
<td>C.1</td>
<td>Average processing time without application logic</td>
</tr>
<tr>
<td>C.2</td>
<td>Average processing time per Fact without application logic</td>
</tr>
</tbody>
</table>
List of Abbreviations

A

AA Authentication and Authorization
AAA Authentication, Authorization, and Accounting
AAAC Authentication, Authorization, Accounting, and Charging
AAC AAClient
AAS AA Server
ACA Accounting Answer
ACR Accounting Request
AD Administrative Domain
AM Audit Management
AMgr Audit Manager
API Application Programming Interface
AR Access Router
AS Audit Specification
ASAF Application Specific Attribute Function
ASCF Application Specific Compliance Function
ASFF Application Specific Filter Function
ASGF Application Specific Grouping Function
ASPF Application Specific Property Function
AT&T American Telephone and Telegraph Corporation
ATP Audit Task Planner
AURIC Auditing Framework for Internet Services
AVP Attribute Value Pair
AVPCM AVPs Calculation Module

B

BR Border Router
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCM</td>
<td>Compliance Calculation Module</td>
</tr>
<tr>
<td>CNSS</td>
<td>Committee on National Security Systems</td>
</tr>
<tr>
<td>CP</td>
<td>Content Provider</td>
</tr>
<tr>
<td>CS</td>
<td>Compliance Specification</td>
</tr>
<tr>
<td>DiffServ</td>
<td>Differentiated Services</td>
</tr>
<tr>
<td>DSCP</td>
<td>Differentiated Services Code Point</td>
</tr>
<tr>
<td>ETHZ</td>
<td>Eidgenössische Technische Hochschule Zürich</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>F.AAS</td>
<td>Foreign AA Server</td>
</tr>
<tr>
<td>F.NRS</td>
<td>Foreign NR Server</td>
</tr>
<tr>
<td>FC</td>
<td>Fact Client</td>
</tr>
<tr>
<td>FFM</td>
<td>Facts Filter Module</td>
</tr>
<tr>
<td>FGM</td>
<td>Facts Grouping Module</td>
</tr>
<tr>
<td>FP</td>
<td>Foreign Provider</td>
</tr>
<tr>
<td>FR</td>
<td>Fact and Report</td>
</tr>
<tr>
<td>FS</td>
<td>Fact Server</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>FT</td>
<td>Fact Transfer</td>
</tr>
<tr>
<td>GAAP</td>
<td>Generally Accepted Accounting Principles</td>
</tr>
<tr>
<td>GAAS</td>
<td>Generally Accepted Auditing Standards</td>
</tr>
<tr>
<td>GGF</td>
<td>Global Grid Forum</td>
</tr>
<tr>
<td>GRAAP</td>
<td>Grid Resource Allocation and Agreement Protocol</td>
</tr>
<tr>
<td><strong>H</strong></td>
<td><strong>I</strong></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>H.AAS</td>
<td>Home AA Server</td>
</tr>
<tr>
<td>H.NRS</td>
<td>Home NR Server</td>
</tr>
<tr>
<td>HP</td>
<td>Home Provider</td>
</tr>
<tr>
<td>HPID</td>
<td>Home Provider Identifier</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IRTF</td>
<td>Internet Research Task Force</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>MPQM</td>
<td>Moving Picture Quality Metrics</td>
</tr>
<tr>
<td>MSC</td>
<td>Message Sequence Chart</td>
</tr>
<tr>
<td>N</td>
<td>Non-repudiation of the Consumption of Internet Services</td>
</tr>
<tr>
<td>NorCIS</td>
<td>Non-repudiation</td>
</tr>
<tr>
<td>NR</td>
<td>NR Client</td>
</tr>
<tr>
<td>NRC</td>
<td>Non-repudiation of Origin</td>
</tr>
<tr>
<td>NRO</td>
<td>NR Server</td>
</tr>
<tr>
<td>NRS</td>
<td>Network Service Provider</td>
</tr>
<tr>
<td>NSP</td>
<td>Network Service Provider</td>
</tr>
<tr>
<td>O</td>
<td>Object Constraint Language</td>
</tr>
<tr>
<td>OCL</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>P</td>
<td>Peer Connection Management and Routing Framework</td>
</tr>
<tr>
<td>PCM&amp;RF</td>
<td>Peer Connection Management and Routing Framework</td>
</tr>
<tr>
<td>PESQ</td>
<td>Perceptual Evaluation of Speech Quality</td>
</tr>
<tr>
<td>PVCM</td>
<td>Property Values Calculation Module</td>
</tr>
<tr>
<td>Q</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>R</td>
<td>Registration Identifier</td>
</tr>
<tr>
<td>RegID</td>
<td>Registration Identifier</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comment</td>
</tr>
<tr>
<td>RGM</td>
<td>Report Generation Module</td>
</tr>
<tr>
<td>RT</td>
<td>Audit Report Transfer</td>
</tr>
<tr>
<td>RTP</td>
<td>Real Time Protocol</td>
</tr>
<tr>
<td>S</td>
<td>SCTP</td>
</tr>
<tr>
<td></td>
<td>SLA</td>
</tr>
<tr>
<td></td>
<td>SLO</td>
</tr>
<tr>
<td></td>
<td>SMI</td>
</tr>
<tr>
<td></td>
<td>SSH</td>
</tr>
<tr>
<td></td>
<td>ST</td>
</tr>
</tbody>
</table>

| T | TCP  | Transmission Control Protocol |
|   | TTP  | Trusted Third Party |

| U | UDP  | User Datagram Protocol |
|   | UID  | User Identifier |
|   | UML  | Unified Modeling Language |
|   | URL  | Uniform Resource Locator |

| V | VID  | Virtual Identifier |
|   | VoIP | Voice over IP |

| W | WSLA | Web Service Level Agreement |

| X | XMI  | XML Meta-data Interchange |
|   | XML  | Extensible Markup Language |
Chapter 1

Introduction

Since its introduction, the Internet has gradually grown into an important medium for a wide variety of business transactions. More and more services are deployed today through and with the aid of the Internet to allow for service access and consumption during 7 days per week and 24 hours per day. In this environment, the Internet is considered as the collection of networks and gateways that use the TCP/IP protocol suite and function as a single, cooperative virtual network [22].

Internet services play an important role in today’s business interactions because many business transactions are concluded with the help of Internet services. Since a uniform and standard definition of Internet services is not available and existing definitions are given from the viewpoint of the providers of specific services, this thesis defines Internet services as those services,
which use the Internet as an infrastructure to transfer data. This covers network and application services as well as content to be accessed over the Internet. A network service enables data to be transferred from one network node to another network node irrespective of the application to which the data belong, whereas an application service is a service provided by a software application which make use of network services to transfer application data among the software components. The future Internet is going to enable service providers to offer secured services with support of mobility and different levels of Quality-of-Service (QoS) [111] to mobile users.

Various interactions over the Internet, especially those business interactions mentioned, impose certain requirements on the quality level of utilized Internet services. To ensure that these requirements are fully met, it is necessary to achieve the desired quality of interactions. Typically, this is solved by negotiating a Service Level Agreement (SLA) between an Internet service provider and a customer to formally specify contractual commitments of the provider on which services and with which measurable quality level the provider will furnish [23].

The TeleManagement Forum's SLA Management Handbook defines an SLA as “a formal negotiated agreement between two parties, sometimes called a Service Level Guarantee. It is a contract (or part of one) that exists between the service provider and the customer, designed to create a common understanding about services, priorities, responsibilities, etc.” [102]. The typical information contained in an SLA consists of: a description of the nature of service to be provided, the expected performance level of the service, the procedure for reporting problems with the service, the time-frame for response and problem resolution, the process for monitoring and reporting the service level, the consequences for the provider not meeting its obligations, and escape clauses and constraints [108]. The committed performance level of a service is specified in a set of Service Level Objectives (SLO). An SLO defines an individual objective in terms of a service metric, threshold values, and tolerances [102]. An example of a typical SLO reads as follows: “The total network unavailability will be less than 5 minutes in any monthly period. A network is considered unavailable for the duration of a test period between any router pair, if a ping test between the router pair results in a 100% packet loss”. Consult [6] and Section 3.3.2 for more examples on SLOs.
1.1 Motivation

In general, verifying that no party in a contract violates her commitments is the major concern of all parties. Therefore, the key objective for this thesis work is to design, specify, and implement a technically feasible approach to ensure that providers and customers do find a secured, fair, and efficient way to deliver and utilize the Internet service negotiated in advance, where contract violations — either intended or not — can be timely detected through an automated mechanism and possible disagreements between the two sides can be efficiently resolved.

1.1 Motivation

As introduced, SLA compliance auditing is necessary and important, especially for commercial transactions or high-value services. Figure 1.1 depicts a motivating example where Internet Service Provider (ISP) P1 is operated as a Content Provider (CP) and ISP P2 is operated as a Network Service Provider (NSP). ISP P1 establishes an SLA with ISP P2 in order to be able to deliver videos with a specific throughput as requested for by end customers. ISP P2 operates a network with QoS-enabled transport services. ISP P1 (as a customer) pays ISP P2 for data transfer depending on the service class used. Each service class defines a certain committed throughput. End customers negotiate an SLA with ISP P2 to achieve network connectivity and negotiate an SLA with ISP P1 to subscribe to a video service.

Figure 1.1: A scenario of cooperating providers.
The video server of ISP P1 requests the necessary service class from the network infrastructure of ISP P2 before delivering a video. Now, if the network infrastructure commits to provide a certain bandwidth, but cannot hold the commitment during the video transfer, this situation determines an SLA violation. Traffic rerouting may avoid a potential SLA violation, and if SLA violations occur too frequently, network redimensioning may be necessary. A customer pays for services and the committed quality, hence, she is interested in having the provider meet all SLOs negotiated.

Having performance data available (i.e., measurement data about the delivered Quality-of-Service, QoS) defines the prerequisite for an SLA compliance audit. The amount of data is generally huge, as it is collected over a relatively long period of time and for all important services consumed by a large number of customers. Thus, an automation of SLA compliance audits is beneficial, since a manual audit task will be very time-consuming and error-prone. Besides this higher efficiency automation also allows for an auditing to be performed continuously or even in real-time in order to detect violations as early as possible. Since each SLO to be audited normally defines a different logic, which depends on the service and, thus, can be arbitrarily complex, a generic framework is necessary to minimize the development effort of auditing applications for different SLOs or different types of SLOs.

Finally, the contractual relation between a service provider and a service user obliges the user to pay for the service consumed. However, a user may deny having consumed a service. Hence, a non-repudiation mechanism to protect against users’ false denial of having consumed specified services is essential. This mechanism is responsible for generating statements on service consumption to be signed by the user and transferring the resulting evidence of service consumption to the provider to be stored. These evidences do not only justify billing, but are also able to protect users and providers against malicious parties.

Thus, Figure 1.2 illustrates in an integrated manner the primary importance of SLA compliance auditing and non-repudiation of service consumption in a business relation between a service provider and a customer.
1.2 Problem Statement

A generic auditing framework, as motivated above, is supposed to be applicable to various auditing objectives, which originate from different services. Therefore, the term auditing as well as auditing objective needs to be defined and a model must be designed, which identifies all key roles directly involved in an audit and those relationships (i.e., the interactions or information exchanges) among these roles.

The underlying architecture of the auditing framework must allow for a flexible distribution of its components, since interactions in SLA compliance auditing may cross administrative domains. In this respect, security and reliability problems of inter-domain interactions need to be considered. A generic auditing framework must implement functionality commonly needed

---

1. Infrastructures within an administrative domain are managed by a single administrative authority. Hence, intra-domain interactions are considered to be secure, since security associations among interacting components are established.
by most of the audits, yet it shall allow for future extensions. It must be *highly reusable* and provides for an *easy integration* of application-specific components.

Additionally, a service consumption requires an evidence to be generated to justify billing of the service usage. In general, a service consumption is not an event happening at a certain point in time, but it has a certain duration. Hence, traditional non-repudiation protocols dealing with non-repudiation of origin or non-repudiation of receipt of messages are not sufficient to prove a service consumption.

A major problem in proving a service consumption is the generation of an evidence, which contains the *real* consumption, if the user or the provider may not play *fair*. On one hand, if an evidence has to be generated *after* a service consumption (proof after consumption, in accordance to pay after use), there is a risk that the user may not send the evidence after consuming the service. On the other hand, if an evidence has to be generated *before* a service consumption (proof before consumption, in accordance to pay before use), the provider may not deliver the service after obtaining the evidence.

Finally, an evidence of service consumption must also identify the user, who has consumed the service. However, in a mobile Internet, a user visiting a foreign domain may not want her identity to be revealed through the generated evidence. Therefore, protocol interactions must be designed to allow for an evidence of service consumption to be verified without revealing the real identity of the user.

### 1.3 Thesis Goal and Contribution

The key driving force of this thesis work was the concern of both service providers and service users with respect to their contractual commitments in the delivery and usage of Internet services. A service provider is concerned about service consumption being repudiated by users, whereas a service user (which may be a service provider as well) is concerned about obligations committed in an SLA being not held by providers.
The key goal of this thesis is, therefore, to develop a generic framework for automated, flexible, and efficient auditing of SLA compliances and technically feasible protocols for non-repudiation of service consumption. The auditing framework is designed to support inter-domain interactions, since SLA compliance auditing must be applicable in a multi-domain environment. The generic framework provides the basis for the development of an SLA compliance auditing application.

This thesis makes the following contributions:

- A generic model and architecture for automated auditing which is applicable to various auditing areas and with a support of a multi-administrative domain environment.
- A concept to decompose an audit task into a sequence of subtasks and an expressive language to specify completely an audit task.
- A generic framework for SLA compliance auditing based on the audit task decomposition and the auditing architecture. The framework and three SLO-specific auditing applications, taken as valuable examples, based on the framework are implemented prototypically and evaluated.
- An accounting architecture and its protocol interactions to support non-repudiation of service consumption and identity privacy. A non-repudiation protocol is proposed for each of the three accounting schemes, item-based, time-based, and volume-based accounting. A comprehensive list of possible disputes and threats in service consumption is presented against which these protocols and their generated evidences are evaluated.

1.4 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 2 provides a general definition of auditing, examines key QoS parameters for services with mobility and security support, and discusses related work in SLA compliance auditing and non-repudiation.
Chapter 3 introduces the generic model and architecture for the automated auditing framework developed and called AURIC (Auditing Framework for Internet Services). The model is shown to be applicable to different auditing areas. This chapter also elaborates the multi-domain scenario introduced in Section 1.1, defines example services and respective SLOs. Additionally, it presents a use case in SLA compliance auditing to show the instantiation of the generic architecture.

Chapter 4 analyzes the audit task in detail and proposes the functional decomposition of this audit task into a sequence of subtasks, each being responsible for a specific function. This chapter also proposes a new and dedicated language — termed Sapta — to specify correct and complete audit subtasks in a flexible and extensible manner.

Chapter 5 presents the detailed architecture of SLA compliance auditing derived from the generic model and architecture. It also defines key interfaces and describes interactions among the main components. Furthermore, this chapter describes a prototypical implementation of the auditing framework.

Chapter 6 develops the necessary non-repudiation architecture termed NorCIS* (Non-repudiation of the Consumption of Internet Services), which provides for a non-repudiation functionality in an environment with Authentication, Authorization, and Accounting (AAA) support. This chapter also proposes a suitable non-repudiation protocol for each of the accounting schemes: item-based, time-based, and volume-based accounting.

Chapter 7 presents a thorough evaluation of AURIC and NorCIS*. A feasibility analysis of the automated auditing framework is done for the multi-domain scenario elaborated in Chapter 3. The design and prototypical implementation of AURIC is evaluated, especially with respect to key dimensions of scalability and reusability. Regarding NorCIS*, all non-repudiation protocols proposed are evaluated with respect to overhead, possible disputes, and attacks.

Chapter 8 summarizes this thesis’ main contributions, discusses economic advantages of AURIC compared to manual audit, draws conclusions with respect to AURIC’s and NorCIS*’s applicability, and presents an outlook to possible future work.
Chapter 2
Definitions and Related Work

Most of existing approaches in auditing are dedicated to specific objectives. Although some proposals show to a certain extent a general approach, a generic auditing model and architecture — to the best knowledge of the author at the moment of writing — is not yet available. This chapter describes key QoS parameters potentially defined in an SLO and standard definitions of auditing before discussing various auditing related work.

2.1 Key QoS Parameters

QoS-enabled mobile Internet will form the future Internet. There is a common view that the current Internet and cellular networks tend to converge to an all IP-based telecommunication network. A large variety of serv-
services will be offered ranging from merely internet connectivity to applications and contents enhanced with mobility and security support. This section describes various important QoS parameters and the impact of mobility and security support to be considered when defining an SLO.

2.1.1 Traditional QoS Parameters

Traditional QoS parameters are defined in this thesis as QoS parameters which are specified for Internet services delivered to a fixed terminal by a fixed service provisioning infrastructure. Internet services are accomplished by transferring data from one location to another location according to a set of communication protocols. Both the transfer and the data may form a service offered to a customer. In case of transfer, QoS parameters comprise various characteristics to determine the quality of a data transfer. The term content is normally used, if data are the service offered to a customer. In this regard, QoS parameters comprise various characteristics to determine the quality of a content. Table 2.1 lists a collection of important QoS parameters of different types of Internet services.

<table>
<thead>
<tr>
<th>Service</th>
<th>Important QoS Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network layer data transfer</td>
<td>Throughput, delay, jitter, loss rate, error rate</td>
</tr>
<tr>
<td>Application layer data transfer</td>
<td>Throughput, application response time</td>
</tr>
<tr>
<td>Content</td>
<td>Resolution, e.g., audio data: highest frequency captured</td>
</tr>
</tbody>
</table>

Generally, a simple logical expression is sufficient to express an SLO for a traditional QoS parameter (e.g., response time less than 100 ms). An auditing application for this simple SLO is easily parameterized. However, an auditing task becomes more complex if the same application has to consider different committed values due to different service classes or if measured values have to be aggregated per service class. In this regard, various aggregation functions need to be supported by an application and an SLO specification.
2.1.2 Impacts of Mobility Support

Mobility can be seen as the capability to access services from different access points and to preserve this access, while moving from one access point to another access point. The most popular solution is provided by Mobile IP [84]. If mobility is to be supported across organizational boundaries, roaming agreements must be concluded, and security in terms of authentication and authorization becomes an important factor to control service accesses [33]. Based on the types of mobile entities, mobility is classified into device, user, and session mobility. Device mobility allows a device to change its network point of attachment while preserving ongoing communications, whereas user mobility allows a user to use the same identity to access services irrespective of the device used and its network point of attachment. In case of session mobility, an ongoing session can be moved from one device to another device. A session is defined here as an instance of a service over time.

Mobility support has two impacts on defining an SLA: definition of new QoS parameters to distinguish between different quality levels of each type of mobility and adaptation of traditional QoS parameters due to mobility. Table 2.2 shows reasonable QoS parameters for device and session mobility since they can be used to assess the quality of a mobility support. With respect to impacts on traditional QoS parameters, the support of mobility allows mobile entities to move to the same cell and overload the access point, thus potentially cause the service provisioning infrastructure to reduce the quality of data transfer. Therefore, the definition of a committed value and its tolerance for a traditional QoS parameter needs to consider mobility. For example, an SLA can specify a higher tolerance in times of network overload, which should be limited to a pre-defined duration or a certain number of occurrences.

Table 2.2: QoS parameters for mobility.

<table>
<thead>
<tr>
<th>Mobility</th>
<th>QoS Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>QoS parameters related to handover between two points of attachment are defined to capture the quality of device mobility: handover success ratio and handover delay.</td>
</tr>
</tbody>
</table>
Table 2.2: QoS parameters for mobility.

<table>
<thead>
<tr>
<th>Mobility</th>
<th>QoS Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Principally, the coverage of a user mobility support can differentiate the quality between two user mobility supports. Without roaming agreements user mobility is restricted to within a single administrative domain. Roaming agreements extend the coverage of a user mobility support. However, the coverage of a user mobility support is fixed and does not change over time during a service usage. Hence, it is not a QoS parameter that needs to be monitored.</td>
</tr>
<tr>
<td>Session</td>
<td>Session handover success ratio, session handover delay.</td>
</tr>
</tbody>
</table>

2.1.3 Impacts of Security Support

Computer security is a broad area, however, it rests on three fundamental properties: confidentiality, integrity, and availability [13]. Other security properties are based on confidentiality and integrity, as shown in Table 2.3. Data confidentiality is the concealment of data. Offering data confidentiality as a security service means providing a service to protect against unauthorized disclosure of data. To provide for anonymity, identities must be kept confidential, whereas to provide for privacy, personal information must be kept confidential. Surely, a security service infrastructure needs to provide more functionality in order to support anonymity or privacy compared to the simply support of data confidentiality.

Table 2.3: Confidentiality and integrity as basis for other security properties.

<table>
<thead>
<tr>
<th></th>
<th>Data Confidentiality</th>
<th>Data Integrity</th>
<th>Origin Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymity</td>
<td>Source identity</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Privacy</td>
<td>Personal information</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Authentication</td>
<td>-</td>
<td>Source identity</td>
<td>required</td>
</tr>
<tr>
<td></td>
<td>Credentials</td>
<td>Source identity</td>
<td>-</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>-</td>
<td>Source identity + message</td>
<td>required</td>
</tr>
</tbody>
</table>
Integrity comprises data integrity and origin integrity. Data integrity refers to the validity of data in the sense of not being modified in unauthorized ways or by an unauthorized user. A security service for data integrity provides for protections against unauthorized modification of data. Note that modification includes insertions and deletions. Origin integrity refers to correct identification of the source of data. A security service for origin integrity provides for protections against falsely claimed identity. It is important to note here that integrity does not imply the truthfulness of the information that the data represent nor the trustworthiness of the source. Truthfulness of an information requires evidences and trustworthiness of a source requires ratings, e.g., by a reputation system.

Availability is the degree to which a resource or a service is accessible when needed. In general, the time at which a user accesses a resource or a service is not known in advance. Hence, a provider needs to make it available too, even no user is currently requesting it. The value of availability is normally expressed in percentage of time within a period of pre-defined length during which the resource or service is accessible. Availability requires all involving resource or service provisioning entities to be operational and an access path to be established. The whole delivery chain is responsible for availability, since any breakdown of a component in the chain results in unavailability. Furthermore, availability across multiple domains requires those domains to cooperate.

As in the case of mobility support, security support also has impacts on defining an SLA. Each security property can be seen as a QoS parameter. Except for availability, these QoS parameters have a binary value, i.e., they are either supported or not. It is possible to detail the technical characteristics of a security support in order to define different quality levels of a security support. For example, with respect to confidentiality and integrity, different quality levels can be achieved through the use of different algorithms or different key lengths. However, offering different quality levels of a security support in this way means configuring security provisioning entities to use a different algorithm with a different key length according to a user request, which is impractical.

In general, auditing security guarantees is difficult, if not impossible, because no proof can be given which shows that they are held until there is a
compromise which means that they are not held. Therefore, auditing the above parameters, except for availability, is restricted to examination of whether the respective security mechanisms, e.g., encryption, are applied or not. Nonetheless, finding criteria to determine this is already a problem in itself. In a specific technology, e.g., the use of IPsec [69] as a security mechanism on network layer, the occurrence of a certain packet header can be used to define the compliance criteria.

With respect to impacts on traditional QoS parameters, security mechanisms require additional data processing time, thus lead to communication overhead. Hence, committed values and tolerances of traditional QoS parameters are to be adapted.

### 2.2 Definition of Auditing

Auditing is a widely applied concept for investigating the adequacy of a system against a set of requirements. Traditional areas of auditing are amongst others financial auditing [104], compliance auditing with respect to governmental laws and regulations, and quality audits [98]. Financial auditing is concerned with the provisioning of a reasonable assurance about whether financial statements are presented fairly in conformity with Generally Accepted Accounting Principles (GAAP) [104], [29] or with a comprehensive basis of accounting other than GAAP. There are approaches to automate certain tasks of financial auditing, e.g., through the use of Artificial Intelligence [105], electronic audit data warehouses [92], and data marts [89]. However, human interventions are still needed in general. In this area, audit software tools are utilized to help a human auditor in the analysis of transactional data [3], [17]. This area of auditing is not the target area of this thesis, and work in this area will not be further discussed.

The wide use of the Internet by research and government institutions, companies, and individuals, as well as the commercialization of Internet services have opened up new and important areas of auditing. The development and deployment of the Internet has bridged the path to the information era, where information becomes a vital resource. Usually information, in form of digital objects, is stored in a computer system, which is connected to the Internet to allow for remote access. Since access to the Internet can be
2.2. Definition of Auditing

gained by anyone, the Internet is subject to attacks. To cope with attacks, various Intrusion Detection Systems (IDS) have been developed. An IDS is a security audit tool to reveal unauthorized access attempts. IDS has been a research area of security auditing since the beginning of the 1980s\(^1\). Note that in the areas related to the Internet the term audit is normally used in conjunction with security audits. However, any activity dealing with examination of something against a specification is principally an audit, and every audit is by its nature a compliance investigation. Therefore, SLA compliance monitoring determines an audit activity also.

Definitions given by the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and the Internet Engineering Task Force (IETF) focus on auditing in the area of security [60], [93], [99]. Both define the term security audit in a similar way. The U.S. Committee on National Security Systems (CNSS), which is chaired by the Department of Defense, defines the term audit instead of security audit and leaves out the clause on “detection of security breaches” [103]. Although this definition does not emphasize security, it is still not a general definition of auditing.

Table 2.4 gives an overview of those different definitions as well as the definitions by the author which are used throughout this thesis. As both ITU-T and IETF documents deal with auditing in the security area, the term security audit is defined instead of the more general term audit. In addition to security audit and security audit trail, the IETF defines an audit service. However, an audit service is not defined as a service which performs a security audit, but rather a service that provides for a security audit trail. The service which performs a security audit is termed Intrusion Detection by the IETF.

Those definitions provided by this thesis are concise and general without mentioning specific purposes of auditing (e.g., adequacy of system controls, policy compliance). With these definitions, an audit is seen as a general mechanism for a compliance investigation. It has to be noted here that the term to comply with or to violate a specification is free from any evaluation of good or bad. To comply with a specification simply means that the re-

\(^1\) Todays, IDS has evolved into Intrusion Protection System (IPS) to actively block network traffic if a possible attack is detected. However, the underlying detection mechanisms are the same.
requirements are fulfilled. Whether this is good or bad for a particular party depends on the expectation or interpretation.

*Table 2.4: Definition of auditing.*

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Definition</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Audit</td>
<td>An independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, to detect breaches in security, and to recommend any indicated changes in control, policy and procedures.</td>
<td>ITU-T X.800.</td>
</tr>
<tr>
<td>Security Audit</td>
<td>An independent review and examination of a system's records and activities to determine the adequacy of system controls, ensure compliance with established security policy and procedures, detect breaches in security services, and recommend any changes that are indicated for countermeasures.</td>
<td>IETF RFC 2828.</td>
</tr>
<tr>
<td>Audit</td>
<td>Independent review and examination of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and operational procedures, and to recommend necessary changes in controls, policies, or procedures.</td>
<td>U.S. CNSS.</td>
</tr>
<tr>
<td>Audit</td>
<td>A systematic and independent examination of facts on system activities to determine the degree of compliance with a pre-defined set of specifications.</td>
<td>This thesis</td>
</tr>
<tr>
<td>Auditing</td>
<td>The process of conducting an Audit.</td>
<td>This thesis</td>
</tr>
<tr>
<td>Auditor</td>
<td>An entity (not necessarily restricted to a human being) that carries out the audit.</td>
<td>This thesis</td>
</tr>
<tr>
<td>Security Audit Trail</td>
<td>Data collected and potentially used to facilitate a Security Audit.</td>
<td>ITU-T X.800.</td>
</tr>
<tr>
<td>Security Audit Trail</td>
<td>A chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence of environments and activities surrounding or leading to an operation, procedure, or event in a security-relevant transaction from inception to final results.</td>
<td>IETF RFC 2828.</td>
</tr>
<tr>
<td>Audit Trail</td>
<td>Chronological record of system activities to enable the reconstruction and examination of the sequence of events and/or changes in an event.</td>
<td>U.S. CNSS.</td>
</tr>
</tbody>
</table>
2.3 Auditing Related Work

Modelling of an auditing system with respect to Internet services has been proposed in security audit. In the area of SLA compliance auditing most of the existing approaches are dedicated to specific services, e.g., web services, or concentrate on the measurements of a certain set of QoS parameters, e.g., availability, round-trip time, and response time [4], [31], [35], [53], [39]. Some focus on efficient algorithms to detect violation, e.g., in network usage [37], while others propose a language to define SLAs [95], [96]. The following subsections briefly describe key characteristics of these related approaches, which are complemented by a comparison of those in Section 2.5. The key characteristics of interest comprise the focus of the related approaches, architectural properties, and the language used to express an SLO.

2.3.1 IDS Modelling and Architecture

In 1987, D. E. Denning presented a model for a real-time intrusion detection expert system, which provides the basis for many IDS [26]. The model can be considered as a rule-based pattern matching system and contains the following six main components: subjects, objects, audit records, profiles, anomaly records, and activity rules. In this model actions performed by subjects on objects are essential, which is valid for an IDS, but not in all areas of auditing. For example, in the case of SLA compliance auditing services delivered by objects to subjects are relevant. Furthermore, profiles in this model, which describe a normal behavior of subjects on objects, are updated based on audit records. Here, profiles define those specifications to be met.

### Table 2.4: Definition of auditing.

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Definition</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Service</td>
<td>A security service that records information needed to establish accountability for system events and for the actions of system entities that cause them.</td>
<td>IETF RFC 2828.</td>
</tr>
<tr>
<td>Intrusion Detection</td>
<td>A security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of attempts to access system resources in an unauthorized manner.</td>
<td>IETF RFC 2828.</td>
</tr>
</tbody>
</table>
However, general auditing does not modify compliance specifications so that they are met by the normal behavior. Therefore, this model is not a general auditing model and not applicable to SLA compliance auditing.

The architecture of a general IDS consists of event generators within a target system, analysis engines, and a response unit [74]. Components can be distributed and analysis engines can form a hierarchy. Recent architectures use autonomous and mobile agents to perform the task of event generators and analysis engines. Distribution of IDS components happens within a single administrative domain, whereas components in SLA compliance auditing can be distributed across multiple domains. Hence, work on IDS are not further discussed.

2.3.2 WSLA Framework

The IBM’s Web Service Level Agreement (WSLA) Framework [68] shows a general concept for SLA management including online monitoring of SLA violation and defines a language to specify SLAs [72], focusing on web services. The WSLA Framework comprises six elementary services, which can interact across multiple domains: SLA Establishment Service, SLA Deployment Service, Measurement Service, Condition Evaluation Service, Management Service, and Service of the Business Entity. The WSLA Framework defines five stages for an SLA management lifecycle. The functionality needed for these various stages is implemented as WSLA services.

The language used to specify an SLA is called WSLA language. A WSLA document is composed of three sections: a description of the involving parties, service definitions, and specifications of obligations. The description of a party includes the definition of its role (provider, customer, or third party) and operations that it can perform when triggered by another party. A service definition section describes services in terms of supported operations, valid SLA parameters, and corresponding metrics to be used. Metrics define how values of SLA parameters are to be calculated or measured. The section on obligations defines Service Level Objectives with respect to SLA parameters defined in the service definitions section.
A condition in a WSLA’s SLO is a logic expression (i.e., logical operators applied to predicates). A predicate is a function that returns true or false, which is used to capture the assertions with respect to an SLA parameter. WSLA does not support specification of compliances using control statements like conditional branches and iterations. The WSLA language is based on XML and defined as an XML schema. Although the WSLA language allows new functions and predicates to be declared (function and predicate in WSLA XML schema definition are abstract types), the framework cannot interpret newly declared functions and predicates.

A measurement metric in WSLA consists of a single field (e.g., ResponseTimeHost1). The framework does not expect to process metered data consisting of more than one field (e.g., <IPAddress, PacketLossRatio>). Therefore, to differentiate between users in SLA compliance audit would require an instance of measurement and condition evaluation service per user which leads to scalability issue. Furthermore, a SLA compliance verification based on WSLA must run in real-time. The timepoint at which the value of a measured metric is transferred is considered as the measurement timepoint. Batch processing of measured data is, therefore, not supported.

2.3.3 Cremona

Cremona [73] is an architecture and library for the creation and monitoring of WS-Agreements developed by researchers of IBM. WS-Agreement is a specification worked out by the Grid Resource Allocation and Agreement Protocol Working Group (GRAAP-WG) of the Global Grid Forum [34]. The goal of WS-Agreement is to standardize the terminology, concepts, overall agreement structure with types of agreement terms, agreement template with creation constraints, and a set of port types and operations for creation, expiration, and monitoring of agreements, including WSDL needed to express the message exchanges and resources needed to express the state [5]. Agreements are expressed in an XML-based language, in which an agreement is composed of a Name to identify the agreement, a Context to identify the involved parties, Service Description Terms, and Guarantee Terms.

SLOs are defined in the Guarantee Terms part of an agreement. An SLO is expressed either using a KPITarget or as a customized expression of service level. The KPITarget specifies a target value for a Key Performance Indi-
cator (KPI) (e.g., average response time, availability) associated with a service. An agreement may contain several SLOs. The customized expression of a service level is not further detailed and is supposed to be domain-specific.

Cremona defines an architecture consisting of multiple layers of agreement management: Agreement Protocol Role Management (APRM), Agreement Service Role Management (ASRM), and Strategic Agreement Management (SAM). Agreements are established between two roles: agreement initiator and agreement provider, which are independent of the roles of service provider and service consumer. APRM is responsible for creating and establishing agreements between agreement provider and initiator. ASRM builds on top of APRM and is responsible for relating APRM to service provisioning or service-consuming system. ASRM includes components for triggering service delivery and the monitoring of compliances at runtime. The agreement provider side’s of APRM implements the status monitor which accesses system instrumentation (measurements) on service provider or service consumer sides. SAM is a meta-layer to APRM and ASRM. It comprises management functions dealing e.g., with policy decisions on choosing role as agreement provider or initiator for specific services. Cremona does not propose a specific component model for the SAM layer.

The Cremona framework library supports the implementation of an Agreement Manager as a Java servlet on the agreement provider side and a Java application on the initiator side. Interfaces to APRM and ASRM are provided in Java. Classes implementing Status Monitor interface are supposed to be implementation specific.

2.3.4 Monitoring Schemes for SLA Violation

Monitoring schemes (methodologies) to detect SLA violations and attacks in a QoS supporting network domain are proposed in [37] to ensure the proper operation of the network. The differentiated services (DiffServ) QoS framework is used as an underlying network. DiffServ allows service differentiation and, thus, different pricing schemes to be applied to different service classes. This attracts attacks by injecting marked packets to steal bandwidth and other network resources. [37] uses network tomography to develop stripe-based and overlay-based distributed network monitoring
schemes that use only edge-to-edge measurements to infer the internal characteristics of a network domain. The exclusion of core routers in the measurement is to achieve efficient and scalable network monitoring. The QoS parameters employed are throughput, delay, and packet loss.

The stripe-based monitoring scheme uses a series of back-to-back packets (referred to as a “stripe”) to infer delay and loss of the internal links of a network domain. The link loss of each individual link can be calculated by probing from any edge router to all other pair of edge routers. The congestion experience by successive packets in a stripe is assumed to be correlated. The packets have similar experience along the path on their journey to the destination. This correlation is used to infer the loss of each internal links using only edge-to-edge measurements. Throughput measurements are only performed when a delay or loss violation is reported. The overlay-based scheme further reduces the communication overhead. In this scheme, the edge routers form an overlay network with their neighboring edge routers. The overlay network is probed intelligently to identify the congested links with high losses in a domain. These links are used to identify the flows that are causing this high loss.

2.3.5 TAPAS SLAng

SLAng [96] is a language for expressing Service Level Agreements developed as part of the European Union’s project TAPAS (Trusted and Quality-of-Service Aware Provision of Application Services) [101]. It is designed to be able to define SLA precisely in order to protect against potential disagreements on whether a violation has occurred. Four different ways of disagreements are identified:

1. Disagreement over the terms of the agreement (by disagreeing over whether a particular piece of monitoring data or aspect of the services configuration is relevant to the calculation of a violation).

2. Disagreement over the conditions of the agreement (by disagreeing over whether a particular behaviour of the service constitutes a violation).

3. Disagreement over the amount of error introduced by the particular process or mechanism for calculating whether a violation has occurred
from a particular set of monitoring data.

4. Disagreement about the amount of error present in any monitoring data, in effect a disagreement over the degree to which a particular set of monitoring data represents the true behaviour of the service.

SLAng is defined using a meta-model, an instance of the Meta-Object Facility (MOF) model [78], in which the relationship between the syntax of the language and its domain of application is explicitly represented. The violation semantics of the language is defined using Object Constraint Language (OCL) constraints. The concrete syntax of the language is the XML Metadata Interchange (XMI) mapping of the syntactic part of the meta-model.

To reduce the possibility of the third type of disagreement from the list above, [95] describes a way to automatically generate a contract checker by using the meta-model of the language and associated constraints as inputs for a generative programming tool. The checker compares the measured performance of a service with a set of SLAs to determine if violations have occurred. The technologies and process applied to generate the contract checker are components of the Model Driven Architecture (MDA) approach under development by the Object Management Group (OMG) [77]. The Java Metadata Interface (JMI) mapping is applied to the meta-model of the language to generate interfaces and classes to create and query SLAs and relevant service monitoring data in memory; and an OCL interpreter is applied to check violation constraints over this data.

2.3.6 Moby Dick’s SLA Compliance Auditing

The EU IST project Moby Dick [76] integrates mobility, AAAC, and QoS [41], [42], [43], [67]. It defines QoS-supporting transport services for mobile users, where QoS differentiation is based on DiffServ Code Point (DSCP) value [66]. Moby Dick SLA compliance auditing focuses on monitoring service availability. To detect service unavailability key service provisioning entities are required to send a log periodically as a sign of life. Besides that, responses to user requests on QoS are also logged [40].

Within the project Moby Dick an SLA compliance auditing architecture has been designed by the author of this thesis and a language has been defined to specify audit rules for the processing of audit trail [45], [46]. An au-
dit rule defines the applicable scope of the rule, a violation condition, and an action. The scope of a rule contains a condition to filter events to be examined, a set of attributes to group events, so that the same rule can be applied to each group separately, and a time frame for violation detection. The violation condition is composed of primitive functions to process events including logical expressions and aggregation functions.

2.3.7 Daidalos’ SLA Compliance Auditing

The EU IST project Daidalos [24] is an Integrated Project covering AAA, mobility management, resource management, QoS, and security, on top of heterogenous access network. It is user centric in the sense that it supports pervasiveness, privacy, personalization, intra- and inter-domain mobility. It is also operator-driven through the design and development of service provisioning platform for multimedia and broadcast services.

In the framework of the Daidalos project, an architecture for SLA compliance auditing has been co-designed by the author of this thesis. The SLOs of interest are delay, packet loss, and throughput of selected service classes. SLOs are defined per service class in XML, requiring the auditing framework to be able to parse the SLOs. The violation detection component is capable of interpreting compliance conditions, but the conditions are only simple relational expressions.

2.4 Non-repudiation Related Work

The underlying model of various non-repudiation related work is based on the IETF or ISO definition of non-repudiation, which is not applicable to all non-repudiation services as shown below.

The definition of repudiation is provided by the IETF in [93] as a “denial by a system entity that was involved in an association (especially an association that transfers information) of having participated in the relationship.” And a non-repudiation service is defined as “a security service that provides protection against false denial of involvement in a communication.” The IETF makes a further remark that “A non-repudiation service does not and cannot prevent an entity from repudiating a communication. Instead, the
service provides irrefutable evidence that can be stored and later presented to a third party to resolve disputes that arise if and when a communication is repudiated by one of the entities involved.” Therefore, designing a non-repudiation protocol requires possible disputes to be studied and analyzed.

The International Organization for Standardization (ISO) non-repudiation model deals with events of creating, sending, receiving, and recognizing the content of a message [52]. Several non-repudiation services are defined with each service related to the specific event or a meaningful combination of those events. E.g., the non-repudiation of origin (NRO) is a non-repudiation service which is intended to protect against an originator’s false denial of having created the content of a message and of having sent the message.

Obviously, these IETF and ISO definitions are related or restricted to communication events. The approach presented in this thesis, however, uses the term non-repudiation in a broader sense. A special non-repudiation service is proposed, i.e., a non-repudiation of service consumption. This is different from non-repudiation in IETF or ISO sense, since it is not the receipt or the sending of a message that needs to be proved, but the usage of a service. The objective is to protect against users’ false denial of having consumed specified services. Hence, statements on service consumption need to be signed by the user and securely stored by the provider.

Current non-repudiation protocols reduce the involvement of Trusted Third Parties (TTP) to deal with keys only rather than with the content of transferred messages. Research has been performed in achieving specific requirements on the property of a non-repudiation protocol, e.g., fairness. Since its introduction in non-repudiation protocols, the definition of fairness has evolved into different flavors: weak, strong, true, and probabilistic fairness [70], [75]. A non-repudiation protocol provides strong fairness if and only if at the end of a protocol execution either A received the non-repudiation of receipt evidence for the message M, and B received the corresponding message M as well as the non-repudiation of origin evidence for this message, or none of them received any valuable information [70]. A fair non-repudiation protocol using an on-line TTP is proposed in [115] and a number of protocols have been developed to improve fairness and security with respect to exchanges of electronic goods [7], [36], [113], [114].
2.4. Non-repudiation Related Work

Proving service requests and access granting determines a more general case, whereas proving service provisioning and service usage is usually application-dependent and sometimes hard to decide without human intervention. [49] shows how non-repudiation methods can be used to prove service requests and access granting for a lease service using public-key cryptography.

[71] proposes a protocol to provide mutual entity authentication, identity privacy, and a limited version of non-repudiation service to secure mobile communications. The protocol is based on the use of conventional secret-key techniques in combination with modern public-key techniques. Upon subscription a mobile user holds the public encryption key of the Home Provider (HP) and a secret-key k shared between the mobile user and the HP. HP’s public-key is used to securely transmit a user's identity and credentials to the HP for authentication. Some shortcomings exist. Since evidences are generated based on k, they cannot be generated by the Foreign Provider (FP), but they can be generated by the HP. Hence, a user is not protected against his malicious HP who generates fake evidences. Furthermore, evidences are generated for service requests not for service consumption. Addressing privacy, a temporary user identity is assigned and sent securely to the user, but done by the FP, and hence, the real identity is known to the FP.

An additional scheme is proposed in [117] by using a combination of a digital signature and a one-way hash chain technique to provide non-repudiation of billing, when a mobile user roams into foreign networks. This scheme aims at improving the abovementioned non-repudiation mechanism by providing evidence to prove a service duration. Mobile users need to submit a digital signature when requesting a call and deliver chained hash values, one after the other in every interval during the session so that the call and its duration are undeniable. The scheme proposed in [117] is, however, limited to time-based accounting schemes. Moreover, the signature and verification keys to be used by the user and FPs respectively are generated by the HP. Therefore, a malicious HP is able to fake evidences.
2.5 Comparison and Conclusion of Related Work

Based on those presentations of related approaches above, the key task for a comparison consists in the definition of relevant and important metrics indicating those areas, which need to be tackled in auditing of today’s and tomorrow’s services. Thus, Table 2.5 compares different auditing related work described above with respect to those metrics.

The comparison shows whether an approach supports secure inter-domain auditing interactions, and whether an audit task is instantiated for each SLA or for each SLO. In general, the number of SLOs defined by a service provider is far smaller than the number of SLAs. For a service provider there are as many SLAs as customers, whereas the number of SLOs depends on the number of services and service classes. Table 2.5 also compares those types of expressions used to specify an SLO, thus, the complexity of an SLO that can be processed by the auditing component. Finally, the reusability of an auditing component or part of its functionality, if applied to another SLA or SLO, is compared. The reusability is low if the whole auditing component needs to be replaced to audit a new SLA or SLO. It is high if common auditing functionality can be reused.

Table 2.5: Comparisons of Auditing Related Work.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Inter-Domain Interaction</th>
<th>Security</th>
<th>Audit Task Instantiation</th>
<th>Complexity of SLO Term</th>
<th>Reusability</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSLA Framework [72]</td>
<td>Yes</td>
<td>Not considered</td>
<td>Per SLA</td>
<td>XML-based logical and relational expression</td>
<td>High</td>
</tr>
<tr>
<td>Cremona [73]</td>
<td>Yes</td>
<td>Not considered</td>
<td>Per WS-Agreement (SLA)</td>
<td>XML-based logical and relational expression</td>
<td>Low</td>
</tr>
<tr>
<td>Habib [37]</td>
<td>No</td>
<td>Considered within domain</td>
<td>For all SLO</td>
<td>Relational expression</td>
<td>Low</td>
</tr>
<tr>
<td>SLAng [96]</td>
<td>No</td>
<td>Not considered</td>
<td>Per SLA</td>
<td>OCL</td>
<td>Low</td>
</tr>
</tbody>
</table>
2.5. Comparison and Conclusion of Related Work

<table>
<thead>
<tr>
<th>Approach</th>
<th>Inter-Domain Interaction</th>
<th>Security</th>
<th>Audit Task Instantiation</th>
<th>Complexity of SLO Term</th>
<th>Reusability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moby Dick' SLA compliance auditing [45]</td>
<td>No</td>
<td>Not considered</td>
<td>Per SLO</td>
<td>Relational and logical expression</td>
<td>Low</td>
</tr>
<tr>
<td>Daidalos' SLA compliance auditing [39]</td>
<td>No</td>
<td>Not considered</td>
<td>Per SLO</td>
<td>XML-based relational expression</td>
<td>Low</td>
</tr>
<tr>
<td>AURIC (the work of this thesis)</td>
<td>Yes</td>
<td>Relevant mechanisms integrated</td>
<td>Per SLO</td>
<td>Control structures, relational and logical expression</td>
<td>High</td>
</tr>
</tbody>
</table>

The primary focus of existing approaches in SLA management is the formal specification of a complete SLA, where SLOs are part of the specification. Here, SLA compliance auditing is recognized as an important function in SLA management. However, in the approaches described, a measurement and an audit task need to be deployed for each SLA concluded between a provider and a customer. This is not efficient for a large number of customers. In general, a service provider has a small number of SLOs defined for all customers. Therefore, only one measurement and one audit task need to be deployed for each specific SLO defined.

Furthermore, a service may have several classes which are characterized by the same performance metrics but different target values. In the existing approaches, only relational and logical expressions can be used to express an SLO, thus each service class must be treated as a different service. Allowing a case differentiation to be expressed in an SLO would enable an audit task to handle several SLOs having the same mathematical formula but different numerical parameters. Indeed, an SLO can be arbitrarily complex, and each new service, especially an application service, may require new SLOs. This means, that a new auditing component is needed, or the employed auditing component must be generally applicable. Hence, *an auditing framework which provides common auditing functionality, reusable for developing an auditing component for each specific SLO, is desirable.*
Existing approaches are aware of the need of inter-domain interactions between various components of SLA management, however, no further considerations are made with respect to the impacts of allowing interactions across domains. These impacts include the security of the system which is now open to external access, as well as the reliability of the interactions. AU-RIC is supposed to support secure inter-domain interactions, be able to process complex SLOs, and highly reusable.

Concerning the second area of work addressed within this thesis, the support of non-repudiation, Table 2.6 summarizes and compares important related non-repudiation approaches with respect to the type of non-repudiation service provided and the support of mobility, privacy, fairness, and security.

*Table 2.6: Comparison of Different Non-repudiation Approaches.*

<table>
<thead>
<tr>
<th>Approach</th>
<th>Non-repudiation</th>
<th>Mobility</th>
<th>Privacy</th>
<th>Fairness</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.g., Zhou-Gollmann [116], Asokan, et al. [7], Zhou-Deng-Bao [113]</td>
<td>Communication event</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Supported (main focus)</td>
<td>Attacks are not considered</td>
</tr>
<tr>
<td>Hasselmeyer, et al. [49]</td>
<td>Lease request</td>
<td>Not considered</td>
<td>By encrypting all communication channels; by using proxy services between the client and the server</td>
<td>Not supported</td>
<td>Attacks are not considered</td>
</tr>
<tr>
<td>Lin-Harn [71]</td>
<td>Call request</td>
<td>Considered</td>
<td>Temporary user identities are generated by the FP; the real identity of a mobile user is known to the FP</td>
<td>Not supported</td>
<td>Malicious HP can generate undetectable fake evidences</td>
</tr>
</tbody>
</table>
As shown in Table 2.6, security and fairness support are missing almost always or are quite weak in existing systems. Furthermore, services considered are restricted to those, whose consumption is accounted for using the time-based scheme only. NorCIS* is able to protect service providers and users from various attacks and it is able to support fairness for service providers and users in service usage with item-based or time-based accounting scheme in a fair manner.
Chapter 3

Generic Model and Architecture

As seen so far, auditing is applied in various areas to examine compliance with requirements imposed on a system. With respect to Internet services, IDS and SLA compliance auditing are the two major areas of auditing. Although different areas of auditing follow different auditing objectives and have different tasks, they share a common model. Hence, this chapter presents the Auditing Framework for Internet Services’ (AURIC) general model and architecture for automated auditing [44], developed by the author of this thesis.

