Grammatikbasierte Zertifikate am Beispiel von Sicherheitsauditierungen in verteilten IT-Systemen

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Matthias Kabatnik

geb. in Stuttgart-Bad Cannstatt

Hauptberichter:	Prof. em. Dr.-Ing. Dr.-Ing. E.h. Dr. h.c. Paul J. Kühn
1. Mitberichter:	Prof. Dr.-Ing. Günter Schäfer, Techn. Univ. Ilmenau
2. Mitberichter:	Prof. Dr.-Ing. Andreas Kirstädter
Tag der Einreichung:	14.06.2013
Tag der mündlichen Prüfung:	25.02.2015

Institut für Kommunikationsnetze und Rechnersysteme
Universität Stuttgart

2015

Inhaltsverzeichnis

Kurzfassung

In dieser Arbeit wird ein neues Verfahren zur Erstellung von signierten Zusicherungen in Form sog. grammatikbasierter Zertifikate (Grammar-based Certificates, GC) vorgestellt und bewertet.

Der dabei betrachtete Anwendungsfall ist die räumlich und zeitlich verteilte Ausführung von Sicherheitsauditierungen von IT-Infrastrukturen. Dabei wird entsprechend eines vorgegebenen Prüfplans eine große Zahl von Informationen zum Zustand der IT-Systeme einschließlich der unterstützenden Infrastruktur und der angewendeten Prozesse erhoben und zur Auswertung manipulationssicher übermittelt. Die Prüfergebnisse werden zusammengeführt und – im Idealfall automatisiert – bewertet. Die angewendeten Bewertungskriterien können dabei Prüfvorschriften wie der *Grundschutz* des Bundesamtes für Sicherheit in der Informationstechnik (BSI), unternehmensweite Sicherheitsrichtlinien und systemspezifische Festlegungen eines Sollzustandes sein.

Die Aufgabe der GC und der zugehörigen Mechanismen ist die Bereitstellung von geeigneten Datenformaten und Verarbeitungsschritten, um eine sichere, verbindliche Übermittlung und Prüfung der im Rahmen der Auditierung erhobenen Informationen zu ermöglichen. Sie müssen dabei verschiedenen Anforderungen des Anwendungsfeldes gerecht werden. Zunächst müssen die Ausdrucksmöglichkeiten der GC ausreichend *flexibel* sein, um die Vielzahl der benötigten Zusicherungen in unterschiedlichsten Umgebungen unterstützen zu können. Daneben müssen gemachte Zusicherungen eine hohe *Verbindlichkeit* aufweisen, da auf Basis von Auditierungsergebnissen teilweise sicherheitsrelevante Entscheidungen getroffen werden.

Daneben gibt es zwei weitere – scheinbar widersprüchliche – Anforderungen. Einerseits sollen zur Übermittlung der Auditierungsergebnisse Datenformate verwendet werden, die eine einfache *automatische Auswertung* ermöglichen, andererseits sollte der Ersteller einer signierten Zusicherung auch ohne Kenntnis der Kodierungsart die Bedeutung der signierten Daten kennen, um die Verantwortung für die gemachten Aussagen übernehmen zu können. Dazu sollte die unterschriebene Zusicherung möglichst *verständlich* und im Idealfall in natürlicher Sprache vorliegen.

Die GC erfüllen diese Anforderungen, indem für natürlichsprachliche Zusicherungen, die einem formalen Aufbau, z. B. entsprechend der Vorgaben des BSI-Grundschutzes, genügen, kontextfreie Grammatiken definiert werden. Die Grammatiken ermöglichen, eine Zusicherung in natürlichsprachlicher Darstellung eineindeutig in einen abstrakten Syntaxbaum zu überführen, der als GC kodiert, signiert und automatisch verarbeitet werden kann.

Ein solches Zertifikat enthält neben dem in XML kodierten abstrakten Syntaxbaum insbesondere eine Referenz auf die verwendete Grammatik in Form eines Hashwertes sowie weitere zertifikatstypische Daten. Der abstrakte Syntaxbaum wird dabei so optimiert, dass zwar redundante, in der Grammatik enthaltene Elemente entfernt werden, jedoch nur so, dass mit Hilfe der Grammatik aus dem Zertifikat der ursprüngliche natürlichsprachliche Text der Zusicherung zurückgewonnen werden kann.

Diese Kodierung der Zusicherungen bietet neben der Automatisierbarkeit zwei weitere Vorteile. Zum einen lässt sich die zu übermittelnde Datenmenge bei Zusicherungen mit langen, statischen Textpassagen deutlich reduzieren. Zum anderen kann durch geeignete Wahl der Nicht-Terminal-Bezeichner und der Produktionsregeln der Grammatik die Namensgebung der XML-Strukturen durch Begriffe des Anwendungsfalles so verständlich ausgeführt werden, dass die Parametrisierung einer automatischen Verarbeitung anwenderfreundlich gestaltet werden kann.