This chapter also develops a scenario for the application of SLA compliance auditing in a multi-domain environment, as well as presents a use case to show the instantiation of the generic architectural components.
3.1 AURIC’s Generic Model

Based on the general definition of auditing the AURIC’s generic model is proposed in this section. The model is important to understand the roles of involving parties, their interactions, and the types of information exchanged between them in an auditing process. Figure 3.1 depicts the class diagram of the generic auditing model in UML (Unified Modeling Language) notation leaving out details on attributes and operations [91]. The model consists of 8 classes divided into 3 roles (subjects), i.e., Auditor, Auditee, and Accountant, and 5 data classes (objects), i.e., Audit References, Compliance Specifications, Activities, Facts, and Audit Reports. Each class and its relationships with other classes in the model are described in detail in the next subsections.

**Legend**

![Legend Diagram](image)

**Figure 3.1: Generic auditing model.**

3.1.1 Model Description

The class Compliance Specifications is a key element in an audit. Compliance Specifications are a set of specifications derived from laws or regulations, contracts or agreements, pre-established policies, procedures, or other sources, which the particular system under audit, i.e., the auditee, has to follow. Note that “follow” means either to meet a specification to achieve an expected state or to avoid meeting a specification to not reach an unexpected
state. Each kind of source from which a Compliance Specification is derived can be considered to represent a specific auditing area.

An auditee carries out one or more activities to achieve a certain goal. The goal itself is irrelevant for this discussion, however, it is expected that in performing those activities, the auditee follows one or more Compliance Specifications. Therefore, those activities are observed by an accountant, who records and stores them as one or several Facts. A Fact is a piece of information presented as having an objective reality. A Fact can be accompanied by an evidence which furnishes a proof, i.e., that ascertains the truth of a matter, thus, increases the informational reliability of a Fact. Evidences are obtained technically through non-repudiation mechanisms, which can be used, e.g., to prove service consumption [48]. While Facts and evidences describe what actually happens, Compliance Specifications describe the required or expected situations. Facts about activities of an auditee reveal whether the auditee has followed the Compliance Specifications.

An auditor conducts an audit by evaluating Facts based on related Compliance Specifications to detect violations. A Fact may be audited by one or more auditors, and one auditor is responsible for auditing at least one Fact. An audit has to be conducted according to valid procedures, standards, laws, or regulations, which are collectively termed Audit References. Hence, Audit References define the legal or generally accepted way to conduct an audit in a particular area of auditing. While an auditee has to follow Compliance Specifications, an auditor has to follow Audit References. Thus, activities of an auditor can be subject to the audit by another auditor.

An auditor generates an Audit Report based on the result of an audit. In certain cases, Audit Reports may need to be generated only if Compliance Specifications are not followed by an auditee. This report gives statements about the (degree of) compliance (of the auditee’s activities) with respect to the pre-defined Compliance Specifications. As a matter of fact, the results of an audit can be considered as new Facts. However, the existence of these Facts is bound to the applied Audit References and Compliance Specifications. A change in Audit References or Compliance Specifications can result in different Facts. Hence, these derived Facts do not hold the same objective reality as the Facts that really occur. Nevertheless, an auditor can make use of Audit Reports of previous audit in conducting current audit. Furthermore,
Audit Reports can be consulted by an auditee to avoid carrying out activities which will result in failing to follow Compliance Specifications.

### 3.1.2 General Auditing Process

Having seen the generic model, the audit as a process is essential, mainly for identifying all types of inputs and outputs, and the steps required to process the inputs to obtain the output. The model identifies the Audit Report as the only output and four types of inputs to an audit, namely:

- **Facts** on activities carried out by an auditee.
- **Compliance Specifications**, *i.e.*, a set of specifications against which Facts are examined.
- **Audit References**, which an auditor has to follow.
- **Audit Reports**, *i.e.*, results of previous audits.

The following steps must be carried out by an auditing process consecutively to complete an audit:

1. Consult Audit References
2. Consult Compliance Specifications
3. Examine Facts and relevant Audit Reports

An auditing process is said to be general if it is capable of processing any Audit References. Normally, Audit References are not available in a language which allows for a computer-based processing. Audit References describe the way to conduct an audit, *i.e.*, how to interpret Compliance Specifications, Facts, and Audit Reports. The technical description of Audit References is called an *Audit Algorithm*. The structure and the *semantics* of Compliance Specifications, Facts, and Audit Reports are known to the audit algorithm. Note that in practically all auditing systems, the audit algorithm is an integral part of the auditing system, *i.e.*, it is not fed into the auditing system as input to be processed. Therefore, these systems begin with the second step above in the auditing process, *i.e.*, with the consulting of the Compliance Specifications. Compliance Specifications describe functions to evaluate Facts and relevant Audit Reports in order to determine compliances. These functions are applied in the last step of the auditing process.
3.1.3 Mapping the Model to Various Auditing Areas

In order to apply the generic model to SLA compliance auditing each generic class of the diagram in Figure 3.1 must be instantiated accordingly. As depicted in Figure 3.2, the auditee is represented by service provisioning entities who deliver services to each customer based on the SLA concluded with them. These activities are metered and accounted for by meter and accounting entities to produce accounting data. These data are then examined and compared to the respective SLA to check for compliances by an SLA compliance auditor. This compliance evaluation must follow a certain SLA audit procedure, and if there is a violation an SLA violation report will be generated.

Moreover, the generic model is also applicable to two well-known areas of auditing: intrusion detection and financial audit, as shown in Table 3.1.

Table 3.1: Application of the generic model to different auditing areas

<table>
<thead>
<tr>
<th>Model</th>
<th>Intrusion Detection</th>
<th>Financial Auditing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auditor</td>
<td>Intrusion detection engine</td>
<td>Auditor</td>
</tr>
<tr>
<td>Auditee</td>
<td>Users of network infrastructures and services</td>
<td>Company, including its accountants</td>
</tr>
</tbody>
</table>
### Table 3.1: Application of the generic model to different auditing areas

<table>
<thead>
<tr>
<th>Model</th>
<th>Intrusion Detection</th>
<th>Financial Auditing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accountant</td>
<td>Sensors (usage monitoring and logging entities)</td>
<td>Company’s accountants</td>
</tr>
<tr>
<td>Audit References</td>
<td>Intrusion detection methods</td>
<td>Auditing standards, \textit{e.g.}, Generally Accepted Auditing Standards (GAAS)</td>
</tr>
<tr>
<td>Compliance Specifications</td>
<td>Signature-based: intrusion rules, patterns (signatures) vs. anomaly-based: heuristic rules, normal states or behavior</td>
<td>Accounting standards, \textit{e.g.}, Generally Accepted Accounting Principles (GAAP)</td>
</tr>
<tr>
<td>Activities</td>
<td>Usage of network infrastructures and services</td>
<td>Accounting of business transactions, in particular generation of financial statements</td>
</tr>
<tr>
<td>Facts</td>
<td>Network traffic data and event logs</td>
<td>Financial statements</td>
</tr>
<tr>
<td>Audit Reports</td>
<td>Intrusion reports and alarms</td>
<td>Audit reports</td>
</tr>
</tbody>
</table>

### 3.2 AURIC’s Generic Architecture

Driven by the general model, the AURIC’s generic auditing architecture is designed, which is applicable to different auditing areas. The components of the generic architecture are supposed to be distributed across administrative domains, if required.

#### 3.2.1 Design

Figure 3.3 depicts the generic auditing architecture developed, having the Auditing Unit as a major component which contains a set of automated auditors. The Auditing Unit implements the audit algorithm of a particular auditing application. An auditee should follow Compliance Specifications in carrying out activities, hence, an auditee is divided into a Controlling Unit and an Executing Unit. The Controlling Unit defines activities to be carried out by the Executing Unit and makes sure that the Compliance Specifications are held. The Controlling Unit provides for Compliance Specifications,
whereas the Accounting Unit delivers Facts about activities performed by the Executing Unit to the Auditing Unit.

![Diagram](image)

**Figure 3.3: Generic auditing architecture.**

The result of an audit is a report with statements on the degree of compliance of auditee’s activities with respect to pre-defined Compliance Specifications. In certain cases, results of previous audits serve as an input to the current audit, therefore, the flow of Audit Reports between Auditing Unit and Report Handling Unit is bi-directional. The Report Handling Unit is responsible for maintaining Audit Reports.

The generic auditing architecture uses a policy-based approach to configure and control the behavior of all different units. Policy is an additional entity in the architecture which is not found in the model. It offers the advantage of being able to influence the behavior of the other units at operational time. As an example, Audit Policies can be defined to influence the behavior of an Auditing Unit in conducting an audit without violating Audit References (i.e., audit algorithm). Policies are defined and managed by a Policy Definition Unit.
In many cases, if the result of an audit reveals something that is unexpected, a pre-defined corrective action needs to be taken. The action is carried out either by the Report Handling Unit, Policy Definition Unit, or Controlling Unit. Therefore, Audit Reports are also accessible by the Policy Definition Unit and Controlling Unit (cf. Figure 3.3). There are three possible causes of a violation: (1) Inappropriately set up Audit Policies, (2) Imprecise Compliance Specifications, and (3) Executing Unit did not perform as expected. In general, cause (1) and (2) should not happen in normal operation, but can happen during learning or experimenting phase. Hence, the generic architecture foresees 3 different feedback control mechanisms to cope with violations:

- Feedback to the auditor through changing Audit Policies. This is useful, when violations have occurred unexpectedly due to Audit Policies being inappropriately set up. The Policy Definition Unit must be able to decide whether an Audit Policy is appropriately set up or not, which is a difficult task without human intervention.

- Feedback to the auditor through modifications of Compliance Specifications. This is useful, when violations have occurred unexpectedly due to imprecise Compliance Specifications. Here, the Controlling Unit must be able to decide, whether a Compliance Specification is precisely specified or not, which is also a difficult task without human intervention.

- Feedback to the Executing Unit by the Controlling Unit through reconfiguration. This should be the case if the Executing Unit did not perform as expected.

Obviously, the use of one mechanism does not preclude the use of other mechanisms at the same time because each mechanism serves a different purpose.

### 3.2.2 Support of Multi-Administrative Domain Environment

SLA compliance auditing is one of the auditing areas that deal with multi-administrative domain (AD) environment. Each provider or customer of Internet services constitutes or manages an administrative domain. Having a contract between them means that they share the same set of Compliance
Specifications. Audits can be performed in each administrative domain inde¬
pendently, but the application of the generic auditing architecture is not re¬
stricted to a single AD, instead, architectural components can be distributed
across several ADs.

Inter-domain auditing interactions are required, if data which are availa¬
ble in one domain, are needed by the other domain to complete an audit. For
example, an SLA between two providers P1 and P2 enables each of the pro-
viders to establish an SLA with a customer C and to specify parameters, e.g.,
one-way network delay, which covers the domain of both providers P1 and
P2. In order to audit the SLO committed to C, inter-domain interactions be-
tween P1 and P2 are required to transfer measurement data or audit results.

An AD can offer an accounting or auditing function as a service to other
ADs. In Figure 3.4, AD 1 provides an auditing service, whereas AD 3 offers
an accounting service. AD 2 uses the auditing service of AD 1, who further
makes use of the accounting service of AD 3. Here, AD 4 implements all the
functions, but still needs additional accounting information from AD 3 to
conduct an audit. Note that the Executing Unit and the feedback arrows are
not shown to avoid overloading the figure.

![Figure 3.4: Multi-administrative domain architecture](image)
Communicating information across domains should involve security mechanisms to protect information and infrastructure from attacks. The following subsections describe two major problems of a multi-domain support which must be considered.

**Open to Attack**

One of the consequences of supporting inter-domain interactions is the opening of the infrastructure to external accesses which may lead to attacks. An auditing infrastructure with distributed architectural components has to deal with the following threats:

- Data theft through eavesdropping or unauthorized access to Facts, Compliance Specifications, and Audit Reports.
- Data interception and manipulation by a Man-In-The-Middle attack which causes an Auditing Unit yielding wrong results.
- Denial-of-Service attack to various components providing a service (e.g., to an Auditing Unit providing an auditing service).

In order to protect an auditing infrastructure against the first two threats, accesses to data and a service must be controlled. In general, this can be achieved by employing a AAA infrastructure, e.g., using the AAA approach of IRTF AAAArch RG [50]. This requires an interaction of a service provisioning entity with AAA entities. Various sequences of interaction are possible as explained in [109], e.g., agent sequence, pull sequence, or push sequence. Furthermore, communication channels need to be secured. A secure communication channel (e.g., an IPsec security association on network layer) can be established as a result of an authentication process.

A AAA infrastructure is designed to authenticate users, to authorize and account usage of services by users [87]. In this respect, services are supposed to interact with a large number of users which are joining and leaving. However, most auditing applications have a pre-defined and very small set of interacting components which does not change during an audit. Besides, an auditing infrastructure is set up for a long time of operation. Here, authentication and authorization are accomplished once between these components. Therefore, security associations among the interacting components can be
pre-established to secure communications during auditing. In this case, inter-
actions with AAA entities are not needed.

To protect against a Denial-of-Service attack an Intrusion Protection Sys-
tem can be employed which is able to identify possible attacks and to block
the respective traffic.

**Higher Latency and Loss Rate**

Since inter-domain communications normally happen through the Inter-
net, latency and loss rate might be high. Hence, an Auditing Unit must con-
sider missing and delayed delivery of Facts and Audit Reports. To cope with
short time data loss, a reliable transport protocol is required if this is not to be
solved on the application layer. Data loss over a longer period can only be
handled on the application layer. Here, an Audit Policy and the audit algo-
ithm must decide on how to proceed. An offline auditing may postpone the
audit and restart it at a later time, whereas an online auditing may send an
alarm and keeps waiting for further inputs. A similar consideration is needed
for handling latency, in particular if online auditing is applied. Latency to a
certain value is tolerable, but latency above this value may be considered as
data loss. In offline auditing, higher latency is not crucial.

The impacts of higher latency and loss rate to auditing results are differ-
ent in online and offline auditing. Since an offline auditing can just be re-
started with the same data again and again, auditing results are not affected.
In online auditing, no result is obtained where inputs are missing. Results can
also be wrong if the audit algorithm is not appropriately defined to handle
delayed or missing inputs.

### 3.3 A Multi-Domain Application Scenario

This section presents a scenario where SLA compliance auditing is ap-
plied in a multi-domain environment. This scenario extends the one intro-
duced in Section 1.1. In this environment, a provider is able to define SLOs
across its domain based on SLAs with other providers. This section also de-
scribes the necessary inter-domain interactions in order to audit such SLOs.
3.3.1 Topology and Business Relation among Parties

Figure 3.5 depicts an example topology for an SLA compliance auditing in a multi-domain environment. It shows three NSPs who offer QoS-supported network connectivity services: SWITCH (the Swiss Academic and Research Network), TeliaSonera, and the American Telephone and Telegraph Corporation (AT&T). In this scenario, SWITCH is a customer of TeliaSonera who peers with AT&T. Thus, an SLA is established between SWITCH and TeliaSonera, as well as between TeliaSonera and AT&T. Furthermore, this example defines ETHZ, University of Zurich (UniZH), and University of Berne (Uni Bern) as customers of SWITCH, and Sun Microsystems Laboratories (Sun Labs) as a customer of AT&T. Each of these business relationships is also held in an SLA.

![Figure 3.5: A multi-domain scenario’s topology.](image)

In order to address auditing of services beyond network connectivity services, ETHZ is supposed to offer a series of high quality tutorials which are to be held in one of its lecture auditoria. However, the tutorials can also be visited remotely with the help of a tele-lecture system installed at ETHZ and participating remote locations. In this scenario, UniZH and Uni Bern provide for a lecture auditorium, equipped with the tele-lecture system, to allow for a remote participation to their students and to public. In this regard, UniZH and Uni Bern are considered customers of ETHZ. Since those tutori-
als are not free of charge, an SLA is concluded between ETHZ and each of the two universities. Here, ETHZ is responsible for generating content (audio, video, and data) with the necessary quality. Furthermore, some tutorials are given by researchers from various institutions around the world. However, they are not required to travel to ETHZ, instead, the tele-lecture system allows them to remotely give tutorials. In this scenario, one of the tutorials is supposed to be given by a researcher working at Sun Labs.

### 3.3.2 Services and SLOs

SLOs specified for a network connectivity service are different from those defined for a tele-lecture service. However, as shown later, they have dependencies. In the following, typical SLOs for network connectivity services are described first, then the ones needed for tele-lecture services.

#### Network Connectivity Service

Network connectivity services enable access to the Internet, or allow different sites of a customer to be inter-connected as an intranet. Following two SLO clauses are typical in an SLA for a network connectivity service [108]:

- The average delay measured monthly across the ISP network between any two access routers of the customer should be less than 200 ms.

- The customer will not have unscheduled connectivity disruption across the ISP network between any two access routers exceeding 5 minutes.\(^1\) Connectivity disruption is defined as 100% packet loss rate when measured by pinging an access router from a machine connected to another access router.

For example, the Internet SLA of Verizon Business covers availability, latency (round trip delay), network packet delivery, and network jitter guarantee [107]. Similar SLAs are defined by AT&T for their Internet businesses, e.g., the Managed Internet Service (MIS) [8]. Its SLOs include site availability, latency, jitter, data delivery, and VoIP call quality. Additionally, perform-

---

\(^1\) In fact, this statement can be misinterpreted, e.g., having a connectivity disruption of at most 4 minutes every half an hour can be argued as not violating the SLA. However, the real intention of this statement may be that within a certain period (e.g., a month) the total unavailability would not exceed 5 minutes.
ance statistics of each month for various SLA parameters are publicly accessible on the web [106], [9]. If the performance guarantees are not satisfied the companies offer to return parts of service charges [107], [8]. The methodology applied by AT&T for the measurement of its network performance is described in [10], [18].

Based on the above descriptions, Table 3.2 defines example SLOs to be used in this scenario for network connectivity services of AT&T, TeliaSonera, and SWITCH. Note that in this example, AT&T is supposed to provide network connectivity within United States (US), whereas TeliaSonera provides for transatlantic connections (London - New York and Copenhagen - New York) and network connectivity within Europe.

Table 3.2: Example SLOs for network connectivity services.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Example SLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T</td>
<td>Monthly average latency within US &lt; 40ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average jitter within US &lt; 1ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average packet loss rate within US &lt; 0.05%.</td>
</tr>
<tr>
<td></td>
<td>100% availability.</td>
</tr>
<tr>
<td>TeliaSonera</td>
<td>Monthly average latency: within Europe &lt; 30ms, transatlantic &lt; 100ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average jitter: within Europe &lt; 2ms, transatlantic &lt; 2ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average packet loss rate: within Europe &lt; 0.1%, transatlantic &lt; 0.1%.</td>
</tr>
<tr>
<td></td>
<td>Total unavailability less than 5 minutes in any monthly period.</td>
</tr>
<tr>
<td>SWITCH</td>
<td>Monthly average latency &lt; 25ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average jitter &lt; 1ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average packet loss rate &lt; 0.05%.</td>
</tr>
<tr>
<td></td>
<td>Total unavailability less than 5 minutes in any monthly period.</td>
</tr>
</tbody>
</table>

Suppose that the above SLOs of AT&T and TeliaSonera are part of their peering agreement, then TeliaSonera is able to offer following SLOs listed in Table 3.3 to SWITCH. The monthly average latency across AT&T’s network and TeliaSonera’s network is the sum of the average latency of each network. The monthly average jitter across both networks in worst case is the sum of the average jitter of each network. However, the monthly average packet loss rate across both networks cannot be appropriately determined using the individual SLOs. A rough upper bound of this total loss rate can be calculated which is equal to 1 - (1 - 0.05%) * (1 - 0.1%) * (1 - 0.1%), namely 0.2498%. But, this value is almost equal to the sum of all individual loss rates. To ob-
tain a more precise value both networks must be analyzed together or measurements across both networks must be performed. Similarly, based on these new SLOs established with TeliaSonera, SWITCH is able to offer SLOs which cover the network of TeliaSonera and AT&T as shown in Table 3.3.

Table 3.3: Defining SLOs based on SLOs with other providers.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Example SLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TeliaSonera</td>
<td>Monthly average latency: Europe - US &lt; 170ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average jitter: Europe - US &lt; 5ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average packet loss rate: Europe - US &lt; 0.25%.</td>
</tr>
<tr>
<td></td>
<td>Total unavailability less than 5 minutes in any monthly period.</td>
</tr>
<tr>
<td>SWITCH</td>
<td>Monthly average latency: Switzerland - US &lt; 195ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average jitter: Switzerland - US &lt; 6ms.</td>
</tr>
<tr>
<td></td>
<td>Monthly average packet loss rate: Switzerland - US &lt; 0.3%.</td>
</tr>
<tr>
<td></td>
<td>Total unavailability less than 10 minutes in any monthly period.</td>
</tr>
</tbody>
</table>

With respect to availability SLO, a satisfying value also cannot be obtained from the individual SLOs. The upper bound of the total unavailability value is again the sum of all individual values, although a situation where the end of unavailability of one network is immediately followed by the start of unavailability of the other network is seldom. Surely, overlapping unavailability can happen and the total unavailability time may exceed the values in the individual SLOs. In this scenario the worst case situation is used in defining SLOs for packet loss rate and unavailability across networks of several interconnected NSPs.

**Tele-Lecture**

As already mentioned, ETHZ is supposed to offer a series of high quality tutorials with the help of a tele-lecture system to allow for a remote participation of the audience as well as the lecturer. In this regard, ETHZ is considered as a content (multimedia data to be transmitted during a tutorial) provider. By requiring the tele-lecture infrastructure used by a remote lecturer to fulfil special requirements prescribed by ETHZ, ETHZ is able to control the quality of multimedia data generated on the lecturer side. Normally, a content or application service provider neither has control over the network performance nor the client side of communication. Thus, Table 3.4 lists those
application level SLOs that can be offered by ETHZ to its customers for a remote participation in a tutorial.

*Table 3.4: An example of application level SLO for a tele-lecture service.*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example SLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio sampling rate and encoding</td>
<td>G.711 [59]</td>
</tr>
<tr>
<td>Video resolution, frame rate, and encoding</td>
<td>Low: 352 × 288, 15 fps, H.264 [54] \Medium: 704 × 576, 15 fps, H.264 \High: 704 × 576, 25 fps, H.264</td>
</tr>
<tr>
<td>Data quality</td>
<td>Graphics resolution (of presentation slides) at least 1024 x 768</td>
</tr>
<tr>
<td>Availability</td>
<td>100% availability during tutorials.</td>
</tr>
</tbody>
</table>

Obviously, a high quality of multimedia data at the source can only be perceived as such by the remote audience if the underlying network is also able to provide for the required bandwidth to transfer the data and meet specific QoS requirements, in particular maximum allowable delay, jitter, and packet loss rate, as well as 100% network availability during the tutorial. Therefore, those network connectivity SLOs which are based on monthly average values, as described in the previous subsection, are generally insufficient to guarantee a good perceived quality of multimedia data transmitted during a tutorial. In many cases customers want to have SLOs defined, which cover guarantees of perceived quality, e.g., as specified in Table 3.5.

The perceived quality is specified in number as a Mean Opinion Score (MOS) which has a range from 1 (bad) to 5 (excellent) [57]. For speech quality, listening-only situations are distinguished from conversational situations, thus requiring different specifications: MOS-LQO (Listening Quality Objective) and MOS-CQO (Conversational Quality Objective) [55], [63].

*Table 3.5: Example SLOs for perceived quality of multimedia data.*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example SLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice quality</td>
<td>MOS-LQO in each tutorial session &gt; 4 \MOS-CQO in each tutorial session &gt; 3.5</td>
</tr>
<tr>
<td>Video quality</td>
<td>MOS in each tutorial session &gt; 3.5</td>
</tr>
</tbody>
</table>
In order to be able to offer these perceived quality guarantees, ETHZ should conclude an SLA with SWITCH that specifies the necessary QoS guarantees for the network traffic generated by the tele-lecture system. This requires SWITCH to identify ETHZ tele-lecture traffic flows and to make available the needed network resources accordingly. One way of doing this is by defining a network service class with its corresponding QoS guarantees, and assigning the tele-lecture traffic flows to this service class, e.g., by marking the flows with a specific Differentiated Service Code Points (DSCP). This identification allows for involving network components to treat the flows according to the SLOs.

Furthermore, network QoS guarantees must also be supported by TeliaSonera and AT&T in their SLAs, if the perceived quality guarantees given by ETHZ should also be valid when the tutorial is given by a Sun Labs researcher. Table 3.6 lists example SLOs for network QoS guarantees of a service class offered by each of the three NSPs. Note that QoS guarantees of a particular service class are NSP specific, and for the sake of simplicity the same service class (in this example the “Gold” service class) is applied to the tele-lecture traffic flows within each NSP network. It can also be seen in this example, that an NSP is able to offer SLOs which cover networks beyond its domain, based on its SLA with other NSPs.

Table 3.6: Example SLOs for network QoS guarantees of a service class.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Example SLO for Service Class “Gold”</th>
</tr>
</thead>
</table>
| AT&T              | **Within US:**  
|                   | • Maximum one way delay < 25ms.                                                                     |
|                   | • Maximum jitter < 3ms.                                                                             |
|                   | • Maximum packet loss rate < 0.1%.                                                                  |
|                   | • 100% availability.                                                                               |
| TeliaSonera       | **Within Europe:**  
|                   | • Maximum one way delay < 20ms.                                                                    |
|                   | • Maximum jitter < 3ms.                                                                             |
|                   | • Maximum packet loss rate < 0.15%.                                                                |
|                   | • 100% availability.                                                                               |
| **Europe - US**   | (based on the SLA with AT&T):  
|                   | • Maximum one way delay < 100ms.                                                                   |
|                   | • Maximum jitter < 10ms.                                                                           |
|                   | • Maximum packet loss rate < 0.4%.                                                                 |
|                   | • Total unavailability less than 5 minutes in any monthly period.                                   |
Table 3.6: Example SLOs for network QoS guarantees of a service class.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Example SLO for Service Class “Gold“</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCH</td>
<td>Within SWITCH network:</td>
</tr>
<tr>
<td></td>
<td>• Maximum one way delay $&lt; 15\text{ms}$.</td>
</tr>
<tr>
<td></td>
<td>• Maximum jitter $&lt; 3\text{ms}$.</td>
</tr>
<tr>
<td></td>
<td>• Maximum packet loss rate $&lt; 0.1%$.</td>
</tr>
<tr>
<td></td>
<td>• 100% availability.</td>
</tr>
<tr>
<td></td>
<td>Within Europe (based on the SLA with TeliaSonera):</td>
</tr>
<tr>
<td></td>
<td>• Maximum one way delay $&lt; 35\text{ms}$.</td>
</tr>
<tr>
<td></td>
<td>• Maximum jitter $&lt; 6\text{ms}$.</td>
</tr>
<tr>
<td></td>
<td>• Maximum packet loss rate $&lt; 0.25%$.</td>
</tr>
<tr>
<td></td>
<td>• 100% availability.</td>
</tr>
<tr>
<td>Switzerland - US</td>
<td>Switzerland - US (based on the SLA with TeliaSonera):</td>
</tr>
<tr>
<td></td>
<td>• Maximum one way delay $&lt; 115\text{ms}$.</td>
</tr>
<tr>
<td></td>
<td>• Maximum jitter $&lt; 13\text{ms}$.</td>
</tr>
<tr>
<td></td>
<td>• Maximum packet loss rate $&lt; 0.5%$.</td>
</tr>
<tr>
<td></td>
<td>• Total unavailability less than 5 minutes in any monthly period.</td>
</tr>
</tbody>
</table>

3.3.3 Measurement

An SLA must define the party who performs measurement tasks. In general, this is the provider, since it has control over the service provisioning infrastructure. However, a customer may also be the party who does the measurements. In fact, measurements may also involve a third party [35]. The main reason for having measurements done by a third party is objectivity (impartiality). In this scenario, the providers are supposed to perform the measurement tasks.

An SLA must specify in detail how a measurement of an SLA parameter is to be done. This includes the timepoints at which a performance measurement should take place, and the duration and sampling rate of each measurement. In each test cycle, current performance of a service must be determined, e.g., current packet loss rate, average and maximum values of latency and jitter within the duration of the test cycle.

For the purpose of SLA compliance auditing, active or passive measurement methods may be used as required by an SLA. Active measurements use probe traffic generated by the measurement system to obtain performance
values of a certain QoS parameter, whereas passive measurements calculate these values from existing customers’ traffic. Hence, performance statements obtained through active measurements are supposed to be valid for all customers having the same SLA. However, this may not coincide with the performance that a customer is experiencing, if the probes are not appropriately designed. In contrast, with more efforts passive measurements allow for a performance statement on a per customer basis. Though an active measurement may disturb existing traffic, its main advantage is the high flexibility in the design of probe streams to obtain properties that match specific measurement requirements [81].

### 3.3.4 Auditing

An SLA must also define the party who performs auditing. This is either the provider, the customer, or a third party. In this scenario, an SLO is supposed to be audited by its provider. For example, SWITCH is responsible for auditing its SLOs specified in the SLA with its customers: ETHZ, UniZH, and Uni Bern. And ETHZ is responsible for auditing its SLOs specifying perceived quality of the tele-lecture service.

With respect to auditing to be done by SWITCH, for those SLOs which cover TeliaSonera network, SWITCH needs to obtain measurement data from TeliaSonera. This requires inter-domain interactions between them. For monthly average guarantees, these interactions occur once a month. However, for guarantees on the maximum value of a particular SLA parameter to be below a certain threshold, network performance data of each test cycle needs to be considered. This leads to more frequent interactions if the test cycle is short, e.g., every five minutes. An inter-domain interaction is also required between ETHZ and SWITCH if ETHZ wants to assess perceived quality based on network performance data from SWITCH. Such interaction is not necessary if ETHZ deploys its own end-to-end performance measurement solution.

Based on the scenario just described, the feasibility of automated auditing approach is evaluated in Section 7.1 and Section 7.2. Furthermore, the next subsection presents a use case which discusses the application of AURIC’s generic architecture to SLA compliance auditing of network throughput guarantees.
3.4 A Use Case: Network Throughput Commitment

To show that the generic auditing architecture is indeed usable for SLA compliance auditing, a non-trivial use case is chosen and various components of the generic architecture are instantiated. Throughput is useful to specify the quality level of services, hence, network throughput commitment is chosen as a use case. Surely, the need for fine granular measurements of network throughput poses a problem to efficient measurements today, however, network service providers may need to measure bandwidth usage in order to be able to bill network traffic caused by content providers and users of peer-to-peer systems which consume a lot of bandwidth capacity [85].

Figure 3.6: Auditing of network QoS commitment.
3.4. A Use Case: Network Throughput Commitment

In this use case, an SLA contains amongst others: provider ID, customer ID, subscription start date, subscription end date, subscribed services and service classes, prices, SLOs, remedies in case of SLA violations, and limiting conditions. Information in an SLA is used for access control, meter configuration, and SLA compliance audit. The scenario in Figure 3.6 provides a basis for the detailed description on the application of the generic architecture to auditing of network QoS commitment. Provider P is a network operator providing QoS-enabled network services to its customers. The provider’s network is connected to the Internet via two Border Routers (BRs), and customers can access the Internet via one of the three Access Routers (ARs). Provider P offers different network service classes to meet different customer and application needs.

![Figure 3.7: Different traffic directions to be metered](image)

The network throughput parameter is part of SLAs between Provider P and its customers, and the level of this QoS parameter shall be committed. In order to be fully precise in the definition of this commitment, downlink and uplink traffic as well as incoming and outgoing traffic need to be distinguished as illustrated in Figure 3.7. The traffic coming from a terminal is the uplink traffic of this terminal and the traffic going to a terminal is the downlink traffic. Traffic entering the network is the incoming traffic, whereas traffic leaving the network is the outgoing traffic. Based on the above description two kinds of network throughput commitment can be defined:
downlink and uplink network throughput commitment. Each commitment is held in an SLO. Further discussion in this section concentrates on downlink traffic in order to be specific. Uplink traffic is treated in a similar way.

Suppose that the downlink incoming traffic from AR1, BR1, and BR2 are destined for the same terminal connected to AR3 as shown in Figure 3.7, and suppose also that the traffic are of the same service class. Since the SLOs of provider P are defined per service class, all downlink incoming traffic of the same service class for the same destination need to be aggregated. Ideally, this total downlink incoming traffic rate is equal to the downlink outgoing traffic rate. Assuming that the network infrastructure is not over-provisioned, congestion may happen due to network overload, thus leading to late arrival of packets or packet lost. This means that the downlink outgoing rate of the traffic at AR3 is less than the total downlink incoming rate of the traffic at AR1, BR1, and BR2.

Downlink network (aggregated edge-to-edge) throughput commitment is defined as follows: “The percentage rate reduction in downlink traffic of a network service class between the incoming and outgoing traffic will be within \( dR_d \) (downlink rate reduction tolerance) in every pre-defined Metering Interval (MI) during the whole session. The rate of the downlink incoming traffic used in the calculation is at most equal to the downlink committed rate \( R_{cd} \).” Note that this definition abstracts away the possible causes of the rate reduction, which may be due to packet losses or network delay. This commitment defines a condition which can be expressed mathematically using the formula given in (3.1). \( R_{od}(t) \) is the downlink outgoing rate, whereas \( R_{id}(t) \) is the downlink incoming rate. Values of \( R_{cd}, dR_d \), and metering interval are determined by the service class. This formula is applied to formally specify this example SLO used throughout this thesis.

\[
1 - \frac{R_{od}(t)}{\text{Min}(R_{cd}, R_{id}(t))} < dR_d
\]  

(3.1)

### 3.4.1 Executing Unit

The provider’s network infrastructure contains QoS-enabled routers and switches as the service provisioning entities for network services. They are
configured to deliver services to customers according to SLAs. Therefore, they constitute the Executing Unit in an SLA compliance audit.

### 3.4.2 Controlling Unit and Compliance Specifications

Provider P defines and manages SLAs with the help of an SLA Manager component. The SLA Manager is consulted by the Network Manager to configure the service provisioning entities to deliver services according to SLAs. In this respect, the SLA Manager and the Network Manager represent the Controlling Unit. In an SLA compliance audit, a Compliance Specification (CS) is derived from an SLO. An SLA may contain more than one SLO, hence, more than one CS. An SLA is valid for a customer and a provider specified in the SLA, however, a provider normally has only a limited set of types of SLA with certain SLOs, based on which each SLA is defined. Therefore, the same SLO may be valid for more than one SLA.

Obviously, an SLA contains more information than needed for defining a CS. Here, it is sufficient if a CS is composed of metrics, service class, and condition expression of an SLO. As Provider P only provides a single service, namely a network service, it is not required to have the name “network service” as subscribed services in the CS. Suppose that there are three service classes offered to customers: gold, silver, and bronze [19], with the values of $R_{cd}$, $dR_d$, and MI as specified in Table 3.7, then an example of a CS may read as follows:

* Specification Number: 001,
* Metrics: Downlink Throughput,
* Service Class: Bronze,
* Conditions: $1 - \frac{R_{od}(t)}{\min(R_{cd}, R_{id}(t))} < dR_d$

<table>
<thead>
<tr>
<th>Service Class</th>
<th>MI [secs]</th>
<th>$R_{cd}$ [Mbps]</th>
<th>$dR_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>10</td>
<td>8</td>
<td>0.001</td>
</tr>
<tr>
<td>Silver</td>
<td>30</td>
<td>4</td>
<td>0.005</td>
</tr>
<tr>
<td>Bronze</td>
<td>60</td>
<td>1</td>
<td>0.005</td>
</tr>
</tbody>
</table>

Table 3.7: Example of service classes offered by provider P.
CS 001 is valid only for customers of service class Bronze, but an explicit list of customer IDs in CS 001 is not needed due to the fact that this case does not deal with auditing of access control or intrusion detection, but with auditing of SLA compliance. Access control entities are responsible to ensure that only authorized customers can access services. Hence, recorded Facts are assumed to contain correct/authorized customer IDs.

CS 001 is a simple CS, where parameter Conditions consists of a single relational expression. Other CSs can be complex, where the parameter Conditions can contain several relational expressions linked by logical operators, and the evaluation is triggered by the occurrence of some other Facts.

### 3.4.3 Accounting Unit and Facts

A meter is deployed in each access router and border router to capture the data rate of customers' traffic. To allow for rate measurements of different service classes, the traffic of each service class must be marked accordingly, e.g., by using Differentiated Service Code Points (DSCP). For the examination of downlink throughput commitments the following information must be collected by meters: router ID, traffic direction, DSCP, destination IP address, meter interval start time, meter interval end time, and volume. The router ID reveals whether a router is a border or an access router. The traffic direction is either “incoming” or “outgoing”. Only traffic at network interfaces connected to other administrative domains are metered.

To ease auditing, related meter data need to be pre-processed by accounting entities. This pre-processing includes data aggregation from different meters and information mapping. In case of downlink throughput commitment, the resulting accounting record contains the following information: record type, customer ID, service class, meter time-stamp, average downlink incoming rate, and average downlink outgoing rate. The record type indicates the kind of information that is contained in the accounting record, in this case, average downlink rate. The service class is obtained from the DSCP. The customer ID of a downlink traffic is determined by the destination IP address of an outgoing traffic at an access router. The time interval and volume of this outgoing traffic determine the average downlink outgoing rate. The average downlink incoming rate is calculated by aggregating all in-
coming traffic to the same destination IP address at all other routers. Surely, aggregation is done per metering interval. Meter time-stamp is either the start time or the end time of a metering interval.

Obviously, meter data or accounting data represent Facts about service deliveries and their quality level. Hence, meters and accounting entities build up the Accounting Unit of the generic architecture. To avoid manipulation of meter data or accounting data by provider (or customer) the measurement and accounting task can be carried out by a third party.

### 3.4.4 Auditing Unit: Automated Auditors

SLA Compliance Auditors are automated auditors which have the task to evaluate whether the accounting data meet the SLOs specified in SLAs. Auditors deal with a set of CSs and a large number of accounting data recorded during service usage of a large number of customers. Therefore, it is reasonable to run a set of SLA Compliance Auditors, where each is responsible for a single CS and a fraction of the accounting data. In the example with Provider P, one way to divide the accounting data is by grouping them based on any combination of metrics, service class, and customer ID. For example, an SLA Compliance Auditor SCA1 is assigned to evaluate downlink throughput of traffic of service class Bronze addressed to customer with ID ranging from 101 to 200. This means, an Accounting Unit must be able to selectively distribute accounting data to each auditor.

SCA1 retrieves customer ID, downlink incoming rate $R_{i,d}(t)$, and downlink outgoing rate $R_{o,d}(t)$ from accounting data with parameters:

- **Metrics** = Downlink Throughput
- **Service Class** = Bronze
- **Customer ID between 101 and 200**

If there is a Fact with $R_{i,d}(t)$ and $R_{o,d}(t)$ values which do not satisfy the specified condition, a violation report is created. The violated SLA is found by matching the customer ID in the SLA with the customer ID of the violating accounting record.
3.4.5 Report Handling Unit and Audit Reports

Audit Reports are maintained and managed by an Audit Report Handler. Normally, only if there is a violation an Audit Report is generated, in which case the violated SLA is identified in the report. An Audit Report Handler is also responsible for determining the actions to be performed if there is a violation. Since this information is a part of an SLA, an Audit Report Handler needs to consult the SLA Manager.

3.4.6 Policy Definition Unit and Policies

Provider P operates various management tools to configure various units and defines policies to be executed by those units, hence, the management tools represent the Policy Definition Unit. One of the management tools is an Audit Task Planner which is used to define audit tasks to be carried out by SLA Compliance Auditors. Each audit task contains configuration items. An example for a configuration item is the communication interface of an SLA Compliance Auditor. An SLA Compliance Auditor communicates with an SLA Manager, a set of Accounting Units, and a set of Audit Report Handlers. Which Accounting Units and Audit Report Handlers have to be contacted depends on which accounting data and previous reports are to be retrieved and where to store all the resulting reports. Configuration parameters include a Uniform Resource Locator (URL), i.e., IP address or hostname, port number, and protocol. The following configuration example states that in order to obtain data on downlink throughput of all customers, the Accounting Unit running at host testbed_db.ethz.ch should be contacted via port 13000 using the Diameter protocol.

*Metrics:* Downlink Throughput
*Customer IDs:* ALL
*Accounting Unit:* diameter:testbed_db.ethz.ch:13000

Audit Policies are useful to influence the behavior of an SLA Compliance Auditor. For example, a policy can be defined for the SLA Compliance Auditor SCA1 to treat specific accounting records differently. Assume that Customer ID 113 is allowed to send traffic with a downlink committed rate 10% above the rate for service class Bronze during the month April 2005. In this regard, an Audit Policy for SCA1 reads as follows:
if (Customer ID = 113 and Meter Timestamp within April 2005) then use $Rc_d = 1.1$ Mbps.

Policies for other units may also be defined, if needed. For example, a meter policy for the Accounting Unit can be specified to adapt metering intervals to network load because the smaller the interval, the more meter data need to be transferred. However, the chosen interval may not lie outside of the range defined in the SLA.

### 3.5 Chapter Summary

AURIC’s generic auditing model, comprising of 3 roles and 5 data classes, has been designed and shown to be applicable to at least the following auditing areas: financial auditing, security audit, and SLA compliance auditing. Based on the generic model, AURIC’s generic auditing architecture has been developed whose components are distributed and may interact across administrative domains.

Furthermore, to illustrate SLA compliance auditing in a multi-domain environment, a scenario with several ISPs linked together by SLAs has been presented. In this respect, example SLOs are described for network connectivity services as well as for a tele-lecture service. The scenario shows the need for inter-domain interactions to transfer measurement data or auditing results. Finally, a use case in the area of SLA compliance auditing has been described in detail to show the application of AURIC’s generic architecture.
Chapter 4

Audit Task Decomposition and Specification

As identified in Section 2.5, an auditing framework which provides common auditing functionality, reusable for developing an auditing component for each specific SLO, is desirable. Therefore, in order to provide a foundation for designing a general applicable auditing framework, this chapter treats an audit as a function and analyzes its inputs and outputs (Section 4.1), determines the detailed sequence of subtasks that compose an audit task (Section 4.2), identifies the required function in each subtask (Section 4.3), as well as isolates those functions which are application-specific, thus, to be defined as *audit specifications* (Section 4.4).
This chapter also specifies the required language constructs to define an audit specification language (Section 4.5) and proposes an expressive language that meets these requirements (Section 4.6). Separating application specific functions from an audit function and specifying them as input to an auditing system are pre-requisites for the design of a general applicable auditing framework.

### 4.1 Audit Function

As already described, an audit is carried out by an Auditing Unit which implements the audit algorithm of a particular auditing application. An audit has been defined as a “systematic and independent examination of Facts on system activities to determine the degree of compliance with a pre-defined set of specifications.” The pre-defined set of specifications is called a Compliance Specification (CS). The result of an audit is stored in Audit Reports which may need to be consulted in the next audit. Therefore, an audit is mathematically a function with a CS, Facts, and previous Audit Reports as its input parameters and new Audit Reports as its results. This is shown as an equation in (4.1). The function $\text{audit}_0$ is termed Audit Function and contains the audit algorithm as illustrated in Figure 4.1.

$$\text{NewReports} = \text{audit}_0(\text{ComplSpec}, \text{NewFacts}, \text{PrevReports}) \quad (4.1)$$
Having a CS (ComplSpec) pre-defined, the audit function $\text{audit}_0$ is applied in each audit to the same CS, a new set of Facts ($\text{NewFacts}$), and all previous Audit Reports ($\text{PrevReports}$). The resulted new Audit Reports ($\text{NewReports}$) are added to previous Audit Reports for the next audit execution. However, the question arises as to whether the audit function $\text{audit}_0$ is applicable to any CS. In other words, does the audit algorithm need to be changed if it is applied to a different CS?

**Definition 4.1:** An audit function is *general* if a change to its inputs does not require a change to its audit algorithm. The audit algorithm of a general audit function is a *general audit algorithm*.

Therefore, if $\text{audit}_0$ is not a general audit function, its audit algorithm contains an input dependent part which needs to be converted into input specifications for the general part of the audit algorithm. This is illustrated in Figure 4.2.

![Figure 4.2: The general audit function $\text{audit}_g$](image)

**Definition 4.2:** An *Audit Specification* is a set of subspecifications which are needed by an Auditing Unit implementing a general audit algorithm to execute an audit.
If $audit_0$ is a general audit function, then a CS is an audit specification, otherwise, it is only a subset of an audit specification. Since Audit Reports can be seen as derived Facts, $PrevReports$ are treated in this chapter as a part of $NewFacts$ in an audit. From this point onwards, if previous Audit Reports are not explicitly mentioned, the term Fact(s) used in this thesis covers derived Fact(s). This simplifies (4.1) to:

$$NewReports = audit_0(ComplSpec, NewFacts) \quad (4.2)$$

The functionality to be implemented by the audit function $audit_0$ comprises subtasks of an audit to be carried out by an Auditing Unit. In order to determine these subtasks, knowledge about the information provided by inputs and outputs of an audit function is a pre-requisite. This is discussed in the following subsections.

### 4.1.1 Compliance Specifications

Each Compliance Specification defines the requirements for a particular Subject Matter (item) of Interest (SMI) to be audited. An example for an SMI in SLA compliance auditing is a certain QoS parameter in an SLA. The requirements imposed on a QoS parameter constitute the SLO of this QoS parameter. Therefore, an SLO is an example of a CS. In order to define requirements for an SMI, thus a CS, the following assumption is made:

**Assumption 4.1:** Each subject matter of interest is characterized by a set of properties, whose values are to be obtained from a list of Facts and are used to determine the degree of compliance with the requirements for that subject matter of interest.

Table 4.1 defines three useful examples of an SMI, its informal CS and properties which are used throughout this thesis:

**Table 4.1: Useful examples of an SMI, its informal CS and properties.**

<table>
<thead>
<tr>
<th>SMI (QoS Parameter)</th>
<th>CS (SLO)</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Service Breakdowns</td>
<td>In every year there are at most 3 breakdowns of service S which are longer than 1 hour.</td>
<td>Number of breakdowns of service S in a year which are longer than 1 hour.</td>
</tr>
</tbody>
</table>
Table 4.1: Useful examples of an SMI, its informal CS and properties.

<table>
<thead>
<tr>
<th>SMI (QoS Parameter)</th>
<th>CS (SLO)</th>
<th>Properties</th>
</tr>
</thead>
</table>
| Success of Service Requests | A service will be available (to the same user) in the next request if current request for this service has been rejected. However, the next request must be made within the next 5 minutes and 1 minute must have been elapsed since the rejected request.³                                                                 | This more complex SLO is defined by the following two properties:  
• Existence of an accepted request within 5 minutes after a rejected request.  
• Existence of requests after 1 minute since the rejected request.  
With the values of these two properties, compliance with the SLO can be determined.                                                                                                                                 |
| Downlink Throughput       | The percentage rate reduction in downlink traffic of a network service class between the incoming and outgoing traffic will be within downlink rate reduction tolerance in every pre-defined metering interval during the whole session.  
The rate of the downlink incoming traffic used in the calculation is at most equal to the downlink committed rate.  
The values of the downlink committed rate and the downlink rate reduction tolerance are determined by the service class.                                                                                     | The necessary properties to determine compliance with the SLO are:  
• Service class.  
• Average downlink incoming rate.  
• Average downlink outgoing rate.                                                                                                                                                                                   |

³ Note that this SLO can be seen as a more detailed way to express service availability.

The above examples show that a CS contains the following information:

- Information on relevant Facts, e.g., in the first SLO, relevant Facts are Facts about service breakdowns of a specific service S.
- Information on grouping of Facts, e.g., in the first SLO, Facts from different years must be processed separately.
- Information on compliance condition.
Suppose $S$ is an SMI to be audited and there are $m$ properties $P_{1S}, P_{2S}, \ldots, P_{mS}$ which are needed to describe $S$, then, mathematically, a compliance condition for $S$ describes a function $c_{pS}$ which transforms the values of the properties to a number $C$ representing the degree of compliance. This function is called \textit{property-based compliance function}, where the properties are variables of this function.

$$C = c_{pS}(P_{1S}, P_{2S}, \ldots, P_{mS}) \quad (4.3)$$

Based on the above description, a CS is necessary but not sufficient as an audit specification, since it does not define, e.g., how to obtain values of a property appeared in a compliance condition expression.

\subsection*{4.1.2 Facts and Audit Reports}

A Fact is a piece of \textit{information} presented as having an objective reality. Beside the objective reality, there are no other restrictions to information or types of information that can be provided by a Fact. Although there is also no restriction on what can be reported, an Audit Report usually stores the following information:

- Information that characterizes currently examined list of Facts.
- Information about the outcome of the compliance examination.
- Information that identifies the Audit Report for later references.

\textbf{Fact Generation}

Normally, an auditing application deals with several SMIs. In order to audit an SMI, a \textit{finite} number of related Facts about the SMI must be generated.

\textbf{Definition 4.3:} Facts are said to be \textit{related} if they relate to the same SMI.

\textbf{Definition 4.4:} A finite number of related Facts in a particular order is called a \textit{Fact-List}, \textit{i.e.}, a \textit{list of related Facts}. 


The typical order applied to a Fact-List is the *chronological order of occurrence*. However, this thesis does not preclude other possible orderings (e.g., ordering based on distance information), if required in an audit.

**Definition 4.5:** A Fact-List about a particular SMI is *complete* if all Facts within the list are necessary and sufficient to determine the degree of compliance of the Facts with the CS of that SMI.

Normally, different complete Fact-Lists about the same SMI are generated due to different settings or different time intervals. This means, that a complete Fact-List $FL$ can be seen as the result of an activity function of an auditee applied to a certain setting and time interval.

$$FL = \text{AuditeeActivity(setting, time interval)}$$ (4.4)

The parameter *setting* captures different “configuration parameters” which remain unchanged during an auditee’s activity, e.g., in SLA compliance auditing the identifier of the user requesting a service and the service requested lead to specific configuration of the service provisioning entities. The parameter *time interval* captures the dynamic conditions during the auditee’s activity, e.g., the current load of the service provisioning entities. These dynamic conditions cause different complete Fact-Lists to be generated for the same setting but at different time.

**Definition 4.6:** A set of all related Facts about a particular SMI is termed a *Fact Base* of that SMI and a set of all complete Fact-Lists about a particular SMI is termed a *Fact-List Base* of that SMI.

Each SMI determines its related Facts, however, this does not preclude a Fact or even a whole Fact Base to serve different SMIs. This implies that a Fact may be *audited several times* due to its usage in different SMIs. Principally, a Fact may belong to different complete Fact-Lists of the same SMI. This is possible, if a Fact is generated for different settings with overlapping time interval. This means, the same Fact may be *audited several times for the same SMI*. 
Fact Representation

In natural languages sentences are used to carry information. Consider the following sentence to express a Fact:

Flow F has a data rate of 1 Mbps.

The construction of a sentence obeys a grammatical structure (syntax) to allow for a correct interpretation. The following syntactical elements have a pre-defined and clear semantics in a natural language: subject, predicate, and object. The semantics\(^1\) of a syntactical element is bound to its name. The information units within the above Fact can be assigned to the syntactical elements as follows:

Subject: flow F
Predicate: has
Object: a data rate of 1 Mbps

This notation of expressing Facts conveys the same information as the sentence. In fact, through the decomposition it provides for easier analysis. This example also shows, that if the semantics of the syntactical elements is clear, a Fact can be expressed by giving only the “value” of its syntactical elements.

The above mentioned syntactical elements of a natural language can be used to capture a wide range of Facts. However, this also makes the processing of Facts difficult. In many applications, the range of expected Facts are restricted, hence, more specific syntactical elements can be used. Suppose that only Facts expressing data rate of flows are expected, then the syntactical elements Flow and DataRate can be defined, and the above Facts can be written as:

Flow: F
DataRate: 1 Mbps

Having the semantics of Flow and DataRate appropriately defined, the same information detail as when using subject, predicate, and object can be

---

\(^1\) To correctly interpret a term (in this case a syntactical element), a description of its semantics must be available to the processing element. In fact, such a description is only able to relate one term to another, and cannot really capture a human perception of the term which should be contained in a semantics.
achieved. Obviously, in order to be able to express a wide range of Facts using specific syntactical elements, the definition of any new syntactical element must be possible. This means a Fact is expressed by stating both the syntactical elements and their values. Based on this consideration, a Fact is expressed using a list of Attribute-Value-Pairs (AVPs)\(^1\), where each attribute represents a syntactical element and the order of the attributes in the list represents the order of the information units in a sentence. Since previous Audit Reports serve as an input to an audit function and are treated the same as Facts, it is reasonable to use the same representation for Audit Reports and Facts.

4.2 Audit Subtasks

Based on the above description about inputs and outputs of an audit function, the following sequence of subtasks are identified to be carried out for each SMI to be audited:

1. Facts filtering:
   In order to audit a specific SMI, only its related Facts are required. The task to obtain these related Facts is called filtering. For example, to audit a Service Breakdown SLO, Facts about downlink throughput are not required. As illustrated in Figure 4.3, only Facts about service breakdown are relevant and need to be processed.

2. Facts grouping:
   For each SMI, auditing is applied to each complete Fact-List of its Fact-List Base, e.g., Facts associated to one user and a service class in a particular time interval during a service consumption are audited separately from Facts associated to other users, other service classes, or other time intervals. The task to sort a list of related Facts to obtain a complete Fact-List for each setting and time interval is called grouping. Fact-Lists which are being built but not yet complete are called open Fact-Lists. Figure 4.4 illustrates the grouping of Facts related to a Service Breakdown SLO into four complete Fact-Lists.

---

\(^1\) A more precise term is Name-Value-Pair, since an attribute has a name and a value. However, since AVP is well-known and not absolutely incorrect, it is used in this thesis.
Chapter 4 Audit Task Decomposition and Specification

Figure 4.3: Illustrating Facts filtering for Service Breakdown SLO.

Figure 4.4: Illustrating Facts grouping for Service Breakdown SLO.
3. Property values calculation:
   This subtask determines the value of each property of the SMI from each complete Fact-List.

4. Compliance calculation:
   For each complete Fact-List, this subtask calculates from the property values the degree of compliance according to the compliance condition of the CS.

5. Report’s AVPs calculation:
   If an Audit Report is to be generated, this subtask determines the name and the value of each attribute of the Audit Report. Some systems may want to generate an Audit Report only if the auditee does not follow the CS. The inputs of this subtask may come from any output of the previous subtasks: Facts grouping, property values calculation, and compliance calculation.

6. Report generation:
   This subtask generates an Audit Report from the AVPs.

Figure 4.5 depicts the above sequence of subtasks and shows inputs and output of each subtask.

### 4.3 Auditability and Subtask Functions

Each of the subtasks identified in the previous subsection implements a specific functionality required in the audit of an SMI. The following notation is introduced to formulate auditability of an SMI mathematically:

- $P(M)$: Given a set $M$, $P(M)$ is a power set of $M$, i.e., the set of all subsets of $M$.
- $M_L$: Given a set $M$, $M_L$ is a list of elements of $M$ in a specific order (e.g., in chronological order of occurrence).
- $P_L(M)$: Given a set $M$, $P_L(M)$ is a set of all sublists of $M_L$. Each sublist must preserve the order in $M_L$, but two consecutive elements of a sublist need not be consecutive in $M_L$. 
Chapter 4  Audit Task Decomposition and Specification

Figure 4.5: The flowchart of audit subtasks.

Suppose $F$ is the set of all available Facts, $S$ is an SMI to be audited, $F_{FS}$ is the Fact Base of $S$, and $F_{S}$ is the Fact-List Base of $S$, then $S$ is *audit-able* if the following pre-condition and the conditions in each subtask are satisfied.
Pre-condition:

At least one complete Fact-List about $S$ exists.

$$\exists \ X \in P_L(\mathbb{F}) : \ X \in \mathbb{F}_S$$  \quad (4.5)

Facts Filtering Condition:

In order to obtain the Fact Base of $S$ from a list of all available Facts, a filter function $f_S$ is needed. Different types of filter functions are possible, e.g.:

- A function that takes a list of Facts as input and returns a sublist of the input as output.

$$\exists \ (f_S : P_L(\mathbb{F}) \rightarrow P_L(\mathbb{F}_S)) :$$

$$\forall \ Y \in P_L(\mathbb{F}) : \ Z = f_S(Y), \ Z \subseteq Y, \ Z \in P_L(\mathbb{F}_S)$$  \quad (4.6)

- A function that takes a Fact as input and returns a boolean value as output. A return value of true means that the Fact belongs to the Fact Base, otherwise the return value is false.

$$\exists \ (f_S : \mathbb{F} \rightarrow \{true, false\}) : \ \forall \ F \in \mathbb{F} :$$

$$f_S(F) = \begin{cases} 
true & \text{if } F \in \mathbb{F}_S, \\
false & \text{if } F \notin \mathbb{F}_S.
\end{cases}$$  \quad (4.7)

Facts Grouping Condition:

Each complete Fact-List about $S$ must be obtainable from the Fact Base of $S$. This requires the existence of a grouping function $g_S$. Different types of grouping functions are possible, e.g.:

- A function that takes a list of Facts as input and returns a set of complete Fact-Lists as output, where each complete Fact-List is a sublist of the input list of Facts.

$$\exists \ (g_S : P_L(\mathbb{F}_S) \rightarrow P(\mathbb{F}_S)) :$$

$$\forall \ Y \in P_L(\mathbb{F}_S) : \ Z = g_S(Y), \ Z \subseteq \mathbb{F}_S, \ \forall \ X \in Z : \ X \subseteq Y$$  \quad (4.8)

- A function that takes a Fact as input and returns a set of identifiers as output, where each identifier uniquely determines a Fact-List that the
Fact belongs to. Suppose that $\mathbb{I}$ is a set of identifiers and each Fact-List is uniquely identified by an element of this set, then
\[ \exists (g_s : F_{fS} \rightarrow P(\mathbb{I})) : \forall F \in F_{fS} : IDs = g_s(F), \quad IDs \in P(\mathbb{I}), \]
\[ (\forall ID \in IDs : (\exists X \in F_S : F \in X) \Rightarrow ID \text{ identifies } X) \]  
(4.9)

**Property Values Calculation Condition:**

The value of each property must be obtainable from each complete Fact-List about $S$. This requires the existence of a set of $m$ property functions $p_{1S}$, $p_{2S}$, ..., $p_{mS}$. The number of property functions is a property of $S$. Note that no assumption is made on the codomains of the property functions. This enables a wide range of property-based compliance functions to be defined.

\[ \forall i = 1..m : \exists (p_{iS} : F_S \rightarrow P_{iS}) : \]
\[ (\forall X \in F_S : P_{iS} = p_{iS}(X), \quad P_{iS} \in P_{iS}) \]  
(4.10)

**Compliance Calculation Condition:**

Obviously, the property-based compliance function $c_{pS}$ must exist. The codomain $\mathbb{R}$ is a set of numbers. Without losing generality, the range of $c_{pS}$ can be defined as all real numbers between and including 0 and 1.

\[ \exists (c_{pS} : P_{1S} \times P_{2S} \times \ldots \times P_{mS} \rightarrow \mathbb{R}) : \]
\[ \forall P_{1S} \in P_{1S}, P_{2S} \in P_{2S}, \ldots, P_{mS} \in P_{mS} : \]
\[ C = c_{pS}(P_{1S}, P_{2S}, \ldots, P_{mS}), \quad C \in \mathbb{R} \]  
(4.11)

**Report’s AVPs Calculation Condition:**

To generate AVPs for an Audit Report, a set of $k$ functions $a_{1S}$, $a_{2S}$, ..., $a_{kS}$ must exist. These functions are called report’s AVP functions and are needed due to two reasons:

- In many cases, an Audit Report requires additional information which cannot be provided by the property functions and the property-based compliance function involved in the audit. Therefore, report’s AVP functions are defined to generate this additional information.
• If the output of a certain function involved in the calculation of the degree of compliance, e.g., a certain property function, is to be stored in an Audit Report, the function needs to be explicitly selected. This is done by referring to this function in a report’s AVP function.

The inputs of a report’s AVP function may come from any output of the functions involved in previous subtasks: the currently examined Fact-List about $S$, the outcome of the property-based compliance function, or even the outputs of the property functions. Normally, a report’s AVP function is also needed to generate an AVP which reflects the fact that it deals with a specific SMI, in this case $S$. This function requires no input. Hence, the domain of a report’s AVP function is either an empty set, the Fact-List Base of $S$, any co-domain of the $m$ property functions, the codomain of the property-based compliance function, or any cartesian product of them. No further assumption is made on the codomain of a report’s AVP function, except that its elements must be AVPs.

**Report Generation Condition:**

To yield an Audit Report, a report generation function is needed to compose the Audit Report from the outputs of report’s AVP functions.

### 4.4 Audit Specifications

The auditability conditions show that six classes of functions are required for each SMI to be audited. However, a report generation function is not SMI-specific, i.e., it is applicable to any SMI. Therefore, for each SMI, only five classes of functions require a specification. The five specifications are as follows:

• Facts filtering function specification ($FFSpec$)
• Facts grouping function specification ($GFSpec$)
• Property functions specification ($PFsSpec$)
• Property-based compliance function specification ($PbCFSpec$)
• Report’s AVP functions specification ($AVPFsSpec$)
Suppose $n_{SMI}$ is the number of SMIs to be audited, then $R_i$, the resulted set of Audit Reports for SMI number $i$, is given by the following equation:

$$R_i = \text{audit}_i(\text{NewFacts}, \text{AVPFsSpec}_i, \text{PbCFSpec}_i, \text{PFsSpec}_i, \text{GFSpec}_i, \text{FFSpec}_i) \quad (4.12)$$

where

- $\text{FFSpec}_i$ is the specification of the filter function to obtain all related Facts about SMI number $i$ from $\text{NewFacts}$,
- $\text{GFSpec}_i$ is the specification of the grouping function to obtain all complete Fact-Lists about SMI number $i$ from its related Facts,
- $\text{PFsSpec}_i$ is the specification of the property functions for SMI number $i$,
- $\text{PbCFSpec}_i$ is the specification of the property-based compliance function for SMI number $i$, and
- $\text{AVPFsSpec}_i$ is the specification of the report’s AVP functions for SMI number $i$.

In the above equation each SMI has its own set of specifications. However, different SMIs may share some of the above mentioned functions, thus some specifications. Therefore, it is reasonable to define the following specifications to be shareable:

- Facts filtering function specification ($\text{FFSpec}$): a specification of a filter function to obtain a certain Fact Base from $\text{NewFacts}$.
- Facts grouping function specification ($\text{GFSpec}$): a specification of a grouping function to obtain complete Fact-Lists from a Fact Base.
- Property function specification ($\text{PFSpec}$): a specification of a property function.
- Property-based compliance function specification ($\text{PbCFSpec}$): a specification of a property-based compliance function.
- Report’s attribute function specification ($\text{AFSpec}$): a specification of a function which returns a report’s attribute value. The function is called a report’s attribute function. A set of $\text{AFSpecs}$ is a shareable part of a $\text{AVPFsSpec}$.
4.4. Audit Specifications

A set of each of these specifications is named accordingly, e.g., $FFSpecs$ is a set of $FFSpecs$. Note that the font of the last ‘s’ in the phrase “a set of $FFSpecs$” is a normal text font. A report’s attribute function needs to be defined only if an attribute value cannot be obtained from a property function or a property-based compliance function. Thus, a $PFSpec$ or a $PbCFSpec$ also serves as an $AFSpec$. Having these shareable specifications, an SMI number $i$ requires two additional specifications:

- $CCSpec_i$: Compliance calculation specification ($CCSpec$) for SMI number $i$, i.e., a specification to calculate the degree of compliance for SMI number $i$ by defining which $FFSpec$, $GFSpec$, $PFSpecs$, and $PbCFSpec$ to be used, and

- $RCSpec_i$: Report composition specification ($RCSpec$) for SMI number $i$, i.e., a specification that defines a list of attribute names which compose an Audit Report for SMI number $i$, and assigns to each attribute a specification which is used to generate the attribute values.

Figure 4.6 shows the use of those specifications in each audit subtask. A $CCSpec$ and an $RCSpec$ can be considered as specifications selectors. An $RCSpec$ also has a property of a switch to connect or disconnect certain input lines to report’s AVPs calculation subtask. The above consideration leads to the following equation for $R_i$:

$$R_i = audit_g(NewFacts, RCSpec_i, CCSpec_i, AFSpecs, PbCFSpecs, PFSpecs, GFSpecs, FFSpecs)$$

(4.13)

Using the term $AuditSpecs_i$ to denote $RCSpec_i$ and $CCSpec_i$ for SMI number $i$, and the term $CommonAuditSpecs$ to denote the entire $AFSpecs$, $PbCFSpecs$, $PFSpecs$, $GFSpecs$, and $FFSpecs$ (4.13) can be rewritten as:

$$R_i = audit_g(AuditSpecs_i, CommonAuditSpecs, NewFacts)$$

(4.14)

and (4.2) as:

$$NewReports = \bigcup_{i=1}^{n_{SMI}} R_i = \bigcup_{i=1}^{n_{SMI}} audit_g(AuditSpecs_i, CommonAuditSpecs, NewFacts)$$

(4.15)
Formula (4.15) states that to audit a set of SMIs requires the availability of Facts (about those SMIs), a set of audit specifications consisting of specifications common to some SMIs \(\text{CommonAuditSpecs}\) and SMI specific specifications \(\text{AuditSpecs}_i\), as well as an Auditing Unit implementing the general audit function \(\text{audit}_g\).

Figure 4.6: The use of audit specifications in audit subtasks

Figure 4.7 depicts the general audit function and the required specifications for auditing. Seven classes of audit subspecifications have been identi-
Audit specifications are classified as task specifications because they give instructions to perform an action, in contrast to property specifications which give information about the property of an object. There are two levels of a task specification: specification of what to do and specification of how to do. Having only a specification of what to do implies that the interpreter knows how to do the task. This level of specification allows a flexible implementation of a task. However, in many situations a specification of what to do is
not sufficient. Hence, a specification of how to do a task is also needed. Specifying how to do a task resembles programming. In this regard, an ASL takes to a certain degree the role of a programming language. The specification of how to do varies in its degree of detailing, depending on how much knowledge the interpreter already has. Principally, something is to be specified only if a different value of the specification can produce a different result of an audit. The following subsections discuss about the things that need to be specified. To allow for referring to a specification, each specification is given a unique identifier.

### 4.4.1 Facts Filtering Function Specification

A filter function enables the generation of a particular Fact Base from a list of available Facts. The Fact Base may be shared by several SMIs. The available Facts are obtained from accountants. The information about which accountants are to be contacted needs not be part of this specification, but considered to be part of the configurations of the module which interprets this specification.

A filter function defines filter criteria. Normally, filter criteria are based on information within and about current Fact being classified. The values of a Fact’s attributes are information carried within the Fact. An example for information about a Fact is the existence of a particular attribute. Some criteria may be based on time information, and their evaluation requires access to current time of audit. Filter criteria are complex if they require information from various Facts. Filter criteria should not be based on information from filtered Facts, i.e., Facts that have passed the filter criteria, since generally, they are not expected to be available all the time. Complex filter criteria lead to inefficient filtering.

Different types of filter functions are distinguished based on their input and output parameter. Table 4.2 discusses main types of filter functions. To allow for an efficient filtering, its criteria should only be based on information carried by the Fact being classified and information provided by the operating system. This leads to the following requirement which is easily met by a proper design of accountants:
**Requirement 4.1:** Each Fact must carry sufficient information to allow for classifying the Fact into certain Fact Bases without the need of information from other Facts.

*Table 4.2: Comparison of different types of filter functions*

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A list of all Facts</td>
<td>Boolean</td>
<td>Input and output do not fit.</td>
</tr>
<tr>
<td>A list of Facts</td>
<td></td>
<td>• Complex filter criteria can be specified using this type of filter function.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• This type of filter function is not suited for real time auditing, since all Facts must be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>collected before being filtered.</td>
</tr>
<tr>
<td>A list of n Facts</td>
<td>Boolean</td>
<td>Input and output do not fit.</td>
</tr>
<tr>
<td>A list of Facts</td>
<td></td>
<td>• No real advantages can be obtained through the use of this type of filter function except</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that it is more suitable for real time auditing than the one which requires a list of all Facts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>as input.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A filter function of this type must have a memory to keep Facts which cannot be processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yet.</td>
</tr>
<tr>
<td>A Fact</td>
<td>Boolean</td>
<td>• The output value denotes whether the input Fact belongs to a particular Fact Base or not.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Since only one Fact is processed at a time, this type of filter function is very well suited</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for real time auditing.</td>
</tr>
<tr>
<td>A list of Facts</td>
<td></td>
<td>• Returning a list of Facts as output for a Fact as input is only possible if the filter function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has a memory to keep previous Facts which couldn’t be processed before current input Fact is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>available. Hence, if the output value is an empty list, a further input Fact is required.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Complex filter criteria can be specified, but normally lead to less efficient filtering.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Also suited for real time auditing.</td>
</tr>
</tbody>
</table>

Based on the above descriptions the type of filter function having a single Fact as input and a boolean value as output is recommended, since it supports real time auditing and does not need to keep Facts. A unique Filter Function Identifier (FFId) is assigned to each specification of a filter function to allow for further references from other specifications.
4.4.2 Facts Grouping Function Specification

A grouping function enables the generation of all complete Fact-Lists from a Fact Base. A grouping function defines grouping criteria. In addition to information within and about current Fact being classified, the criteria may be based on information about an open Fact-List and information within and about any Fact of the open Fact-List. The number of Facts within a Fact-List is an example for information about the Fact-List.

There are two types of grouping criteria, namely criteria to determine group memberships of Facts, which are called group membership criteria, and criteria to determine whether a Fact-List is complete, which are called group completeness criteria. In principle, group completeness criteria are not necessary, since based on the group membership criteria all related Facts can be grouped and after that, all resulted Fact-Lists are complete. However, to allow for early detection of completeness, thus supporting real time auditing, group completeness criteria are required. Generally, group completeness criteria are based on a certain event, i.e., the occurrence of a certain Fact or the arrival of a certain timepoint. If the arrival of a certain timepoint determines the completeness of a Fact-List, periodic evaluation of the group completeness criteria and information on current time is needed.

Table 4.3 discusses main types of grouping functions which are distinguished based on their input and output parameter. The type of grouping function having a single Fact as input and a set of identifiers as output is recommended, since it does not need to keep Facts and can be extended to support real time auditing by defining additional constructs to specify group completeness criteria. To allow for an efficient grouping its criteria may not be based on information from complete Fact-Lists, since they are not expected to be available all the time, especially not after the audit. This leads to the following requirement:

**Requirement 4.2:** Each Fact of a certain Fact Base must carry enough information to allow for classifying the Fact into particular Fact-Lists of the corresponding Fact-List Base. The classification may not need any information from complete Fact-Lists. It may however use information from any open Fact-List.
### Table 4.3: Comparison of different types of grouping functions

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A list of Facts</td>
<td>A set of identifiers</td>
<td>A set of identifiers as output for a list of Facts as input is useless, unless the corresponding Fact-Lists are also returned as output.</td>
</tr>
</tbody>
</table>
|                        | A set of Fact-Lists     | • If the input list is the entire list of related Facts (the Fact Base), the resulted set of Fact-Lists contains the whole complete Fact-Lists of the Fact-List Base.  
  • If the input list is not the entire list of related Facts, the grouping function must keep Facts which cannot be processed yet for later function invocation, unless the output may contain open Fact-Lists.  
  • If the output may contain open Fact-Lists, they need to have identifiers for later references. This means, the output set must also contain an identifier for each open Fact-List. Moreover, open Fact-Lists need to be accessible from within the function to allow later determination of completeness.  
  • This type of grouping function can define complex grouping criteria, but is not well suited for real time auditing. |
| A Fact                 | A set of identifiers    | • Each identifier in the output set denotes a particular Fact-List to which the input Fact belongs. To obtain the Fact-Lists the input Fact of each function invocation must be collected according to the output set.  
  • Complex grouping criteria are possible if access to information about any Fact-List being built and information within and about any Fact of the Fact-List is allowed.  
  • A Fact-List can be considered complete only after all related Facts have been processed, unless an additional construct is provided to allow for early detection of completeness.  
  • This type of function is very well suited for real time auditing if an additional construct to evaluate completeness of a Fact-List is provided. |
| A set of Fact-Lists    |                         | • Assuming that normally a complete Fact-List consists of more than one Fact, this type of grouping function must have a memory to keep Facts unless the output may contain open Fact-Lists.  
  • If the output may contain open Fact-Lists, they need to have identifiers for later references. This means, the output set must also contain an identifier for each open Fact-List. Moreover, open Fact-Lists need to be accessible from within the function to allow later determination of completeness.  
  • This type of grouping function can define complex grouping criteria and is well suited for real time auditing. |
A unique Grouping Function Identifier (GFIId) is assigned to each specification of a grouping function to allow for further references from other specifications. Additionally, an identifier called Open Fact-List Identifier (OFLId) is assigned to each open Fact-List. The grouping function must ensure that this identifier is unique among the open Fact-Lists obtained from the same Fact Base. After an open Fact-List is complete its OFLId can be assigned to a new open Fact-List.

### 4.4.3 Property Function Specification

A specification of a property function describes the processing steps on a complete Fact-List to obtain a value of a property. It has a complete Fact-List as an input parameter. The processing steps are the algorithm of the applied property function. To describe these processing steps following information must be accessible:

- Information about current complete Fact-List being processed.
- Information about a particular Fact within this set.
- Information within a Fact of this set.

The complexity of a property function is largely reduced if each complete Fact-List is composed only of a single Fact and the value of the property can be directly extracted from the value of its attributes. This means, the complexity depends on how well the Facts are prepared by the accountant for auditing. In cases where Facts are not well prepared aggregation or statistical functions need to be applied to the attribute values of the Facts, e.g., summation, averaging, counting, and calculation of extreme values. Each specification of a property function is assigned a unique identifier called Property Function Identifier (PFId) to allow for later references.

### 4.4.4 Property-Based Compliance Function Specification

A specification of a property-based compliance function describes the algorithm, i.e., the processing steps on a set of property values to obtain a number representing the degree of compliance. The properties are the formal parameters of this specification. The number and the types of these parameters can vary among the specifications. The required operations include ma-
nipulation of property values and comparison of values with certain target values. In general, complex operations are not expected to be performed within this function, since the complexity of an audit should have been placed in the calculation of property values. Each specification of a property-based compliance function is assigned a unique identifier called Compliance Function Identifier (CFId).

4.4.5 Compliance Calculation Specification

A compliance calculation specification is defined for each SMI. This specification selects:

- One of the \textit{FFSpecs} by referring to its FFId, to be used to obtain the Fact Base.

- One of the \textit{GFSpecs} by referring to its GFId, to be used to obtain complete Fact-Lists from the Fact Base.

- Some of the \textit{PFSpecs} by referring to their PFIds, to be used to calculate property values from each complete Fact-List.

- One of the \textit{PbCFSpecs} by referring to its CFId, to be used to calculate the degree of compliance from the property values. Each formal parameter of the selected \textit{PbCFSpec} is bound to a certain \textit{PFSpec}.

The identifier assigned to this specification is called Compliance Calculation Identifier (CCId).

4.4.6 Report’s Attribute Function Specification

A specification of a report’s attribute function defines the processing steps to obtain a value of an Audit Report’s attribute. Each specification has its own number and types of formal input parameters. The currently examined complete Fact-List, a property value, or a compliance value can be an input to a report’s attribute function. Therefore, the required operations cover those identified for property functions and property-based compliance functions.

The complexity of a report’s attribute function varies. Some functions need only to generate a constant, others need to process the currently exam-
ined complete Fact-List. Each specification of a report’s attribute function is assigned a unique identifier called Report’s Attribute Function Identifier (AFId) to allow for later references.

### 4.4.7 Report Composition Specification

A report composition specification defines the attributes that compose an Audit Report. It defines the name of each attribute and assigns a specification to each attribute which is to be used to obtain the attribute values. Any of the following specifications can be assigned:

- **An AFSpec.**
  If the \textit{AFSpec} requires any inputs, its input parameters must be specified as well, which may include any specifications used in the related compliance calculation specification.

- **A PFSpec.**
  The input parameter of the \textit{PFSpec} must be the \textit{GFSpec} used in the related compliance calculation specification, hence, needs not be specified here.

- **A PbCFSpec.**
  This must be the same specification as used in the related compliance calculation specification.

Each SMI has its own report composition specification, which is assigned an identifier called Report Composition Identifier (RCId).

### 4.4.8 Examples

This section presents examples of \textit{informal} audit specifications using the three SMIs defined in Section 4.1.1. These informal audit specifications are presented in a tabular form where the text in the right column should be interpreted as follows, according to the entry in the left column:

- **CS:** The description of the SLO.

- **Properties:** The properties whose values are used to calculate the compliance value.

- **Fact’s Attributes:** The fields of accounting records.
• Facts Generation: The frequency of accounting records generation.
• Filter: The criteria for selecting accounting records related to the SLO.
• Group: The criteria to group accounting records.
• Property Function: The functions to calculate property values.
• Compliance Function: The function to calculate a compliance value.
• Interpretation of Compliance Value: Describes the value that determines violation to or compliance with SLO.
• Report’s Attributes: The fields of report records.
• Report’s Attribute Functions: The functions to calculate report’s attribute values.

Table 4.4: Audit specification of SMI 1 (Number of Service Breakdowns)

<table>
<thead>
<tr>
<th>CS</th>
<th>In every year there are at most 3 breakdowns of service S which are longer than 1 hour.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Number of breakdowns of service S in a year which are longer than 1 hour.</td>
</tr>
<tr>
<td>Fact’s Attributes</td>
<td>The following attributes are proposed to capture Facts for this SMI: Type, Timepoint, Service, and Duration. The Service attribute is assigned the value “S” and the Type attribute is assigned the value “Service Breakdown” to associate the Fact with this SMI.</td>
</tr>
<tr>
<td>Facts Generation</td>
<td>Each time there is a service breakdown.</td>
</tr>
</tbody>
</table>
| Filter | Criteria:
• The Type attribute must have the value “Service Breakdown”, the Service attribute must have the value “S”, and the Duration attribute must have a value greater than 1 hour. |
| Group | Group membership criteria:
• Related Facts having the same year in the Timepoint attribute are grouped together.
Group completeness criteria:
• All Fact-Lists with current year in the Timepoint attribute are open sets, i.e., incomplete.
• All Fact-Lists from previous years are considered complete. |
| Property Functions | A property function is needed to count the number of Facts. |
Table 4.4: Audit specification of SMI 1 (Number of Service Breakdowns)

<table>
<thead>
<tr>
<th>Compliance Function</th>
<th>If the property value is greater than 3 then the compliance degree is 0, otherwise, the compliance degree is 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpretation of Compliance Value</td>
<td>If the compliance degree is 0, then the CS is violated. If it is 1 then the CS is held.</td>
</tr>
</tbody>
</table>

**Report’s Attributes**

The following attributes are recommended for an Audit Report: **Type**, **Year**, **Service**, **NumberOfCriticalBreakdowns**, and **ComplianceValue**.

The Type attribute is assigned the value “Service Breakdown Audit Report” and the NumberOfCriticalBreakdowns attribute is assigned the value obtained from the property function, which is the number of breakdowns that are longer than 1 hour.

The value of the ComplianceValue attribute is obtained from the property-based compliance function.

**Report’s Attribute Functions**

The following report’s attribute functions need to be specified:

- A function which returns the value “Service Breakdown Audit Report”.
- A function to obtain the Year value from a complete Fact-List.
- A function to obtain the Service value from a complete Fact-List.

Table 4.5: Audit specification of SMI 2 (Success of Service Requests)

<table>
<thead>
<tr>
<th>CS</th>
<th>A service will be available in the next request if current request for this service has been rejected. However, the next request must be made within the next 5 minutes and 1 minute must have been elapsed since the rejected request.</th>
</tr>
</thead>
</table>
| Properties | P1 = Existence of an accepted request within 5 minutes after a rejected request.  
P2 = Existence of requests after 1 minute since the rejected request. |
| Fact’s Attributes | The following attributes are proposed to capture Facts for this SMI: **Type**, **UserID**, **Timepoint**, **Service**, **Result**.  
The value of the Type attribute is “Service Request” and the value of the Result attribute is either “Rejected” or “Accepted”. |
| Facts Generation | Each time there is a service request. |
| Filter | Criteria:  
- The Type attribute must have the value “Service Request”.

---

The Type attribute is assigned the value “Service Breakdown Audit Report” and the NumberOfCriticalBreakdowns attribute is assigned the value obtained from the property function, which is the number of breakdowns that are longer than 1 hour.

The value of the ComplianceValue attribute is obtained from the property-based compliance function.
### Table 4.5: Audit specification of SMI 2 (Success of Service Requests)

| **Group** | Group membership criteria: All related Facts having the same values of the UserID and the Service attribute are grouped together. Group completeness criteria: A group is complete if one of the following conditions is true.  
- Current Fact being grouped is a first Fact in the group and the Result attribute of this Fact has the value “Accepted”.  
- The Timepoint value of current Fact being grouped is within 5 minutes after the Timepoint value of the first Fact in the group, and the value of the Result attribute of current Fact is “Accepted” whereas the value of the Result attribute of all previous Facts in the group is “Rejected”.  
- The difference between current time and the value of the Timepoint attribute of the first Fact exceeds 5 minutes, and the value of the Result attribute of all Facts in the group is “Rejected”. |
|---|---|
| **Property Functions** | Facts in a group are available in order of occurrence. Two property functions are needed:  
- A property function to obtain P1: If the Result attribute of the last Fact has the value “Accepted”, then P1 is true, otherwise it is false.  
- A property function to obtain P2: If the difference between the values of the Timepoint attribute of the first and the last Fact is greater than 1 minute, then P2 is true, otherwise it is false. |
| **Compliance Function** | The formula for compliance determination is defined as follows:  
P1 == false and P2 == false => compliance degree = 1  
P1 == false and P2 == true => compliance degree = 0  
P1 == true and P2 == false => compliance degree = 1  
P1 == true and P2 == true => compliance degree = 1 |
| **Interpretation of Compliance Value** | A compliance degree of 1 means that the CS is held, otherwise it is violated. |
| **Report’s Attributes** | The following attributes are recommended for an Audit Report: **Type**, **UserID**, **Timepoint**, **Service**.  
The Type attribute is assigned the value “Service Request SLO Violation Report”. |
| **Reports Generation** | Only when there is a violation to the CS. |
| **Report’s Attribute Functions** | The following report’s attribute functions need to be specified:  
- A function which returns the value “Service Request SLO Violation Report”.  
- A function to obtain the UserID value from a complete Fact-List.  
- A function to obtain the Timepoint value from the first Fact in a complete Fact-List.  
- A function to obtain the Service value from a complete Fact-List. |
### Table 4.6: Audit specification of SMI 3 (Downlink Throughput)

| CS | The percentage rate reduction in downlink traffic of a network service class between the incoming and outgoing traffic will be within downlink rate reduction tolerance in every pre-defined metering interval during the whole session. The rate of the downlink incoming traffic used in the calculation is at most equal to the downlink committed rate. The values of downlink committed rate and downlink rate reduction tolerance are determined by the service class. |
| Properties | ServiceClass, AvgDownlinkIncomingRate, and AvgDownlinkOutgoingRate. |
| Fact’s Attributes | Following attributes are recommended to capture the related Facts: Type, ServiceClass, UserID, Timestamp, AvgDownlinkIncomingRate, and AvgDownlinkOutgoingRate. The value “Avg Downlink Rate” is assigned to the Type attribute. Each Fact carries information obtained from a metering interval. |
| Facts Generation | Periodically every metering interval. |
| Filter | Criteria: • All Facts having “Avg Downlink Rate” as the value of the Type attribute are related to this SMI. |
| Group | Group membership criteria: • The values of the ServiceClass, UserID, and Timestamp attribute determine the different Fact-Lists. Group completeness criteria: • Each Fact constitutes a complete Fact-List. |
| Property Functions | The values of each property can be directly extracted from the values of the respective attribute. Three property functions are required: • A property function which returns the value of the ServiceClass attribute. • A property function which returns the value of the AvgDownlinkIncomingRate attribute. • A property function which returns the value of the AvgDownlinkOutgoingRate attribute. |
4.5 Required Language Constructs for Audit Specifications

Audit subtasks of the examples in the previous section are informally specified. However, a formal specification is needed to allow for an automated auditing. Formal specifications can be written in a specific programming language (e.g., C/C++), which normally requires a gut programming skill. Hence, a language tailored for the specification of audit subtasks is highly desirable. To allow for the definition of such a language Table 4.7 lists language constructs needed to support each of the seven classes of audit subSpecifications.
**Table 4.7: Required language constructs for audit specifications**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Required Language Constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FFSpec</strong></td>
<td>An ASL must support:</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare a filter function.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to access information within and about current Fact being classified.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to specify current time of Audit.</td>
</tr>
<tr>
<td></td>
<td>Additionally, complex criteria may require further processing of the above mentioned information and specify relationships between those information. Therefore, a set of operators and functions to manipulate those information should be supported by an ASL.</td>
</tr>
<tr>
<td><strong>GFSpec</strong></td>
<td>An ASL must support:</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare a grouping function.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare group membership criteria.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare group completeness criteria.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to access information within and about current Fact being classified.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to specify current time of Audit.</td>
</tr>
<tr>
<td></td>
<td>To allow complex grouping criteria to be specified an ASL should support:</td>
</tr>
<tr>
<td></td>
<td>• A language construct to access information about an open Fact-List.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to access information within and about any Fact of an open Fact-List.</td>
</tr>
<tr>
<td></td>
<td>Additionally, complex criteria may require further processing of the above mentioned information and specify relationships between those information. Therefore, a set of operators and functions to manipulate those information should be supported by an ASL.</td>
</tr>
<tr>
<td><strong>PFSpec</strong></td>
<td>An ASL must support:</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare a property function.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to access information about current complete Fact-List being processed.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to access information within and about a particular Fact of current complete Fact-List being processed.</td>
</tr>
<tr>
<td></td>
<td>• A set of pre-defined operators and functions to process the above information.</td>
</tr>
<tr>
<td></td>
<td>• Various data types.</td>
</tr>
<tr>
<td></td>
<td>• A specific assignment statement to return a value of a certain data type as a result of the property function.</td>
</tr>
<tr>
<td></td>
<td>More complex language constructs, including sequences of statements, conditional branches, and iterations, as in a general purpose programming language, are required to specify complex processing steps. It should be noted that the purpose of an ASL is not to provide a programming language, therefore, Facts should be prepared for auditing whenever possible to reduce programming to the minimum.</td>
</tr>
</tbody>
</table>
### Table 4.7: Required language constructs for audit specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Required Language Constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PbCFSpec</strong></td>
<td>An ASL must support:</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare a property-based compliance function and its formal parameters.</td>
</tr>
<tr>
<td></td>
<td>• Various data types and a set of pre-defined operators and functions for these data types in order to allow processing of property values of various data types.</td>
</tr>
<tr>
<td></td>
<td>• Language constructs to specify different value of the degree of compliance depending on the processing result of the parameters. The required language constructs in this respect are conditional branches for distinguishing cases and specific assignment statements for returning values. Conditional branches contain relational expressions to specify conditions.</td>
</tr>
<tr>
<td></td>
<td>An ASL may support additional language constructs to allow specification of more complex processing steps:</td>
</tr>
<tr>
<td></td>
<td>• Assignment statements for variables.</td>
</tr>
<tr>
<td></td>
<td>• A sequence or a block of statements.</td>
</tr>
<tr>
<td></td>
<td>• Iterations.</td>
</tr>
<tr>
<td><strong>CCSpec</strong></td>
<td>An ASL must support:</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare a compliance calculation.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to refer to an FFSpec.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to refer to a GFSpec.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to refer to a set of PFSpecs.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to refer to a PbCFSpec and bind each of its formal parameter with a PFSpec.</td>
</tr>
<tr>
<td><strong>AFSpec</strong></td>
<td>In addition to language constructs that have been defined for PFSpec and PbCFSpec an ASL must support a language construct to declare a report’s attribute function and its formal parameters.</td>
</tr>
<tr>
<td><strong>RCSpec</strong></td>
<td>An ASL must support:</td>
</tr>
<tr>
<td></td>
<td>• A language construct to declare a report composition.</td>
</tr>
<tr>
<td></td>
<td>• A language construct to specify a set of attribute names and bind each name with a specification together with the actual parameters of this specification.</td>
</tr>
</tbody>
</table>

### 4.6 Sapta: an Audit Specification Language

Based on those requirements specified in the previous section, an audit specification language called Sapta (sapta is a Sanskrit word for seven, reflecting the seven classes of audit subspecifications) is defined. Compared to C/C++, the syntax of Sapta is simpler, since it is tailored to the specification of audit tasks. Hence, for automated auditors who need to interpret audit subspecifications at run time, a Sapta interpreter is easier to develop than a
C/C++ interpreter. Furthermore, the following principle is followed in the design of Sapta: The management (storage and transport) of Facts and Fact-Lists should be transparent to a programmer of an audit specification. Accesses to and manipulations of Facts and Fact-Lists are to be supported through specific language constructs.

This section describes the grammar and the semantics of Sapta’s main language constructs and presents examples of audit specifications in Sapta. The syntax is specified by means of a variant of Extended Backus-Naur Form (EBNF). Syntax notation and the complete grammar of Sapta is given in Appendix B. The syntax of an audit subspecification in Sapta is given by the following production:

\[<\text{AuditSubspecification}> ::= <\text{FilterFunctionSpec}> | <\text{GroupingFunctionSpec}> | <\text{PropertyFunctionSpec}> | <\text{ComplianceFunctionSpec}> | <\text{AttributeFunctionSpec}> | <\text{ComplianceCalculationSpec}> | <\text{ReportCompositionSpec}>\]

As defined in the above production, Sapta supports all seven classes of audit subspecifications (cf. Section 4.4 and Section 4.5). Each subspecification consists of a header and a body. The header contains a specific reserved word representing the class name to capture the usage intention of the subspecification. This is shown in the alternative production of an audit subspecification below:

\[<\text{AuditSubspecification}> ::= <\text{Header}> <\text{Body}> \]
\[<\text{Header}> ::= <\text{Class}> <\text{SpecId}> ["(" <\text{FormalParameters}> ")"]\]
\[<\text{Class}> ::= "\text{FilterFunction}" | "\text{GroupingFunction}" | "\text{PropertyFunction}" | "\text{ComplianceFunction}" | "\text{AttributeFunction}" | "\text{ComplianceCalculation}" | "\text{ReportComposition}"\]

\[<\text{SpecId}> ::= <\text{Identifier}>\]
\[<\text{FormalParameters}> ::= <\text{FormalParameter}> \{""," <\text{FormalParameter}>\}\]
4.6. Sapta: an Audit Specification Language

<FormalParameter> ::= <Identifier>

An audit specification in Sapta for a specific SMI comprises typically:

- A FilterFunction specification to define how to filter Facts.
- A GroupingFunction specification to define how to group Facts.
- A number of PropertyFunction specifications to define how to calculate the values of the properties.
- A ComplianceFunction specification to define how to calculate the degree of compliance.
- A ComplianceCalculation specification to select the FilterFunction specification, the GroupingFunction specification, the PropertyFunction specifications, and the ComplianceFunction specification, which are to be consulted in conducting an audit.
- A number of AttributeFunction specifications to define how to calculate the values of Audit Report’s attributes.
- A ReportComposition specification to define the names of the Report’s attributes and select the specifications which are to be consulted to obtain the value of each attribute.

Before describing each subspecification in detail, the language building blocks supported by Sapta are presented first.

4.6.1 Data Types and Constants

Sapta supports the following types of data: boolean, number (integer and real), string, list, fact, and factgroup. The four data types: boolean, number, string, and list are sufficient to capture most of attribute values a Fact normally has. The data type fact is used to represent a Fact, while the data type factgroup is used to represent a Fact-List. The elements of a list can be of any data type, except a fact or a factgroup. The elements of a list need not be of the same data type. This flexibility allows the use of a list to construct many other data structures. Data of type fact and factgroup can only be generated through specific expressions. Constants can be defined to express data of
type boolean, number, string, or list. A constant list is a list whose elements are constants. Table 4.8 shows some examples of constants in Sapta.

Table 4.8: Examples of constants in Sapta.

<table>
<thead>
<tr>
<th>Type of Constant</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>true</td>
</tr>
<tr>
<td>Number</td>
<td>5, 3e2, -2.5e-3</td>
</tr>
<tr>
<td>String</td>
<td>&quot;sapta&quot;</td>
</tr>
<tr>
<td>List</td>
<td>[&quot;Gold&quot;, [8, 0.001]]</td>
</tr>
</tbody>
</table>

Sapta also supports the notation to denote a physical quantity, i.e., a dimensional value (a number with unit). In any operation a dimensional value is involved, the respective dimensionless value is used after an implicit conversion to a pre-defined dimension. The dimensionless value must be a number, therefore a dimensional value is compatible to numbers. To allow this conversion to happen at parsing time and to manipulate the value at execution time without bothering about its unit, a dimensional value is not defined as a separate data type. In auditing application dealing with time information, e.g., in SLA compliance auditing, this notation provides a convenient way to express time interval. Sapta recommends a second as the predefined dimension to convert to, in case time unit is used.

```plaintext
<DimensionalValue> ::= 
  "" <DecimalNumberConstant> <Unit> ""

<Unit> ::= <Identifier>
```

4.6.2 Expressions

The purpose of an expression is to specify a formula. Building blocks of an expression consist of list expressions, fact expressions, factgroup expressions, operators, attribute value expressions, function calls, variables, and constants. Principally, an operator can be seen as a function call using a special pre-defined notation. The value of an expression is obtained through evaluation of the expression. In the evaluation, all variables are replaced by their values. A list expression evaluates to a value of type list. The syntax of a list expression is given by the following productions:
4.6. Sapta: an Audit Specification Language

<ListExpression> ::= <ListConstructionExpression> | <FunctionCall> | <Variable>

<ListConstructionExpression> ::= "["
[<ListElemExpression> {"," <ListElemExpression>}] ""]"

<ListElemExpression> ::= <SimpleExpression> | <RelationalExpression> | <ListExpression>

The evaluation of a fact expression results in a Fact, i.e., a value of type fact, whereas the evaluation of a factgroup expression results in a group of (related) Facts, i.e., a value of type factgroup. An attribute value expression is an expression to obtain the value of an attribute of a Fact. Fact expressions, factgroup expressions, and attribute value expressions are described in detail in Section 4.6.4 on filter function specification and Section 4.6.5 on grouping function specification.

4.6.3 Statements

Within a specification sequences of statements are specified whose syntax is given by the following production:

<StatementSequence> ::= <Statement> {";" <Statement>}

Note that a semicolon (";") is used in Sapta as a statement separator and not as a statement terminator. Sapta supports several types of statements including assignment statement, conditional branch, iteration, and I/O statement. Table 4.9 describes statements supported by Sapta.

<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Statement</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional branch</td>
<td>If statement</td>
<td>An if statement contains an expression, a sequence of statements which is executed if the expression evaluates to true, and an optional sequence of statements which is executed if the expression evaluates to false.</td>
</tr>
</tbody>
</table>
Table 4.9: Statements supported by Sapta

<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Statement</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment statement</td>
<td>Variable assignment statement</td>
<td>A variable assignment statement is used to assign the value of an expression to a variable. A variable is implicitly declared at the first time of usage, where a value is assigned to it in an assignment statement. The data type of the assigned value determines the data type of the variable. The scope of a variable is limited within each specification.</td>
</tr>
<tr>
<td>Return statement</td>
<td></td>
<td>A return statement is a special type of assignment statements which is used to return the value of an expression to a particular language construct.</td>
</tr>
<tr>
<td>Iteration</td>
<td>While statement</td>
<td>A while statement contains an expression and a sequence of statements which is executed as long as the expression evaluates to true.</td>
</tr>
<tr>
<td></td>
<td>Foreach statement</td>
<td>A foreach statement contains a variable, an expression, and a sequence of statements. The expression must evaluate to a list or a factgroup. The foreach statement iterates through each element of the list or each fact in a factgroup, and assigns in each iteration the next element to the variable. The sequence of statements is executed after the variable assignment in each iteration. The scope of the variable is limited to the foreach statement. In particular, it is accessible within the sequence of statements.</td>
</tr>
<tr>
<td>I/O statement</td>
<td>Print statement</td>
<td>Print a text. The destination is a configuration parameter of the interpreter.</td>
</tr>
</tbody>
</table>

4.6.4 FilterFunction Specification

A FilterFunction specification defines a sequence of statements to determine for a particular Fact Base whether a Fact belongs to the Fact Base. It has an implicit formal parameter of type fact which is omitted in the specification definition, but considered as the first formal parameter. The implicit formal parameter is named CurrentFact, which represents the Fact to be classified. Further formal parameters can be specified, if needed. The actual parameters are specified in ComplianceCalculation specifications. Table 4.10 describes the sequence of statements in detail. The syntax of a FilterFunction specification in Sapta is given by the following production:
<FilterFunctionSpec> ::= "FilterFunction" <SpecId> ["(" <FormalParameters> ")"]
{"<StatementSequence>
"}"

The <SpecId> following the reserved word FilterFunction is an FFId.

Table 4.10: Description of the statement sequence in a FilterFunction specification.

<table>
<thead>
<tr>
<th>Input</th>
<th>The formal parameters of the specification are the inputs to the sequence of statements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>The sequence of statements is executed if there is a new value of CurrentFact.</td>
</tr>
<tr>
<td>Result</td>
<td>An execution ends at a return statement. The expression in the return statement must evaluate to a boolean value, where a value of true means that the CurrentFact belongs to the Fact Base, whereas a value of false means the opposite.</td>
</tr>
</tbody>
</table>

Filtering of Facts requires access to the value of the attributes, which is expressed using <AttributeValueExpression>.

<AttributeValueExpression> ::= <FactExpression> "." <AttributeName>

The formal parameter CurrentFact is a FactVariable. To obtain information about a particular Fact a set of pre-defined functions is defined. Those functions include a function to determine whether an attribute exists in a Fact.

4.6.5 GroupingFunction Specification

A GroupingFunction specification defines sequences of statements to obtain Fact-Lists. It has an implicit formal parameter of type fact which is omitted in the specification definition, but considered as the first formal pa-
rameter. The implicit formal parameter is named \texttt{CurrentFact}, which represents the Fact to be classified. Further formal parameters can be specified, if needed. The actual parameters are specified in ComplianceCalculation specifications. The syntax of a \texttt{GroupingFunction} specification in Sapta is given by the following production:

\begin{verbatim}
<GroupingFunctionSpec> ::= \\
"GroupingFunction" <SpecId> ["(" <FormalParameters> ")"] \\
"{" \\
[ <GroupState> ] [ <On_Timer> ] <StatementSequence> \\
"}"

<GroupState> ::= "GroupState" \\
"{" \\
<VariableAssignmentStatement> \\
{";" <VariableAssignmentStatement>}
"}"

<On_Timer> ::= "On_Timer" <ScheduleList> \\
"{" \\
<StatementSequence> \\
"}"
\end{verbatim}

The \texttt{<SpecId>} following the reserved word \texttt{GroupingFunction} is a GFId. The \texttt{<StatementSequence>} immediately within this specification defines group membership criteria as well as group completeness criteria which are based on the occurrence of a Fact. This means, that the sequence of statements determines complete Fact-Lists due to \texttt{CurrentFact} and open Fact-Lists that \texttt{CurrentFact} belongs to. Table 4.11 describes this sequence of statements. Sapta supports an optional construct within this specification, namely an \texttt{On_Timer} construct, to define group completeness criteria which are based on the arrival of a certain timepoint. The timepoints are scheduled in \texttt{<ScheduleList>}. The sequence of statements of this construct is described in Table 4.12.

\textit{Table 4.11: Description of the statement sequence in a \texttt{GroupingFunction} specification.}

<table>
<thead>
<tr>
<th>Input</th>
<th>The formal parameters of the specification are the inputs to the sequence of statements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>Each time there is a new Fact to be grouped the sequence of statements is executed.</td>
</tr>
</tbody>
</table>
Table 4.11: Description of the statement sequence in a GroupingFunction specification.

| Result | An execution is terminated by a return statement, whose expression must evaluate to a list value. The list may contain any number of elements. An empty list as a return value is allowed and means that CurrentFact should be dropped, although it has passed the filter criteria. An element of the returned list is either of the following:
|        | • An OFLId: It identifies the open Fact-List to which CurrentFact belongs.
|        | • A list:
|        |   If this list is empty, it denotes that CurrentFact constitutes a new complete Fact-List, i.e., a complete Fact-List with CurrentFact as its only element.
|        |   If this list is not empty, its first element must be an OFLId which identifies an open Fact-List that is now complete, and its second element must be a boolean value. A value of false means that the Fact-List identified by the first element is complete if CurrentFact is added to the Fact-List, whereas a value of true means that the Fact-List is already complete.
|        | The following productions show the syntax of the evaluated return value:
|        | ReturnValue ::= "[" [ ReturnElem {"," ReturnElem} ] "]"
|        | ReturnElem ::= <OFLId> | <CompleteFL>
|        | CompleteFL ::= "[" [ <OFLId> "," <BooleanConstant> ] "]"
|        | <OFLId> ::= <StringConstant>

Table 4.12: Description of the statement sequence in an On_Timer construct

| Input | The formal parameters of the specification except CurrentFact are the inputs to the sequence of statements. Instead of CurrentFact this construct has an implicit formal parameter of type factgroup called CurrentGroupedFacts.
|       | The timepoints at which the sequence of statements of this construct is to be executed on each open Fact-List are specified in the expression <ScheduleList> which evaluates to a value of type list.
|       | <ScheduleList> ::= "[" <Start>, <Period> ["," <Stop>] "]"
|       | <Start> ::= <SimpleExpression>
|       | <Period> ::= <SimpleExpression>
|       | <Stop> ::= <SimpleExpression>
| Execution | The sequence of statements should be executed periodically starting at <Start> and at every <Period> after <Start>. The last time of execution is at <Stop>, if specified. The sequence of statements is executed on one Fact-List at a time which is referred to by CurrentGroupedFacts.
Table 4.12: Description of the statement sequence in an On_Timer construct

| Result | An execution is terminated by a return statement, whose expression must evaluate to a boolean value. A value of true means that the examined open Fact-List is complete, otherwise, it still misses some Facts. |

Grouping of Facts requires access to the value of the attributes, which is expressed using `<AttributeValueExpression>`.

```
<AttributeValueExpression> ::=  
    <FactExpression> "." <AttributeName>
```

```
<FactExpression> ::= <FactVariable>
    | <FactGroupExpression> "[" <n> "]"
```

```
<FactGroupExpression> ::= <FactGroupVariable>
    | "GroupedFacts" "{" <OFLId> "}"
```

```
<FactVariable> ::= <Identifier>
<FactGroupVariable> ::= <Identifier>
<AttributeName> ::= <Identifier>
<OFLId> ::= <SimpleExpression>
<n> ::= <SimpleExpression>
```

The formal parameter `CurrentGroupedFacts` is a FactGroupVariable. The reserved word `GroupedFacts` represents the list of all open Fact-Lists which is accessible to all sequences of statements in this specification. Each Fact-List in `GroupedFacts` can be accessed in a foreach statement as follows:

```
foreach g elem_of GroupedFacts do  
    ... // do something with g
endfor
```

The variable `g` in the above statement is a Fact-List, thus of type `factgroup`. Each Fact in a Fact-List, e.g., in `g` or in `CurrentGroupedFacts` can be accessed chronologically in a foreach statement as follows:

```
foreach f elem_of CurrentGroupedFacts do  
    ... // do something with f
endfor
4.6. Sapta: an Audit Specification Language

The value of an attribute of \( f \) in the above statement is accessed through the notation:

\[ f.\text{<AttributeName>} \]

Table 4.13 describes the possible notations to denote a particular Fact in addition to \text{CurrentFact}.

\textit{Table 4.13: Possible notations to denote a particular Fact in a Fact-List}

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{GroupedFacts{&lt;OFLId&gt;}[&lt;n&gt;]}</td>
<td>This notation denotes a particular Fact within an open Fact-List identified by \text{&lt;OFLId&gt;}, an expression that must evaluate to a string.</td>
</tr>
<tr>
<td>\text{CurrentGroupedFacts[&lt;n&gt;]}</td>
<td>In \text{On_Timer} construct to denote a specific Fact within current Fact-List. Note that \text{CurrentGroupedFacts} is an implicit formal parameter in this construct.</td>
</tr>
</tbody>
</table>

The value of \(<n>\) defines a specific Fact within a Fact-List according to Table 4.14:

\textit{Table 4.14: Interpretation of index \(<n>\) of a FactGroupExpression}

<table>
<thead>
<tr>
<th>Value of (&lt;n&gt;)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Most current Fact within the Fact-List.</td>
</tr>
<tr>
<td>(&gt; 0)</td>
<td>Fact number (&lt;n&gt;) in the Fact-List according to the chronological order of occurrence.</td>
</tr>
<tr>
<td>(&lt; 0)</td>
<td>The (</td>
</tr>
</tbody>
</table>

All variables defined within a sequence of statements are accessible only within that sequence of statements. They are released after each execution of the sequence of statements and are redefined in the next execution. To allow certain values to be kept for next executions, if necessary, special variables called state variables can be defined within an optional construct called \text{GroupState}. Each Fact-List has its own set of state variables. All state variables of a Fact-List will be initialized when the Fact-List is created. Obviously, to be useful these state variables must be accessible within the
sequences of statements in this specification. The following notation is used to access a state variable:

\[
\text{<FactGroupExpression> "." <StateVariable>}
\text{<StateVariable> ::= <Identifier>}
\]

To obtain information about a particular Fact-List a set of pre-defined functions is defined. Those functions include functions to determine the number of Facts within a Fact-List, and the identifier of a Fact-List.

### 4.6.6 PropertyFunction Specification

A PropertyFunction specification defines a sequence of statements to calculate a property value from a complete Fact-List. The sequence of statements expresses the algorithm of a property function. It has an implicit formal parameter of type factgroup which is omitted in the specification definition, but considered as the first formal parameter. The implicit formal parameter is named CurrentGroupedFacts which represents the complete Fact-List to be processed. Further formal parameters can be specified, if needed. The actual parameters are specified in ComplianceCalculation specifications. The syntax of a PropertyFunction specification in Sapta is given by the following production:

\[
\text{<PropertyFunctionSpec> ::=}
\]

\[
\text{"PropertyFunction" <SpecId> ["(" <FormalParameters> ")"]}
\]

\[
\text{"{"}
\text{<StatementSequence>}
\text{"}"}
\]

The <SpecId> following the reserved word PropertyFunction is a PFId. The sequence of statements in this specification is executed on each complete Fact-List. A statement gains access to an attribute value of a particular Fact in the current Fact-List by using the following notation:

\[
\text{"CurrentGroupedFacts" "}\text{<n> "}\text{." <AttributeName>}
\]

The number <n> has the same interpretation as specified in Table 4.14. The value of the property function is returned using a return statement. The data type of the return value must be boolean, number, string, or list. Note
that, a return value of type list may not contain element of type fact or factgroup.

4.6.7 ComplianceFunction Specification

A ComplianceFunction specification defines a sequence of statements to obtain a compliance value from a set of property values. The sequence of statements expresses the algorithm of a property-based compliance function. Each relevant property must be specified as a formal parameter of the function. Further formal parameters can be specified after the property parameters in the parameter list. The actual parameters are specified in ComplianceCalculation specifications. The syntax of a ComplianceFunction specification in Sapta is given by the following production:

\[
<\text{ComplianceFunctionSpec}> ::= \\
"\text{ComplianceFunction}" <\text{SpecId}> "(" <\text{FormalParameters}> ")" \\
"\{" \\
<\text{StatementSequence}> \\
"\}" \\
\]

The \(<\text{SpecId}>\) following the reserved word ComplianceFunction is a CFId. The statement sequence in this specification is executed on each set of property values. All property values in a set are obtained from the same complete Fact-List. The execution of the statement sequence ends at a return statement. The evaluation of its expression yields the compliance value.

4.6.8 ComplianceCalculation Specification

A ComplianceCalculation specification defines a sequence of specifications to be used to determine the degree of compliance of Facts against the CS of a specific SMI. Each of the specifications is referred to by its identifier. The syntax of a ComplianceCalculation specification in Sapta is given by the following production:

\[
<\text{ComplianceCalculationSpec}> ::= \\
"\text{ComplianceCalculation}" <\text{SpecId}> \\
"\{" \\
<\text{SpecificationSequence}> \\
"\}" \\
\]
<SpecificationSequence> ::=  
  <GFInputSpec>  
  ">>" <PFInputSpec>  
  ">>" <CFInputSpec> \{"," <CFInputSpec>\}  
  ">>" <CFId> ["(" <ActualConstParams> ")"]  

<GFInputSpec> ::= <FFId> ["(" <ActualConstParams> ")"]  
<PFInputSpec> ::= <GFId> ["(" <ActualConstParams> ")"]  
<CFInputSpec> ::= <PFId> ["(" <ActualConstParams> ")"]  

<FFId> ::= <SpecId>  
<GFId> ::= <SpecId>  
<PFId> ::= <SpecId>  
<CFId> ::= <SpecId>  
<ActualConstParams> ::= <Constant> \{"," <Constant>\}  

The <SpecId> following the reserved word ComplianceCalculation is a CCId. Since FFIds, GFIds, PFIds, and CFIds are all specification identifiers, the above syntax cannot prohibit illegal sequences of specifications. However, based on the information about the specification class that a specification identifier refers to, a parser can generate error messages. Suppose that:

- FF1 identifies the FilterFunction specification,
- GF1 identifies the GroupingFunction specification,
- PF1, PF2, and PF3 identify three PropertyFunction specifications, and
- CF1 identifies the ComplianceFunction specification,

which are to be used to calculate degrees of compliance for an SMI, then the following <SpecificationSequence> is to be specified in the ComplianceCalculation specification:

FF1 >> GF1 >> PF1, PF2, PF3 >> CF1

The results of FF1 are forwarded to GF1 and each result obtained from GF1 serves as input to PF1, PF2, and PF3. The results of the three property functions are the inputs to CF1.
4.6.9 AttributeFunction Specification

An AttributeFunction specification defines a sequence of statements to obtain a report’s attribute value. The sequence of statements expresses the algorithm of a report’s attribute function. Certain functions do not require any parameters. If a formal parameter is specified, it can be of type factgroup, or any possible type of a property value or a compliance value. The actual parameters are specified in ReportComposition specifications. The syntax of an AttributeFunction specification in Sapta is given by the following production:

```
<AttributeFunctionSpec> ::= 
  "AttributeFunction" <SpecId> ["(" <FormalParameters> ")"] 
  
  <StatementSequence> 
  
"
```

The <SpecId> following the reserved word AttributeFunction is an AFId. In each execution of the sequence of statements the value of each parameter comes from the same complete Fact-List. The attribute value is obtained from the return statement which ends the execution of the sequence of statements.

4.6.10 ReportComposition Specification

A ReportComposition specification defines a list of attribute names and assigns to each attribute a sequence of specifications to be used to obtain the attribute value. Each of the specifications is referred to by its identifier. The resulted AVPs are used to compose an Audit Report for a specific SMI. The syntax of a ReportComposition specification in Sapta is given by the following production:

```
<ReportCompositionSpec> ::= 
  "ReportComposition" <SpecId> 
  
  <AVPSpecs> 
  
"
```

```
<AVPSpecs> ::= 
  <AVPSpec> 
  
  <AVPSpec> {"," <AVPSpec>}
```

The `<SpecId>` following the reserved word `ReportComposition` is an RCId. The rightmost `<SpecId>` in the production of `<SpecSeqForAttribute>` is either an AFId, a PFId, or a CFId, whereas a `<SpecId>` in the production of `<InputSpec>` is either a GFId, a PFId, or a CFId. If a GFId, a PFId, or a CFId is selected as a `<SpecId>` in a `ReportComposition` specification, then it must also be selected in the corresponding (related to the same SMI) `ComplianceCalculation` specification and its actual parameters are implicitly specified. Suppose that an Audit Report of an SMI is composed of the following attributes:

- Type, whose value to be obtained from AF1,
- UserID, whose value to be obtained from the sequence GF1 >> RAF2,
- Prop, whose value to be obtained from the sequence PF1, PF2 >> AF3, and
- Compliance, whose value to be obtained from CF1,

then the following `<AVPSpecs>` is to be specified in the `ReportComposition` specification:

```
[Type eq AF1],
[UserID eq GF1 >> AF2],
[Prop eq PF1, PF2 >> AF3],
[Compliance eq CF1]
```

In the above example, GF1, PF1, PF2, and CF1 must be specified in the corresponding `ComplianceCalculation` specification.
4.6.11 Pre-defined Functions

A list of pre-defined functions to be supported by Sapta is described in Table 4.15. This list is not supposed to be complete, but it should cover most of the functions needed by an audit specification. New functions can be introduced and their support requires an extension of the respective Sapta interpreter.

*Table 4.15: Pre-defined functions supported by Sapta*

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substr(S1, S2)</td>
<td>Return true if S1 is a substring of S2, return false otherwise. S1 and S2 are strings.</td>
</tr>
<tr>
<td>InsertElem(E, L, P)</td>
<td>Insert the element E to the list L at position P. If P is a negative number, add the element E to the end of the list L. Return L.</td>
</tr>
<tr>
<td>DeleteElem(L, P)</td>
<td>Delete the element at position P from the list L. If P is a negative number, delete the last element from the list L. Return the deleted element.</td>
</tr>
<tr>
<td>DeleteAllElem(L)</td>
<td>Delete all elements of L. Return an empty list L.</td>
</tr>
<tr>
<td>ViewElem(L, P)</td>
<td>Return the element at position P in the list L. If P is a negative number, return the last element in the list L.</td>
</tr>
<tr>
<td>ListLength(L)</td>
<td>Return the length of the list L.</td>
</tr>
<tr>
<td>Subset(L1, L2)</td>
<td>Return true if L1 is a subset of L2, return false otherwise. L1 and L2 are lists (ordered sets) which are treated as sets.</td>
</tr>
<tr>
<td>Elem(E, L)</td>
<td>Return true if E is an element of the list L, return false otherwise.</td>
</tr>
<tr>
<td>InsertMapElem(M, K, V)</td>
<td>M is a list of [Key, Value]. Insert [K, V] into M at any position if K does not yet exist as a Key in M, otherwise overwrite its corresponding Value with V.</td>
</tr>
<tr>
<td>ViewMapValue(M, K)</td>
<td>Return the Value if K is a Key in M, otherwise return an empty list.</td>
</tr>
<tr>
<td>DeleteMapElem(M, K)</td>
<td>Delete the element of M with K as a Key. Return the deleted element or an empty list.</td>
</tr>
<tr>
<td>MinElem(L)</td>
<td>Return the element with the smallest value. All elements of L are of the same type, either boolean, number, or string. L may not be empty.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MaxElem(L)</td>
<td>Return the element with the greatest value. All elements of L are of the same type, either boolean, number, or string. L may not be empty.</td>
</tr>
<tr>
<td>AvgElem(L)</td>
<td>Return the average value of all elements of L. All elements of L are numbers. L may not be empty.</td>
</tr>
<tr>
<td>Time(S)</td>
<td>Convert string S containing time information into a number representing number of seconds since epoch. Return the result of the conversion.</td>
</tr>
<tr>
<td>Now()</td>
<td>Return current time as number of seconds since epoch.</td>
</tr>
<tr>
<td>GetYear(T)</td>
<td>Return the year of T. T is a number representing number of seconds since epoch.</td>
</tr>
<tr>
<td>GetMonth(T)</td>
<td>Return the month of T. T is a number representing number of seconds since epoch.</td>
</tr>
<tr>
<td>GetDay(T)</td>
<td>Return the day of T. T is a number representing number of seconds since epoch.</td>
</tr>
<tr>
<td>GetHour(T)</td>
<td>Return the hour of T. T is a number representing number of seconds since epoch.</td>
</tr>
<tr>
<td>GetMinute(T)</td>
<td>Return the minute of T. T is a number representing number of seconds since epoch.</td>
</tr>
<tr>
<td>GetSecond(T)</td>
<td>Return the second of T. T is a number representing number of seconds since epoch.</td>
</tr>
<tr>
<td>AttributeExist(F, A)</td>
<td>Return true if there is an attribute named A in Fact F, otherwise return false. A is a string. F is a fact.</td>
</tr>
<tr>
<td>GroupExist(Id)</td>
<td>Return true if there is an open Fact-List with Id as OFLId, otherwise return false. Id is a string.</td>
</tr>
<tr>
<td>GetGroupSize(G)</td>
<td>Return the number of Facts within a Fact-List G. G is a factgroup.</td>
</tr>
<tr>
<td>GetGroupId(G)</td>
<td>Return the OFLId of a Fact-List G. G is a factgroup.</td>
</tr>
<tr>
<td>MinAttrVal(G, A)</td>
<td>Return the minimum value of all attributes named A in all Facts of Fact-List G. A is a string. G is a factgroup.</td>
</tr>
<tr>
<td>MaxAttrVal(G, A)</td>
<td>Return the maximum value of all attributes named A in all Facts of Fact-List G. A is a string. G is a factgroup.</td>
</tr>
</tbody>
</table>
Table 4.15: Pre-defined functions supported by Sapta

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvgAttrVal(G, A)</td>
<td>Return the average value of all attributes named A in all Facts of Fact-List G. A is a string. G is a factgroup.</td>
</tr>
<tr>
<td>Composeld(...)</td>
<td>Compose and return a unique OFLId from the input parameters. Any number and types of parameters are allowed. The return value is a string.</td>
</tr>
<tr>
<td>Min(...)</td>
<td>Return the minimum of all parameters. All parameters are of the same type, either boolean, number, or string.</td>
</tr>
<tr>
<td>Max(...)</td>
<td>Return the maximum of all parameters. All parameters are of the same type, either boolean, number, or string.</td>
</tr>
<tr>
<td>Avg(...)</td>
<td>Return the average of all parameters. All parameters are numbers.</td>
</tr>
</tbody>
</table>

4.6.12 Examples

The expressiveness of Sapta is demonstrated by formally defining audit specifications for the three examples (cf. Section 4.4.8) in Sapta.

Audit Specifications for SMI “Number of Service Breakdowns”

FilterFunction FF_SvcBreakdown (Service, Threshold) {
    return (CurrentFact.Type = "Service Breakdown")
    and (CurrentFact.Service = Service)
    and (CurrentFact.Duration > Threshold)
}

GroupingFunction GF_SvcBreakdown {
    NewId := Int2Str(GetYear(CurrentFact.Timepoint));
    if ListLength(GroupedFacts) > 0 then
        Id := GetGroupId(GroupedFacts[1]);
        if NewId > Id then
            return [ [Id,true], NewId ]
        else
            return [ Id ]
        endif
    else
        return [ NewId ]
    endif
In the GroupingFunction the year value of the Timepoint attribute is used as OFLId. The group completeness criteria (NewId > Id) is useful to allow early detection of completeness if auditing is applied to data collected for quite a while, in this case over years. By applying this criteria, only one open Fact-List can exist at any time, since as soon as a Fact with a newer year value occurs, the currently open Fact-List is complete. However, if auditing is applied with no data in the beginning, then this leads to inefficient grouping, since completeness evaluation on Facts within the same year is unnecessary. An On_Timer construct is more suitable in this case. Assuming that current year is 2006, the next completeness evaluation will be January 1st, 2007, as shown in the On_Timer construct within the following alternative specification of the GroupingFunction:

```java
GroupingFunction GF_SvcBreakdown {
    On_Timer ["01-01-2007", '1 year'] {
        return GetYear(Now()) > Str2Int(GetGroupId(CurrentGroupedFacts))
    }
    return [ Int2Str(GetYear(CurrentFact.Timepoint)) ]
}

PropertyFunction PF_NOfCriticalBreakdowns {
    return GetGroupSize(CurrentGroupedFacts)
}

ComplianceFunction CF_SvcBreakdownSLO ( NOfSvcBreakdowns, Threshold ) {
    if NOfSvcBreakdowns > Threshold then
        return 0
    else
        return 1
    endif
}

ComplianceCalculation SBreakdownCompliance {
    FF_SvcBreakdown("S", '1 hour')
    >> GF_SvcBreakdown
    >> PF_NOfCriticalBreakdowns
    >> CF_SvcBreakdownSLO(3)
}

AttributeFunction AF_SvcBreakDownType {
    return "Service Breakdown Audit Report"
}
```
AttributeFunction AF_Year (FG) {
    return GetYear(FG[1].Timepoint)
}

AttributeFunction AF_Service (FG) {
    return FG[1].Service
}

ReportComposition SBreakdownReport {
    [Type eq AF_SvcBreakDownType],
    [Year eq GF_SvcBreakdown >> AF_Year],
    [Service eq GF_SvcBreakdown >> AF_Service],
    [NumberOfCriticalBreakdowns eq PF_NOfCriticalBreakdowns],
    [ComplianceValue eq CF_SvcBreakdownSLO]
}

Audit Specifications for SMI “Success of Service Requests”

The definition of the filter function is straightforward as shown below.

FilterFunction FF_SvcReq {
    return CurrentFact.Type = "Service Request"
}

In order to write the GroupingFunction specification the group completeness criteria have to be studied carefully. A new Fact $F_n$ which would belong to a Fact-List with id as OFLId may cause the Fact-List to be complete. Figure 4.8 presents the three completeness criteria: C1, C2, and C3, which are described below:

- C1: If the new Fact is a first Fact and the value of its Result attribute is "Accepted", then this Fact constitutes a complete Fact-List. Therefore, the return value of the grouping function is [[]]. An OFLId for this complete Fact-List is not necessary.

- C2: If the new Fact is not the first Fact of a Fact-List with id as OFLId, but it occurs within 5 minutes after the first Fact of this Fact-List and the value of its Result attribute is "Accepted", then the new Fact completes the Fact-List. Hence, the return value is [[id, false]]. id identifies a complete Fact-List because it is an element of a list within a list. The boolean value false indicates that the new Fact belongs to the complete Fact-List.
• C3: If the new Fact occurs after 5 minutes have elapsed since the occurrence of the first Fact of the Fact-List with Id as OFLId, then the Fact-List is complete and the new Fact belongs to a new Fact-List with the same OFLId. This new Fact-List is also complete if the value of the Result attribute of the new Fact is "Accepted".

C1: \( F_n.\text{Result} = \text{"Accepted"} \)

C2: \( F_n.\text{Timepoint} - F_1.\text{Timepoint} \leq 5 \) and \( F_n.\text{Result} = \text{"Accepted"} \)

C3: \( F_n.\text{Timepoint} - F_1.\text{Timepoint} > 5 \) if \( F_n.\text{Result} = \text{"Accepted"} \) then return \( [[\text{Id}, \text{true}], []] \) else return \( [[\text{Id}, \text{true}], \text{Id}] \)

Figure 4.8: Completeness criteria for SMI “Success of Service Request”

GroupingFunction GF_SvcReq (Threshold) {
    Id := ComposeId(CurrentFact.UserID, CurrentFact.Service);
    if GroupExist(Id) then
        if CurrentFact.Timepoint - GroupedFacts{Id}[1].Timepoint > Threshold then
            if CurrentFact.Result = "Accepted" then
                return \( [[\text{Id}, \text{true}], []] \)
            else
                return \( [[\text{Id}, \text{true}], \text{Id}] \)
            endif
        else
            if CurrentFact.Result = "Accepted" then
                return \( [[\text{Id}, \text{false}] ] \)
            else
                return \( [[\text{Id}] ] \)
            endif
        endif
    else
        if CurrentFact.Result = "Accepted" then
            return \( [[\text{Id}, \text{true}], []] \)
        else
            return \( [[\text{Id}, \text{false}] ] \)
        endif
    endif
}
4.6. Sapta: an Audit Specification Language

In normal cases, the above GroupingFunction specification will produce a lot of complete Fact-Lists consisting of a single Fact whose Result attribute has the value "Accepted". What makes this example special is the fact that two of the completeness criteria (C1 and C2) reveal compliances. Since an evaluation of the complete Fact-Lists obtained through C1 and C2 always yields no SLO violation and the fact that no Report is to be created in case of no SLO violation, it is more efficient to drop these Fact-Lists before they undergo an evaluation. By eliminating each empty list from the elements of the return value in the above specification, the new Fact can be dropped, thus avoid the creation of a complete Fact-List due to C1. This version of Sapta does not support dropping of open Fact-Lists.

PropertyFunction PF_Accepted {
    return CurrentGroupedFacts[0].Result = "Accepted"
}

PropertyFunction PF_RequestExist (Threshold) {
    return CurrentGroupedFacts[0].Timepoint -
        CurrentGroupedFacts[1].Timepoint > Threshold
}

ComplianceFunction CF_SvcReqSLO (Accepted, HasRequest) {
    if not Accepted and HasRequest then
        return 0
    else
        return 1
    endif
}

ComplianceCalculation SvcReqCompliance {
    PF_SvcReq
    >> GF_SvcReq('5 minute')
    >> PF_Accepted, PF_RequestExist('1 minute')
AttributeFunction AF_SvcReqType {
    return "Service Request SLO Violation Report"
}
AttributeFunction AF_UserID (FG) {
    return FG[1].UserID
}
AttributeFunction AF_Timepoint (FG) {
    return FG[1].Timepoint
}
AttributeFunction AF_Service (FG) {
    return FG[1].Service
}
ReportComposition SvcReqReport {
    [Type eq AF_SvcReqType],
    [UserID eq GF_SvcReq >> AF_UserID],
    [Timepoint eq GF_SvcReq >> AF_Timepoint],
    [Service eq GF_SvcReq >> AF_Service]
}

Audit Specifications for SMI “Downlink Throughput”
FilterFunction FF_DlinkThroughput {
    return CurrentFact.Type = "Avg Downlink Rate"
}
GroupingFunction GF_DlinkThroughput {
    return [[]]
}
PropertyFunction PF_ServiceClass {
    return CurrentGroupedFacts[0].ServiceClass
}
PropertyFunction PF_AvgDlIncomingRate {
    return CurrentGroupedFacts[0].AvgDownlinkIncomingRate
}
PropertyFunction PF_AvgDlOutgoingRate {
    return CurrentGroupedFacts[0].AvgDownlinkOutgoingRate
}
ComplianceFunction CF_DlThroughputSLO
    ( SvcClass, AvgDIR, AvgDOR, SvcTable ) {
    V := ViewMapValue(SvcTable, SvcClass);
    if ListLength(V) != 0 then

DlCommittedRate := ViewElem(V,1);
DlRateRedTol := ViewElem(V,2);
if 1 - AvgDOR / Min(DlCommittedRate, AvgDIR) < DlRateRedTol then
    return 1
else
    return 0
endif
else
    print "SLO unspecified for Service Class ";
    print SvcClass;
    return 0
endif

ComplianceCalculation DlThroughputCompliance {
    FF_DlinkThroughput
    >> GF_DllinkThroughput
    >> PF_ServiceClass, PF_AvgDlIncomingRate, PF_AvgDlOutgoingRate
    >> CF_DlThroughputSLO ( [
        ["Gold", [8, 0.001]],
        ["Silver", [4, 0.005]],
        ["Bronze", [1, 0.005]]
    ]
    )
}

AttributeFunction AF_DlThroughputType {
    return "Downlink Throughput SLO Violation Report"
}

AttributeFunction AF_UserID (FG) {
    return FG[1].UserID
}

AttributeFunction AF_Timestamp (FG) {
    return FG[1].Timestamp
}

ReportComposition DlThroughputReport {
    [Type eq AF_DlThroughputType],
    [ServiceClass eq PF_ServiceClass],
    [UserID eq GF_DllinkThroughput >> AF_UserID],
    [Timestamp eq GF_DllinkThroughput >> AF_Timestamp]
}
4.7 Chapter Summary

This chapter has shown an audit as a function that produce new Audit Reports from new Facts, a set of Compliance Specifications, and previous Audit Reports. By analysing the inputs provided to an Auditing Unit and the output to be produced, a sequence of subtasks of an audit has been identified, where each subtask implements a certain functionality. The auditability conditions of an SMI reveal what functions must be specified in each subtask and the resulted subspecifications constitute an audit specification which defines the logic to be executed by a general audit algorithm.

Seven classes of audit subspecifications have been identified which are divided into shareable audit subspecifications and SMI specific audit subspecifications. Shareable audit subspecifications are function definition subspecifications, whereas SMI specific audit subspecifications are function invocation subspecifications. Furthermore, Compliance Specifications become parts of audit specifications.

Audit specifications must be formally defined to allow for automated auditing. This can be done through the use of a programming language which is normally inconvenient and error-prone. In order to conveniently and formally define audit specifications, a general purpose audit specification language named Sapta has been designed. The expressiveness of Sapta has been demonstrated successfully in three representative examples of SLA compliance auditing.
AURIC’s generic auditing architecture provides for a basis to design an auditing architecture for any specific objective, thus also for SLA compliance auditing. This chapter presents the AURIC’s architecture for SLA compliance auditing and defines the interactions required between the core component (i.e., the Auditing Unit) with other architectural components that may be distributed across domains. Therefore, open interfaces are utilized whenever suitable, e.g., Diameter.

This chapter also describes the prototypical implementation of AURIC as an application oriented middleware. The evaluation of this auditing framework with respect to technical and economic feasibility, when applied to the multi-domain scenario presented in Section 3.3, is given in Chapter 7. Other required properties are also evaluated there as well.
5.1 Design

AURIC’s architecture for SLA compliance auditing consists of metering, accounting, and auditing functionality as depicted in Figure 5.1. Accounting can be enhanced with non-repudiation functionality to obtain irrefutable evidence regarding the accounting information [47]. NorCIS*, the non-repudiation extension to accounting of service consumption, developed by this thesis, is presented in detail in Chapter 6.

![Figure 5.1: AURIC’s architecture for SLA compliance auditing.](image)

As described in Chapter 1, a service provider and a customer agree on a set of services and their respective quality levels in an SLA. A service is made available by the service provisioning entities on the provider side (service provisioning domain), and is requested and consumed by the service
consuming entities on the customer side (service consuming domain). Here, access control is applied, so that only authorized customers may consume services as specified in the SLA. The service is delivered with the help of a service delivery infrastructure. Note that access control and configurations of service provisioning entities are based on SLA, and thus require interactions with the SLA Manager. However, this is not within the scope of this thesis.

5.1.1 Metering and Accounting

The quality level of the service delivered is metered to allow for the auditing of its SLA compliance. Metering can be performed at various locations as shown in the architecture. Indeed, metering of specific QoS parameters can also be done to a certain accuracy through probing from a location outside the real service delivery path. Metered data are collected and aggregated by accounting components to generate accounting data (Facts\(^1\)). The architecture shows metering and accounting on both the service consuming and provisioning domains. Normally, metering and accounting are done on the service provisioning domain, however, the author of this thesis identifies that metering and accounting on the service consuming domain may be necessary due to the following reasons:

- Customers want to perform SLA compliance audit independently.
- For the purpose of non-repudiation: either customers want to generate evidence of service consumption based on their own accounting data or customers want to verify the correctness of provider’s accounting data before generating evidences based on these data.
- Certain QoS parameters can be measured more precisely on the customer side.

On each side, accounting data are passed to the non-repudiation (NR) module to generate evidence of service consumption. Generation and transfer of evidences require interactions between NR modules from both sides. Accounting data and evidences are stored in the accounting database and the respective Fact Server is notified, so that they are transferred to the SLA Compliance Auditor.

\(^1\) The term accounting data, accounting records, and Facts are used interchangeably throughout this Chapter. A list of related accounting records is also called a Fact-List.
5.1.2 Auditing

An Auditing Unit provides for an auditing service through an Audit Manager which waits for incoming audit requests, receives audit tasks, and delivers each task to an SLA Compliance Auditor to be conducted. The Audit Manager also accepts requests relating to an audit task being conducted, e.g., requests on status of an audit task being conducted and requests to stop an audit task. An Audit Task Planner represents an entity which requests an auditing service from an Auditing Unit. An SLA Compliance Auditor retrieves data to be processed from various sources as specified in the audit task and sends the audit results to an entity which is also specified in the audit task. The data sources for SLA compliance auditing are Accounting Units, SLA Managers, and Audit Report Handlers. Each of these components provides for a service to access its data through a data server component, namely a Fact Server, an SLA Server, and an Audit Report Server respectively. Note that an Audit Report Server also receives requests to store Audit Reports.

All SLOs committed by a provider are assumed to be specified in a language which allows for an automated auditing. The resulted specifications are called SLA Audit Specifications (SLA AS). Therefore, the relevant information for SLA compliance auditing to be obtained from an SLA Manager is SLA AS. Other SLA information, e.g., user profile, service profile, measurement directive, is not relevant, and thus is not explicitly listed in the figure. An SLA AS written in Sapt (cf. Chapter 4) consists of function definition subspecifications and function invocation subspecifications. Sapt function definition subspecifications are FilterFunction, GroupingFunction, PropertyFunction, ComplianceFunction, and AttributeFunction specification, whereas Sapt function invocation subspecifications are ComplianceCalculation and ReportComposition specification.

An SLA Compliance Auditor comprises the following components:

• Control Module:
  This component configures and controls other components in carrying out their functions based on the audit task.

• SLA Client:
  This component retrieves SLA AS and delivers them to the Control
Module. Information about which SLA Manager to be contacted and which SLA AS to be retrieved is obtained from the Control Module.

- **Fact Client:**
  This component retrieves accounting data and delivers them to the Compliance Evaluator. Information about which Accounting Unit to be contacted and which accounting data to be retrieved is obtained from the Control Module. Information about the accounting data to be retrieved is part of an SLA AS.

- **Audit Report Client:**
  This component retrieves Audit Reports and delivers them to the Compliance Evaluator. It also sends Audit Reports obtained from the Compliance Evaluator as audit results to an Audit Report Handler. Information about the Audit Report Handlers to be contacted and the Audit Reports to be retrieved is obtained from the Control Module. Information about the Audit Reports to be retrieved is part of an SLA AS.

- **Compliance Evaluator:**
  This is the core component of an SLA Compliance Auditor which evaluates the SLA AS obtained from the Control Module by examining accounting data and Audit Reports obtained from the Fact Client and the Audit Report Client respectively. The evaluation result is given to the Audit Report Client to be delivered to a certain Audit Report Handler.

AURIC’s SLA compliance auditing architecture provides for important properties required, as specified in Chapter 2: support of multi-domain, highly reusable, and scalable. As described, Auditing Unit, SLA Manager, Accounting Unit, and Audit Report Handler can be distributed across administrative domains. Thus, their interfaces are described in the next section.

### 5.2 Interfaces

As mentioned in the previous section, an Auditing Unit provides for an auditing service, therefore, the description of its interfaces is substantial for implementation consideration and for supporting interoperable implementations. The architecture shows 4 external interfaces of an Auditing Unit which
are specified in all relevant detail in the following subsections. These interfaces are:

- AM (Audit Management) Interface
- ST (SLA AS Transfer) Interface
- FT (Fact Transfer) Interface
- RT (Audit Report Transfer) Interface

### 5.2.1 AM Interface

An Auditing Unit provides an auditing service through the AM interface. This interface aims at transferring and controlling audit tasks to be or being conducted by an Auditing Unit. As shown in the architecture this interface is used by the Audit Task Planner (ATP) to communicate with the Audit Manager (AMgr). Following two communication patterns are sufficient to enable management interactions in normal and erroneous situations:

- **Request-Answer pattern:** A request message can only be sent by an Audit Task Planner to an Audit Manager. An answer message is sent as a response to a valid request message. An answer message may contain error description, if any.

  \[
  \text{ATP} \rightarrow \text{AMgr}: \text{Request} \\
  \text{AMgr} \rightarrow \text{ATP}: \text{Answer}
  \]

- **Notification pattern:** A notification message can be sent at any time by an Audit Manager to an Audit Task Planner. The main reason to have a notification message is to allow an Audit Manager to inform the respective Audit Task Planner of any error occurred during an Audit. The need to inform unsolicitedly that an audit task has been completed is another reason to have a notification message.

  \[
  \text{AMgr} \rightarrow \text{ATP}: \text{Notification}
  \]

A set of message types is proposed for the communication through the AM interface, which is described in detail in Table 5.1 and Table 5.2. While Table 5.1 defines Request-Answer message types, Table 5.2 defines Notification message types. Additionally, Table 5.3 proposes various error, status,
and result codes to be used in a communication between an Audit Task Planner and an Audit Manager.

Table 5.1: Request-Answer message types for audit management

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
<th>Request Parameters:</th>
</tr>
</thead>
</table>
| Audit            | Purpose: To transfer an audit task. | - AuditStartTime: The time at which the audit task must be started. This parameter is optional. If not specified, the audit task is to be started immediately.  
- AuditStopTime: The time at which the audit task must be stopped. This parameter is optional. If not specified, the audit task is to be stopped through AuditTermination request message.  
- AS_URL: The URL of the SLA AS.  
- FactServer: The IP address or hostname, and port number of the entity which keeps the accounting data. More than one FactServer may be specified.  
- IReportServer: The IP address or hostname, and port number of the entity which keeps the Audit Reports to be examined. This parameter is optional and more than one IReportServer may be specified.  
- OReportServer: The IP address or hostname, and port number of the entity which accepts Audit Reports to be stored.  
- SortingInfo: The sort order (ascending or descending) and the name of the attributes based on which Facts and Audit Reports from different sources are to be sorted. Normally, it is the timestamp attribute in chronological order. This parameter is optional. |
| AuditTermination | Purpose: To terminate an audit task. | - TaskID: The identifier of the audit task to be terminated. |

Answer Parameters:  
- TaskID: The identifier for the requested audit task.  
- ResultCode: A number to denote the result of processing the request message.
### Table 5.1: Request-Answer message types for audit management

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AuditStatus</strong></td>
<td>Purpose: To obtain the status of an audit task.</td>
</tr>
<tr>
<td></td>
<td><strong>Request Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td>• TaskID: The identifier of the audit task whose status information is</td>
</tr>
<tr>
<td></td>
<td>requested.</td>
</tr>
<tr>
<td></td>
<td><strong>Answer Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td>• TaskID: The identifier of the requested audit task.</td>
</tr>
<tr>
<td></td>
<td>• StatusCode: A number representing the status of the audit task.</td>
</tr>
<tr>
<td></td>
<td>• ResultCode: A number to denote the result of processing the request</td>
</tr>
<tr>
<td></td>
<td>message.</td>
</tr>
<tr>
<td><strong>AuditTaskList</strong></td>
<td>Purpose: To obtain a list of audit tasks.</td>
</tr>
<tr>
<td></td>
<td><strong>Request Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td><strong>Answer Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td>• TaskID: The identifier of an audit task to be or being conducted.</td>
</tr>
<tr>
<td></td>
<td>There may be none, one, or more TaskID in the answer message.</td>
</tr>
<tr>
<td></td>
<td>• ResultCode: A number to denote the result of processing the request</td>
</tr>
<tr>
<td></td>
<td>message.</td>
</tr>
</tbody>
</table>

### Table 5.2: Notification message types for audit management

<table>
<thead>
<tr>
<th>Message</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error</strong></td>
<td>Purpose: To inform the respective Audit Task Planner of errors during an</td>
</tr>
<tr>
<td></td>
<td>audit.</td>
</tr>
<tr>
<td></td>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td>• ErrorCode: A number denoting the type of error.</td>
</tr>
<tr>
<td></td>
<td>• TaskID: The identifier of the affected audit task.</td>
</tr>
<tr>
<td></td>
<td>This parameter is optional and more than one TaskID may be specified.</td>
</tr>
<tr>
<td></td>
<td>• UnreachableHost: The hostname or IP address of the host which is</td>
</tr>
<tr>
<td></td>
<td>unreachable. This parameter is optional and more than one UnreachableHost may be specified.</td>
</tr>
<tr>
<td><strong>AuditCompleted</strong></td>
<td>Purpose: To inform the respective Audit Task Planner of audit tasks</td>
</tr>
<tr>
<td></td>
<td>which have been completed.</td>
</tr>
<tr>
<td></td>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td>• TaskID: The identifier of the completed audit task.</td>
</tr>
<tr>
<td></td>
<td>More than one TaskID may be specified.</td>
</tr>
</tbody>
</table>
Table 5.3: Various codes in ATP - AMgr communication.

<table>
<thead>
<tr>
<th>Type of Code</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorCode</td>
<td>ParsingError, InvalidURL, HostUnreachable,</td>
</tr>
<tr>
<td>ResultCode</td>
<td>Success, UnknownTaskID,</td>
</tr>
<tr>
<td>StatusCode</td>
<td>Scheduled, InProgress, Finished, Idle_NoData, Suspended</td>
</tr>
</tbody>
</table>

An audit task can be time-consuming or long-lived by its nature, e.g., real-time auditing of availability. Figure 5.2 depicts two typical Message Sequence Charts (MSC) in audit management sessions, where one MSC is usually found in cases with short-lived audit tasks and the other in long-lived audit tasks. In cases of long-lived audit tasks, audit status retrieval becomes useful. If the communication uses connection oriented transport protocol, e.g., Transmission Control Protocol (TCP), the connection needs not be preserved during the long idle time. This means a disconnection with the Audit Task Planner may not affect the audit task being conducted by the Auditing Unit. Furthermore, in order to send notification messages to the right ATP, a mapping of each audit task to the IP address and the port number used by the ATP who has submitted the audit task is required.

![Audit Task Planner](image1)
![Audit Manager](image2)

(a) short-lived audit tasks

![Audit Task Planner](image3)
![Audit Manager](image4)

(b) long-lived audit task

Figure 5.2: Typical MSCs in audit management sessions.
5.2.2 ST Interface

The purpose of this interface is to enable an Auditing Unit to retrieve a particular SLA AS from an SLA Manager. A URL is used to locate a particular SLA AS. Existing protocols such as HTTPS [88] and SSH File Transfer Protocol [32] are very well suited to be used to transfer SLA AS securely from an SLA Manager to the Auditing Unit. Thus, no further detail description is necessary.

5.2.3 FT Interface

This interface is used to transfer Facts from a Fact Server to a Fact Client. Since a Fact Server provides an interface to retrieve any type of Facts a mechanism must be defined to tell the Fact Server which Facts are requested. This can be seen as doing Facts filtering at the side of Fact Server, which is reasonable to reduce traffic. The Request-Answer pattern is proposed to transfer the Facts selection criteria as well as the Facts themselves.

The Fact Client sends a FactSelection_Request message to the Fact Server containing criteria to select certain accounting data. The language to define the criteria is transparent to the protocol. The Fact Server is expected to be able to identify the language based on the language code and thus correctly parse the criteria. Language code consists of language ID and version number. If the Fact Server does not support the language or the selection criteria cannot be parsed correctly, the respective result code is sent in the answer message, e.g., UnsupportedLanguage, UnsupportedLanguageVersion, InvalidSelectionCriteria.

\[
\text{FC} \Rightarrow \text{FS}: \text{FactSelection \_Request}(<\text{LanguageCode}>, \text{<SelectionCriteria>})
\]

\[
\text{FS} \Rightarrow \text{FC}: \text{FactSelection \_Answer}(<\text{ResultCode}>)
\]

To transfer the Facts the following message pair is used:

\[
\text{FS} \Rightarrow \text{FC}: \text{FactTransfer \_Request}(<\text{Fact}>, \text{<Fact>})
\]

\[
\text{FC} \Rightarrow \text{FS}: \text{FactTransfer \_Answer}(<\text{ResultCode}>)
\]

For the purpose of transferring Facts Diameter protocol [16] is very well suitable. The Base Accounting message pair Accounting-Request / Accounting-Answer (ACR/ACA) is sufficient to implement FactTransfer message.
pair. However, to allow for FactSelection message pair a new Diameter command must be defined.

### 5.2.4 RT Interface

Since Audit Reports have the same data representation as Facts, this interface does not differ from the FT Interface, except that an Audit Report Client must be able to send FactTransfer_Request messages to an Audit Report Server in order to store audit results.

### 5.3 Security Consideration

As described in Chapter 3, in order to support inter-domain interactions security must be considered. Since in an SLA all parties involved are defined, thus interacting components are known in advance, security associations among those components can be established as the result of the SLA concluded, before auditing interactions take place. Having these security associations in place, authentication and authorization in inter-domain interactions can be accomplished.

For example, accounting data are possibly distributed across administrative domains, thus, access control to the data are needed. There are several ways of doing access control, e.g., through static configurations of firewall once it is known which hosts and communication ports are going to be accessed by which other hosts from other administrative domains. Certainly, more dynamic solutions are applicable. Solutions to inter-domain access control have been widely discussed in research community, e.g., by IRTF AAAArch RG [50], and in EU research projects Moby Dick [76] and Daidalos [24]. Therefore, access control is outside the scope of this thesis. Furthermore, access control functionality can indeed be provided without intervening auditing functionality except the interception of the data flow as shown in Figure 5.3.

In this solution, an Auditor Proxy is inserted between Audit Task Planner and Auditing Unit. The Auditor Proxy is able to analyse audit tasks and requests admission to access the relevant Accounting Unit from the Access Control module of the respective administrative domain. If there is a security
association between the two domains, the access request is accepted and the firewall is configured to allow data flows between the Auditing Unit and the Accounting Unit. On receipt of a positive response from the Access Control module, the Auditor Proxy forwards the audit task to the Auditing Unit.

![Diagram](image)

**Legend:**
- *Control* direction
- *Data* direction

**Administrative Domain Boundary**

**Figure 5.3: Access control to accounting data.**

Normally, an Audit Task Planner and an Auditing Unit reside within the same administrative domain. However, in case they reside in different domains access control to the Auditing Unit is also required. Since an Audit Task Planner and an Auditing Unit have a security association, a secure communication channel can be established. This prevents their interactions from a Man-In-The-Middle attack and only authorized Audit Task Planners can gain access to the auditing service. However, an authorized Audit Task Planner may still be malicious. Hence, counter measures to following serious threats from an authorized Audit Task Planner are described:

- **An Audit Task Planner tries to control audit tasks owned by the others:** To protect against this threat a mapping of audit tasks to their owner must be maintained by an Auditing Unit. Surely, the sender of each request message must be identifiable.

- **An Audit Task Planner submits an audit task where the data sources (SLA Server, Fact Server) are not according to the SLA:**
To avoid this threat, the SLA Manager specified in the audit task must verify the address of involving data sources. It is possible, since each SLA Manager is supposed to keep tuples consisting of QoS parameter, the corresponding SLA AS, the IP addresses (or hostnames) and port numbers of the involving Fact Servers and Audit Report Servers.

5.4 Implementation

Based on the proposed SLA compliance auditing architecture, a prototypical implementation of an SLA compliance auditing framework is provided. The framework is an application-oriented middleware [38] which provides for a set of basic functionality to allow developers to quickly implement applications for auditing of SLA compliances. The middleware is implemented in C++, applying the object oriented programming paradigm. The prototypical implementation aims at showing that developing an SLA Compliance Auditor can be done basically through specialization of the framework’s base classes which virtually implement function definition subspecifications of an audit specification. Each specialization of the base class contains the concrete implementation of a specific function definition subspecification. It should be noted that current implementation of the framework is a possible, but not the only application area of AURIC. As a proof of the generic auditing concept proposed, three specific SLA Compliance Auditors have been implemented on top of the common auditing framework, whose implementation architecture is described in the following subsection.

5.4.1 Implementation of an SLA Compliance Auditor

Figure 5.4 depicts the implementation architecture of a specific SLA Compliance Auditor. The auditor is specific since the application logic to audit a specific SLO is implemented as an integral part of the auditor. Thus, the auditor does not require an SLA Client component to retrieve the SLA AS (cf Figure 5.1). However, to allow for a flexible configuration of an auditor, function invocation subspecifications written in Sapta (i.e., ComplianceCalculation specification and ReportComposition specification) are used. The architecture consists of a configuration file containing function invocation subspecifications, a Control Module, a Sapta Parser, a Fact and Report Client, and a Compliance Evaluator. Since the data representation of Facts and
Audit Reports are the same, and the FT and RT Interface use the same protocol, the functionality of a Fact Client and a Report Client (cf Figure 5.1) is merged into a single entity called a Fact and Report Client.

![Diagram of SLA Compliance Auditor](image)

**Figure 5.4: Implementation architecture of an SLA Compliance Auditor.**

Facts and Audit Reports are transferred using the Diameter protocol [16]. Diameter is selected because its accounting message is suitable for carrying a list of AVPs and because Diameter is extensible. Therefore, the Fact and Report Client consists of a Fact and Report (FR) Transfer Module and the Open Diameter Framework which is an Open Diameter [79] implementation of the Diameter protocol. The detailed description of the Open Diameter implementation and its software architecture is given in [28]. The Compliance Evaluator consists of two parts: Subtask Modules and Application Logic. The Subtask Modules implement functionality which is common to all auditing application in executing audit subtasks, whereas the Application Logic implements function definition subspecifications of an audit specification. The Control Module, the Sapta Parser, the FR Transfer Module, and the Subtask Modules build together the AURIC middleware.

The Control Module reads the ComplianceCalculation and ReportComposition specification from the configuration file, gives it to the Sapta Parser
to be parsed, and based on the resulted parse tree configures the Compliance Evaluator. Then the Control Module instructs the Fact and Report Client to establish connection with the respective Fact Servers and Report Servers for the transfer of Facts and Audit Reports.

![Implementation Architecture Diagram]

**Figure 5.5: Implementation architecture of a Fact and Report Client.**

**Connection Establishment**

Figure 5.5 depicts in detail the implementation architecture of a Fact and Report Client. Diameter runs over reliable transport protocols, *e.g.*, TCP, Stream Control Transmission Protocol (SCTP), as defined in [2]. Both TCP
and SCTP are connection-oriented, thus require connection establishment. The FR Transfer Manager is responsible for establishing connections with Fact Servers and Audit Report Servers as specified in the audit task. The FR Transfer Manager provides an interface to the Open Diameter’s Peer Connection Management and Routing Framework (PCM&RF) in order to expose only relevant functionality of Open Diameter and to offer new functionality currently not supported by Open Diameter, e.g., notification of established connections. The PCM&RF comprises a set of objects responsible for connection establishment, managing peer objects, implementing peer’s finite state machine, sending and receiving of messages, and message routing. Note that in Diameter terminology, a Fact Server or an Audit Report Server is a Diameter peer of a Fact and Report Client and vice versa. A Diameter peer is a Diameter node to which a given Diameter node has a direct transport connection [16].

Transfer of Facts and Audit Reports

Facts and Audit Reports are transferred within FR Transfer Sessions. An FR Transfer Session is to be generated for every connection with a Fact Server or an Audit Report Server. FR Transfer Sessions are derived from Open Diameter accounting sessions in order to provide methods to transfer Facts or Audit Reports within accounting messages. There are two situations in which a new session is generated:

-Receipt of an accounting message containing a new session identifier. The Session Delivery is responsible for delivering each incoming message which is obtained from the PCM&RF to the respective session. The session to which a message belongs is determined by querying the message's session identifier. The Session Delivery searches the local Session Database for a matching session. If no matching session is found, the Session Delivery will lookup a matching Session Factory that has an application identifier matching the application identifier of the message. Note that every Diameter message contains an application identifier, e.g., the application identifier of Diameter Base Accounting is 3. If there is a registered Session Factory, then the Session Delivery requests the factory to create a new session and delivers the message to the newly created session. In this respect, the FR Transfer Manager is responsible for registering the Session Factory, i.e., the factory of FR Transfer Sessions to the Session Factory Register. Upon creation of a
new session the Session Factory notifies the FR Transfer Manager of that session, so that it can relate the new session with the Compliance Evaluator Interface to handle incoming messages.

- Explicit creation of a new session by the application in order to send accounting messages.
A Fact and Report Client sends the resulted Audit Reports to a particular Audit Report Server. This requires the FR Transfer Manager to generate an FR Transfer Session which will register itself to the Session Database. The FR Transfer Manager also relates the Compliance Evaluator Interface with the generated FR Transfer Session to allow audit results to be delivered through the FR Transfer Session.

An Accounting Session Finite State Machine (FSM) is responsible for implementing the accounting state machine as described in [16]. After processing each message to update its internal state, an Accounting Session FSM passes the message to the respective handler. Incoming messages from Session Delivery are passed to the respective FR Transfer Sessions, whereas messages from FR Transfer Sessions are passed to the Session Delivery.

The Compliance Evaluator Interface collects Facts and Audit Reports to be examined from different FR Transfer Sessions and delivers them to the Compliance Evaluator. The interface is also responsible for sorting the data before delivery, if required. Furthermore, it receives audit results from the Compliance Evaluator and forwards them to a particular FR Transfer Session.

Diameter specifies a generic accounting message pair Accounting-Request / Accounting-Answer (ACR/ACA), where ACR messages are supposed to carry accounting data. Since accounting data are application specific, Diameter Base only specifies for an ACR message two mandatory AVPs Accounting-Record-Type and Accounting-Record-Number, and two optional AVPs User-Name and Event-Timestamp, which are useful in most cases to carry accounting data, in addition to other AVPs which are necessary but not for carrying accounting information. In order to use ACR messages to carry any Facts and Audit Reports this thesis defines three new AVPs: Fact-Structure-ID, Attribute-Name, and Attribute-Value.
An ACR message can contain multiple Attribute-Name AVPs whose values reflect the attribute names of the same type of Facts. A Fact-Structure-ID is a number used to denote a certain list of attribute names. The list of attribute names should have been defined in a previous ACR or it is contained within the Attribute-Name AVP in the same ACR. Using a Fact-Structure-ID AVP eliminates the need to always send Attribute-Name AVPs. In an ACR message only one Fact-Structure-ID AVP may exist. Each ACR message contains one or more Attribute-Value AVPs. The number of Attribute-Value AVPs must be a multiple of the number of Attribute-Name AVPs in the same ACR or in the ACR containing the same Fact-Structure-ID.

In order to correctly relate multiple Attribute-Name AVPs with multiple Attribute-Value AVPs, the Diameter implementation must preserve the ordering of AVP insertion into a message. Otherwise the values of the Attribute-Name AVPs and the Attribute-Value AVPs must be encoded with a sequencing number, e.g., as follows:

““<a string of digits><a separator symbol><a string of characters>“”

Compliance Evaluation

The Compliance Evaluator consists of a series of subtask modules with each being responsible for a particular audit subtask as identified in the previous chapter. The outputs of one module serve as inputs to the other modules as shown in Figure 5.6. The Control Module has the task to interconnect these modules according to specific application needs. In order to be flexible, interconnection information is specified in Sapta as ComplianceCalculation specification and ReportComposition specification.

The application logic resides in the following components (application specific subtask functions):

- ASFF (Application Specific Filter Function)
- ASGF (Application Specific Grouping Function)
- ASPF (Application Specific Property Function)
- ASCF (Application Specific Compliance Function)
- ASAFAF (Application Specific Attribute Function)
The architecture of each subtask module is depicted in Figure 5.7. Each Specific Subtask Function implements a particular Application Specific Function. Based on the information from the Control Module, the Subtask Engine instructs the Subtask Function Factory to instantiate a Specific Subtask Function. The Producer Interface provides a method for data sources to deliver their data to be processed by this subtask module. These data are forwarded to the Subtask Engine which will invoke each Specific Subtask Function to process them. However, if the Subtask Engine is processing previous data, then new data are put in the Queue first. Based on the results of the Specific Subtask Functions the Subtask Engine generates data to be delivered to other modules. The Control Module is responsible for configuring the Subtask Engine to deliver the subtask results to the right modules. The Subtask Engine of each subtask module has a different logic which corresponds to the functionality to be provided by the subtask module. For example, a Facts Grouping Module must maintain open Fact-Lists (i.e., a set of lists of related accounting records which are not yet complete for compliance evaluation), whereas the other modules not.
The auditing framework API provides for the following base classes to implement the application logic:

```cpp
class FilterFunction : public SubtaskFunction {
public:
    virtual ~FilterFunction() {}
    virtual bool Process(const Fact& currentFact) = 0;
};

class GroupingFunction : public SubtaskFunction {
public:
    virtual ~GroupingFunction() {} 
    virtual void Process(const Fact& currentFact, 
                         OpenFactLists& ofl) = 0;
};

class PropertyFunction : public SubtaskFunction {
public:
```
virtual ~PropertyFunction() {}
virtual prop_value_t* Process(FactList& currentFactList) = 0;
};
class ComplianceFunction : public SubtaskFunction {
public:
virtual ~ComplianceFunction() {}
virtual compliance_value_t Process(const PropertyValues& propertyValues) = 0;
};
class AttributeFunction : public SubtaskFunction {
public:
virtual ~AttributeFunction() {}
virtual void Process(attribute_value_t& attrValue, FactList& currentFactList,
const PropertyValues& propertyValues, compliance_value_t complianceValue) = 0;
};

Each base class offers a method Process() which should be implemented by the developer of the auditing application as described in Table 5.4. The methods differ in their input parameters and the type of the return value. For example, the Process() method of the class FilterFunction has an input parameter of class Fact and should return a boolean value. An object of the class Fact provides for methods to get information about the accounting record encapsulated in the object, e.g., the value of a particular attribute (i.e., a field of the record). Based on this information the Process() method returns either true or false. A return value of true indicates the auditing framework to pass the Fact to the GroupingFunction, whereas a return value of false means to drop the Fact.

Table 5.4: The use of the API’s Process() methods.

<table>
<thead>
<tr>
<th>Class</th>
<th>The use of Process() method</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterFunction</td>
<td>Examine the accounting record encapsulated in the Fact object and return true or false to denote whether the accounting record is related to the SLO being audited.</td>
</tr>
<tr>
<td>GroupingFunction</td>
<td>Examine the accounting record encapsulated in the Fact object and assign this record to one or more Fact-Lists (according to the SLO) with the help of OpenFactLists object.</td>
</tr>
</tbody>
</table>
Table 5.4: The use of the API’s Process() methods.

<table>
<thead>
<tr>
<th>Class</th>
<th>The use of Process() method</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyFunction</td>
<td>Calculate a property value from the list of related accounting records encapsulated in the FactList object.</td>
</tr>
<tr>
<td>ComplianceFunction</td>
<td>Calculate a compliance value from the list of property values encapsulated in the PropertyValues object.</td>
</tr>
<tr>
<td>AttributeFunction</td>
<td>Calculate a report attribute value from the list of related accounting records (encapsulated in FactList object), the list of property values (encapsulated in the PropertyValues object), and the compliance value.</td>
</tr>
</tbody>
</table>

The Process() method of the class GroupingFunction has an additional parameter of class OpenFactLists. An object of the class OpenFactLists provides for methods to manipulate open Fact-Lists managed by the auditing framework, e.g., to add a Fact into an open Fact-List and to close an open Fact-List. This allows the developer to examine accounting records and to group them as required by the application logic.

The Process() method of the class PropertyFunction has a parameter of class FactList. An object of the class FactList provides for methods to manipulate and to access information about the accounting records encapsulated in the object, e.g., the number of accounting records, the sum of the value of a particular field of the accounting records. The return value is of type pointer to prop_value_t. The developer needs to derive from prop_value_t a class to store application specific type of a property value. This way, the auditing framework allows for any type to be used for a property value. The auditing framework maintains the pointer to this value which will be passed to the Process() method of the ComplianceFunction.

The Process() method of the class ComplianceFunction has a parameter of class PropertyValues. An object of the class PropertyValues provides for methods to access property values encapsulated in the object. The return value is of type compliance_value_t which is a C++ float.

Finally, the Process() method of the class AttributeFunction is used to generate a value of type attribute_value_t which is a string in this implementation. To allow the Process() method to calculate an attribute value three ob-
jects are made available: a FactList, a PropertyValues, and a compliance_value_t, which are obtained from Process() methods of the other functions. Of course, the property values and the compliance value must stem from the same Fact-List.

The parent class SubtaskFunction provides for methods to parameterize application specific subtask functions. Parameterization allows the function to be configurable. The methods will be invoked by the auditing framework after the application specific subtask function is created based on the Sapta specification in a configuration file. The definition of this class is as follows:

```cpp
class SubtaskFunction {
public:
    virtual ~SubtaskFunction() {} // paramNo starts at 1
    virtual bool SetStringParam(unsigned int paramNo, const std::string& paramVal) {return false;}
    virtual bool SetNumberParam(unsigned int paramNo, float paramVal) {return false;}
    virtual bool SetBooleanParam(unsigned int paramNo, bool paramVal) {return false;}
    virtual bool SetListParam(unsigned int paramNo, const SaptaList& paramVal) {return false;}
};
```

5.4.2 Implementation of a Fact/Report Server

As described, an FT interface and an RT interface differ only in the direction of the data transfer. Therefore, it is beneficial to implement a component which is configurable to act as a Fact Server or as an Audit Report Server. This component is called a Fact/Report Server.

The implementation of a Fact/Report Server does not differ significantly from the implementation of a Fact and Report Client as shown in Figure 5.8. The difference lies in the one-to-one relationship between Data Access Interfaces and FR Transfer Sessions. Interactions among the components follow the same pattern as for the Fact and Report Client. They are described below for the two possible roles of a Fact/Report Server in a session: as a data source or as a data repository.
1. Fact or Report Server as a data source

- Upon connection establishment FR Transfer Manager creates an FR Transfer Session and notifies F/R Server Control Module that a new session is created.

- F/R Server Control Module creates a Data Access Interface and passes this object to FR Transfer Manager.

- FR Transfer Manager relates the new Data Access Interface with the new FR Transfer Session.

*Figure 5.8: Implementation of a Fact/Report Server.*

Legend:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>Acctg</td>
<td>Accounting</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>FR</td>
<td>Facts and Reports</td>
</tr>
</tbody>
</table>

- Creation
- Control
- Data (Facts or Reports)
5.5. Chapter Summary

- Data Access Interface retrieves data and sends them within the FR Transfer Session.

2. Report Server as a data repository

- Upon connection establishment an accounting message with a new session identifier is expected.
- Upon reception of this message, Session Factory creates a new FR Transfer Session and notifies FR Transfer Manager of the new session.
- FR Transfer Manager notifies F/R Server Control Module that a new session is created.
- F/R Server Control Module creates a Data Access Interface and passes this object to FR Transfer Manager.
- FR Transfer Manager relates the new Data Access Interface with the new FR Transfer Session.
- Data Access Interface waits for data to be stored from the new FR Transfer Session.

5.5 Chapter Summary

This chapter has presented AURIC’s architecture for SLA compliance auditing which supports inter-domain interactions between its main components: Audit Task Planners, Auditing Units, Accounting Units, Audit Report Handlers, and SLA Managers. The interfaces for inter-domain interactions have been described and solutions to security issues identified in Chapter 3 have been discussed. In this regard, Diameter has been proposed to be used to transfer Facts and Audit Reports.

The prototypical implementation of an SLA Compliance Auditor which is the core component of an Auditing Unit as well as the implementation of a Fact/Report Server which is configurable to operate as a Fact Server or a Report Server have been described in detail. At the same time, AURIC’s middleware components and its API are described as well. For the purpose of Facts and Audit Reports transfer, the OpenDiameter implementation of Diameter protocol has been employed in a modular way by the AURIC’s middleware.
Chapter 6

Non-repudiation of Service Consumption

While auditing of SLA compliances addresses the concern of customers about providers’ contractual commitments in service delivery, non-repudiation of service consumption addresses the concern of providers about the customers’ obligation to pay for services consumed. A customer of a service provider may be a service provider to another party. Since a party can have these two roles at the same time, non-repudiation of service consumption and SLA compliance auditing complement each other.

This chapter develops the architecture that extends the Accounting Unit of AURIC’s architecture for SLA compliance auditing (cf. Figure 5.1) with detailed components for providing non-repudiation of service consumption
in a AAA supported mobile IP-based environment. It is a refinement of the NorCIS (Non-repudiation of the Consumption of Internet Services) architecture [48] developed by the author for the same environment, hence, termed NorCIS*. All architectural components have the task to generate and transfer irrefutable evidence of service consumption to interested parties. Thereby, non-repudiation protocols required are proposed in this thesis for three widely applied accounting schemes: item-based, time-based, and volume-based scheme.

To ease the process of describing, discussing, and defining correct and complete protocols for the support of non-repudiation functions, the following notation is used throughout this chapter:

\( P_A \) Public encryption key of party A.

\( P^{-}_A \) Private decryption key of party A.

\( V_A \) Public verification key of party A.

\( S_A \) Private signature key of party A.

\( sS_A(X) \) Party A’s digital signature on information X with the private signature key \( S_A \). The encryption is performed on the hash of X.

\( eK(X) \) Information X encrypted with key K.

\( dK(X) \) Information X decrypted with key K.

\( H(X) \) A one-way hash of information X.

\( X_1, X_2 \) Information \( X_1 \) concatenated with information \( X_2 \).

\( A \rightarrow B: M \) Party A sends message M to party B.

\( A \leftarrow B: M \) Party A retrieves message M from party B.

\([X]\) Information of type X is optional

\([X]^*\) Information of type X may appear 0, 1, or more times.

\( f_m \) A flag indicating the intended purpose of a message.
6.1 Roaming User Scenario

Figure 6.1 depicts the scenario considered for a mobile environment where users are able to consume IP-based services not only from the home, but also from foreign domains. It is considered as a part of Figure 3.5 in the multi-domain scenario described in Section 3.3. User U has a contract with his Home (network) Provider HP. This contract allows user U via his mobile terminal to reach network access from the home domain operated by HP. To allow for user U also gaining network access from different foreign domains, roaming agreements are established between HP and Foreign (network) Providers (FP), who operate those domains independently. The deployment of Authentication, Authorization, and Accounting [25] and a Mobile IP [65], [83] infrastructure is assumed to enable user mobility.

In such a mobile environment accounting data are important to justify charges for the consumption of services. Users are expected to rely on accounting data that a provider collects. However, a user can deny having consumed a service, if there is no proof of this usage. In practice, this dispute can be solved by having written terms and conditions, "forcing" the user to...
accept the correctness of all providers' accounting information, mainly if the user cannot prove the opposite. This solution places the user in a disadvantage compared to the provider. Therefore, a solution for users and providers has to be achieved by applying non-repudiation (NR) mechanisms to generate evidence of service consumption or henceforth, "evidence" in short.

There are different types of services, e.g., connectivity, voice communication, data transfer, and content (streaming and non-streaming), that a provider can offer. Depending on the type of a service consumed, charging of a service usage can be based on a variety of schemes, in particular, data volume, session time, and service event (i.e., item consumed) [1]. Accordingly, the accounting schemes required are volume-based, time-based, and item-based accounting. Considering these schemes, new non-repudiation protocols are required, since existing approaches for non-repudiation in a message transfer process are not always applicable for transferring evidence of service consumption in a way that is fair for service providers as well as users.

6.2 Requirements

As mentioned, generating an evidence of service consumption faces the problem of fairness for the service provider and the user with respect to its timepoint of delivery. Both alternatives, proof after consumption and proof before consumption, as depicted in Figure 6.2 are not fair. Hence, a fair non-repudiation protocol for transferring evidence of service consumption is required, where fairness is defined as follows:

Definition 6.1: A protocol for non-repudiation of service consumption is fair, if in case a requested service is consumed it provides for the service provider and the service user evidences capable of proving the real service consumption after completion of the protocol, and if in case a requested service is not consumed the service provider cannot obtain evidences to prove a service consumption.

Requirement 6.1: Fairness:
The non-repudiation protocol to be designed must be fair for the service provider and the user in the sense of business risk.
In high risk situations a non-repudiation protocol may need the involvement of a TTP. A Trusted Third Party (TTP) can assist the other parties in various ways as identified by different roles a TTP can play, such as a Certification Authority (CA), a Notary, a Delivery Authority, a Time-Stamping Authority (TSA), and an Adjudicator [112]. Of all these roles a Delivery Authority represents the strongest TTP involvement. A Delivery Authority is an inline TTP intervening in each transaction. Furthermore, as shown in Figure 6.3, a TTP can help in all four phases of a non-repudiation service: (1) evidence generation, (2) evidence transfer, (3) evidence verification and storage, and (4) dispute resolution [112]. The approaches presented in this thesis deal with phases (1) to (3). Dispute resolution is to be solved with human interactions. In principle, the involvement of a TTP is not desirable, but might not be avoidable in high risk situations. Therefore, the following requirement is valid.

---

**Figure 6.2: Two alternatives for evidence delivery.**

(a) Proof after consumption

(b) Proof before consumption
Requirement 6.2: *Minimal Involvement of TTP*:
The involvement of a TTP, if needed, should be kept to a minimum. In particular, the involvement of a TTP taking the role of a Delivery Authority is to be avoided.

Finally, although evidence of service consumption requires identification of the user who has consumed or is consuming the service, a mechanism must be found which allows for a protection of the identity privacy of a user, since principally, only the party who bills the user needs to know her identity. Thus, the following requirement is essential in a mobile environment:

**Requirement 6.3: Identity Privacy:**
Except for the HP, who is supposed to bill a user for services consumed, no other party needs to know the real identity of the user.

### 6.3 The NorCIS* Model

The NorCIS* model is a model for non-repudiation of service consumption in the roaming user scenario. The model identifies all parties involved in
the non-repudiation process as well as their roles and functions. In this model there are four parties involved in a non-repudiation process: an FP, an HP, a user, and a TTP as shown in the class diagram in Figure 6.4. In fact, FP and HP are two different roles that a service provider can take in an interaction with users and with the other service providers. However, to clearly show their different functions, they are represented as two different classes in the class diagram. A service provider must implement all functionality assigned to both the HP and the FP.

![Figure 6.4: The NorCIS* model.](image)

The model distinguishes three types of evidence based on the creator: user, HP, and TTP evidence. As a party who delivers a service an FP is not re-
required to create an evidence. The model is described in detail in the following subsections with respect to service delivery and consumption, generation and verification of statements on service consumption, and generation and verification of evidences.

### 6.3.1 Service Delivery and Consumption

Services are offered by service providers and consumed by users. To be able to consume services a user must present her identifier and credentials. Only after a successful authentication and authorization by the HP a user is allowed to consume services. The model in Figure 6.4 depicts FP as a service provider and HP as a provider who is responsible for authenticating and authorizing users. In this respect, an HP acts as an identity provider.

A service may be encrypted before being delivered. However, service encryption is only possible for content. If a content is encrypted the decryption key must be made available at some other time to the user. In this model the key can be submitted directly to the user or via a TTP who will then generate a TTP evidence, which proves the submission and contains the key. The party interested in this evidence is expected to retrieve it from the TTP.

A service consumption is accounted for by the FP as well as the user. The model assumes that a TTP does not account for service consumption because a strong involvement of a TTP is not desired. The accounting data generated by an FP is called provider accounting data, whereas the accounting data generated by a user is called user accounting data. Provider accounting data are used by an FP to charge a user via her HP for the services consumed. User accounting data are used to verify provider’s accounting.

### 6.3.2 Generation and Verification of Statements on Service Consumption

A statement on service consumption identifies the service provider, the service user, and the service, and contains information about the consumption of the service with respect to accounting parameters. Statements on service consumption can be generated with or without using accounting data. Without using accounting data these statements are generated based on the
service consumption intention. Therefore, such statements should actually be called statements on service consumption intention. This model does not distinguish between these two kinds of statements because they are to be signed by a user to generate evidence of service consumption. A signed statement on service consumption intention is treated in this thesis as a signed statement on service consumption.

A verification of a statement on service consumption should be based on accounting data, if available. Otherwise, it should be based on the service consumption intention. A party who has generated a statement needs not verifying it, while another party should verify it.

Two issues are of importance with respect to the generation of non-repudiation statements: identification of the party who generates statements and the frequency of such statement generations. Three alternatives exist with respect to the generation of non-repudiation statements as summarized in Table 6.1. The model foresees the FP or the user as the party who generates a statement on service consumption.

Table 6.1: Different alternatives of parties generating statements.

<table>
<thead>
<tr>
<th>Generator</th>
<th>Verifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>User</td>
<td>If verifications are based on user accounting data, they need not be precise to reduce complexity and processor’s load doing accounting. In this regard, a deviation from provider’s statements should be tolerated to a certain degree.</td>
</tr>
<tr>
<td>User</td>
<td>FP</td>
<td>Since the user herself generates the statements and signs them, this leads to simpler interactions and reduced non-repudiation traffic in the network. However, if a statement generation is based on accounting data, precise accounting is required to avoid deviation from provider’s accounting data.</td>
</tr>
<tr>
<td>HP</td>
<td>User</td>
<td>If a statement is generated based on the provider accounting data, the accounting data collected by the FP must be sent to the HP. This approach does not have advantages compared to the other approaches.</td>
</tr>
</tbody>
</table>

The frequency of statement generations is determined by the accounting scheme for the particular service as shown in Table 6.2. For time-based and volume-based accounting schemes this frequency is determined by a trade-off between communication overhead and business risk.
If a statement on service consumption is not generated by the user, the correctness of this statement should be verified by the user before signing it. If this verification fails, the user must not sign such statements while facing the consequence of service termination. The correctness of a statement signed by a user should also be verified by the service provider. If this verification fails, the respective service may need to be terminated.

To support statement verifications, accounting of service consumption at the user’s side and at the provider’s side must be synchronized. This is possible if the same time scale (e.g., UTC (Coordinated Universal Time)) is used and the computers’ clocks are synchronized (e.g., using Network Time Protocol). As clocks cannot be fully synchronized in distributed systems, users and providers have to agree on a small deviation.

Table 6.2: Frequency of statement generations.

<table>
<thead>
<tr>
<th>Accounting Scheme</th>
<th>Frequency of Statement Generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-based</td>
<td>Every $t$ unit of time</td>
</tr>
<tr>
<td>Volume-based</td>
<td>Every $n$ unit of traffic volume consumed by the user</td>
</tr>
<tr>
<td>Event-based</td>
<td>Every item sent/received</td>
</tr>
</tbody>
</table>

6.3.3 Generation and Verification of Evidences

Non-repudiation of service consumption aims at providing various evidences related to a service consumption, which are classified into three types: user evidence, HP evidence, and TTP evidence. As already mentioned, an FP charges a user via her HP. This is guaranteed by an HP to an FP through an HP evidence which binds an HP to a user. An HP evidence also contains the verification key of a user in order to allow an FP to verify user signatures. However, an HP evidence is not enough to charge a user. An FP also requires a user evidence of service consumption, i.e., a user signature applied to a statement on service consumption. The timepoint at which this evidence is to be delivered is not predefined by the model. Hence, it may be before, during, or after a service consumption.

A user evidence may not be sufficient to correctly prove a service consumption. This may be due to the fact that the statement signed does not con-
tain the whole consumption information, or the user evidence only proves
the receipt of an encrypted service. In this case, interactions with TTPs are
expected to be able to deliver additional evidences. The model defines these
interactions to be of type event notifications and key submissions. The pur-
pose is to reduce a TTP’s involvement to only dealing with keys and events,
since a stronger involvement (e.g., as a service proxy or to account for serv-
ice consumption) is not desirable.

Instead of generating an unencrypted evidence, a user can generate an en-
crypted evidence. An encrypted evidence is useful in cases where an evi-
dence has to be delivered but should not be valid before the service has been
consumed. The encryption key is known only to the user before it is also
made known to a TTP, who must publish the key after the respective service
has been consumed.

It is in the interest of a service provider to verify the correctness of evi-
dences (i.e., the correctness of statements as well as signatures). An HP must
be able to verify the correctness of signatures of her users. An FP is able to
verify the correctness of signatures of a visiting user with the help of the re-
spective HP evidence.

Using asymmetric keys is actually not the only way to generate an evi-
dence. An evidence can also be generated using a symmetric key. However,
this requires the key to be shared with a TTP who will be responsible for evi-
dence verification. NorCIS* uses asymmetric keys for evidence generations
to avoid this kind of TTP’s involvement.

6.4 The Design of NorCIS* Non-repudiation Protocols

In principle, a party issuing an invoice must be able to prove service con-
sumption. Generally, it is the HP which charges a user for her service con-
sumption in the home as well as in foreign domains. Therefore, an HP is
interested in having user evidences and their related TTP evidences. When a
user is in a foreign domain, it is the FP that delivers services to the user. Nor-
mally, an FP does not charge the visiting user directly, instead, she will
charge the HP of this user. Therefore, an FP is interested in having HP evi-
dences, user evidences, and the related TTP evidences. Furthermore, a user
needs to keep TTP evidences to resolve a possible dispute. Table 6.3 summarizes evidence of interest of each party.

The HP receives her evidences of interest from the FP when the FP wants to charge. At this point of time all evidences are already available. Therefore, no non-repudiation protocol is needed for this phase. However, non-repudiation protocols are required to transfer evidences to users and FPs.

Table 6.3: Parties and evidences of interest

<table>
<thead>
<tr>
<th>Party</th>
<th>Evidences of Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>User evidences, HP evidences, and TTP evidences.</td>
</tr>
<tr>
<td>HP</td>
<td>User evidences and TTP evidences.</td>
</tr>
<tr>
<td>User</td>
<td>TTP evidences.</td>
</tr>
</tbody>
</table>

6.4.1 Transfer of an HP Evidence

An HP evidence is a special evidence not directly related to service consumptions. Hence, its transfer does not need a non-repudiation protocol. An FP needs an HP evidence to verify a user signature. Since a user must be authenticated and authorized before she may consume services, the authentication and authorization interaction is best suited to be used to piggyback an HP evidence to the FP, if the user is in a foreign domain.

6.4.2 Transfer of User and TTP Evidences

User and TTP evidences are directly related to service consumptions. Their transfer requires non-repudiation protocols. As already mentioned, two alternatives exist with respect to the timepoint at which a user evidence should be delivered: proof before consumption and proof after consumption. None of these alternatives supports fairness as one party is always in advantage over the other. In general, any of these two alternatives can be applied if the party in disadvantage is willing to accept the risk of being cheated.

One way to reduce the risk of unfairness is by dividing the whole duration of service consumption into smaller intervals depending on the accounting
needs, and requiring an evidence to be delivered in each interval. However, this does not really solve the problem of fairness in each interval.

In [112] various approaches for a fair non-repudiation protocol in a message transfer process are described with respect to obtaining evidence of origin and evidence of receipt of the message. In this regard, [112] defines that a non-repudiation protocol is fair if it provides the originator and the recipient of a message with valid irrefutable evidence after completion of the protocol, without giving a party an advantage over the other party in any possible incomplete protocol run.

The approaches described in [112] cannot be generally applied to non-repudiation of service consumption due to the fact that a service consumption cannot always be treated as a message transfer. Those approaches are, however, suitable for consumption of content which are not streaming, since the receipt of a non-streaming content can be considered the same as the receipt of a message.

For a service consumption which is accounted continuously as time passes by, a different approach to achieve fairness is needed. In fact, different accounting schemes require different approaches to achieve fairness in non-repudiation of service consumption. The following subsections propose and describe approaches for fair non-repudiation in item-based, time-based, and volume-based accounting schemes. In the description a user U, identified by a User Identifier (UID), interacts with an FP as a service provider, identified by a Service Provider Identifier (SPID). If a TTP is involved, she is identified by a TTP Identifier (TTPID).

**Adapted ZG97 for Item-based Accounting**

Item-based accounting scheme is useful to account for consumption of non-streaming contents. A fair non-repudiation of service consumption can be designed based on one of the approaches for fair non-repudiation in message transfer. In order to show how this works, an efficient and fair non-repudiation protocol with offline TTP presented in [116] is applied as an example.

The general idea of the protocol is to allow parties to exchange messages and evidences directly (in normal case) and to use TTP only if there is a
problem (recovery phase). The other idea of splitting a message into a commitment (encrypted message) and a key is adopted from [115]. The originator sends an encrypted message instead of the plain message to the recipient. The decryption key is sent to the recipient after the originator has received an evidence of receipt of the encrypted message from the recipient. After receiving the key the recipient is expected to send an evidence of receipt of the key. If the originator does not obtain the evidence of receipt of the key she should submit the key to a TTP and retrieves a confirmation from the TTP. The recipient is assumed to be able to retrieve the key from the TTP and hence, cannot deny having obtained the key or being able to obtain the key.

Since non-repudiation of service consumption is interested in evidence of receipt only, the protocol is slightly modified by eliminating the need of evidence of origin. The resulted protocol is termed Adapted ZG97. Furthermore, the message of interest is replaced by the non-streaming content. Note that the communication channels between the TTP and the transacting parties are assumed to be not permanently broken. The following notation is used additionally to specify the protocol:

\[ M \] Non-streaming content.

\[ K \] Key defined by FP to encrypt M.

\[ C = e^k(M) \] Non-streaming content encrypted with key K.

\[ L = H(M, K) \] A unique label linking C and K.

\[ EOR_C = s_{SU}(f_{EOR_C}, \text{SPID}, \text{UID}, L, C, T_{sub}) \] Evidence of receipt of C.

\[ EOR_K = s_{SU}(f_{EOR_K}, \text{SPID}, \text{UID}, L, K) \] Evidence of receipt of K.

\[ \text{sub}_K = s_{FP}(f_{\text{sub}_K}, \text{SPID}, \text{UID}, L, K, T_{sub}) \] Authenticator of K provided by FP.

\[ \text{con}_K = s_{STTP}(f_{\text{con}_K}, \text{SPID}, \text{UID}, L, K, T_{con}) \] Evidence of confirmation of K issued by the TTP.

\[ T_{sub} \] The deadline that FP should either send the key K to U or TTP.
The timepoint at which K is confirmed by the TTP and placed in a publicly accessible repository.

The protocol steps in the normal case are as follows:

1. U => FP: f_{ContentReq}, UID, ContentId
2. FP => U: f_{Content}, L, C
3. U => FP: f_{EOR_C}, SPID, UID, L, T_{sub}, EOR_C
4. FP => U: f_{Key}, L, K
5. U => FP: f_{EOR_K}, SPID, UID, L, EOR_K

U requests a content in step 1. In step 2, FP sends to U the encrypted content C and a label L to identify C throughout the interaction. Upon receiving of C, U sends to FP in step 3 an evidence of receipt of C and a deadline T_{sub} for FP to submit the decryption key K. In step 4, FP sends the key K to U, and U sends the evidence of receipt of K as a response in step 5. Assuming U does not execute the fifth step, FP can enter the recovery phase. The protocol steps in the recovery phase are as follows:

4’. FP => TTP: f_{sub_K}, SPID, UID, L, K, T_{sub}, sub_K
5’. FP <= TTP: f_{con_K}, SPID, UID, L, K, T_{con}, con_K
6’. U <= TTP: f_{con_K}, SPID, UID, L, K, T_{con}, con_K

FP submits K to TTP before T_{sub} together with other information to relate K to the content and the user. As a response the TTP generates an evidence relating all the information and put it in a repository accessible to U and FP. Table 6.4 lists evidences to be kept by an FP.

Table 6.4: Evidences to keep — as generated in the protocol Adapted ZG97.

<table>
<thead>
<tr>
<th>Party</th>
<th>Evidences to keep</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>Two evidences are to be kept:</td>
</tr>
<tr>
<td></td>
<td>• f_{EOR_C}, SPID, UID, L, C, T_{sub}, sS_U(f_{EOR_C}, SPID, UID, L, C, T_{sub}) and either</td>
</tr>
<tr>
<td></td>
<td>• f_{EOR_K}, SPID, UID, L, K, sS_U(f_{EOR_K}, SPID, UID, L, K) or</td>
</tr>
<tr>
<td></td>
<td>• f_{con_K}, SPID, UID, L, K, T_{con}, sS_TTP(f_{con_K}, SPID, UID, L, K, T_{con})</td>
</tr>
<tr>
<td>User</td>
<td>None</td>
</tr>
</tbody>
</table>
Improved HS05 for Time-based Accounting

A time-based accounting scheme is used to account for the duration of a service consumption. A consumption of a streaming service or a voice communication is typically accounted using this scheme. A fair non-repudiation of service consumption must be able to provide an evidence at any instance of time during the service consumption.

Allowing services to be consumed prior to evidence delivery won’t support fairness. Services cannot be encrypted since they must be consumed in real-time or encryption is not possible due to the nature of the services. Hence, an evidence must be generated and delivered prior to service consumption. The solution proposed in this thesis involves an online TTP. It is an improved version of the solution presented by the author in [48]. There, an encrypted evidence is sent by U to FP, and the decryption key is lodged at a TTP during the service consumption.

The improved version, termed Improved HS05, reduces the participation of a TTP in the protocol interactions and the amount of information that a TTP has to deal with. The key idea is to split an evidence of service consumption into an evidence of start consumption and an evidence of end consumption. The evidence of start consumption is sent unencrypted to FP and a TTP is requested to time-stamp a service termination event which can happen during the service consumption. The protocol assumes that a secure communication channel is available between U and FP. It also assumes that the TTP is always reachable during the service consumption. The following notation is used to specify the protocol:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoSC</td>
<td>Statement on service consumption, contains amongst others SPID, UID, service identifier, the intended start time $T_{\text{start}}$, and end time $T_{\text{end}}$.</td>
</tr>
<tr>
<td>SoSC$_{\text{Req}}$</td>
<td>SoSC without $T_{\text{start}}$ and $T_{\text{end}}$, but contains $T_{\text{request}}$.</td>
</tr>
<tr>
<td>SoSC$_{\text{Start}}$</td>
<td>SoSC without $T_{\text{end}}$.</td>
</tr>
<tr>
<td>$L$</td>
<td>$= H(\text{SoSC}<em>{\text{Start}})$. A label identifying SoSC$</em>{\text{Start}}$.</td>
</tr>
<tr>
<td>$T_{\text{end}}$</td>
<td>End time is also a deadline for TTP to time-stamp and publish L.</td>
</tr>
</tbody>
</table>
6.4. The Design of NorCIS* Non-repudiation Protocols

\[
\begin{align*}
\text{EoSC}_{\text{Req}} &= s_{U}(f_{\text{ServiceReq}}, \text{SoSC}_{\text{Req}}) \\
\text{Evidence of service consumption request.} \\
\text{sub}_R &= s_{FP}(f_{RegistrationReq}, T_{\text{end}}, \text{SPID}, \text{UID}, V_U, L) \\
\text{Authenticator of L provided by FP.} \\
T_{\text{reg}} &= \text{The timepoint at which TTP confirms a registration of L.} \\
\text{RegNo} &= \text{A locally unique registration number generated by the TTP.} \\
\text{con}_L &= s_{TTP}(f_{RegistrationConfirmed}, T_{\text{reg}}, T_{TPID}, \text{RegNo}, T_{\text{end}}, \text{SPID}, \text{UID}, V_U, L) \\
\text{Evidence of registration confirmation issued by the TTP.} \\
\text{EoSC}_{\text{Start}} &= s_{U}(f_{StartReq}, T_{TPID}, \text{RegNo}, \text{SoSC}_{\text{Start}}) \\
\text{Evidence of start consumption.} \\
T_{\text{term}} &= \text{The timepoint at which TTP confirms a termination request.} \\
\text{con}_T &= s_{TTP}(f_{TerminationConfirmed}, T_{\text{term}}, T_{TPID}, \text{RegNo}, \text{SPID}, \text{UID}, L) \\
\text{Evidence of confirmation of termination request issued by the TTP (evidence of end consumption).}
\end{align*}
\]

The protocol steps before the start of a service consumption are as follows:

1. U => FP: f_{ServiceReq}, SoSC, EoSC_{Req} [, TTPID]*
2. FP => U: f_{ServiceResp}, L, TTPID
3. FP => TTP: f_{RegistrationReq}, T_{end}, SPID, UID, V_{U}, L, sub_R
4. U, FP <= TTP: f_{RegistrationConfirmed}, T_{reg}, TTPID, RegNo, T_{end}, SPID, UID, V_{U}, L, con_L
5. U => FP: f_{StartReq}, TTPID, RegNo, L, EoSC_{Start}

U sends a service request message to FP in step 1. The message contains information about the service the user wants to consume including the intended start and end time of consumption. She also sends an evidence of service request in the same message, however, the evidence does not contain the start and end time of service consumption. She may suggest TTPs to be involved. The list of TTPs should be predefined by FP, and is accessible to U offline. In step 2, FP sends a response message to U containing information
about the TTP to be involved and a label L to identify the service consumption throughout the interaction.

In the next step, FP registers SPID, UID, $V_U$, L, and $T_{end}$ to the TTP in a registration request message. This step can be performed in parallel with step 2. This message requests the TTP to time-stamp L latest at $T_{end}$ and publish the time-stamped evidence thereafter. This registration by FP allows the TTP to notify service termination event to FP, if U terminates the service consumption via the TTP before $T_{end}$. The message contains $V_U$ (i.e., the public verification key of U), which the TTP needs to know in order to verify service termination request of U.

The registration request is confirmed by the TTP by publishing a confirmation evidence which is retrieved by U and FP in step 4. This TTP confirmation contains a locally (within the TTP domain) unique registration number RegNo and a registration time-stamp $T_{reg}$. TTPID and RegNo are to be used to link an evidence of start consumption with an evidence of end consumption. Knowing now that the TTP holds a valid $V_U$, U is certain to be able to terminate the service consumption once it starts. In step 5, U sends a start request message to FP containing an evidence of start consumption. Now, service consumption may start at $T_{start}$ as defined in SoSC. Note that the TTP in Improved HS05 does not know about the start time of the service consumption, as opposed to [48].

During service consumption either U or FP may stop the consumption and notifies this to the TTP. The protocol steps to stop the service consumption by U are as follows:

6. U $\Rightarrow$ TTP: \text{$f_{TerminationReq}$, TTPID, RegNo, SPID, UID, L, s$S_U$(f$_{TerminationReq}$, TTPID, RegNo, SPID, UID, L)}

7. TTP $\Rightarrow$ FP: \text{$f_{TerminationNotification}$, TTPID, RegNo, SPID, UID, L, s$S_U$(f$_{TerminationReq}$, TTPID, RegNo, SPID, UID, L)}

8. FP, U $\Leftarrow$ TTP: \text{$f_{TerminationConfirmed}$, $T_{term}$, TTPID, RegNo, SPID, UID, L, con$_T$}

In step 6, U notifies the TTP that she wants to terminate the service consumption identified by the tuple TTPID, RegNo, SPID, UID, and L. This message is signed by U. After successfully verifying the validity of this mes-
sage, the TTP forwards this notification to FP in step 7 and publishes an evidence of service termination containing all related information.

The protocol steps to stop the service consumption by FP are as follows:

6'. FP $\Rightarrow$ TTP: $f_{\text{TerminationReq}}$, TTPID, RegNo, SPID, UID, L, $s_{FP}(f_{\text{TerminationReq}},$ TTPID, RegNo, SPID, UID, L)

7'. FP, U $\Leftarrow$ TTP: $f_{\text{TerminationConfirmed}}$, Tterm, TTPID, RegNo, SPID, UID, L, con_T

In this case, the TTP is notified by FP in step 6', but U won’t be notified by the TTP, since it was not U who has registered L. It is assumed that U is able to detect service termination and fetches the evidence from the TTP. The value of $T_{\text{term}}$ marks the real end of the service consumption. Therefore, this protocol is able to provide evidence of real service consumption, if the communication channels are reliable, and the communication delay is negligible. Table 6.5 lists evidences to be kept by FP and U.

**Table 6.5: Evidences to keep — as generated in the protocol Improved HS05.**

<table>
<thead>
<tr>
<th>Party</th>
<th>Evidences to keep</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>$f_{\text{ServiceReq}}$, SoSC, $s_{SU}(f_{\text{ServiceReq}},$ SoSC Req)</td>
</tr>
<tr>
<td></td>
<td>$f_{\text{StartReq}}$, TTPID, RegNo, L, $s_{SU}(f_{\text{StartReq}},$ TTPID, RegNo, SoSC Start)</td>
</tr>
<tr>
<td></td>
<td>$f_{\text{TerminationConfirmed}}$, $T_{\text{term}}$, TTPID, RegNo, SPID, UID, L,</td>
</tr>
<tr>
<td></td>
<td>$s_{TTP}(f_{\text{TerminationConfirmed}},$ $T_{\text{term}}$, TTPID, RegNo, SPID, UID, L)</td>
</tr>
<tr>
<td>User</td>
<td>$f_{\text{TerminationConfirmed}}$, $T_{\text{term}}$, TTPID, RegNo, SPID, UID, L,</td>
</tr>
<tr>
<td></td>
<td>$s_{TTP}(f_{\text{TerminationConfirmed}},$ $T_{\text{term}}$, TTPID, RegNo, SPID, UID, L)</td>
</tr>
</tbody>
</table>

**Improved ZL98 for Volume and Time-based Accounting**

The advantage of time-based accounting is the fact that time dimension is measurable regardless of location and can be synchronized among different clocks. This is not the case with volume-based accounting. Unless a TTP is made responsible for the accounting task, a protocol for a fair non-repudiation of service consumption with volume-based accounting cannot be designed. As already mentioned, the risk of unfairness can be reduced by dividing the whole duration of service consumption into smaller intervals, in each of which an evidence is to be generated. Such solution is of course also valid for time-based accounting.
As described, statements can be generated by the user. This possibility leads to unsolicited evidence generation, which allows for an efficient and simple interaction between the provider and the user. For example, a user can sign the service consumption statement and send it unsolicited each time a certain time has elapsed or a certain volume has been achieved.

For time-based and volume-based accounting, unsolicited evidence generation can make use of chained hashes, which work as follows:

1. The user generates nonce $r$; calculates $H_0 = H(r)$ and chained hashes $H_n = H(H_{n-1})$.
2. The user sends to FP a user evidence containing the last hash in the chained hashes:
   $$U \Rightarrow FP: f_{\text{LastHash}}, \text{SoSC}, H_m, sS_U(f_{\text{LastHash}}, \text{SoSC}, H_m).$$

   For a time-based accounting SoSC must state the time interval, whereas for a volume-based accounting the volume interval. Surely, SoSC must also state all information related to the service consumed. This includes SPID, UID, service identifier, session identifier, and start time of consumption.

3. For all subsequent evidences of the same session: $i=1\ldots n$, $n \leq m$ :
   $$U \Rightarrow FP: f_{\text{PrevHash}}, H(\text{SoSC}), H_{m-i}.$$  

4. If the session is not yet terminated after the first hash $H_0$ in the chained hashes has been sent, repeat from step 1.

Table 6.6 lists evidences to be kept by FP. This protocol improves the protocol proposed in [117] by extending the evidence with service consumption information.

Table 6.6: Evidences to keep — as generated in the protocol Improved ZL98.

<table>
<thead>
<tr>
<th>Party</th>
<th>Evidences to keep</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>$f_{\text{LastHash}}, \text{SoSC}, H_m, sS_U(f_{\text{LastHash}}, \text{SoSC}, H_m)$, $H_{m-i}$</td>
</tr>
<tr>
<td>User</td>
<td>None</td>
</tr>
</tbody>
</table>
6.5 The NorCIS* Architecture

Based on the NorCIS* model the architecture has been developed, including interactions of its key components to allow for the transfer of evidences to the interested party. The architecture as depicted in Figure 6.5 consists of Non-repudiation (NR) components, Authentication and Authorization (AA) components, and Accounting components. The architectural components are grouped into four domains representing the four parties in the model: the home domain, the foreign domain, the TTP domain, and the user’s mobile terminal.

![Figure 6.5: The NorCIS* Architecture.](image)

6.5.1 Architectural Components

To ensure a clear functional description, each of those three groups of components in place are outlined and described as follows.
Non-repudiation Components

The NR components are mainly responsible for non-repudiation interactions, in particular for the generation, transfer, verification, and storage of evidences. These components are an NR Client in the user’s mobile terminal, a Foreign NR Server in the foreign domain, a Home NR Server in the home domain, an NR TTP residing in the TTP domain, Cryptography Modules, and repositories for evidences and keys. A Cryptography Module is responsible for performing cryptographic functions. Furthermore, the NR TTP manages keys submitted by the NR Server or the NR Client for a duration determined by the non-repudiation protocol, and provides for publicly accessible TTP evidences.

Authentication and Authorization Components

The AA components are responsible for authentication of identities claimed by a mobile user and for authorization of service accesses. Those components are an AA Client in the mobile terminal, AA Agents and a Foreign AA Server in the foreign domain, and a Home AA Server in the home domain. The User Profiles contain information of users required for authentication and authorization. The authentication process must also support the establishment of a security association between the mobile terminal and the network point of attachment.

Authentication and Authorization are not within the scope of this thesis, but since an evidence verification deals with user identifiers and failure in evidence verification may lead to a service termination, NR components need to interact with AA components.

Accounting Components

Accounting components are User and Provider Accounting module as well as the repositories for the accounting data. Accounting data which are needed to generate and to verify the statements on service consumption are accessed by the NR components via the User and Provider Accounting module respectively.
6.5.2 User Identifiers

The user’s identity generated upon contract establishment (subscription) is identified uniquely by a Registration Identifier (RegID). The user’s identity contains all information of a user which is required to conclude a contract. In interactions with service providers a user may use other identifiers instead of the RegID. These other identifiers are called Virtual Identifiers (VID) and must be mapped to a RegID prior to their usage. A mapping of a VID to a RegID is called a UID mapping. The concept of VID is based on the research work performed within the Daidalos project [24], [39], in which the author of this thesis is actively involved.

A UID mapping is known only to the HP (and of course also to the user) of the RegID. A VID is generated either by the HP or the user, however a UID mapping must be signed by the user. A digital signature applied to a UID mapping is an evidence of a UID mapping. This evidence gives an HP the guarantee that the signing user, identified by the RegID, agrees to be responsible for payment of services consumed by a user identified by the VID.

The lifetime of a VID is limited, and an HP must ensure that a VID is unique in her domain during its lifetime. Each service consumption is linked to a VID. Therefore, stealing a signature key associated with a VID and impersonating a VID are attractive to a malicious user or provider. To reduce the risk of a VID misuse, several restrictions to the use of a VID can be defined in addition to its lifetime, e.g.:

- Services that a VID may consume.
- Amount of expenses that a VID may spend.

These restrictions are to be defined by the user in agreement with the HP.

6.5.3 User Signature and Verification Key

A user generates a public-private key pair, keeps secret the private key to be used for signing a statement, and gives the public key to the HP on contract establishment. This public key is bound to the user’s RegID and is used for verifying digital signatures of the user. This public verification key must be certified in one of these two ways:
• With TTP: the verification key of a user must be certified by a CA.
• Without TTP: upon conclusion of a contract with an HP a user signs a paper document specifying her verification key.

A user uses several VIDs in consuming services to protect her real identity. However, all these VIDs may not be linked to the same verification key, otherwise, this linkability of VIDs will reveal that it was the same user behind those various VIDs. Hence, each VID is assigned a new public-private key pair for signature generation and verification. The lifetime of a signature and verification key pair bound to a VID is equal to the lifetime of the VID. To avoid being compromised the signature key of a VID can be safely destroyed after its lifetime is expired or if it is not used anymore.

Taking the above description into considerations, a UID mapping must also contain the verification key of the VID. Figure 6.6 illustrates a signed UID mapping which must be sent to the HP, before a user can use a VID.

![Figure 6.6: Illustrating a signed UID mapping.](image_url)
6.5.4 Authentication

Before services can be consumed the user has to be authenticated and authorized by the HP. This subsection describes the integration of the transfer of an evidence of a UID mapping and the transfer of an HP evidence into the authentication interaction. The following notation is used in the description of the authentication process:

AAC  AA Client
AAS  AA Server
H.AAS  Home AA Server
F.AAS  Foreign AA Server
NRC  NR Client
NRS  NR Server
H.NRS  Home NR Server
F.NRS  Foreign NR Server

EoM \[= sS_U(V_{VID}, VID, RegID, Time-stamp, Lifetime)\]
Evidence of UID mapping.

signed_M \[= V_{VID}, VID, RegID, Time-stamp, Lifetime, EoM\]

EoA \[= sS_{HP}(V_{VID}, VID, HPID, Time-stamp, Lifetime)\]
Evidence of payment assurance (HP evidence).

signed_A \[= V_{VID}, VID, HPID, Time-stamp, Lifetime, EoA\]

In the authentication process an Authentication Request message is sent to the AA Server via an AA Agent. An example of an AA Agent is a PANA Agent [30]. The AA Agent has a mediation function and is transparent to the NR components, hence, not further described in this thesis. The Authentication Request message contains amongst others a VID, an HP identifier (HPID), and credentials. The exact protocol message exchange depends on the authentication protocol used, however, the authentication message is expected to be able to carry a signed UID mapping encrypted using the public encryption key of the HP. Note that, the user signature also acts as credentials.
1. AAC => AAS: $f_{AuthReq}$. VID, HPID, $e_{PHp}(signed\_M)$

   If the AA Server is the Home AA Server of the user, then the signed UID mapping is delivered to NRS, after it has been verified by the AA Server and the user is authenticated.

2. AAS => NRS: $f_{UIDMapping}$. signed\_M

   However, if the AA Server is not the Home AA Server of the user, the message will be forwarded to the Home AA Server based on HPID.

2'. F.AAS => H.AAS: $f_{HomeAuthReq}$. VID, HPID, $e_{PHp}(signed\_M)$

   After the user is authenticated, an HP evidence is to be generated.

3'. H.AAS => H.NRS: $f_{HPEvidenceReq}$. signed\_M

4'. H.NRS => H.AAS: $f_{HPEvidenceResp}$. signed\_A

5'. H.AAS => F.AAS: $f_{HomeAuthResp}$. ResultCode, signed\_A

   The response of the Home AA Server contains an HP evidence which assures payment for services consumed by the user. This evidence also certifies the verification key of the VID in order to enable the Foreign NR Server to verify user signatures in a service consumption phase.

6'. F.AAS => F.NRS: $f_{HPEvidence}$. signed\_A

   After authentication the NR Client needs to identify the NR Server to whom evidences are to be sent. This is best accomplished through the use of Authentication Response message to convey the information on NR Server.

7. AAS => AAC: $f_{AuthResp}$. VID, ResultCode, NRServerInfo, Y

   where NRServerInfo contains all necessary information to establish a communication with NR Server, and Y is the service provider’s signature of the message: $Y = s_{SP}(VID, ResultCode, NRServerInfo)$.

   The NRServerInfo is then communicated to the NR Client:

8. AAC => NRC: $f_{NRServerInfo}$. VID, NRServerInfo
6.5.5 Non-repudiation Interaction

While Section 6.4 presents the design of NorCIS* non-repudiation protocol interactions at the user-provider level, this section shows these interactions at the component level. In order to run a non-repudiation protocol, interactions among the NR components, as well as between the NR components and other components in the architecture are needed.

Without TTP

Without involving a TTP only the two unfair alternatives are possible as depicted in Figure 6.7 and Figure 6.8. The interaction in proof after consumption is straightforward. Upon a successful service authorization, the service is started. Accounting data are collected and a statement on service consumption is generated based on the accounting data, then sent to the user to be signed. If the user wants to play fair, she accounts for the consumption, verifies the statement, and signs it if the verification result is positive. However, the user might not want to play fair and NR Client is configured to deliver invalid evidences after the service is consumed.
The interaction in proof before consumption is rather different. Here, accounting data are not yet available, hence, a statement on service consumption is generated based on information about the intended service usage. This information is assumed to be obtained from AA Server which is responsible for authorization of service requests. Accounting at service provider’s side aims at determining the right time to terminate the service or to request the next evidence if the service is to be continued. Accounting at user’s side is useful to determine whether the service provider plays fair. A variant of these interactions is possible, where evidences are delivered unsolicited as defined in Improved ZL98.

With Offline TTP for Item-based Accounting

Figure 6.9 shows the interactions of the architectural components when Adapted ZG97 is used as the non-repudiation protocol for consumptions of non-streaming contents. Here, NR Client and NR Server are also responsible for the transfer of the content. In this regard, NR components take over the job of the service provisioning components. However, one may consider that it is the service provisioning components whose functionality is extended with non-repudiation functions.
6.5. The NorCIS* Architecture

Figure 6.9: NR interaction with offline TTP for item-based accounting.
In the recovery phase the interactions between the NR TTP and the other NR components (i.e., the NR Client and the Foreign NR Server) happen independently. Since there is a time limit set by the user for the provider to submit the decryption key, the NR Client needs not to query the NR TTP endlessly.

**With Online TTP for Time-based Accounting**

The detail interactions of the architectural components in time-based accounting using Improved HS05 are shown in Figure 6.10 and Figure 6.11. While Figure 6.10 depicts the interactions before service consumption, Figure 6.11 depict the interactions to terminate a service consumption either by a user or a provider respectively. The interactions show that in order to support fairness in time-based accounting, NR Client and NR Server need to take over the accounting function.
Figure 6.11: Service consumption termination in Improved HS05.
6.5.6 Time-stamping of Evidences

A signature key can be compromised and used by a malicious party. Therefore, a public verification key must be revocable. This has the consequence that an evidence generated after the revocation of the key is invalid. This also means that an evidence must carry information about its generation time. However, this information must be generated by a TSA.

Time-stamping of user evidences by a TSA is highly recommended for a consumption of an expensive\(^1\) service. A compromised signature key of an HP can be misused by a malicious FP. Therefore, an HP evidence also needs to be time-stamped by a TSA.

6.5.7 Lifetime of Evidences

Most of evidences generated are signed using the signature key of a VID. Requiring each of these evidences to be time-stamped by a TSA is not always desirable, in particular, if the service is cheap and the applied non-repudiation protocol weakly involves or does not involve a TTP (e.g., Adapted ZG97 and Improved ZL98).

However, abandoning a time-stamp by a TSA gives a malicious party a chance to misuse a compromised signature key of a VID, even though the lifetime of a VID is limited. Particularly, a malicious provider will be able to generate fake evidences for services which are supposed to be consumed before the revocation of the verification key and within the time period where the VID is valid. This risk can be reduced by limiting the lifetime of an evidence, e.g., to a billing period. In this case, evidences generated for previous billing periods are worthless.

Figure 6.12 illustrates the benefit of limiting the lifetime of an evidence, if time-stamping of evidences by a TSA is left out. An evidence can only be generated for a service consumed between \(T_{g\text{VID}}\) and \(T_{e\text{VID}}\), and is valid until \(T_{e\text{Evd}}\). Note that, \(T_{e\text{VID}}\) does not need to be smaller than \(T_{e\text{Evd}}\).

\(^1\) Expensive is a relative rate and depends strongly on a consumer’s spending power.
If an evidence is not time-stamped by a TSA, a key compromised before $T_{Evd}$ can be misused even after revocation. Although an evidence is not time-stamped by a TSA, there is no risk if a key is compromised after $T_{Evd}$.

![Diagram showing lifetime of evidence and VID](image)

Legend:

- $T_{g_{VID}} =$ VID generation time
- $T_{g_{Evd}} =$ Evidence generation time
- $T_{e_{VID}} =$ VID expiration time
- $T_{e_{Evd}} =$ Evidence expiration time

Figure 6.12: The benefit of limiting the lifetime of an evidence.

### 6.5.8 Mobility Considerations

A clear impact of mobility on the non-repudiation service exists with respect to two areas: layer-3 handover and inter-domain interoperability.

#### Layer-3 Handover

The NR Server is located within the HP’s and FP’s network, while the NR Client is located within the user’s Mobile Terminal. Due to the fact, that the terminal is mobile its Care-of Address (CoA) can change. Two kinds of addresses are available for an NR Client to use: Home Address and CoA. Table 6.7 compares the use of a Home Address with the use of a CoA, if User Datagram Protocol (UDP) or TCP is chosen as the transport protocol for the communication between the NR Server and the NR Client.

If an NR Server should not rely on Mobile IP and should not be aware of mobility, an NR Agent can be designed to make mobility transparent to the NR Server. Assume that the component in the network which is aware of CoA changes is called an Handover-aware Component (HO-aware Compo-
To hide mobility from this NR Server an NR Agent needs an interface to the HO-aware Component and to other NR Agents.

Table 6.7: Home Address versus CoA.

<table>
<thead>
<tr>
<th></th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Home Address</strong></td>
<td>The use of Home Address makes mobility transparent to transport layer. However, without route optimization (communicating through the Home Agent) it is inefficient due to triangular routing. With route optimization means that the NR Server machine must have a Mobile IP stack (In Mobile IP terminology, NR Server acts as a Correspondence Node).</td>
<td>Recommended because TCP is reliable. UDP is not reliable. Since the transfer of non-repudiation evidences must be reliable (the evidences must not be lost in the transmission), message retransmission and duplicate detection mechanisms must be implemented if using UDP.</td>
</tr>
<tr>
<td><strong>Care-of Address</strong></td>
<td>In order to send a packet to an NR Client, the NR Server must know the current CoA of the Mobile Terminal. This CoA can be obtained from: • Components in the network that know this information (aware of Mobile Terminals’ layer-3 handover), e.g., location server, handover module, AA Server. • The previous packet sent by the NR Client to the NR Server. To obtain a fresh information on CoA in this packet, the NR Client is required to send a notification message to the NR Server after each handover. In this case the NR Client needs to be aware of handover, i.e., it must be told of CoA changes. • The accounting data. To obtain a fresh information on CoA in this data, the component responsible for accounting must send accounting data to the Accounting module after each handover.</td>
<td>Not recommended because of possible TCP session interruption. Recommended because there is no such session interruption. However, UDP is not reliable. Since the transfer of non-repudiation evidences must be reliable (evidences must not be lost in transmission), message retransmission and duplicate detection mechanisms must be implemented if using UDP.</td>
</tr>
</tbody>
</table>

The specification of these interfaces are summarized in Table 6.8. For easier management, each Access Router hosts an NR Agent. In turn, each NR Agent supports a TCP connection with the NR Server and a UDP connection with each of the NR Clients of the Mobile Terminals connected to the respective Access Router. Since an NR Agent acts as a forwarding entity
between an NR Server and an NR Client, the functionality of NorCIS* non-repudiation protocols is not affected.

Table 6.8: NR Agent Interfaces.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>HO-aware Component - NR Agent</td>
<td>Communicating the new CoA of a Mobile Terminal to the NR Agent.</td>
</tr>
<tr>
<td>NR Agent - NR Agent</td>
<td>Forwarding messages from NR Server to NR Clients having changed their points of attachment.</td>
</tr>
<tr>
<td>NR Agent - NR Client</td>
<td>Forwarding messages from NR Server or NR Agent to NR Client. Receiving messages from NR Client to be forwarded to NR Server.</td>
</tr>
<tr>
<td>NR Agent - NR Server</td>
<td>Receiving messages from NR Server to be forwarded to NR Clients. Forwarding messages from NR Client to NR Server.</td>
</tr>
</tbody>
</table>

Inter-Domain Interoperability

For non-repudiation to be effective it must be applicable across administrative domains. To achieve this inter-domain interoperability, a standard transport protocol and a standardized format for these statements and evidences have to be used. This allows for the NR Client to communicate with the NR Server of a different administrative domain. In addition, this also allows for NR Servers of different administrative domains to interact with each other across such domain boundaries.

6.6 Implementation

An NR Client and an NR Server have been prototypically implemented in order to study their non-repudiation interactions and the effort required to provide a non-repudiation service. An NR Client is connected to an NR Server of a service provider, from whom the user is consuming a service. In order to have a reliable communication, TCP is used as the transport protocol.

For each accounting session of a service being consumed an NR protocol session (i.e., an instance of an NR protocol at the NR Client and NR Server)
is run. There can be several NR protocol sessions run in parallel between an NR Client and an NR Server as shown in the example in Figure 6.13. Here, at time $t_1$ the user is consuming 3 services: $S_1$ and $S_3$ from provider $P_1$, and $S_2$ from provider $P_2$. Hence, the NR Client is running 3 NR protocol sessions in parallel at $t_1$ for provable accounting of consumption of $S_1$, $S_2$, and $S_3$. An NR protocol session ends, when the service consumption ends. The following subsections describe the implementation architecture of the NR Client and the NR Server.
6.6.1 NR Client

As shown in Figure 6.14, an NR Client comprises the following components:

- **Control Module:**
  This module has the task to establish connections with NR Servers with the help of the Connection Management module. It is also responsible for controlling each NR Client Protocol Session object.

- **Connection Management:**
  This module listens for incoming connection requests from NR Servers or attempts to connect to an NR Server as requested by the Control Module. Each successfully established connection results in an NR Entity Connection object.

- **Connection Database (Conn DB):**
  This database is used by the Connection Management module to keep information about each connection.

- **NR Entity Connection:**
  This object maintains a connection to an NR Server and provides a method to send NR protocol messages. Incoming messages are delivered to the Dispatcher.

- **Dispatcher:**
  Each message carries the session identifier of an NR protocol session. If a message is received from an NR Entity Connection object, this module looks up in the Session Database, based on the session identifier, to find the correct NR Client Protocol Session object which should process the message. If no matching session can be found, the Session Factory is used to create a new NR Client Protocol Session object, and the message is delivered to this newly created object. A new entry is put into the Session Database.

- **Session Database:**
  This database is used to keep references to existing NR Client Protocol Session objects indexed by session identifiers.

- **Session Factory:**
  This object creates NR Client Protocol Session objects and notifies the Control Module of each newly created object.
NR Client Protocol Session:  
This object is responsible for running the client part of an NR protocol. If the protocol requires a statement to be generated by the NR Client, this object invokes the respective method of the Statement Generator & Verifier to generate the statement. This statement is then signed by the Cryptography Module. If this object receives a statement to be signed from an NR Server, it invokes the method of the Statement Generator & Verifier to verify the statement before signing it. Furthermore, if this object is requested by the Control Module to terminate, it removes its entry from the Session Database and terminates.

- Statement Generator & Verifier:  
  This object consults accounting data if requested to generate or to verify statements. However, in this implementation, pre-assembled statements are generated and statement verifications always succeed.

Figure 6.14: Implementation architecture of the NR Client.
6.6.2 NR Server

The implementation architecture of an NR Server is similar to the one of NR Client as shown in Figure 6.15. An NR Server does need a Session Factory object, since NR Server Protocol Session objects are directly created by the Control Module. This means, that a session is always initiated by an NR Server. An NR Server Protocol Session object runs the server part of an NR protocol. Furthermore, evidences collected need to be stored by an NR Server Protocol Session object. All the other components behave as described in the previous subsection.

Figure 6.15: Implementation architecture of the NR Server.
6.6.3 Cryptography Module

Within the framework of the Daidalos project a shared library called Crypto Interface Library (CIL) and a Key Manager (KM) component to manage identities (in particular VIDs) and their keys have been developed [94]. CIL wraps around OpenSSL [80] to provide a simpler programming interface than OpenSSL provides. Figure 6.16 depicts the implementation architecture of the Cryptography Module whose detail description is given in [94].
6.6. Implementation

On the user side, the Key Store is used to store private and public keys of the user’s VIDs and public keys of service providers, whereas on the provider side, it is used to store the private and public keys of the provider, and public keys of VIDs and other providers. The User Interface allows for manual adding and removing of an identifier and its keys. The NR Wrapper provides an NR Client and an NR Server with the following methods:

```c
int Init(); /* to initialize the internal structures of NR Wrapper */

void Exit(); /* to free memory used by NR Wrapper before termination */

int Sign(unsigned char *data, int datalen, const char *usrId, int alg, unsigned char **signature, int *siglen);
/*
 * data == buffer with data to be signed
 * datalen == length of the above buffer
 * usrId == identifier of the user which does the signing
 * alg == signature algorithm. Defined values:
 *   0 == RSA with SHA1 (PKCS#1 v1.5)
 * signature == will be allocated and filled with signature
 * caller should free() this buffer when not needed
 * siglen == length of the above buffer
 */

int Verify(unsigned char *data, int datalen, const char *usrId, int alg, unsigned char *signature, int siglen);
/*
 * data == data whose signature is to be verified
 * datalen == length of the above buffer
 * usrId == identifier of the user which does the signing
 * alg == signature algorithm. Defined values:
 *   0 == RSA with SHA1 (PKCS#1 v1.5)
 * signature == signature (over data) to be verified
 * siglen == length of the above buffer
 */
```

The prototypical implementation of NR Client and NR Server is able to generate and transfer evidence of service consumption, conforming to the NR protocol designed and implemented in NR Client and NR Server Protocol Session objects.
6.7 Chapter Summary

The NorCIS* architecture extends the Accounting Unit of AURIC’s architecture for SLA compliance auditing with components and functionality to support non-repudiation of service consumption. NorCIS* architecture and its protocol interactions allow for the reliable and secure generation and transfer of irrefutable evidence of service consumption in an inter-domain mobility environment offering Internet services.

This thesis adapts and extends two existing non-repudiation protocols; one to be used for item-based and the other one for volume-based accounting. Moreover, this thesis develops a new non-repudiation protocol for use in time-based accounting, which is fair for the service provider and the service user in the sense that evidences generated can correctly identify the duration of a service consumption and neither the provider nor the user can gain advantage over the other in the evidence transfer. Although a TTP involvement is required to support the above mentioned fairness, the involvement is kept as minimal as necessary.
Chapter 7

Evaluation

The technical and economic feasibility of the auditing framework, developed by the author of this thesis, is analyzed in this chapter on the multi-domain scenario presented in Section 3.3. The feasibility evaluation investigates the efforts to set up and operate a metering, accounting, and auditing infrastructure for verifications of SLA compliances and non-repudiation of service consumption.

This chapter also evaluates the scalability, reusability, processing delay, and memory requirements of AURIC middleware implementation described in Chapter 5. Finally, NorCIS* non-repudiation protocols are evaluated with respect to overhead, possible disputes, and attacks as well as requirements to be met: fairness for providers and users, privacy support, and a minimum TTP involvement.
7.1 Technical Feasibility

In order to allow for compliance verification of SLOs, performance data must be available. An SLO should only be defined if its performance metrics are measurable or computable from measurement data. The performance metrics defined in the multi-domain scenario for network connectivity services are latency, jitter, packet loss rate, and availability, whereas MOS has been specified for perceived quality of transmitted contents.

Furthermore, to support non-repudiation of service consumption, metering of service usage must be done. This should not be confused with performance metering, not only because they serve different purposes, but also because they meter different parameters. With respect to network usage measurements, the amount of traffic volumes generated by each customer is of interest. In the scenario, customers of an NSP are assumed to be always connected to the provider’s network. Hence, it does not make much sense to measure durations of network sessions, i.e., how long a customer is connected. In case of the tele-lecture service, the duration of a participation in a tutorial session is the parameter to be measured, assuming this participation duration is considered in charge calculations.

7.1.1 Metering

The required technical efforts for metering consist of:

• **Efforts to develop metering tools needed, if not already available:** These efforts depend on the complexity of the metric definition and the accuracy required. Although not all metering tools required for measurements of network performance and perceived quality are simple, it is technically feasible to develop them, as shown later in this section. With respect to usage metering, measurements of durations and traffic volumes are of common practice and technically feasible.

• **Deployment efforts (meter installation, configuration, and operation):** To obtain network performance data, a meter is to be deployed on each access router of interest. For assessing the perceived quality of transmitted content, a meter is deployed at each customer side and ETHZ. Finally, the deployment of usage meters is to be done on each access router for network connectivity service and on the central unit of the
tele-lecture service at ETHZ. Figure 7.1 illustrates those measurement points with different types of meter just described. The efforts described for SWITCH are also valid for TeliaSonera and AT&T.

![Diagram of network connectivity service](image)

*Figure 7.1: Deployment of performance and usage measurements.*

**Network Connectivity Service**

Verizon Business measures its network performance using data collected by pings via ICMP. In 5-minute intervals data are collected from designated routers in key network hubs world-wide. All samples from the previous month are used to calculate the monthly average latency and packet delivery statistics [106]. ICMP ping-based measurement is widely used mainly due to its easy deployment.

In order to audit a certain packet loss rate SLO, there is a minimum number of packets to be transmitted in each test cycle, e.g., to audit a 0.2% packet loss rate SLO at least 500 packets are to be transmitted so that one packet is allowed to be lost. Latency and jitter measurements do not have this restriction, and can therefore use measurement configuration as for packet loss rate. This allows the use of the same ICMP ping measurement data set to derive packet loss rate, latency, and jitter.
Suppose $s$ is the packet size at link layer and $b$ is the bandwidth allowed to be used for sending ping requests, then for a packet loss rate $r$, at least $1/r$ packets need to be sent in a test cycle $T$ with an inter-packet interval $I$ greater than $s/b$. The maximum packet rate $p_{max}$ is thus $b/s$. The values of $I$ and $T$ are illustrated in Figure 7.2. Furthermore, equation (7.1) gives the minimum duration of a test cycle $T$, where $rtt_{max}$ is the maximum round trip time.

$$T_{min} = I \times (1/r - 1) + rtt_{max}$$

Figure 7.2: Illustrating inter-packet time interval and test cycle length.

Based on the above description, an example is presented to show how to determine a ping-based measurement configuration for a given SLA. Suppose the packet loss rate should be less than 0.1% and the latency (round trip time) below 40 ms. Suppose also that only 50 kbps is allowed to be used for this measurement and that a 600 bytes link layer packet size$^1$ is chosen as a typical Path Maximum Transmission Unit (PMTU) in the communication path under examination. Therefore, the maximum packet rate $p_{max}$ is 50 kbps / 4800 bits = 10.42 packets/s. This value can also be determined from the graphics in Figure 7.3 (a), which shows maximum packet rate as a function of packet transmission bandwidth for various packet sizes. Suppose now that 10 packets/s (inter-packet interval = 100ms) is chosen for the packet

---

$^1$Assuming the use of an ethernet link, the ICMP payload of a 600 bytes ethernet packet is 546 bytes based on the following calculation: the length of an ICMP header is 8 bytes and the length of an IP header without options is 20 bytes. The ethernet header (without the 4 bytes VLAN-tag) and the Frame Check Sequence require together 26 bytes.
rate, then based on (7.5) the minimum duration of a test cycle for a 0.1% packet loss rate and a maximum latency of 40 ms is $100 \text{ms} \times 999 + 40 \text{ms} = 99940 \text{ms}$, or approximately 100s. The same value can be obtained from Figure 7.3 (b). This configuration means that in each test cycle 1000 packets are to be transmitted during 100 seconds or longer with a data rate of 10 packets/s. The measurement data obtained from a test cycle are used to calculate current packet loss rate, average latency, and average jitter.

![Diagram](image)

(a) Maximum packet rate for various packet sizes  
(b) $T - rtt_{max}$ for various packet rates

Figure 7.3: Determining ping-based measurement configuration.

ICMP ping is also widely used to determine network availability. If in a test cycle a 100% packet loss is experienced, then the network is assumed to be unavailable for the duration of the test cycle $T$, and not only $T_{min}$. Assuming an SLO with a total unavailability of less than 5 minutes in a month, a test cycle length of 100 seconds would allow 100% packet loss in two test cycles. The third failure in a month (summing up to 5 minutes) counts as an SLO violation. Obviously, the same measurement data obtained for the calculation of packet loss rate can also be used to determine network unavailability. Furthermore, if $(T - T_{min})$ is a lot greater than $rtt_{max}$, it is beneficial to send a ping request again shortly before the end of a test cycle, if no ping response has been received so far, since a single ping response in a test cycle already counts as network availability for the duration of the test cycle.\(^1\)

---

1. The statement that a single ping response counts as availability for the whole duration of a test cycle is valid if a 100% packet loss is required to decide on unavailability. However, this is a very debatable statement.
In order to obtain timely and more precise performance data, AT&T develops and deploys an active measurement method using two test sequences: a Poisson and a periodic probe sequence, according to RFC 2330 [82] and RFC 3432 [86] respectively [18]. AT&T divides each 24-hour day into 96 test cycles of 15 minutes. Between two measurement servers of any pair a periodic probe sequence of UDP packets is sent by each of the servers independently in each test cycle. The probe sequence has a random start time within the 15 minute cycle and lasts for 1 minute. The UDP packet size is 60 bytes and the inter-packet interval is 20ms. The periodic probe sequence mimics a real-time VoIP application and can accurately characterize effects of repeating events on real-time application performance [18].

However, the dense test of the periodic probe sequence covers only a small fraction of the test cycle. To increase the probability of detecting possible performance degradation the Poisson probe sequence is designed. A transmission performance degradation which lasts at least 10 seconds is significant and can lead to unavailability from the user’s perspective. Aiming at detecting such performance degradation the Poisson probe sequence is run by each measurement server pair throughout the length of a test cycle, but at lower density. Thus, the average interarrival time of the Poisson distribution is 3.3 seconds and the UDP packet size is 278 bytes.

Surely, to obtain performance data for traffic of a specific service class, the probe sequence must use packets which correspond with the service class to be examined. For example, to denote traffic of the service class “Gold”, the corresponding DSCP is used to mark packets in the probe sequence. This way, the performance information obtained is supposed to be valid for this specific service class.

Finally, monthly average latency measurements of each of the three NSPs: SWITCH, TeliaSonera, and AT&T, can be used to calculate monthly average latency across the three NSPs. However, to obtain monthly average packet loss rate, average jitter, or unavailability information across those NSPs, only worst case estimation can be given by summing up the monthly average packet loss rate, average jitter, or unavailability from each NSP. A much better estimation is possible by using each test cycle’s measurement data of each NSP. Surely, the highest accuracy can be achieved by having measurements carried out between routers of SWITCH and AT&T. The same
consideration is valid for determination of *maximum* one way delay, jitter, or packet loss rate across several NSPs.

**Tele-Lecture**

As already mentioned, customers want to have SLOs which specify guarantees for a high perceived quality of multimedia data at the receiver side. In order to be effective, perceived quality must be measurable. There exist measurement methods which are based on subjective tests, *i.e.*, by involving human subjects for quality evaluations [57], as well as methods based on objective evaluations [20], [56], [58], [61], [62], [64], [100], [110]. Since subjective tests are not feasible for real-time applications, this scenario assumes the use of objective evaluation mechanisms.

Perceived quality can be determined either by comparing the distorted signal (data) with the original or reference signal [12], [56], [58], [90], or by calculating the signal impairment caused by various factors including codec and network performance [20], [61], [62], [100], [110]. In real-time applications, original signals are not available at the receiver side for comparison. Thus, measurements of perceived quality in the tele-lecture scenario must be based on the latter approach.

With respect to objective measurements of voice quality, the ITU-T E-Model [62] can be used to predict voice quality nonintrusively and directly from the network and other system parameters, whereas ITU-T P.563 [61] can be used to estimate score from analysis of the degraded voice signal [100]. The E-Model is a computational model for assessing the combined effects of variations in several transmission parameters that affect conversational quality of 3.1 kHz handset telephony. The primary output from the model is the "Rating Factor" \( R \). Although this can be transformed to MOS to give estimates of customer opinion, ITU-T emphasizes that such estimates are only made for transmission planning purposes and not for actual customer opinion prediction. The calculation of this transmission rating factor is given in (7.2).

\[
R = R_0 - I_s - I_d - I_{e-eff} + A \quad (7.2)
\]

\( R_0 \) is the base factor (basic signal-to-noise ratio) determined from send and receive loudness ratings, the circuit and room noise. \( I_s \) represents im-
pairments that simultaneously occur with the voice signal, e.g., too loud speech level, non-optimum sidetone, quantization noise. \( I_d \) represents impairments caused by delay and echo effects, and \( I_{e-eff} \) is the effective equipment impairment factor caused by low bit-rate codecs and packet losses of random distribution (determined subjectively for each codec and for each packet loss rate). The advantage factor \( A \) allows for compensation of impairment factors when there are other advantages to the user, e.g., the ability to be mobile using cellular networks adds a certain value to \( R \).

The calculation of the base factor and each impairment factor utilizes complex formulae and is restricted to a number of codecs. Hence, a new methodology was proposed by L. F. Sun and E. C. Ifeachor to develop new and efficient models to predict conversational voice quality nonintrusively for different codecs [100]. The new models are regression models whose parameters are obtained from measurements using the combined ITU-T Perceptual Evaluation of Speech Quality (PESQ) [12], [58], [90] and E-Model structure. The advantage of regression models is that they are efficient, straightforward and can be easily used in voice quality monitoring/prediction and perceived quality-driven QoS control (e.g., jitter buffer control and adaptive sender bit-rate control) [100]. The models allow for direct voice quality assessments from network and other relevant system parameters (e.g., delay, jitter, packet loss, and codec type), which are obtained from packet headers (e.g., Real Time Protocol (RTP) headers) analysis. For example, MOS-CQO for different codecs can be estimated to a certain accuracy from network end-to-end delay \( d \) and packet loss rate \( r \) using a general polynomial equation as in (7.3), where the coefficients \( a_i \) to \( a_{10} \) are codec dependent.

\[
\text{MOS-CQO} = a_1 + a_2 r + a_3 d + a_4 r^2 + a_5 d^2 + a_6 rd \\
+ a_7 r^3 + a_8 d^3 + a_9 rd^2 + a_{10} r^2 d
\]  

(7.3)

While perceived voice quality can be easily assessed in a relative accurate way using network performance data, proper assessments of perceived video quality are more difficult to achieve. For example, the IETF proposes a measurement method called Media Delivery Index (MDI) to assess network performance with respect to its real-time delivery [110]. The MDI consists of two components: the Delay Factor (DF) and the Media Loss Rate (MLR). Their values are supposed to provide necessary information to detect all network-induced impairments for streaming video or voice-over-IP applica-
tions. However, using these values alone is insufficient to assess the video quality as perceived by users, since it does not consider impairments caused by codec being used, ignores error control mechanisms in place, and treats all types of video frames the same, although they might be of different importance. For example, an I-frame in Moving Picture Experts Group (MPEG)\(^1\) encoding has a greater entropy (information quantity) than a B- or a P-frame. Also, the size of the Group of Pictures determines the impact of the loss of an I-frame.

In order to assess in real-time the video quality as perceived by users, Symmetricom developed V-Factor [20] which is based on the Moving Picture Quality Metrics (MPQM)\(^2\) model [11]. The information required to calculate the V-Factor quality score is the packet loss rate probability \(P_{lr}\), the image entropy \(\Gamma_r\), and the quantizer value as shown in (7.4). \(Q_{er}(qs)\) is a function of the quantizer value, and it measures the impairments coming from the encoder for a given codec. The packet loss rate probability is a function of various parameters including video jitter buffer length, program clock reference jitter, network jitter and network losses. The image entropy depends on the type and size of the video frame, the ratio of reference frames and non reference frames, bandwidth and bandwidth variation, quantizer and quantizer variation [21]. The V-Factor quality score has a range from 0 to 5, similar to MOS.

\[
V-\text{Factor} = Q_{er}(qs) \times \left(1 - e^{\frac{P_{lr}}{\Gamma_r}}\right) \quad (7.4)
\]

Obviously, in addition to network performance parameters, various transport or application level performance parameters need to be measured in order to obtain a good estimation of perceived video quality. This requires knowledge of the content transport protocol being used, \textit{a.o.}, MPEG-2 Transport Stream or RTP.

---

1. MPEG compression uses 3 types of frames: I-frame (intra-coded picture), P-frame (predictive-coded picture), and B-frame (bidirectionally predictive-coded picture). I-frames are encoded without reference to any other frames. P-frames are encoded with reference to the nearest previous I- or P-frame. This technique is called forward prediction. B-frames are encoded with reference to both the previous and the next P-frame or I-frame. This is called bidirectional prediction. An I-frame followed by a certain sequence of B- and P-frames before the next I-frame define together a Group of Pictures (GOP). The size of the GOP can be set to 8, 12, or 16 to suit various needs.
2. The MPQM model is a spatio-temporal model of the human visual system.
In this scenario, as part of the SLA specification between SWITCH and ETHZ, SWITCH is supposed to measure its network performance and make the data available for auditing purposes. Thus, ETHZ can make use of these data to assess the perceived quality of multimedia data of its tele-lecture service, but as mentioned before, application level performance information is also needed for a good assessment of video quality. In order to obtain this additional information, ETHZ may request a special measurement service from SWITCH or it may integrate its own measurement solution with the tele-lecture infrastructure at its customers’ side.

Having analyzed various measurement techniques as described above, it can be concluded, that it is technically feasible to develop and deploy performance measurements as well as usage measurements for network connectivity services and tele-lecture services.

### 7.1.2 Accounting

The required technical efforts for accounting consist of:

- **Efforts to develop an accounting unit:**
  An accounting unit is needed to aggregate data obtained from one or several metering tools and to calculate usage and performance information over a longer time interval or a larger coverage. Moreover, an accounting unit with non-repudiation support must be able to generate and verify evidences. Hence, an accounting task encompasses in particular data transfer and statistical as well as cryptographical computations. The complexity of a statistical computation lies in formulae to be applied and the amount of data to be processed. The statistical formulae needed by accounting in this scenario are generally simple, namely those for the calculation of totals, average, minimum, and maximum values. The amount of data to be processed is to be investigated in this section. With respect to data transfer, standard protocols are available for use, except for non-repudiation of service consumption, thus NorCIS* is designed. For developing software modules to perform cryptographic functions, security libraries exist to ease implementation.

- **Efforts to deploy an accounting unit:**
  Technically, the deployment of an accounting unit does not require much efforts in general. There might be inter-operability problems in
case of inter-domain data transfer, if protocol implementations of various parties do not fully comply with the specification. Hence, test phases are needed, and those inter-operability problems are technically not really difficult to solve.

In this scenario, those NSPs involved support QoS-enabled network services, and their customers are assumed to be always connected to the network. Thus, for network usage measurements, only the amount of traffic volumes, generated by each customer per service class, is interesting. In this regard, the number of accounting records generated in each month (billing period) is equal to the number of customers, and each record contains the traffic volume of each service class. Concerning usage measurement of the tele-lecture service, the maximum number of records per billing period (normally a semester) equals the number of participating customers multiplied by the number of tutorial sessions. However, for the purpose of non-repudiation, evidence of service consumption needs to be collected. The evaluation of this thesis work on non-repudiation of service consumption, i.e., NorCIS*, is given in detail in Section 7.4.

The amount of performance measurement data generated at a measurement point mainly depends on the SLO. Consider for example an access router pair, one at ETHZ and the other at Uni Bern, to measure network performance between them. Assuming the use of a ping-based measurement method, in every test cycle a number of packets are to be transmitted, captured, and analysed. The outcome consists of one value for packet loss rate and one average value for each metric: latency and jitter. If maximum values are also needed, then one maximum value for latency and one for jitter are to be calculated. Thus, 5 values are obtained per test cycle for a router pair. However, these 5 values can be stored in a single measurement record.

The smaller the length of this test cycle, the more data will be generated in an audit period, e.g., in a month for monthly average guarantees. The minimum length of a test cycle depends on several factors as given in (7.1). The nearer the length of a test cycle chosen to the minimum length, the more data are generated in an audit period, thus the more accurate is the performance evaluation. Choosing the proper length is a trade-off between accuracy and effort. Suppose the following typical constraints must be held:
At most 0.1% of the access link bandwidth $B$ may be used for measurements. For the sake of simplicity, assume uplink and downlink bandwidth are the same.

- Packet loss rate $r$ of 0.05% must be measurable.
- Link layer packet size $s$ for ping measurements is at most 600 bytes.

Further assume that the maximum round trip time (typical value is in milisecond range) is negligible compared to the length of a test cycle (typical value is in minute range), then (7.5) is an approximation of (7.1), if the packet loss rate $r \ll 100\%$.

$$T_{min} = \frac{s}{b \times r} \quad (7.5)$$

Since there are availability and usage measurements beside packet loss rate, latency, and jitter measurements, only 10% of the 0.1% link bandwidth will be used here. Applying this consideration and the abovementioned constraints to (7.5) yields (7.6). Suppose a 1 Gbps link connects the access router to the SWITCH network, then the minimum test cycle length is 96 seconds. This is a feasible value when considering the 5 minutes test cycle length chosen by Verizon [106], which surely does not want to have much measurement traffic load in the backbone.

$$T_{min} = \frac{600 \times 8}{0.1 \times 0.001 \times B \times 0.0005} = 96 \times 10^9 / B \quad (7.6)$$

As stated before, 1 measurement record is generated in each test cycle $T$. Let $c$ be the number of QoS classes supported, and $k$ be the number of router pairs to be tested, then the maximum number of measurement records $N_{max}$, generated in a calendar month, is given by (7.7). Figure 7.4 shows $N_{max}$ as a function of $T$ for different number of router pairs, if there are 3 different service classes to be measured.

$$N_{max} = \frac{31 \times 24 \times 60}{T} \times c \times k, \text{ where } T \text{ is in minute.} \quad (7.7)$$

For a large number of router pairs (e.g., $k = 200$), the number of records generated can be reduced significantly by doubling the test cycle length from 5 to 10 minutes. In fact, the number of records generated in a month, with a 5 minute test cycle length for 200 router pairs, i.e., $5'356'800$ records, is actually not much.
7.1. Technical Feasibility

With respect to availability SLO, the number of records generated is expected to be low. As discussed, in order to allow for 100% packet loss in 2 test cycles, a 100 seconds test cycle length is chosen, if the total unavailability should be less than 5 minutes in a month. To reduce the number of records, only events with 100% packet loss in a test cycle will be stored. As soon as a response to a ping request is received, the running test can be broken until the next cycle. Furthermore, the number of ping requests to be sent in a test cycle need not be many to not overload the network unnecessarily. One ping request every 5 to 10 seconds is sufficient for detecting availability.

![Figure 7.4: N_max in network performance measurements.](image)

Concerning the tele-lecture service, a small test cycle length is to be chosen, in order to react as soon as possible, if there is a performance degradation. In each test cycle, MOS values for video, audio, and data are determined respectively. By using passive measurement method, real traffic...
can be examined, and enough data can be collected and analysed during a short test cycle in a tutorial session. Suppose $L$ is the duration of a tutorial session, $h$ is the number of customers, then the number of measurement records $N$, generated in a tutorial session, is:

$$N = (L / T) \times h$$

(7.8)

Figure 7.5 shows $N$ as a function of $T$ for various number of customers, if the duration of a tutorial session is 90 minutes. As can be seen in the figure, the amount of data generated is also not large for a simple accounting task.

![Figure 7.5: Number of records generated in perceived QoS measurements.](image)

Obviously, accounting efforts in this scenario can be considered low, since neither a complex algorithm nor a high speed processing of large amount of data is required. This is true for accounting of performance data as well as service usage.
7.1.3 Auditing

The required technical efforts for auditing consist of:

- **Efforts to develop an auditing application for each SLO:**
  These efforts are largely reduced with the help of AURIC, the generic auditing framework proposed by this thesis (cf. Section 7.3.2 for the reusability evaluation of AURIC). Without automated auditing performance measurement data must be manually accounted for and audited. If network performance data from two domains are to be audited together, and the data are aggregated over several test cycles to reduce transmission traffic, then care must be taken, because audit results may not be correct. For example, SWITCH’s packet loss rate SLO specifies a maximum value below 0.25% within Europe (based on SLA with TeliaSonera) and 0.1% within SWITCH network. In test cycle $T_1$ and $T_2$ the maximum packet loss rate is estimated as the sum of the rates in both networks. For values given in Figure 7.6, this results in no violation to the SLO. Now, if the maximum packet loss rate is aggregated over $T_1$ and $T_2$, and over both networks, the worst case estimation yields a maximum value of 0.25%, which means an SLA violation. Thus, aggregations are fine as long as no violation is found based on the aggregated values. However, if a potential violation is detected, then a more granular analysis on the level of test cycle is needed.

- **Efforts to deploy an automated auditing infrastructure:**
  A set of automated auditors is deployed to audit various SLOs. The required software installation is straightforward and the use of Sapta function invocation subspecifications simplify a lot the configuration of audit tasks. Therefore, deployment efforts are supposed to be small.

Based on the above descriptions, it is technically feasible to develop and deploy an automated auditing infrastructure in a multi-domain environment. It is not necessary to have all service providers to participate at once in auditing, but only those who have concluded SLAs among each others. This allows for an incremental deployment of automated auditing infrastructure. If a service provider wants to base an SLO on an SLA concluded with another service provider, then it is necessary to have the performance data of the other SLO or its audit results.


<table>
<thead>
<tr>
<th>TeliaSonera Network</th>
<th>$r_{max}(T_1+T_2) = 0.20%$</th>
<th>$r_{max}(T_1) = 0.20%$</th>
<th>$r_{max}(T_2) = 0.05%$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCH Network</td>
<td>$r_{max}(T_1+T_2) = 0.05%$</td>
<td>$r_{max}(T_1) = 0.00%$</td>
<td>$r_{max}(T_2) = 0.05%$</td>
<td>$t$</td>
</tr>
</tbody>
</table>

Legend:
- $r_{max} =$ maximum packet loss rate
- $T_i =$ $i^{th}$ test cycle

*Figure 7.6: Auditing of aggregated measurements across domains.*

### 7.2 Economic Feasibility

This section discusses the costs and gain in setting up and operating a non-repudiation of service consumption and an automated auditing infrastructure in order to justify their economic feasibility.

#### 7.2.1 Economic Efforts

The economic efforts are composed mainly of costs for procurement ($C_P$), installation ($C_I$), and operation ($C_O$) as shown in Table 7.1. In general, procurement and installation costs are spent once as initial costs for the capacity planned for an infrastructure, whereas operational costs are future costs. Procurement costs are determined by the price of each item (hardware or software), their quantity, and the transport costs. Installation costs normally depend on the time required to install, configure, and test all the items needed and the costs of a working hour. Note that, the relation between working hours and number of items to be installed is in general not linear. Operational costs are also a function of the number of items under operation and various properties like robustness and reliability of each item. Moreover,
for each type of costs there are personnel wages and administrative efforts, 
*e.g.*, for correspondences and documentations, to be covered.

Table 7.1: Economic efforts.

<table>
<thead>
<tr>
<th>Type of Costs</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procurement</td>
<td>Hardware and software must be acquired. Hardware include computers, measurement devices if not implemented as software, and storage devices for various databases. Software for metering, accounting, and auditing can be purchased or a software development team is paid to develop the required software. Surely, operating systems and a database management system are needed as well.</td>
</tr>
<tr>
<td>Installation</td>
<td>Hardware and software must be installed, configured correctly, and well tested before operation.</td>
</tr>
<tr>
<td>Operation</td>
<td>During operation, hardware and software need to be maintained, upgraded, repaired, and replaced where necessary. The purposes are, <em>a.o.</em>, to keep or increase the level of efficiency and to adapt to changes in operational conditions, <em>e.g.</em>, SLA redefinitions, tariff structures.</td>
</tr>
</tbody>
</table>

The number of items needed is partly determined by the capacity planned for the metering, accounting, and auditing infrastructure as shown in Table 7.2. This capacity can be seen as the number of customers to be supported.

Table 7.2: Items required by a service provider for automated auditing.

<table>
<thead>
<tr>
<th>Type of Item</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance meter</td>
<td>NSP: 1 per access router</td>
</tr>
<tr>
<td></td>
<td>Tele-lecture Provider: 1 per tele-lecture infrastructure</td>
</tr>
<tr>
<td>Usage meter</td>
<td>NSP: 1 per access router</td>
</tr>
<tr>
<td></td>
<td>Tele-lecture Provider: 1 per tele-lecture infrastructure</td>
</tr>
<tr>
<td>Accounting Unit</td>
<td>1</td>
</tr>
<tr>
<td>Auditing Unit</td>
<td>1</td>
</tr>
<tr>
<td>SLA Manager</td>
<td>1</td>
</tr>
<tr>
<td>Audit Report Handler</td>
<td>1</td>
</tr>
<tr>
<td>Audit Task Planner</td>
<td>1</td>
</tr>
<tr>
<td>Database Management System</td>
<td>1</td>
</tr>
<tr>
<td>(including database storage)</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 7.2: Items required by a service provider for automated auditing.

<table>
<thead>
<tr>
<th>Type of Item</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host computer</td>
<td>1 per meter if it is implemented as software + 6 or more for accounting and auditing (depends on how many instances of architectural components are replicated and distributed to be run on several hosts)</td>
</tr>
<tr>
<td>Operating system</td>
<td>1 per host computer</td>
</tr>
</tbody>
</table>

a. Confer Section 5.1 for details on architecture design.

Suppose \( n_i \) is the amount of item of type \( i \) and \( Price_i \) is the price of this type of item, then (7.9) can be used to calculate procurement costs. Personnel costs in (7.9) are wages to be paid to personnel responsible for procurements. Normally, personnel, administrative, and transport costs are negligible compared to the total item costs, hence, (7.10) is a good approximation of (7.9).

\[
C_P = C_{Personnel} + C_{Admin} + C_{Transport} + \sum_{i=1}^{n_{ItemType}} (n_i \cdot Price_i) \quad (7.9)
\]

\[
C_P \approx \sum_{i=1}^{n_{ItemType}} (n_i \cdot Price_i) \quad (7.10)
\]

Assume that the installation time of an item is approximately the same for any type of item, and assume that the time needed for an installation can be modelled as a function of the number of items, which has a linear and a logarithmic component. The linear component models installation activities that must be carried out sequentially item per item. The logarithmic component models installation activities that can be performed partly in parallel on each item. If administrative costs are negligible, then (7.11) is a good approximation for installation costs. In (7.11), \( n_{Item} \) is the total number of item which is at least 1, and \( a_1, a_2, \) and \( a_3 \) are factors to obtain an appropriate curve.

\[
C_I \approx a_1 \cdot \ln(a_2 \cdot n_{Item} + 1) + a_3 \cdot n_{Item} \quad (7.11)
\]

With respect to operational costs, (7.12) is an approximation of these costs, where:

- \( C_{upg,i} \), \( C_{rpr,i} \), and \( C_{rpl,i} \) are the average costs for an upgrade, a repair, and a replacement of item type \( i \) respectively,
7.2. Economic Feasibility

- $F_{upg,i}$, $F_{rpr,i}$, and $F_{rpl,i}$ is the upgrade, repair, and replacement frequency of item type $i$ respectively, and
- $T_O$ is the duration of operation.

$$C_O \approx T_O \sum_{i=1}^{n_{Item_Type}} (C_{upg,i} \cdot F_{upg,i} + C_{rpr,i} \cdot F_{rpr,i} + C_{rpl,i} \cdot F_{rpl,i})$$ (7.12)

In (7.12), administrative and related personnel costs for update, repair, and replacement administration, are considered low and therefore neglected. Note that, no further personnel are required for the daily operation of an automated auditing infrastructure.

Finally, an example cost calculation for an NSP with 100 access routers to operate an automated auditing infrastructure is presented, assuming the numbers given in Table 7.3. In this example, item replacements are not foreseen for the sake of simplicity and since all items, except host computers and storage, are software which do not need to be replaced, but upgraded. Repair of a software is to be understood as fixes or updates provided at no charge by its manufacturer.

**Table 7.3: Example costs for cost calculation of automated auditing.**

<table>
<thead>
<tr>
<th>Item</th>
<th>Amount</th>
<th>Price [CHF/Item]</th>
<th>Yearly Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>New</td>
<td>Upgrade</td>
</tr>
<tr>
<td>Network performance meter</td>
<td>100</td>
<td>200$^a$</td>
<td>50</td>
</tr>
<tr>
<td>Network usage meter</td>
<td>100</td>
<td>0$^b$</td>
<td>0</td>
</tr>
<tr>
<td>Accounting Unit</td>
<td>1</td>
<td>10'000</td>
<td>250</td>
</tr>
<tr>
<td>Auditing package$^c$</td>
<td>1</td>
<td>15'000</td>
<td>300</td>
</tr>
<tr>
<td>Database Management System (including database storage)</td>
<td>1</td>
<td>2'000</td>
<td>200</td>
</tr>
<tr>
<td>Host computer (including operating system)</td>
<td>110</td>
<td>2'000</td>
<td>250</td>
</tr>
</tbody>
</table>

$a$. The performance meter is assumed to be a software, where one license is needed per host and the price given is the price of a license.

$b$. Commercial router capability normally includes usage metering.

$c$. An auditing package is supposed to consist of an Auditing Unit, an SLA Manager, an Audit Report Handler, and an Audit Task Planner.
Based on (7.10), the procurement costs are approximately CHF 267'000. Note that most of these costs are generated by those metering computers attached to access routers in order to carry out network performance measurements. If those access routers used are programmable, a reduction of the costs of the 100 computers are possible, which is CHF 200’000.

Regarding installation costs, suppose that the following prices are set to obtain the three parameters $a_1$, $a_2$, and $a_3$ of equation (7.11):

- Installation costs for one item are CHF 200.
- Installation costs for 2 items are CHF 350.
- Installation costs for 10 items are CHF 1’000.

Entering these values into (7.11) results in 3 equations with 3 unknowns, whose solution yields: $a_1 = 510.4376$, $a_2 = 0.4398$, and $a_3 = 13.9371$. Therefore, the total installation costs of all those 216 items are approximately CHF 5’340. As explained, there are costs to be paid during operation. For example, the operational costs after 5 years of operation are about CHF 56’000 based on (7.12). Thus, the total costs after 5 years of operation, $C_{\text{Total}(5)}$, are:

$$C_{\text{Total}(5)} = C_P + C_I + C_O(5) = \text{CHF 328’340}.$$

These total costs to be paid by an NSP for setting up and operating an automated auditing infrastructure to monitor network performance with 100 access routers for 5 years can be considered low compared to the potential risk of late handling of SLA violations as discussed in the following subsection.

### 7.2.2 Economic Gain

The economic gain of an operational automated auditing of SLA compliances appears in forms of:

- **Support of marketing**: It is valuable to have the ability to show to customers that commitments are held. Publishing performance data can be seen as a *marketing strategy* to attract more customers in a highly competitive world of
service providers, if a provider is able to show a superior performance of its infrastructure.

- **Faster corrective actions in case of SLA violations:**
  A timely detection of an SLA violation is able to avoid potential greater loss caused by more customer claims.

- **Reduced efforts in Customer Relationship Management (CRM):**
  This is a direct consequence of the previous benefit. Less customer claims also means less CRM efforts.

- **Reduced efforts of technical support team:**
  Technical support team does not need to manually audit the performance of its service infrastructure if there is a customer claim.

- **Support of capacity planning:**
  Knowledge about load and performance of own service infrastructure is useful for capacity planning.

As already mentioned, an SLA also defines reimbursements if a provider does not meet a performance objective. Suppose SWITCH (and the other NSPs as well) specifies the following terms in its SLA concluded with a customer:

- If SWITCH does not meet availability, packet loss rate, delay, or jitter SLO, Customer will be eligible for a credit of 1/30 of total charges Customer has to pay to SWITCH in that month.

- SWITCH requires at most five\(^1\) minutes for the Time to Restore (TTR) from a network outage. An outage occurs if Customer is not able to transmit IP packets through SWITCH network for more than one minute. Measurement of TTR begins when SWITCH Customer Care opens Customer’s trouble ticket and ends with the first attempt to notify Customer on successful recovery. If SWITCH requires a longer TTR, Customer is entitled to a credit of an amount determined by the length of the TTR as given in Table 7.4.

If there is an SLA violation and both of the above terms apply, only the one which results in a higher credit is used to calculate reimbursements. Furthermore, a total reimbursement to a customer in a month is at most equal to

\(^{1}\) AT&T defines a TTR of one minute in its Business Service Guide [8].
the charges to be paid by the customer for that month. Suppose $G$ is the average charges a customer pays in a month and $h$ is the number of customers eligible for a credit due to SLA violations, then (7.13) calculates the total credit to be paid. In (7.13), $C_{TTR}$ is the credit in percentage of monthly charges, which is a function of TTR, as given in Table 7.4.

$$Credit_{Total} = h \times G \times \text{Max}(1/30, C_{TTR})$$

(7.13)

<table>
<thead>
<tr>
<th>TTR</th>
<th>Credit ($C_{TTR}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>From 5 minutes to less than 1 hour</td>
<td>3%</td>
</tr>
<tr>
<td>From 1 to less than 2 hours</td>
<td>5%</td>
</tr>
<tr>
<td>From 2 to less than 4 hours</td>
<td>10%</td>
</tr>
<tr>
<td>From 4 to less than 8 hours</td>
<td>25%</td>
</tr>
<tr>
<td>From 8 to less than 16 hours</td>
<td>50%</td>
</tr>
<tr>
<td>Over 16 hours</td>
<td>100%</td>
</tr>
</tbody>
</table>

For example, a network outage is reported by customers and it takes 4 hours to find the cause and fix it. Now, if there are 20 customers who are eligible for a credit and each pays in average CHF 40’000\(^1\) per month for a 1 Gbps connection and data traffic, then $Credit_{Total} = 0.25 \times 20 \times \text{CHF 40’000} = \text{CHF 200’000}^2$. If this happens more than once in 5 years, the resulted credits will exceed the costs of operating an automated auditing infrastructure described in the previous section. Having an operational automated auditing infrastructure in place, an alarm can be raised if the network is becoming overloaded, hence, counter measures can be taken in time.

---

1. As a comparison, AT&T offers to corporate customers a Managed Internet Service (MIS) with an NxE1 - 6 Mbps port speed for a monthly fee of around US$ 1’000 without Local Access. Local Access is the connection between the customer premises and the nearest AT&T point of presence (POP). However, the price does not increase linearly with increasing bandwidth.

2. Unlike SWITCH, a commercial ISP of similar network size or capacity will have thousands, if not millions, of residential or small business customers, but each customer has a lower bandwidth available and pays much less per month. Therefore, the revenue will approximately be of the same order.
Moreover, in case of an outage, CRM and technical support team have to spend some hours to process trouble tickets, perform auditing, solve the problem, and notify customers on results. However, the related costs are normally negligible compared to the amount of credit. Assume, for example, 12 person-hours are required with a wage of CHF 40 per person-hour, then an additional cost of only CHF 480 is to be accounted.

Concluding, since the economic gain is potentially higher than the efforts, it is economically feasible to setup and operate an automated auditing infrastructure for SLA compliance verification.

7.3 AURIC

Methodologies for the evaluation of software systems have been established to enable an efficient and thorough evaluation from different viewpoints reflecting the interest of various parties to a software system developed. [14] proposes an evaluation framework which is organized along three interconnected dimensions: a software system may be subject to several projects during its lifetime and it operates in an organizational environment. The three dimension system, project, and environment relate to various factors including amongst others product, performance, technology, compliance, usability, and tool. These factors are the target of the software system evaluation. Since this thesis is not about software engineering, the evaluation focuses on the requirements on the properties to be fulfilled by the implemented auditing framework. Thus, the auditing framework is evaluated with respect to the following properties: scalability, reusability, processing time, memory requirements, and reliability.

7.3.1 Scalability

This section evaluates the scalability of AURIC architecture with respect to the number of providers and customers, number of SLAs concluded, number of services a provider offers, number of SLOs, and number of Facts (i.e., accounting records) to be audited.

Normally, in an SLA concluded between two parties, one party takes the role of a service provider and the other of a customer (which can be a service
provider in another SLA). If an SLA is concluded between three or more parties, then only a party takes the role of a service provider. It will be very unusual, though not impossible, to have in an SLA two or more parties taking the role of a service provider. Suppose that there are \( p \) parties in a multi-domain environment and two SLAs are concluded between any two parties (in one SLA a party takes the role of a service provider, in the other SLA the role of a customer). This full mesh relationship results in \( p \times (p-1) \) SLA. However, this does not mean that a single auditing infrastructure is employed to audit performance data against all these SLAs, since from the point of view of each party only \( 2 \times (p-1) \) SLAs are relevant to her. Therefore, an auditing infrastructure is applied to audit performance data against at most \( 2 \times (p-1) \) SLAs. This means, an SLA between \( P_1 \) and \( P_2 \), and an SLA between \( P_3 \) and \( P_4 \) are audited by different auditing infrastructure.

In fact, the number of SLAs is not important, since auditing is performed to detect violations to SLOs and the number of SLOs does not depend on the number of SLAs, but on the number of services. This means, concluding an SLA with another party does not necessarily add new SLOs. Assume that each service has a maximum of \( c \) SLOs and there are \( n_{svc} \) services, then the total number of SLOs to be audited is bounded by \( n_{SLO} = c \times n_{svc} \). The auditing framework supports multiple instances of auditors and each auditor is normally responsible for a particular SLO. Let \( n_A \) = number of auditors required, then \( n_A = n_{SLO} = c \times n_{svc} \), hence the order of \( n_A \) is \( O(n) \). This allows for scaling out the auditor with increasing number of SLOs or services.

Now, for an SLO to be audited a number of Accounting Units and Audit Report Handlers are required to interact with the auditor. Normally, this depends on the SLA which determines the parties responsible for measurements and sending the data to the auditor, and the parties who receive the audit results. In practice, a service provider having millions of customers is not supposed to send the audit results to all customers. Accounting is also not supposed to be performed by each of these customers and the data be sent to the auditor. Instead, the provider (or a third party) is supposed to meter the performance, and audit results are made available to customers via another mechanism, e.g., web access. Thus, the number of Accounting Units and Audit Report Handlers connected to an auditor is generally small (up to 5 Accounting Units and 1 Audit Report Handler) and does not pose a scalability problem to an auditor.
With respect to load scalability of an auditor, the number of Facts to be audited is crucial. There is a limit to the processing speed of an auditor which determines the allowed amount of Facts per time unit (i.e., the Facts rate). The amount of Facts can increase due to e.g., more sessions are generated. The processing speed depends on the complexity of the SLO and the capacity of the hardware resources. Therefore, by scaling up the auditor more Facts can be audited. However, this problem can also be solved by scaling out the auditor, since accounting data for the same SLO can be partitioned (e.g., based on CustomerID) and delivered to several instances of auditors, all responsible for the same SLO.

Therefore, AURIC architecture is scalable based on the above explanations.

7.3.2 Reusability

Evaluating the reusability of the auditing framework aims at showing how generic the auditing framework is, when applied to different use cases. High reusability is one of the most important properties to be fulfilled by the auditing framework. Reusability of the auditing framework is shown in this section by demonstrating that most of the auditing components do not need to be adapted or replaced when developing a new auditing application based on the framework. Assume as an example the following application logic to determine compliances of Facts with a certain SLO:

- If a Fact belongs to the SLO to be audited then \( ff_1(\text{Fact}) \) is true.
- The value of \( gf_1(\text{Fact, open Fact-Lists}) \) identifies the Fact-List to which the Fact belongs (e.g., all accounting records about (un)availability of service S within a month are to be grouped in order to decide on SLO compliance).
- If a Fact-List is complete (ready for compliance evaluation), then \( gf_2(\text{Fact-List}) \) is true.
- A Fact-List complies with the SLO if \( cf_1(pf_1(\text{Fact-List}), pf_2(\text{Fact-List})) \) is 1 (e.g., if service S may down at most 3 times which are longer than 5 minutes, and the total downtime may not exceed 30 minutes, then \( pf_1() \) would count the number of breakdowns longer than 5 minutes and \( pf_2() \) would calculate the total downtime).
• If a Fact-List does not comply with the SLO a report consisting of $	ext{pfi}(\text{Fact-List}), \text{pf}_2(\text{Fact-List}), \text{af}_1(\text{Fact-List}), \text{and cf}_1(\text{pf}_1(\text{Fact-List}), \text{pf}_2(\text{Fact-List}))$ is to be generated.

In order to implement this logic into the framework, none of the components of the SLA Compliance Auditor needs to be modified except the five application specific components which basically consist of subclasses of the five base classes FilterFunction, GroupingFunction, PropertyFunction, ComplianceFunction, and AttributeFunction. This is indeed true as shown in the following code snippets\(^1\), thus the framework is highly reusable.

```cpp
class FF_SL01 : public FilterFunction {
public:
    bool Process(const Fact& currentFact) {
        return ff1(currentFact);
    }
};

class GF_SL01 : public GroupingFunction {
public:
    void Process(const Fact& currentFact, OpenFactLists& ofl) {
        thisFactListId = gf1(currentFact, ofl);
        ofl.Assign(thisFactListId, currentFact);
        if (gf2(ofl.GetFactList(thisFactListId))) {
            ofl.CloseFactList(thisFactListId);
        }
    }
};

class PV_SL01 : public prop_value_t {
    // define variables required to store a property value;
};

class PF_1_SL01 : public PropertyFunction {
public:
    prop_value_t* Process(FactList& currentFactList) {
        PV_SL01* pv = new PV_SL01;
        // assign the result of pf1(currentFactList) to
        // variables in pv

\(^1\)The code is simplified to ease reading (e.g., variable declarations and method definition to create an object of the class are omitted).
class PF_2_SLO1 : public PropertyFunction {
    public:
    prop_value_t* Process(FactList& currentFactList) {
        PV_SLO1* pv = new PV_SLO1;
        // assign the result of pf2(currentFactList) to
        // variables in pv
        return ((prop_value_t*)pv);
    }
};

class CF_SLO1 : public ComplianceFunction {
    public:
    compliance_value_t Process(const PropertyValues& propertyValues) {
        PV_SLO1& pvl =
            (PV_SLO1&) propertyValues.GetPropertyValue(1);
        PV_SLO1& pv2 =
            (PV_SLO1&) propertyValues.GetPropertyValue(2);
        return (cf1(pvl, pv2));
    }
};

class AF_SLO1 : public AttributeFunction {
    public:
    void Process(attribute_value_t& attrValue,
        FactList& currentFactList,
        const PropertyValues& propertyValues,
        compliance_value_t complianceValue) {
        attrValue = af1(currentFactList);
    }
};

Having defined these subclasses, the programming job is done and an executable SLA Compliance Auditor for this specific SLO can be compiled. All other functionality is provided by the framework: interactions with Fact/Report Servers to obtain Facts and to deliver Audit Reports, management of
Facts, Fact-Lists, property values, and execution of methods invoked by audit subtasks, as well as transfer of data between two audit subtasks.

Before invoking the newly developed SLA Compliance Auditor to audit Facts according to the integrated application logic determined by the SLO, a configuration file written in Sapta needs to be created. The framework consults this file to determine which subclasses are to be used by each audit subtasks and to determine the composition of an Audit Report. For the example given above, the content of the configuration file is as follows:

```java
ComplianceCalculation CC_SLO1 {
    FF_SLO1
    >> GF_SLO1
    >> PF_1_SLO1, PF_2_SLO1
    >> CF_SLO1
}
ReportComposition RC_SLO1 {
    [Field1 eq GF_SLO1 >> AF_SLO1],
    [Field2 eq PF_1_SLO1],
    [Field3 eq PF_2_SLO1],
    [Field4 eq CF_SLO1]
}
```

This very flexible configuration possibility allows a developer to define in a modular way various FilterFunctions, GroupingFunctions, PropertyFunctions, ComplianceFunctions, and AttributeFunctions, then configure each instance of the SLA Compliance Auditor to audit a specific SLO. It is likely that several SLOs share the same application logic for specific functions, e.g., a PropertyFunction to determine the average value of a certain field in the accounting records. This subclass needs to be coded once and can be used for various SLOs. Thus, the framework also supports reuse of application logic without code duplication in addition to the reuse of its own components.

In fact, the framework also supports parameterization of the five application specific functions. For example, service S1 may down at most 3 times, while service S2 at most 5 times in a month. It would be very beneficial to be able to use the same application to audit availability of S1 and S2, and without having to hard-coded the value 3 and 5 into the auditing logic. If the
same instance of SLA Compliance Auditor is used to audit the availability of both services S1 and S2, following changes are required:

1. The subclass GF_SLO1 of the GroupingFunction must be able to group accounting records by service name and month, instead of only by month.

2. A new subclass PF_3_SLO1 of the PropertyFunction is required to obtain the service name from the list of accounting records.

3. The subclass CF_SLO1 of the ComplianceFunction must be able to distinguish between the two services being audited and define a method to set its parameters.

4. The configuration file written in Sapta must be adapted.

If the availability of S1 and S2 are audited by different instances of an SLA Compliance Auditor only CF_SLO1 and the configuration file need to be changed to allow the value 3 or 5 to be parameterized. Assuming S1 and S2 are audited by the same instance of an SLA Compliance Auditor the Sapta ComplianceCalculation specification is modified to:

```cpp
ComplianceCalculation CC_SLO1 {
   FF_SLO1
   >> GF_SLO1
   >> PF_1_SLO1, PF_2_SLO1, PF_3_SLO1
   >> CF_SLO1([["S1",3],["S2",5]])
}
```

The changes to CF_SLO1 is as follows:

```cpp
class CF_SLO1 : public ComplianceFunction {
public:
   compliance_value_t Process(const PropertyValues& propertyValues) {
      PV_SLO1& pv1 = (PV_SLO1&) propertyValues.GetPropertyValue(1);
      PV_SLO1& pv2 = (PV_SLO1&) propertyValues.GetPropertyValue(2);
      PV_SLO1& pv3 = (PV_SLO1&) propertyValues.GetPropertyValue(3);
      return (cf2(pv1, pv2, pv3, param));
   }
}
```
bool SetListParam(unsigned int paramNo,
    const SaptaList& paramVal) {
    param = paramVal;
    return true;
}

private:
    SaptaList param;
};

The SetListParam() method will be invoked by the framework immediately after an object of the subclass CF_SLO1 is instantiated. This method will receive as input an index and an object of the class SaptaList containing the values specified in the configuration file and methods to access these values. The auditing logic of CF_SLO1 needs to be adapted as denoted by the function cf2() to be coded by the developer. After these changes the resulted SLA Compliance Auditor will be flexibly configurable. It may even be used to audit availability of new services with other maximum number of downtime without being changed again except the configuration file. Thus, the auditing framework supports the creation of configurable auditors and its components are highly reusable.

7.3.3 Processing Time

Certain auditing applications require fast processing of Facts in order to react immediately based on the auditing results. For example, in a mobile environment, a handover of mobile terminals may need to be initiated if the risk of an SLO violation is getting higher. In this case, a real-time and fast auditing application is required. This section evaluates the implemented auditing framework with respect to the processing time, first with an empty application logic, then on 3 use cases:

- Service Breakdown SLO
- Service Request SLO
- Downlink Throughput SLO

The evaluation is done on a host with a Pentium 4 CPU 1.80 GHz, 512 MB Main Memory, 8 K L1 D Cache, and 256 K L2 Cache. The Facts to be processed are delivered at once to the FFM. The experiments are carried out
with different numbers of Facts: 100, 1000, 2000, 4000, 6000, 8000, 10000, 20000, 40000, 60000, 80000, and 100000. An auditor is normally responsible for a particular SLO, hence, the processing time of an auditor is determined for an SLO, not a set of SLOs. The complexity of an SLO determines the time requirements of an auditor to accomplish its task.

All subtask modules in the subtask chain are implemented as separate threads, hence, switching between subtasks during a run of an audit task is possible. Therefore, duration measurements make sense only between the first input to the first subtask module and the last output of the last subtask module in a chain. To obtain the processing time in each subtask, each experiment is started with only the first subtask module being activated, and then the experiment is repeated 5 times, in which the next module is activated.

![Diagram of subtask chain with measurement points](image.png)

*Figure 7.7: Measurement points for modules’ processing time.*

The resulted duration measurement is the total processing time from FFM to the last activated subtask module in each of the repeated experiment. This duration is denoted \( T_{tp}(M) \), where \( M \) is the last activated subtask module. This is visualized in Figure 7.7 containing only those parts of the implementation architecture of the auditor (cf. Figure 5.4) which are relevant for meas-
uring the processing time. The Control Module is required here to put all subtask modules in action. Each \( T_{tp}(M) \) is measured \( N \) times. The average processing time \( T_{p}(M_i) \) in subtask module \( M_i \) is given by:

\[
T_{p}(M_i) = \text{average}(T_{tp}(M_i) - T_{tp}(M_{i-1}))
\] (7.14)

with a squared standard deviation given by:

\[
\sigma^2_{T_{p}(M_i)} = \text{Var}(T_{tp}(M_i)) + \text{Var}(T_{tp}(M_{i-1})) - 2 \times \text{Cov}(T_{tp}(M_i), T_{tp}(M_{i-1}))
\] (7.15)

Note that \( M_{i-1} \) is the subtask module preceding \( M_i \), and the variance and covariance function used in (7.15) in this thesis normalize by \( N-1 \).

**Empty Logic**

The first evaluation on the auditing framework is done without implementing any auditing application logic. In this regard, all application specific functions, except the grouping function, immediately return upon invocation. The grouping function may not immediately return, since FGM needs to feed PVCM with Fact-Lists in order to allow PVCM and thus, also the following subtask modules to work. Hence, this function composes a complete Fact-List from each Fact. This means, the number of Fact-Lists is equal to the number of Facts.

This evaluation aims at providing an insight into the time requirements of each subtask module merely in the management of Facts. Figure 7.8 shows the average processing time of each subtask module for various number of facts based on measurement data in Appendix C.1. The average value is obtained from 10 runs with the same configurations. The values of the resulted standard deviation are available in Appendix C.1 as well. Averaging is done to reduce the impact of possible variations of the auditing process runtime environment. The relative high processing time of RGM is due to the fact, that RGM deals with I/O operations (in this case writing an empty string to a file for each fact processed). Of the rest of the subtask modules FGM is the most time-consuming, whereas FFM is the least. This is reasonable based on the description of a subtask to be carried out by the respective module.
Figure 7.9 depicts the average processing time per Fact (cf. Appendix C.2 for data). It exhibits an asymptotical decrease of the total processing time per Fact. Note that the x-axis has a logarithmic scale. This behavior can be explained as follows: the total processing time comprises a fixed cost and a variable cost with respect to the number of Facts. As the number of Facts increases, the impact of the fixed cost decreases. Surely, there is a limit to the maximum number of Facts which still shows the decrement of processing time per Fact. If the next subtask module is slower and its queue to accept incoming data is full, then the module delivering the Facts will block. This leads to an increasing processing time per Fact for large number of Facts.
The graph also shows that on the specified testbed machine the overhead of facts handling in the auditing framework is about 90 μsec per Fact for large number of Facts, at least until 100000 Facts. It is also important to note here that about half of this overhead is caused by I/O operations in the last subtask module RGM.

![Graph showing average processing time per Fact](image)

*Figure 7.9: Average processing time per Fact without application logic.*

**The Three Use Cases**

Figure 7.10 depicts the average processing time per Fact in each use case, including the empty logic use case for easier comparison. In all use cases for large number of Facts (until 100000 Facts) the processing time per Fact exhibits a relative constant value as expected. In Service Request SLO Audit,
the application logic (property functions) to be executed by PVCM is time consuming and dominates the overhead of the auditing framework for Facts management. This leads to more idle phases in subsequent subtask modules waiting for data to be processed. Therefore, the graph shows a relative flat run compared to the other use cases.

Table 7.5 provides an overview of all use cases regarding the processing time per Fact for 100000 Facts. Note that the structure as well as the content
of a Fact is different in each use case. Note also that the number of Fact-Lists generated by FGM depends on the application logic and may be different between two use cases. The number of Fact-Lists determines the workload of PVCM and all subsequent subtask modules.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty Logic</td>
<td>100000</td>
<td>&lt;0.5</td>
<td>18</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>46</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Service Breakdown</td>
<td>57674</td>
<td>3</td>
<td>19</td>
<td>6</td>
<td>5</td>
<td>23</td>
<td>24</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>Service Request</td>
<td>99481</td>
<td>1</td>
<td>45</td>
<td>89</td>
<td>32</td>
<td>48</td>
<td>11</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Downlink Throughput</td>
<td>100000</td>
<td>1</td>
<td>21</td>
<td>65</td>
<td>37</td>
<td>42</td>
<td>11</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

Concluding, the processing time of an auditor is strongly influenced by the application logic (i.e., application specific subtask functions to be executed by each subtask module). However, the processing time per Fact decreases with increasing number of Facts, at least until 100000 Facts in the prototypical implementation of the auditing framework.

### 7.3.4 Memory Requirements

This section evaluates the memory requirements of an auditor in relation to the number of Facts for the three use cases. The information on the memory usage is obtained from the /proc files [27]. Here, the virtual memory usage of heap is of interest. All other memory usage information is left out. This includes the static libraries, shared libraries, code, and stack. They all hold the executable code of the application and not the Facts to be processed.

Figure 7.11 shows the heap memory requirements of an auditor. There is a linear increase of memory requirements with increasing number of Facts. Note that in all of the experiments all Facts are delivered at once to the auditor. In real situations, Facts are generated with a certain rate. As long as this
7.3. AURIC

rate is not faster than the processing speed of the auditor, a large number of Facts can be processed.

![Graph showing heap memory usage of an auditor.](image)

Figure 7.11: Heap memory usage of an auditor.

7.3.5 Reliability

The reliability of a middleware can only be evaluated with respect to the functionality to be provided to the application, not the functionality to be provided by the application. Since the logic to audit an SLO is not implemented within the framework, the reliability of the auditing framework cannot be evaluated by examining the audit results when inputs are invalid. However, tests in various use cases show that the framework reliably per-
forms its main functions: transport of accounting data and audit reports, and execution of audit subtasks defined by the application logic.

Furthermore, an auditor is instantiated for each SLO to be audited. If an auditor fails to work, only the respective SLO is affected. However, in cases where the result of the failed auditor is used to audit other SLOs, those SLOs are also affected. This is crucial for auditing applications that have to work in real time, otherwise an audit can be repeated as long as the input Facts and Audit Reports are available. There is however a restriction to this possibility. Auditing logic which uses current time of the day information might need to be adapted.

7.4 NorCIS

The main purpose of collecting non-repudiation evidences is to help resolution in case of a dispute. Possible disputes in billing a service consumption are investigated and evidences collected by each party are evaluated whether they are able to resolve a dispute. In this regard, the fairness support of the proposed protocols is also evaluated.

The support of identity privacy is another requirement to be met (cf. Section 6.2 for the key requirements). To show this support in NorCIS*, component interactions in its authentication phase as well as in its service consumption phase are evaluated. In addition to fairness and privacy support, the involvement of a TTP is required to be minimal, and hence, evaluated and discussed for each protocol proposed and each role of a TTP. Furthermore, the protocol overhead and the costs of unsuccessful transactions are evaluated to show the efforts required for the security gained. Finally, NorCIS* architecture and protocol interactions are also evaluated with respect to reliability, scalability, and possible attacks.

7.4.1 Dispute Resolution and Fairness

This subsection investigates possible disputes and evaluates whether the collected evidences can be used to resolve them. Disputes are possible due to any of the following disagreements:
Disagreements on Identity Relations

Table 7.6 lists possible relations among a RegID, a VID, and an SPID in a service consumption. Any of these relations can be denied by any of the parties involved in a service consumption.

**Dispute 1:** A user, identified by a RegID, denies being a particular VID during a certain time interval.

This dispute is easily solved by using the evidence of a UID mapping: \( s_{U}(V_{VID}, \text{VID}, \text{RegID}, \text{Time-stamp}, \text{Lifetime}) \), which is kept by the user’s HP. This evidence is obtained from the user in the authentication process.

**Table 7.6: Possible relations among a RegID, a VID, and an SPID.**

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegID, VID</td>
<td>A RegID uses a VID for a service consumption.</td>
</tr>
<tr>
<td>VID, SPID</td>
<td>Two possible relations:</td>
</tr>
<tr>
<td></td>
<td>• A VID consumes a service delivered by an SPID.</td>
</tr>
<tr>
<td></td>
<td>• An SPID is the Home Provider of a VID.</td>
</tr>
<tr>
<td>RegID, SPID</td>
<td>In a service consumption this relation is splitted into two relations: a relation between a RegID and a VID, and a relation between a VID and an SPID.</td>
</tr>
</tbody>
</table>

**Dispute 2:** A user, identified by a VID, denies that it was the Service Provider identified by an SPID, who has delivered the consumed service.

In this dispute, the user does not deny having consumed a service, but denies that the service is delivered by a particular service provider. A service provider is able to prove the opposite by presenting an evidence signed by the user in which the SPID and the service are mentioned. Table 7.7 lists required evidences for each of the three NR protocols. Note that an SoSC contains amongst others an SPID, a VID, and service information.
Table 7.7: Required evidences to solve Dispute 2.

<table>
<thead>
<tr>
<th>NR Protocol</th>
<th>Required Evidences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapted ZG97</td>
<td>( f_{\text{EOR}<em>C}, \text{SPID}, \text{UID}, L, C, T</em>{\text{sub}}, sS_U(f_{\text{EOR}<em>C}, \text{SPID}, \text{UID}, L, C, T</em>{\text{sub}}) )</td>
</tr>
<tr>
<td>Improved HS05</td>
<td>( f_{\text{StartReq}}, \text{TTPID}, \text{RegNo}, \text{SoSC}, sS_U(f_{\text{StartReq}}, \text{TTPID}, \text{RegNo}, \text{SoSC}_{\text{Start}}) )</td>
</tr>
<tr>
<td>Improved ZL98</td>
<td>( f_{\text{LastHash}}, \text{SoSC}, \text{H}<em>m, sS_U(f</em>{\text{LastHash}}, \text{SoSC}, \text{H}_m) )</td>
</tr>
</tbody>
</table>

**Dispute 3:** A user, identified by a VID, denies being the customer of a Home Provider identified by an SPID.

An HP can prove the opposite if she can present the following evidences:
- A signed mapping of the VID to the RegID of the user
- A contract document showing the RegID of the user

If the RegIDs of both evidences coincide then the user is proven wrong.

**Dispute 4:** A provider, identified by an SPID, denies being the Home Provider of a user identified by a VID.

This dispute can occur between an FP and an HP. An FP can prove the opposite if she holds an HP evidence showing the relation between the HP and the VID. This evidence is obtained from the HP in the authentication process.

**Disagreements on Services Consumed**

This type of disagreement includes disagreement on service identifiers, time of consumption, and quantity of consumption (i.e., number of items, duration, and volume). Note that, a disagreement on the number of items (contents) consumed is solved by letting FP present an evidence of receipt for each content delivered.

**Dispute 5:** A user, identified by a VID, denies having consumed a particular service.
Evidences collected by an FP and a user in all three non-repudiation protocols are able to identify the consumed service. In Adapted ZG97 a label L identifies the content and links it to the key. In Improved HS05 and Improved ZL98 it is the statement on consumption SoSC that identifies the service consumed. This information is contained in the respective evidence and signed by a VID. Hence, having these evidences at hand, a dispute on a service identifier can be easily solved.

In item-based accounting, consuming a particular service means receiving a particular item (content). Therefore, this dispute requires an evidence of receipt of the content to solve. An evidence of receipt is composed of an evidence of receipt of encrypted content and an evidence of receipt of key. The evidence of receipt of key is obtained either directly from the user or from the TTP. Only if an FP can present both types of evidence of receipt a content is considered received.

A user cannot deny having obtained the key, since she is supposed to retrieve it from the TTP if she does not get it from FP. An FP won't get the evidence of receipt of key if she does not submit the key, either directly to the user or via the TTP. In this regard, the protocol is fair.

**Dispute 6:** A user, identified by a VID, denies a service consumption at a particular time.

Time of consumption is also identified both in Improved HS05 evidences as well as in Improved ZL98 evidences. This information is signed by a VID, therefore, cannot be denied by the signing VID. In Improved HS05 an evidence of start consumption contains this information, whereas in Improved ZL98 it is the SoSC. Evidences in Adapted ZG97 do not contain time of consumption. However, the protocol can be easily extended so that an evidence of receipt also contains the time of receipt.

**Dispute 7:** A service provider and a user, identified by a VID, disagree on the consumption duration of a service.

Consumption duration is only relevant in time-based accounting. Improved HS05 and Improved ZL98 are applicable to this accounting scheme. Therefore, both protocols are evaluated.
In Improved HS05 an evidence is generated for the start time as well as for the end time of a service consumption. The start time is signed by a VID, so it cannot be denied by the signing VID. This evidence is available to the provider before a user consumes a service. Hence, if a provider did not agree with the start time she wouldn’t have delivered the service. The end time of consumption is signed by a TTP, hence, is not deniable by a provider as well as a user.

However, both evidences must be related. A TTPID and a RegNo, which is generated by the involved TTP, link both evidences. Since a RegNo is locally unique in a TTP domain, neither a provider nor a user can obtain from a TTP another evidence of end consumption containing the same TTPID and RegNo. Otherwise, a malicious provider could obtain an evidence with a later end time of consumption by replaying the interaction with the TTP at a later time. Furthermore, Improved HS05 is fair, since evidences generated can correctly identify the start and end time of a service consumption.

In Improved ZL98 an evidence is generated by a user in every time interval of a service consumption. The whole duration is determined by the first and the last evidence. A user cannot deny having consumed a service in this duration, although the real duration may be shorter by one interval. In this regard, fairness is not supported. This evaluation is also valid for a disagreement between a provider and a user with respect to the data volume accounted for a service consumption.

**Disagreements on the Validity of a Signature**

A signature is invalid if the verification key is invalid. A verification key is invalid if it does not belong to the signatory or if it has been revoked before an evidence is generated using the corresponding signature key.

**Dispute 8:** A user, identified by a VID, denies having signed the evidence.

As described in Section 6.5.4 an HP obtains the signed mapping of a VID to the RegID of a user which also contains the public verification key of the VID. The user can be proven wrong if this public verification key can be used to verify the refused evidence.
Dispute 9: A user, identified by a VID, rejects an evidence, which is supposed to be signed using her signature key that had been revoked.

This dispute is evaluated for both of the cases where a time-stamp by a TSA is either needed or not for an evidence. Figure 7.12 depicts the time at which valid evidences can be generated using a compromised signature key.

If each evidence requires a time-stamp by a TSA, then valid evidences cannot be generated after revocation of a signature key, since a provider holding a valid VID will be immediately notified of a revocation by a user. In this case, only the time between key compromise (TcVID) and key revocation (TrVID) allows for generation of valid evidences. This assumes that the signature key is compromised during the lifetime of the VID. If the key is compromised after the lifetime of the VID is expired, then no valid evidence can be generated. Hence, a user will win the dispute if she can prove the time of the key revocation.

The situation is different if a time-stamp by a TSA is not required for an evidence. Here, valid evidences can be generated for any time in the whole lifetime of the VID irrespective of when the key is compromised. A key rev-
ocation also does not help, if it is the provider who is malicious. Hence, a user has to be very careful, if an evidence is allowed to not have a TSA time-stamp. However, a user can win the dispute if she can prove the time of the key revocation and if the generated evidence shows a service consumption after the key revocation. Otherwise, a user will lose the dispute in this case.

### 7.4.2 Identity Privacy

Billing requires accounting data to be assigned correctly to a user, which means, that an accounting record must contain a user’s identifier. Normally, billing is done by the HP. Hence, the HP needs to know the identity of the user. However, an FP does not need to know who is consuming his service. He only needs to know the HP of the user who is responsible for the service consumption. Yet FP and HP must refer to the same user of a certain UID.

A RegID uniquely identifies a user throughout the contract lifetime with her HP. While a user is visiting a foreign domain, this RegID is never revealed to the FP in an authentication process as well as in a service consumption phase. This privacy support is enabled by the following mechanisms:

- In the authentication process an FP only sees the VID of a user. The RegID is encrypted using the public encryption key of the HP.
- During a service consumption a user is identified by her VID and the resulted accounting data are linked to this VID.
- Each VID of the same user has its own public verification key. Thus, two or more VIDs of the same user are not linkable merely through the use of accounting information.

### 7.4.3 TTP Involvement

CAs are generally needed in a system comprising of multi providers, if public and private keys are applied for signature generation and verification. The involvement of a CA is therefore unavoidable. In Adapted ZG97, an offline TTP taking the role of a notary is required, whereas in Improved HS05 an online notary is needed. The notary in Improved HS05 has to provide for a precise time-stamp, since the protocol is used for time-based accounting. Unlike the other two protocols Improved ZL98 does not involve
7.4 NorCIS 229

a TTP. However, as required, none of these protocols needs the involvement of a TTP taking the role of a Delivery Authority.

### 7.4.4 Non-repudiation Protocol Overhead

A non-repudiation protocol increases the number of messages exchanged in addition to service consumption. These protocol messages are an overhead and reduce the effective bandwidth for users’ data traffic. In a wideband access, this overhead is not critical, but in narrow band access technologies with smaller speeds, the overhead of these messages can be relevant. This subsection analyzes the number of message exchanges caused by each proposed non-repudiation protocol for the duration of a service consumption.

Service request and response messages are considered a part of service consumption interaction and do not count as overhead. A retrieval of an information from a TTP is counted as two messages: a message to send a query and a message to send a query result. Table 7.8 summarizes the number of additional message exchanges caused by each of the three non-repudiation protocols. However, these additional message exchanges must be seen in relation to the security gain that users and service providers obtain, and which can be very significant.

**Table 7.8: Non-repudiation protocol overhead.**

<table>
<thead>
<tr>
<th>Parties</th>
<th>Adapted ZG97 [#messages]</th>
<th>Improved HS05 [#messages]</th>
<th>Improved ZL98 [#messages]</th>
</tr>
</thead>
<tbody>
<tr>
<td>U &lt;=&gt; FP</td>
<td>3</td>
<td>1</td>
<td>k (^a)</td>
</tr>
<tr>
<td>FP &lt;=&gt; TTP</td>
<td>3</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>U &lt;=&gt; TTP</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

\(^a\) In Improved ZL98, k is the number of hashes to be sent by a user to the FP during the whole duration of a service consumption.

In Improved ZL98 several evidences need to be generated and sent in regular intervals during service consumption. These intervals between two successive evidences depend on the pricing, hence, the accounting scheme. One way to reduce the impact of these messages is to control their intervals. On one hand, the larger the interval between two successive evidences, the less traffic they generate. On the other hand, a smaller interval has a lower risk of
loss for both the provider and the user. Hence, defining the right interval is a trade-off between communication overhead and business risk.

7.4.5 Evidence Size

Storage space is required to keep evidences generated by a non-repudiation protocol. Hence, this section investigates the size of evidences generated by the three non-repudiation protocols. Particularly of interest is the size of information that needs to be kept in addition to statements on service consumption. Therefore, evidence size considered here refers to this additional information only. This means, those information which have to be there anyway for accounting of a service consumption, e.g., Service Provider Identifier, User Identifier, Service Identifier, start and end time of a service consumption are excluded. For the calculation of an evidence size, typical size of information elements are assumed as listed in Table 7.9.

<table>
<thead>
<tr>
<th>Information Element</th>
<th>Typical Size [bytes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>4</td>
</tr>
<tr>
<td>Hash</td>
<td>32</td>
</tr>
<tr>
<td>Encryption key</td>
<td>24</td>
</tr>
<tr>
<td>Time</td>
<td>4</td>
</tr>
<tr>
<td>Signature</td>
<td>128</td>
</tr>
</tbody>
</table>

Table 7.10: Size of evidences kept by FP and U per service consumption.

<table>
<thead>
<tr>
<th></th>
<th>Adapted ZG97</th>
<th>Improved HS05</th>
<th>Improved ZL98</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size @ FP [bytes]</td>
<td>320</td>
<td>428</td>
<td>192</td>
</tr>
<tr>
<td>Size @ U [bytes]</td>
<td>0</td>
<td>172</td>
<td>0</td>
</tr>
<tr>
<td>Size @ FP [bytes]</td>
<td>192</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Size @ U [bytes]</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Based on NorCIS* non-repudiation protocols description in Section 6.4, a summary of evidence size is given in Table 7.10. As can be seen, the evidence size per service consumption is very small compared to the storage capacity of an optical disk. Assuming 1 million service consumption in a month, those evidences require an additional storage space of less than 500 MB to backup, in addition to traditional accounting records.
7.4.6 Costs of Unsuccessful Transactions

Unsuccessful transactions are rejected service requests or cancelled services before being consumed. They may be caused by DoS attacks. In this subsection, the generated costs in terms of time and space requirements are analysed for each party involved in each non-repudiation protocol proposed. The purpose here is not to compare the different protocols, since they are not intended to be used for the same accounting scheme. Instead, costs generated at each party are to be compared, in particular between the user and the other parties. In this respect, a misbehaving user may not cause too much costs. A protocol starts with a user requesting a service, and the costs accounted for on the user side are only the minimum required costs of a misbehaving user. This means, signature generation costs on the user side are not accounted for.

In Adapted ZG97, a user can cancel a transaction by sending an invalid evidence of receipt of encrypted content. In Improved HS05, cancellation is done by sending an invalid evidence of start consumption. A misbehaving user can go further in the protocol by requesting the TTP to publish an evidence of end consumption which is useless for FP. In Improved ZL98, a user has to deliver an evidence to request a service, hence, a transaction cannot be cancelled if a service request is valid.

The following units are used to denote the costs of processing time of different functions:

- \( T \) = message (with content) transmission or reception
- \( t \) = message (without content) transmission or reception
- \( h \) = hash calculation
- \( e \) = symmetric encryption
- \( s \) = signature generation
- \( v \) = signature verification
- n.a. = not applicable

The following units are used to denote the costs of memory space:

- \( hv \) = hash value
- \( id \) = identifier
- \( ev \) = evidence = signature + statement
Table 7.11 summarizes the costs generated by a cancelled transaction, as analyzed by the author of this thesis. It shows that the costs on the user side are smaller than the costs on the other sides, except on the side of the TTP in Adapted ZG97.

Table 7.11: The costs of a cancelled transaction.

<table>
<thead>
<tr>
<th>Party</th>
<th>Adapted ZG97</th>
<th>Improved HS05</th>
<th>Improved ZL98</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Space</td>
<td>Time</td>
</tr>
<tr>
<td>U</td>
<td>2t+T</td>
<td>0</td>
<td>6t+2s</td>
</tr>
<tr>
<td>FP</td>
<td>T+2t+h+e+v</td>
<td>id+hv</td>
<td>7t+3v+s</td>
</tr>
<tr>
<td>TTP</td>
<td>0</td>
<td>0</td>
<td>7t+2v+2s</td>
</tr>
</tbody>
</table>

The very high costs on the side of the FP in Adapted ZG97 are due to the transfer of the encrypted content. The high costs on the side of the FP and TTP in Improved HS05 are due to the fact, that before a service can be consumed, an evidence is to be generated, transferred, and verified. And also a TTP involvement is to be requested. Therefore, a cancellation of a transaction is expensive. Transaction cancellations can be discouraged by applying cancellation fee, which is usual in many business transactions. Considering only invalid service requests as possible loss, the resulted costs are now much lower as shown in Table 7.12.

Table 7.12: The costs of an invalid service request.

<table>
<thead>
<tr>
<th>Party</th>
<th>Adapted ZG97</th>
<th>Improved HS05</th>
<th>Improved ZL98</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Space</td>
<td>Time</td>
</tr>
<tr>
<td>U</td>
<td>t</td>
<td>0</td>
<td>t</td>
</tr>
<tr>
<td>FP</td>
<td>t</td>
<td>0</td>
<td>t+v</td>
</tr>
<tr>
<td>TTP</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Assume the following values for the above parameters: $T = 20$ s, $t = 50$ ms, $h = 100$ ms, $e = 5$ s, $s = 15$ ms, and $v = 15$ ms. Based on these values, time costs on each party’s side are shown in Figure 7.13 and Figure 7.14 for the two protocols Adapted ZG97 and Improved HS05 respectively. Note the potential costs reduction if cancellations are avoidable.
due to a cancelled transaction

$T + 2t + h + e + v$

due to an invalid service request

Figure 7.13: Adapted ZG97’s time costs of unsuccessful transactions.

Figure 7.14: Improved HS05’s time costs of unsuccessful transactions.
7.4.7 Reliability

One important characteristic of reliability is the ability to restore states after a failure. If an NR component is restarted after a crash, it must continue the protocol run where collected evidences are incomplete, thus, no party can gain advantage over the other. Table 7.13 discusses the behavior of NR components after a restart to achieve reliability.

Table 7.13: Behavior of NR components after a restart.

<table>
<thead>
<tr>
<th>Crashed NR Component</th>
<th>Adapted ZG97</th>
<th>Improved HS05</th>
<th>Improved ZL98</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR Client</td>
<td>If the decryption key of an encrypted content has not been received from NR Server, try retrieving it from NR TTP.</td>
<td>Retrieve the evidence of end consumption from NR TTP if this evidence is missing for a service already terminated.</td>
<td>Normal start.</td>
</tr>
<tr>
<td>NR Server</td>
<td>If the evidence of receipt of decryption key has not been received from NR Client, send key to NR TTP and retrieve the evidence of confirmation of key from NR TTP.</td>
<td>Retrieve the evidence of end consumption from NR TTP if this evidence is missing for a service already terminated by NR Client via NR TTP. If the service was terminated by the provider herself (AA Server), send termination request to NR TTP if evidence of end consumption are not yet obtainable from NR TTP.</td>
<td>Normal start.</td>
</tr>
</tbody>
</table>

Improved HS05 is fair for the service provider and the user with respect to correct accounting of the service consumption duration. However, this relies on the assumption that the TTP involved is always reachable during a service consumption and that the communication delay is negligible. Therefore, if a user is forced to stop consuming a service earlier than the end time pre-defined, but cannot notify TTP to generate an evidence of end consumption, then an unfair situation will happen. Such a situation can occur if a user lose network connectivity for a long period. In this special case, the risk of unfairness is unavoidable. However, a well-behaved provider will gain a
good reputation if despite of this situation the service usage is charged properly.

Furthermore, if for any reason an NR component fails to work permanently no further evidence will be generated. AA components will be aware of this due to missing responses, and service usage will be terminated. However, this is an expected behavior: a service can be consumed if the evidence of consumption can be obtained. Finally, in order to improve reliability, several instances of an NR Server or an NR TTP can be deployed within the respective domain. This reduces the load of an NR component as well as prevents it from becoming a single point of failure for all sessions. Based on the above descriptions NorCIS* architecture and protocols allow for a reliable transfer of evidences.

7.4.8 Scalability

NorCIS* architectural components are distributed across administrative domains and its protocol interactions are defined not only to transfer evidences during service consumptions, but also during user authentications with the help of AA components. Hence, the scalability of the underlying AAA architecture is also relevant to mention. Discussions on the scalability of a AAA architecture are concerned with the number of security associations among service providers. Thereby, the current approach to address this scalability issue is through the use of AAA Brokers [16], [33].

NorCIS* defines for each session of a service consumption non-repudiation protocol interactions between an NR Client, an NR Server, and an NR TTP (if a TTP involvement is required by the protocol). An NR Server is not supposed to know all existing NR TTPs, but only a few ones selected by the service provider to be involved in a non-repudiation interaction, if required. This constant number of NR TTPs involved does not pose a scalability problem to the performance of an NR Server.

Since the purpose of a non-repudiation protocol is not to generate, transfer, or process evidences as many and fast as possible, load scalability of an NR component is not a crucial issue. The timepoint at which an evidence is to be generated and delivered is determined by the protocol state. Nevertheless, it is worthwhile to know the memory requirements of an NR component
with increasing number and duration of sessions. Based on the description of the three non-repudiation protocols, the size of the state information to be kept by each NR component for an on-going session does not grow with increasing duration of the service consumption. This state information which contains information about the protocol state and session state does have a maximum size k. Suppose n is the total number of sessions, then the total size s of state information for all on-going sessions is k*n. Thus, s is O(n) for each NR component, irrespective of how long a session lasts. Therefore, NorCIS* architecture and protocols are scalable.

### 7.4.9 Possible Attacks

This subsection analyzes possible attacks to NorCIS* protocol interactions either by a malicious provider or a malicious user. An attack by a malicious user is motivated by the hope to consume a service without having to pay or to destruct the provider’s service infrastructure, whereas an attack by a malicious provider is motivated by the hope to charge for a fictitious service usage or to charge more for a real service usage.

**Man-In-The-Middle (MITM) Attack**

There are four parties involved in NorCIS* protocol interactions: User, FP, HP, and TTP. Principally, in any interactions between any two of them an MITM attack is possible. In general, this kind of attack is possible if the communication channel is not secured. Table 7.14 discusses counter measures to protect NorCIS* protocol interactions against this attack for all communication channels between these parties.

In fact, a secure communication channel between a user and an FP can be provided on the network layer based on the following reasons. Normally, as a result of a successful user authentication an IPsec tunnel is established between a mobile terminal and the access router. And the communication path between the access router and the NR Server can be secured since it is located within the domain of the FP. Therefore, based on the above descriptions, NorCIS* protocol interactions are protected against MITM attacks.
Table 7.14: Counter measures against MITM attacks.

<table>
<thead>
<tr>
<th>Communication Channel</th>
<th>Counter Measures against Man-In-The-Middle Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP - HP</td>
<td>A secure communication channel can be established between an FP and an HP since two providers who have concluded an agreement share a secret or they are able to obtain each other’s public verification key.</td>
</tr>
<tr>
<td>User - FP</td>
<td>A user has access to or is able to obtain the public verification key of a provider, in this case an FP. However, an FP does not have the public verification key of a visiting user. The public verification key of a visiting user can be securely transfer to the FP via the HP since the HP holds the public verification key of the user. Therefore, a secure communication channel can be established between a user and an FP.</td>
</tr>
<tr>
<td>FP - TTP</td>
<td>Both FP and TTP have access to each other’s public verification key either through a business relation or a PKI. Hence, a secure communication channel can be established between them.</td>
</tr>
<tr>
<td>User - TTP</td>
<td>A user is able to obtain the public verification key of a TTP. However, to secure the communication channel between a user and a TTP, the public verification key of the user needs to be transferred to the TTP. In Improved HS05 this happens with the help of the FP. However, in Adapted ZG97, this is not necessary since a user does not need to send a secured message to a TTP.</td>
</tr>
</tbody>
</table>

**Impersonation Attack**

A malicious provider can try to impersonate a user and generates fake evidences to prove fictitious service consumption. In order to be able to do this, the malicious provider has to generate a public-private key pair, then generates fake evidences using the private key of this pair, and claims that the public key of this pair is a verification key of a certain user. This allows for the provider to charge this user for services she did not consume. However, this is not possible because the verification key will not be valid without being signed by a CA or by the user herself in a contract (paper document).

Fake evidences can also be generated if the signature key of a user is compromised. To reduce this risk several ways have been proposed:

- Require evidences to be time-stamped by a TSA.
• Limit the lifetime of an evidence.
• Limit the use of a VID by limiting its lifetime, its rights to use specific services, and the amount of expenses.
• Destroy the signature key of a VID if it is no longer in use, even if its lifetime has not yet been expired.

Replay Attack

This type of attack can be accomplished successfully if a message or part of it can be retransmitted at a later time without being recognized as such. Replay attacks can be prevented by using nonces (e.g., time-stamps or random numbers) in a message. Table 7.15 describes possible replay attacks, the intention behind these attacks, and how NorCIS* protocols are protected against these attacks.

Table 7.15: Protection against replay attacks.

<table>
<thead>
<tr>
<th>Intention</th>
<th>Replay Attack</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>To obtain an HP evidence to charge the HP.</td>
<td>By replaying an encrypted signed UID mapping.</td>
<td>This replay attack cannot obtain a different HP evidence than the first one, since the digital signature of the user (RegID) is applied to a RegID, a VID, the public verification key of the VID, a time-stamp and a lifetime, which — except the RegID — are contained in an HP evidence. A duplicate HP evidence has no use to an FP. Furthermore, an HP evidence is only useful if there is a statement on service consumption signed by the VID.</td>
</tr>
</tbody>
</table>
| To use the same service twice but to pay only once. | By replaying an evidence of service consumption. | This attack cannot succeed due to the following reasons:  
• In Adapted ZG97, each session of a content transfer is identified by a unique label determined by the FP which is contained in each evidence.  
• In Improved HS05, an evidence of start consumption contains session information including the start time of consumption.  
• In Improved ZL98, each evidence contains session information including the start time of consumption. |
7.5. Chapter Summary

Table 7.15: Protection against replay attacks.

<table>
<thead>
<tr>
<th>Intention</th>
<th>Replay Attack</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>To obtain for an existing evidence of start consumption the corresponding evidence of end consumption at a later time from the TTP in order to account a session duration longer than the real duration.</td>
<td>By replaying the termination request message sent to the TTP or the whole interactions of the FP with the TTP.</td>
<td>This is an attack to Improved HS05. Since an evidence of start consumption is linked to an evidence of end consumption by a unique information comprising of a TTPID and a locally unique RegNo determined by the TTP, a different evidence of end consumption for the same evidence of start consumption cannot be obtained.</td>
</tr>
</tbody>
</table>

**Denial-of-Service**

Having direct communications between an NR Server and NR Clients can endanger the NR Server. One or some malicious NR Clients can generate many invalid evidences which overload the NR Server. This is a Denial-of-Service attack which is hardly avoidable.

**7.5 Chapter Summary**

This chapter has evaluated the auditing framework for usage in a multi-domain environment where several ISPs are linked by SLAs. In this evaluation, the efforts for setting up and operating an automated auditing infrastructure as well as its potential economic gain are investigated. The results of the analysis show that AURIC is technically and economically feasible. Besides, the automated auditing framework developed in this thesis allows a provider to reduce the costs of auditing software development due to its high reusability. Since an auditing application built using the auditing framework is easily reconfigured and adapted to SLO changes, operational costs can also be reduced.

Furthermore, AURIC scales well with increasing number of SLOs to be audited and number of Facts to be processed. The framework functionality is also highly reusable. Implementing an SLA Compliance Auditor using the
framework merely requires the implementation of five subtask functions reflecting the auditing application logic. SLA Compliance Auditors for three specific SLOs have been implemented to evaluate their processing time and memory requirements. Evaluation results show that the processing time per Fact is relative constant to a certain number of Facts, and if all Facts to be processed are delivered at once to an auditor, the memory needs increase only linearly with increasing number of Facts.

NorCIS* protocols and their evidences generated allow for an efficient resolution of possible disputes. One of the major advantages of NorCIS* is its support of fairness for service providers and users in item-based and time-based accounting of service consumption. Besides, the use of VIDs in evidence of service consumption provides for a support of identity privacy, which is often important for mobile users consuming a service in a foreign domain. NorCIS* non-repudiation protocols do generate overhead due to evidence transfer and storage, however, the gain in security, in the sense of being able to prove service consumption, constitutes the main benefit. Furthermore, NorCIS* protocols and evidences collected are able to protect service providers and users against various attacks, including impersonation and replay attacks. Finally, the evaluation shows that NorCIS* architecture and protocols are scalable. In this regard, the memory requirements of an NR component increases linearly with increasing number of sessions, irrespective of their durations.
Chapter 8

Summary and Conclusions

The work on an automated auditing framework for the compliance verification of Internet Service Level Agreements (SLA) has shown in this thesis that such an automated approach is technically feasible, highly valuable for providers and customers, and secure for all participating parties. These features have been demonstrated by AURIC and NorCIS*.

8.1 Summary

Internet Service Providers (ISP) and service consumers negotiate SLAs to define obligations of both parties in their business relationships. A service provider is obliged to deliver services with the committed quality level, whereas a service consumer is obliged to pay for services consumed. Both,
service providers and service consumers, are concerned about whether the other party will fulfill her obligations. The author successfully addresses these concerns in this thesis. The supervision of such obligations is solved by the author with the help of an automated auditing system to examine whether a service provider holds her commitments and a non-repudiation mechanism to prove service consumption.

Existing approaches in SLA compliance auditing lack a general applicability and concentrate on formal specifications of SLAs rather than on the auditing of Service Level Objectives (SLO). These pure specification approaches lead to the potential unawareness of system designers on how complex an SLO for application services can be beyond a guarantee of traditional QoS parameters. This becomes even more important as soon as impacts of mobility and security are to be taken into account. Hence, the SLA Compliance Auditor, developed by the author in this work, is capable of processing complex compliance specifications, which use control structures, relational and logical expressions.

Motivated by the fact that auditing is a task required in various areas of high value service offerings in a commercialized world, where facts of a service delivery are to be examined in an automated way against a set of pre-established specifications, and where their compliance needs to be determined, the author develops the AURIC’s generic model for auditing. The model identifies all key roles involved in an audit and the data they exchange, process, or generate.

The AURIC’s generic and distributed auditing architecture has been designed based on its generic model. Therefore, this architecture determines the basis for the development of the overall SLA compliance auditing architecture and the auditing framework, considered as an application-oriented middleware. To obtain a general applicable auditing framework, a thorough analysis of the auditing process has been performed by the author, which leads to the functional decomposition of an audit task into a chain of subtasks and the identification of various audit specifications required to define those audit subtasks. The auditing framework was designed to implement the common audit functionality, which is necessary to execute audit subtasks. The author shows that this approach is highly reusable, thus applicable to a range of providers. Furthermore, the Audit Specification Language termed Sapta
has been developed by the author, in order to allow audit subtasks to be specified conveniently, in a complete manner, and contradiction-free for providers and customers.

As mentioned, providers are concerned about whether users will fulfill their obligations, especially paying for services consumed. To protect a provider against possible user’s denial of having consumed a service, evidence of service consumption need to be generated by users and transferred to the provider. However, the required non-repudiation mechanism cannot be solved by traditional approaches of non-repudiation for a message transfer process. This is due to the fact that except for non-streaming contents, a service consumption cannot be treated as a message to be transferred, where an evidence of its receipt can be requested. Thus, new mechanisms are essential.

Since a service consumption has a certain duration, non-repudiation of service consumption faces the problem of supporting fairness for the provider and the customer. It turns out that each specific accounting scheme requires a different approach to achieve fairness. Therefore, this thesis designs a fair non-repudiation protocol for time-based (or stream-based) accounting with the involvement of an online Trusted Third Party (TTP). The key idea of the author is the split of an evidence of service consumption into an evidence of start consumption and an evidence of end consumption. The evidence of start consumption can be delivered directly by a user to a service provider without having a risk to have to pay. To end an ongoing service consumption one of the party, either the user or the provider notifies the service termination to the TTP involved to obtain an evidence of end consumption. Since this end time is signed by a TTP, fairness for the provider and the user can be guaranteed. Without a TTP’s involvement, either the provider or the user may have an advantage over the other, which is unfair. In addition, to cover the two other important, but different Internet service types, item-based and volume-based ones, this thesis adapts two existing non-repudiation protocols to address the key requirements for the item-based and volume-based accounting.

Furthermore, the NorCIS* architecture proposed supports also identity privacy for service consumption in a Mobile IP-based environment. This is achieved through the use of Virtual Identifiers (VID), which a user can gen-
erate by herself. However, each VID must be mapped to a Registration Identifier (RegID) prior to its usage, and the mapping is to be sent encrypted to the Home Provider. This thesis proposes the efficient way to piggyback the encrypted signed mapping in an authentication request message, when the user registers to the Home Provider. In addition to the support of this identity privacy, NorCIS* evidences are able to resolve various disputes between a user and a service provider, and its protocol interactions are able to protect users and service providers from various attacks.

8.2 Discussion of Results

Since AURIC’s generic model, architecture, and framework for automated auditing of SLA compliances proposed are complemented with a language to specify audit tasks, and the accounting architecture and process have been extended with non-repudiation functionality to provide for evidence of service consumption, the solutions developed add complexity to the traditional accounting and auditing processes. However, this effort, possibly considered as a drawback, brings a large benefit into the system: the reliability that neither a provider nor a user can cheat each other without being recognized by the other party. Thus, this complexity is necessary to meet all major requirements on system properties, and especially to reduce or to solve efficiently much more costly conflicts that might arise in a business relation between a provider and a customer. Hence, the approach developed saves efforts and costs, very much like insurances, which require regular premiums to be paid, but which cover in cases of necessity the current costs involved.

As shown, SLAs define obligations of contractual parties. Conflicts arise, if a party is not able to fulfil her obligations. Measurement data on usage and performance are pre-requisites to solve a conflict, however, without an additional mechanism for conflict resolution this becomes complicated and costly. Having an operational SLA compliance auditing infrastructure and an accounting of service consumption enhanced with non-repudiation capability in operation, the author shows that following efforts can be largely reduced:

- **Improvements of Customer Relationship Management (CRM):**
  Since an SLA violation or a potential SLA violation is detectable in an automated fashion the first time it appears, corrective actions can be taken in time to avoid further violations. This leads to a smaller
number of claims from customers, which in turn lowers the workload of CRM to address customer claims. Less claims regarding billing or disagreements for usage-based accounting schemes is also achieved, since evidences of service consumption can be presented. If there is a dispute regarding usage-based accounting schemes, all collected evidences allow for a fast and efficient dispute resolution.

- **Economic efforts to reimburse for customer loss:**
  Since further SLA violations can be avoided in time, reimbursement costs do not grow.

- **Efforts of technical support:**
  Without the help of an automated SLA compliance auditing, the staff of the technical support team have to examine performance data manually to find SLA violations or to prove a customer claim. This task encompasses operations dealing with database queries and analysis of related performance records, which might be very complex, if the corresponding SLO is complex. Automation takes over this task and the audit reports resulting automatically ease the task to prove a customer claim, if it comes this far.

- **Efforts in accounting and billing:**
  Accounting staff do not expect to have to recheck accounting data, since usage evidences are available and can be made accessible to customers.

- **Efforts of technical developments:**
  Once an automated SLA compliance auditing infrastructure based on a generic framework is established, adding or changing SLOs requires far less efforts than the initial setup in the development of an auditing application. Thus, the framework provides for an extensible and flexible approach in support of various Internet service auditing tasks.

As mentioned above the complexity added to the traditional accounting and auditing process leads to the gain of those advantages just discussed. Moreover, additional important properties have been achieved. These valuable properties — and at the same time sources of complexity — include:

- **Generality:**
  In general, especially when dealing with application services, SLOs can be arbitrarily complex. Therefore, a simple logical expression is in-
sufficient to express compliance conditions. In order to be generic, *procedural statements* must be allowed in the specification of audit tasks to be accomplished by an auditing application. The need to describe *in detail* the steps to perform an audit task is a source of complexity. Given a set of measurement data a generic auditing framework is not able to know, without being told, how to relate those data, how to compute the required values for compliance verifications, and what information is to be reported. All these require a *precise* specification in a computer language. Thus, complexity shows the gain of precision.

- **Security support:**
  Security support also adds complexity to a system, which is true for any distributed system in general. However, reducing complexity at the expense of security is not recommended for an open system. There are two meanings of security support: (a) the system requires security support to function properly or (b) the system provides for a security support. In case of SLA compliance auditing, access control and secured data transfers are required for inter-domain interactions. Since digital signatures using asymmetric keys are applied in non-repudiation of service consumption an operational Public Key Infrastructure (PKI) is required to provide for certification and certificate revocation functionality. Moreover, the support of identity privacy adds complexity in the authentication protocol interactions. However, the benefit lies on the secured system itself.

- **Fairness support:**
  Another source of complexity is driven by the requirement to support fairness between the provider and the user, especially in case of accounting the service usage. Without an involvement of a TTP the risk of unfairness can be reduced by considering smaller interval of evidence delivery, however, not eliminated at all. Thus, the newly developed non-repudiation protocol for time-based accounting as well as the adapted non-repudiation protocol for item-based accounting eliminate any unfairness.

An automated auditing infrastructure based on the generic framework proposed by the author consists of machines to host the SLA compliance auditing application and to store auditing results. Since metering and accounting infrastructure is supposed to be available, they are required irrespective
of whether auditing is automated or not, the additional hardware effort in support of the automated auditing approach is kept minimal.

The technical effort to develop an auditing application for a specific SLO is minimal as well, since developers need to be concerned about application specific logic only. Operational efforts required include the setup of security associations among interacting application components, which are distributed across domains. Although this requires communications between the technical staff of providers involved, once the automated auditing infrastructure is put into operation, human intervention is needed only, if an SLA violation requires corrective actions.

The economic efforts for an automated auditing framework to be put into operation comprise the costs to accomplish the technical work, in particular the investment to establish and operate such an automated auditing system. Costs are also required to train technical staff to operate and maintain the auditing system. These costs will be amortized in the same manner as high value service usage will be accounted for correctly and potential provider as well as user misuse will be minimized. If these technical and economic efforts are small compared to the costs that can be saved, once the infrastructure is operational, the employment of an automated auditing infrastructure is justifiable. Based on the feasibility evaluation by the author and the descriptions given above, this is the case.

### 8.3 Conclusions

AURIC’s auditing model and architecture proposed in this thesis identify components and key interactions common to auditing in various areas, therefore, the model and architecture are generic. With respect to inter-domain interactions the security and reliability considerations described in this thesis are feasible and allow for the generic architecture to be applied to a multi-domain environment.

AURIC’s architecture for automated auditing of SLA compliances has been designed based on the generic model and architecture. The architecture neither assumes specific services nor specific SLOs. In this respect, the architecture is general and applicable to various services and SLOs. Thus, the
full range of Internet service types can be supported, such as network services, application services, and contents.

The detailed auditing framework has been designed based on this architecture and the audit task decomposition. The auditing framework implements the required common audit functionality and offers an Application Programming Interface (API) to implement the application logic for auditing a specific SLO. Additionally, the API allows for a modular structuring of the application logic. This results in a highly reusable auditing framework. Furthermore, the API is simple and a set of special classes is provided to ease manipulation of accounting data (Facts). A developer also does not need to be concerned about the control of data flow, management of audit data, and data transport. Therefore, the efforts to develop an auditing application based on the auditing framework are largely reduced.

Based on the discussion above the technical and economic efforts are small, but the advantages gained through the operation of an automated SLA compliance auditing infrastructure are larger. Furthermore, since a provider only needs to specify SLOs, whose metrics are measurable, a provider can deploy her own automated auditing infrastructure, irrespective of whether the other providers, who have SLAs negotiated with this provider, also operate an auditing infrastructure. Hence, establishing and operating an automated auditing infrastructure based on the proposed architecture is feasible.

In order to be able to develop a specific auditing application for any SLO conceivable a list of language constructs required to specify an audit task has been compiled by the author. Sapta is the Audit Specification Language tailored for this need, and it is designed to support all these constructs with a simple syntax. Thus, Sapta provides for a convenient way to specify an audit task.

Finally, with respect to non-repudiation of service consumption NorCIS* non-repudiation protocols are able to support fairness or they reduce the risk of unfairness. Furthermore, its authentication interactions and evidence composition support the use of VIDs, thus identity privacy of a user. Therefore, the NorCIS* architecture and its protocols are suitable for providing non-repudiated consumption of Internet services in an environment with a low trust, e.g., when a mobile user is visiting a foreign domain.
8.4 Future Work

While many open issues have been solved and solutions have been provided as well as discussed, open issues always can be found in a thesis. Therefore, the most important ones are briefly introduced at this stage. While SLOs are defined in an SLA to capture customers’ requirements on service performance (QoS), ideally, each SLO for a particular service defined together with a customer may be specific to that customer; or even more specific, if for the same service a customer is allowed to define various SLOs for various usage situations. Pushing this further to the very extreme of flexibility means to allow for the dynamic negotiation of SLOs within a lifetime equalling to a session (service usage) lifetime.

There are some consequences of enabling this dynamic establishment of SLOs to the whole service provisioning and auditing system, namely:

- A protocol for SLO negotiations will be required. This will be most probably similar to a QoS negotiation approach, but the protocol has to consider the legal binding aspect of an SLO negotiation, which adds complexity to the protocol, and thus also to session setup.

- In addition to an SLO negotiation protocol, performance of service provisioning entities must be configurable for each session of a service according to the negotiated SLO. However, this is already the case as soon as several classes of a service are to be supported. It is irrelevant for the service provisioning entities whether configurable target performance values are the result of a negotiation or pre-defined.

- Session data must contain the negotiated SLOs so that performance measurement data can be correctly linked to the corresponding SLO. In this regard, an SLO becomes a Session Level Objective.

- A dynamic reconfigurable auditing system will be required. Static configurable auditing systems, where each SLO defines a certain configuration, are insufficient to handle SLO changes during auditing.

Another problem to be solved is load balancing of auditors, which may need to run hundreds or even thousands of tasks. Since different auditors may have different workload depending on the complexity of the auditing application logic and the amount of Facts to be processed, it is beneficial to
have an appropriate load balancing support in case Facts can be partitioned while still allowing for an auditing, or if an auditor is able to execute different application logic.

The need for a TTP involvement in current non-repudiation protocols to support fairness may be considered as a drawback that requires further, most probably completely new approaches, to solve the fairness problem between providers and users. However, the effort to support fairness may not exceed the risk of unfairness. Solutions by reducing unfairness risks seem to be more feasible than inefficient solutions to acquire complete fairness.

Current solutions to protect against misuses of digital signatures are provided through time-stamping of evidences by a Time-Stamping Authority (TSA) and revocation of the verification key. This causes an overhead in signature generation and verification. Hence, more efficient mechanisms to protect against misuses of digital signatures are necessary. Here as well, mechanisms to reduce the risk of misuses, e.g., through limitations of the lifetime of a signature and an evidence may be more feasible.
Appendix A

Mathematical Notation

The following notation is used in NorCIS* protocol specifications:

A => B: M  Party A sends message M to party B.
A <= B: M  Party A retrieves message M from party B.

con_K = s_{TTP}(f_{con_K}, SPID, UID, L, K, T_{con})
Evidence of confirmation of K issued by the TTP.

con_L = s_{TTP}(f_{RegistrationConfirmed}, T_{reg}, TTPID, RegNo, T_{end}, SPID,
UID, V_U, L)
Evidence of registration confirmation issued by the TTP.

con_T = s_{TTP}(f_{TerminationConfirmed}, T_{term}, TTPID, RegNo, SPID, UID,
L)
Evidence of confirmation of termination request issued by the
TTP (evidence of end consumption).

dK(X)  Information X decrypted with key K.
eK(X)  Information X encrypted with key K.
EoA = s_{HP}(V_{VID}, VID, HPID, Time-stamp, Lifetime)
Evidence of payment assurance (HP evidence).
EoM = sSu(VVID, VID, RegID, Time-stamp, Lifetime)
Evidence of UID mapping.

EOR_C = sSu(fEOR_C, SPID, UID, L, C, Tsub)
Evidence of receipt of C, where C = eK(M), i.e., a non-streaming content M encrypted with key K.

EOR_K = sSu(fEOR_K, SPID, UID, L, K)
Evidence of receipt of K.

EoSCReq = sSu(fServiceReq, SoSCReq)
Evidence of service consumption request.

EoSCStart = sSu(fStartReq, TTPID, RegNo, SoSCStart)
Evidence of start consumption.

fm A flag indicating the intended purpose of a message.

H(X) A one-way hash of information X.

Hi The i-th hash in a hash chain.

K A key.
In protocol Adapted ZG97, K is a key defined by FP to encrypt non-streaming content M.

L A unique label in a protocol run.
In protocol Adapted ZG97, L = H(M, K), where M is the non-streaming content and K is the key to encrypt M.
In protocol Improved HS05, L = H(SoSCStart).

PA Public encryption key of party A.

P^A Private decryption key of party A.

RegNo A locally unique registration number generated by the TTP.

SA Private signature key of party A.

sSA(X) Party A’s digital signature on information X with the private signature key SA. The encryption is performed on the hash of X.

signed_A = VVID, VID, HPID, Time-stamp, Lifetime, EoA

signed_M = VVID, VID, RegID, Time-stamp, Lifetime, EoM
Statement on service consumption, contains amongst others SPID, UID, service identifier, the intended start time $T_{\text{start}}$, and end time $T_{\text{end}}$.

**SoSC**

**SoSC** without $T_{\text{start}}$ and $T_{\text{end}}$, but contains $T_{\text{request}}$.

**SoSC** without $T_{\text{end}}$.

$\text{sub}_K = s_{\text{FP}}(f_{\text{sub}_K}, \text{SPID}, \text{UID}, L, K, T_{\text{sub}})$

Authenticator of $K$ provided by FP.

$\text{sub}_R = s_{\text{FP}}(f_{\text{Reg}}(\text{Registration Req}, T_{\text{end}}, \text{SPID}, \text{UID}, V_U, L))$

Authenticator of $L$ provided by FP.

$T_{\text{con}}$ The timepoint at which $K$ is confirmed by the TTP and placed in a publicly accessible repository.

$T_{\text{end}}$ End time of service consumption. It is also a deadline for TTP to time-stamp and publish $L$.

$T_{\text{reg}}$ The timepoint at which TTP confirms a registration of $L$.

$T_{\text{sub}}$ The deadline that FP should either send the key $K$ to $U$ or TTP.

$T_{\text{term}}$ The timepoint at which TTP confirms a termination request.

$V_A$ Public verification key of party $A$.

$[X]$ Information of type $X$ is optional

$[X]^*$ Information of type $X$ may appear 0, 1, or more times.

$X_1, X_2$ Information $X_1$ concatenated with information $X_2$. 
Appendix B

Sapta Grammar

The grammar of Sapta is specified in this appendix by means of a variant of Extended Backus-Naur Form (EBNF). Terminals are quoted ("..."), whereas non-terminals are enclosed in angle brackets (<...>). Alternatives are either separated by vertical bars (|) or are given in different productions. Components that can occur at most once are enclosed in square brackets ([...]), and components that can occur any number of times (including zero) are enclosed in braces ({...}). Note that whitespaces are ignored in the productions here.

B.1 Audit Subspecification

<AuditSubspecification> ::= <FilterFunctionSpec>
| <GroupingFunctionSpec>
| <PropertyFunctionSpec>
| <ComplianceFunctionSpec>
| <AttributeFunctionSpec>
| <ComplianceCalculationSpec>
| <ReportCompositionSpec>
B.1.1 Filter Function Specification

\[
\langle \text{FilterFunctionSpec} \rangle ::= \\
\text{"FilterFunction" } \langle \text{SpecId} \rangle \ [\text{"(" } \langle \text{FormalParameters} \rangle \text{")"}] \\
\{" \\
\langle \text{StatementSequence} \rangle \\
\}" \]

\langle \text{SpecId} \rangle ::= \langle \text{Identifier} \rangle

\langle \text{FormalParameters} \rangle ::= \\
\langle \text{FormalParameter} \rangle \{\text{"," } \langle \text{FormalParameter} \rangle\}

\langle \text{FormalParameter} \rangle ::= \langle \text{Identifier} \rangle

B.1.2 Grouping Function Specification

\[
\langle \text{GroupingFunctionSpec} \rangle ::= \\
\text{"GroupingFunction" } \langle \text{SpecId} \rangle \ [\text{"(" } \langle \text{FormalParameters} \rangle \text{")"}] \\
\{" \\
\ [ \langle \text{GroupState} \rangle ] \\
\ [ \langle \text{On_Timer} \rangle ] \\
\langle \text{StatementSequence} \rangle \\
\}" \]

\langle \text{GroupState} \rangle ::= \\
\text{"GroupState"} \\
\{" \\
\langle \text{VariableAssignmentStatement} \rangle \\
\{"\;" \langle \text{VariableAssignmentStatement} \rangle\} \\
\}" \]

\langle \text{On_Timer} \rangle ::= \\
\text{"On_Timer" } \langle \text{ScheduleList} \rangle \\
\{" \\
\langle \text{StatementSequence} \rangle \\
\}" \]
B.1.3 Property Function Specification

<PropertyFunctionSpec> ::= "PropertyFunction" <SpecId> "(" <FormalParameters> ")"

B.1.4 Compliance Function Specification

<ComplianceFunctionSpec> ::= "ComplianceFunction" <SpecId> "(" <FormalParameters> ")"

B.1.5 Compliance Calculation Specification

<ComplianceCalculationSpec> ::= "ComplianceCalculation" <SpecId>

<SpecificationSequence> ::= <GFInputSpec> ">>" <PFInputSpec>

">>" <CFInputSpec> "," <CFInputSpec>

">>" <CFId> ["(" <ActualConstParams> ")"]

<GFInputSpec> ::= <FFId> ["(" <ActualConstParams> ")"]
<PFInputSpec> ::= <GFId> ["(" <ActualConstParams> ")"]
<CFInputSpec> ::= <PFId> ["(" <ActualConstParams> ")"]

<FFId> ::= <SpecId>
<GFId> ::= <SpecId>
<PFId> ::= <SpecId>
<CFId> ::= <SpecId>
<ActualConstParams> ::= <Constant> "," <Constant>}
B.1.6 Attribute Function Specification

```xml
<AttributeFunctionSpec> ::= 
  "AttributeFunction" <SpecId> ["(" <FormalParameters> ")"] 
  
  <StatementSequence>
```

B.1.7 Report Composition Specification

```xml
<ReportCompositionSpec> ::= 
  "ReportComposition" <SpecId> 
  
  <AVPSpecs>
```

```xml
<AVPSpecs> ::= <AVPSpec> {"," <AVPSpec>}
<AVPSpec> ::= 
  "[" <AttributeName> "eq" <SpecSeqForAttribute> "]"
```

```xml
<SpecSeqForAttribute> ::= 
  [ <InputSpecs> ">>"]
  <SpecId> ["(" <ActualConstParams> ")"]
```

```xml
<InputSpecs> ::= <InputSpec> {"," <InputSpec>}
<InputSpec> ::= <SpecId>
```

B.2 Statements

```xml
<StatementSequence> ::= <Statement> {";" <Statement>}
<Statement> ::= <AssignmentStatement>
  | <ConditionalBranch> | <Iteration>
  | <IOStatement>
```

```xml
<AssignmentStatement> ::= <VariableAssignmentStatement>
  | <ReturnStatement>
```

```xml
<VariableAssignmentStatement> ::= 
  <Variable> ":=" <ExtConventionalExpr>
```
<ReturnStatement> ::= "return" <ExtConventionalExpr>

<ConditionalBranch> ::= <IfStatement>
<IfStatement> ::= "if" <ConventionalExpression> "then" <StatementSequence>
[ "else" <StatementSequence> ]
"endif"

<Iteration> ::= <WhileStatement> | <ForEachStatement>
<WhileStatement> ::= "while" <ConventionalExpression> "do"
[<StatementSequence>]"endwhile"
<ForEachStatement> ::= "foreach" <Variable>
"elem_of" ( <ListExpression> | <FactGroupExpression> )
"do" <StatementSequence>
"endfor"

<I0Statement> ::= <PrintStatement>
<PrintStatement> ::= "print" <Expression>

B.3 Expressions

<Expression> ::= <ExtConventionalExpression>
| <FactExpression> | <FactGroupExpression>

<ExtConventionalExpression> ::= <ConventionalExpression>
| <ListExpression>

<ConventionalExpression> ::=<SimpleExpression> | <RelationalExpression>

<ListExpression> ::= <ListConstructionExpression>
| <FunctionCall> | <Variable>

<ListConstructionExpression> ::= "[
[<ListElemExpression> {"," <ListElemExpression>}]"
"]"
<ListElemExpression> ::= <ExtConventionalExpression>

<RelationalExpression> ::= <SimpleExpression>
   <RelationalOperator> <SimpleExpression>

<RelationalOperator> ::= "<" | "<=" | ">" | ">=" | "==" | ".="

<SimpleExpression> ::= <Term>
   | <SimpleExpression> "+" <Term>
   | <SimpleExpression> "-" <Term>
   | <SimpleExpression> "or" <Term>

<Term> ::= <Factor>
   | <Term> "*" <Factor>
   | <Term> "/" <Factor>
   | <Term> "div" <Factor>
   | <Term> "mod" <Factor>
   | <Term> "and" <Factor>

<Factor> ::= "(" <ConventionalExpr> ")"
   | "+" <Factor>
   | "-" <Factor>
   | "not" <Factor>
   | <AttributeValueExpression>
   | <FunctionCall>
   | <Variable>
   | <SimpleConstant>

<FunctionCall> ::= <Identifier> "(" [<Expression> {"," <Expression>}] ")"

<Variable> ::= <Identifier>
<Identifier> ::= <Letter> { <Letter> | <Digit> | "_" }

<AttributeValueExpression> ::= <FactExpression> "." <AttributeName>
<FactExpression> ::= <FactVariable>
   | <FactGroupExpression> "[" <n> "]"

<FactGroupExpression> ::= <FactGroupVariable>
   | "GroupedFacts" "{" <OFLId> "}"
<FactVariable> ::= <Identifier>
<FactGroupVariable> ::= <Identifier>
<AttributeName> ::= <Identifier>
<OFLId> ::= <SimpleExpression>
<n> ::= <SimpleExpression>

B.4 Constants

<Constant> ::= <SimpleConstant> | <ListConstant>

<SimpleConstant> ::= <BooleanConstant>
   | <DecimalNumberConstant> | <StringConstant>
   | <DimensionalValue>

<BooleanConstant> ::= "true" | "false"
<DecimalNumberConstant> ::= 
   <IntegerNumberConstant> | <RealNumberConstant>

<IntegerNumberConstant> ::= [<SignPart>] <IntegerCoeffPart>
   [("e" | "E") [<SignPart>] <ExponentPart>]

<RealNumberConstant> ::= [<SignPart>] <CoefficientPart>
   [("e" | "E") [<SignPart>] <ExponentPart>]
<StringConstant> ::= "" {<ASCII_Symbol>} ""
<ListConstant> ::= "" [<Constant> {"," <Constant>}] ""

<SignPart> ::= "+" | "-
<CoefficientPart> ::= 
   <IntegerCoeffPart> "." <FractionalCoeffPart>
<ExponentPart> ::= <Digit> {<Digit>}
<IntegerCoeffPart> ::= <Digit> {<Digit>}
<FractionalCoeffPart> ::= <Digit> {<Digit>}

<Digit> ::= "0" | "1" | "2" | ... | "9"
<ASCII_Symbol> ::= An element of the ASCII character set
<Letter> ::= "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"

<DimensionalValue> ::= 
   "" <DecimalNumberConstant> <Unit> ""
<Unit> ::= <Identifier>
Appendix C

AURIC Processing Time

C.1 Average Processing Time

Table C.1: Average processing time without application logic.

<table>
<thead>
<tr>
<th>#Facts</th>
<th>$T_p$(FFM) [sec]</th>
<th>$T_p$(FGM) [sec]</th>
<th>$T_p$(PVCM) [sec]</th>
<th>$T_p$(CCM) [sec]</th>
<th>$T_p$(AVCM) [sec]</th>
<th>$T_p$(RGM) [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.000152 ± 0.000001</td>
<td>0.001314 ± 0.000007</td>
<td>0.006410 ± 0.000300</td>
<td>0.002905 ± 0.001003</td>
<td>0.001131 ± 0.000943</td>
<td>0.001278 ± 0.000244</td>
</tr>
<tr>
<td>1000</td>
<td>0.000191 ± 0.000003</td>
<td>0.031080 ± 0.012598</td>
<td>0.037643 ± 0.0012696</td>
<td>0.032603 ± 0.010793</td>
<td>0.019929 ± 0.009852</td>
<td>0.010215 ± 0.009979</td>
</tr>
<tr>
<td>2000</td>
<td>0.000213 ± 0.000026</td>
<td>0.054657 ± 0.011195</td>
<td>0.056318 ± 0.018243</td>
<td>0.073672 ± 0.013419</td>
<td>0.028467 ± 0.020057</td>
<td>0.050046 ± 0.024310</td>
</tr>
<tr>
<td>4000</td>
<td>0.000247 ± 0.00031</td>
<td>0.080758 ± 0.007215</td>
<td>0.117924 ± 0.040212</td>
<td>0.102447 ± 0.036775</td>
<td>0.069114 ± 0.039007</td>
<td>0.119258 ± 0.050683</td>
</tr>
<tr>
<td>6000</td>
<td>0.000273 ± 0.00002</td>
<td>0.135692 ± 0.008229</td>
<td>0.117170 ± 0.034756</td>
<td>0.120478 ± 0.055276</td>
<td>0.104723 ± 0.074743</td>
<td>0.219837 ± 0.091720</td>
</tr>
<tr>
<td>8000</td>
<td>0.000331 ± 0.00001</td>
<td>0.146666 ± 0.026690</td>
<td>0.159636 ± 0.052302</td>
<td>0.143506 ± 0.097053</td>
<td>0.118082 ± 0.092042</td>
<td>0.252000 ± 0.079188</td>
</tr>
<tr>
<td>10000</td>
<td>0.000395 ± 0.00002</td>
<td>0.203045 ± 0.038746</td>
<td>0.143625 ± 0.058979</td>
<td>0.155223 ± 0.077923</td>
<td>0.080405 ± 0.057402</td>
<td>0.469174 ± 0.097315</td>
</tr>
</tbody>
</table>
Table C.1: Average processing time without application logic.

<table>
<thead>
<tr>
<th>Facts</th>
<th>T_p(FFM) [sec]</th>
<th>T_p(FGM) [sec]</th>
<th>T_p(PVCM) [sec]</th>
<th>T_p(CCM) [sec]</th>
<th>T_p(AVCM) [sec]</th>
<th>T_p(RGM) [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>0.000697 ± 0.000009</td>
<td>0.409641 ± 0.043145</td>
<td>0.222856 ± 0.063217</td>
<td>0.262253 ± 0.071052</td>
<td>0.233373 ± 0.096263</td>
<td>0.799401 ± 0.116544</td>
</tr>
<tr>
<td>40000</td>
<td>0.001293 ± 0.000009</td>
<td>0.731729 ± 0.044432</td>
<td>0.414083 ± 0.071247</td>
<td>0.385831 ± 0.105872</td>
<td>0.330710 ± 0.151345</td>
<td>1.814635 ± 0.119581</td>
</tr>
<tr>
<td>60000</td>
<td>0.001899 ± 0.000011</td>
<td>1.090432 ± 0.035982</td>
<td>0.628717 ± 0.200047</td>
<td>0.500135 ± 0.260981</td>
<td>0.458931 ± 0.208030</td>
<td>2.750905 ± 0.120930</td>
</tr>
<tr>
<td>80000</td>
<td>0.002501 ± 0.000011</td>
<td>1.432259 ± 0.049943</td>
<td>0.776995 ± 0.188381</td>
<td>0.608563 ± 0.288203</td>
<td>0.715198 ± 0.341088</td>
<td>3.613384 ± 0.258536</td>
</tr>
<tr>
<td>100000</td>
<td>0.003117 ± 0.000006</td>
<td>1.801235 ± 0.059232</td>
<td>0.84594 ± 0.125101</td>
<td>0.731991 ± 0.124370</td>
<td>0.805063 ± 0.212496</td>
<td>4.560584 ± 0.123237</td>
</tr>
</tbody>
</table>

C.2 Average Processing Time per Fact

Table C.2: Average processing time per Fact without application logic.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>2</td>
<td>13</td>
<td>64</td>
<td>29</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>1000</td>
<td>&lt; 0.5</td>
<td>31</td>
<td>38</td>
<td>33</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>2000</td>
<td>&lt; 0.5</td>
<td>27</td>
<td>28</td>
<td>37</td>
<td>14</td>
<td>25</td>
</tr>
<tr>
<td>4000</td>
<td>&lt; 0.5</td>
<td>20</td>
<td>29</td>
<td>26</td>
<td>17</td>
<td>30</td>
</tr>
<tr>
<td>6000</td>
<td>&lt; 0.5</td>
<td>23</td>
<td>20</td>
<td>20</td>
<td>17</td>
<td>37</td>
</tr>
<tr>
<td>8000</td>
<td>&lt; 0.5</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>15</td>
<td>32</td>
</tr>
<tr>
<td>10000</td>
<td>&lt; 0.5</td>
<td>20</td>
<td>14</td>
<td>16</td>
<td>8</td>
<td>47</td>
</tr>
<tr>
<td>20000</td>
<td>&lt; 0.5</td>
<td>20</td>
<td>11</td>
<td>13</td>
<td>12</td>
<td>40</td>
</tr>
<tr>
<td>40000</td>
<td>&lt; 0.5</td>
<td>18</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>45</td>
</tr>
<tr>
<td>60000</td>
<td>&lt; 0.5</td>
<td>18</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>46</td>
</tr>
<tr>
<td>80000</td>
<td>&lt; 0.5</td>
<td>18</td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>100000</td>
<td>&lt; 0.5</td>
<td>18</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>46</td>
</tr>
</tbody>
</table>
Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Prof. Dr. Burkhard Stiller, for his continuous support and many invaluable and motivating discussions. With his constructive advice he guided me to always be on the right track and to keep an eye on the future. I would also like to thank my examiner, Prof. Dr. Bernhard Plattner and co-examiner Prof. Dr. Georg Carle, for supporting me and taking the time to comment on my thesis.

Furthermore, I would like to thank ETH Zurich, in particular the Department of Information Technology and Electrical Engineering (D-ITET) and the Computer Engineering and Networks Laboratory (TIK), for providing me a great infrastructure and workplace. I would also like to thank the Schweizerisches Bundesamt für Bildung und Wissenschaft BBW, which was renamed to Staatssekretariat für Bildung und Forschung SBF, and has funded ETH Zurich in the EU IST project “Moby Dick” and “Daidalos”, in the framework of which my work has partially been performed.

My special thanks go to my colleagues who have shared the same office space, frustration, and success with me on the H-floor, Placi Flury, Dr. Jan Gerke, Dr. David Hausheer, Kostas Katrinis, Pascal Kurtansky, Dr. Jan Mischke, Arno Wagner, and all the other colleagues in my research group, Dr. Sai Anand, Rainer Baumann, Marcel Baur, Dr. Matthias Bossardt, Daniela Brauckhoff, Dr. Thomas Dübendorfer, Károly Farkas, Dr. Ulrich Fiedler, Stefan Frei, Simon Heimlicher, Dr. Ralph Keller, Dr. Vincent Lenders, Dr. Martin May, Jorgos Parissidis, Dr. Marc Rennhard, Dr. Lukas Ruf, Daniel Sigg, Caterina Sposato, Mario Strasser, Svetlena Taneva,
Bernhard Tellenbach, Dr. Nathalie Weiler, as well as all the other people in my institute for a great time at TIK.

Surely the contributions from the students under my supervision also deserve my acknowledgements, especially Morten Vinje and Stephan Kiene for the implementation of the initial idea in SLA compliance auditing, and Egon Burgener for the initial work in non-repudiation of service consumption. I would also like to thank all Moby Dick and Daidalos project partners, especially Dr. Rui Aguiar, José M. Fernandes, Dr. Francisco Fontes, Jürgen Jähnert, and Carsten Schmoll for many fruitful discussions.

Finally, I would like to thank my wife Widawhasti Lampita and my beloved daughters Stella and Vanessa for their support, patience, understanding, and encouragement through all the ups and downs of writing a thesis. Great thanks also to my mom and dad, my brother and sisters, and to all my friends, especially Dr. Daniel Bauer and Peter Imhof for their support over the past many years and in future.
References


[85] Personal communications at: Cisco meeting with CSG, University of Zurich, August 31, 2006.


Curriculum Vitae

I was born in Binjai, Indonesia, on March 29, 1964. After 3 years study in the Department of Electrical Engineering at Neu Technikum Buchs I graduated in 1989. In 1990, I continued with the diploma study in computer science at the Swiss Federal Institute of Technology, ETH Zurich, after a transition year in HTL Winterthur. I received the diploma degree from ETH Zurich in March 1993.

I was employed as a research assistant at the Computer Engineering and Networks Laboratory from May 1993 to November 1994 before I left for Indonesia and worked as a lecturer at the State Polytechnic Malang until September 2000. I returned to Switzerland in September 2000 and worked as a research assistant again in the same laboratory at ETH Zurich where I got the chance to participate in various EU IST projects: M3I, Moby Dick, and Daidalos. In 2003, I started my PhD thesis work under the supervision of Prof. Dr. Burkhard Stiller on the generic auditing framework for compliance verification of Internet SLAs.