Als weitere Anforderung an die GC wird in dieser Arbeit die Möglichkeit zur Delegation der Berechtigung, Zusicherungen ausstellen zu dürfen, betrachtet. Aufbauend auf dem Grundmechanismus der GC wurde ein Verfahren entwickelt, das zur Beschreibung von Anforderungen an Zusicherungen und zur Festlegung von Beschränkungen der Rechte im Rahmen einer mehrstufigen Delegation dient. Das Verfahren wird mit Hilfe einfacher Pfadangaben realisiert, die nach dem XPath-Standard aus den Namen der in den abstrakten Syntaxbäumen verwendeten Knoten gebildet werden.

Da für die Bearbeitungsschritte *Kodierung* und *Rückgewinnung* die Kenntnis der jeweils verwendeten Grammatik notwendig ist und die Grammatiken teilweise selbst vertraulich sein können, wurde ein Grammatikverteildienst konzipiert, der es ermöglicht, Grammatiken ggf. verschlüsselt abrufbar zu halten ohne selbst vertrauenswürdig im Hinblick auf die Vertraulichkeit der Grammatiken sein zu müssen. Dabei garantiert der Dienst unter Verwendung kryptographischer Authentisierung eine Trennung der administrativen Zugriffe verschiedener Anwender.

Mit den drei vorgestellten Mechanismen lässt sich die gesamte Kette der Aufgaben bei der Durchführung einer Auditierung unterstützen. Ausgehend von der Festlegung von möglichen Zusicherungstexten durch die Grammatiken können die spezifischen Anforderungen z. B. in Form von Sicherheitsrichtlinien und die zu untersuchenden Systeme spezifiziert werden. Es können im Zusammenspiel mit einer Public Key Infrastruktur die notwendigen Berechtigungen zur Erstellung von Zusicherungen erteilt und die benötigten Grammatiken sicher verteilt werden. Die angeforderten Zusicherungen können so erstellt werden, dass sie für alle Beteiligten verbindlich in natürlichsprachlicher Form anzeigbar sind und verlustfrei als strukturierte Daten im XML-Format übertragen werden können. Schließlich lassen sich die Authentizität, die vorliegende Kompetenzzusicherung und die Erfüllung der anfangs definierten Anforderungen automatisiert prüfen.

Das Verfahren wurde prototypisch implementiert. Die Implementierung umfasst die Spezifikation einer Meta-Grammatik zur Erstellung einer anwendungsspezifischen Grammatik (Grammar-based Certificate Grammar, GCG), ein einfaches Verfahren, um aus Beispieltexten einer Zusicherung eine passende Grammatik für den jeweiligen Prüfbereich abzuleiten, und Generatoren, die eine GCG in Grammatiken für den existierenden ANTLR-Parser-Generator übersetzen.

Zu jeder Grammatik wird ein Parser-Paar erzeugt. Der erste Parser gewinnt aus dem natürlichsprachlichen Text der Zusicherung den abstrakten Syntaxbaum, der in ein Zertifikat aufgenommen wird. Der zweite Parser ist ein sog. Baumparser, der den im Zertifikat enthaltenen optimierten Syntaxbaum wieder in die ursprüngliche Zusicherung zurück wandeln kann. Ebenso können automatisiert validierende Editoren bereitgestellt werden, die durch Vorgaben vordefinierter Textbausteine und kontinuierliche Syntaxprüfung die Eingabe grammatikkonformer natürlichsprachlicher Zusicherungen unterstützen.

Der Grammatikverteildienst wurde als Web-Dienst mit dahinter liegender Datenbank implementiert. Es wurde ein dazu passendes Client-Programm als Kommandozeilenwerkzeug realisiert.

Durch die Implementierung konnte die Funktionsfähigkeit der Mechanismen und die Praktikabilität des Ansatzes gezeigt werden. Aus der Implementierung konnten darüber hinaus Erkenntnisse über die Grenzen der Einsatzmöglichkeiten des Verfahrens abgeleitet werden.

Die grammatikbasierten Zertifikate und der Delegationsmechanismus beruhen auf einer Reihe von Interpretationsvereinbarungen, die festlegen, wie Datenstrukturen umgewandelt und ausgewertet werden müssen. Da es sich dabei um sicherheitskritische Informationen handelt, ist es von entscheidender Bedeutung, dass diese Interpretationen präzise durch eine formale Semantik festgelegt werden. Es muss außerdem gezeigt werden, dass bei korrekter Anwendung dieser Interpretation auch in Gegenwart eines Angreifers korrekte Schlüsse gezogen werden können.

Daher wurden die neu entworfenen Mechanismen zur Zertifizierung und Delegation einer formalen Analyse auf Basis der SVO-Logik, einer sog. Belief-Logik von Syverson und van Oorshoot, unterzogen. Hierzu wurde die Logik so erweitert, dass sie allgemein die Zusicherung von Aktualität von Nachrichten beschreiben kann. Zur Modellierung und Analyse der neuen Mechanismen wurden neue spezifische Schlussregeln eingeführt, die die Verarbeitung und Interpretation von grammatikbasierten Zertifikaten beschreiben.

Mit diesen Regeln wurde für die GC eine eindeutige Semantik in der Terminologie der SVO-Logik festgeschrieben. Dadurch war es möglich, auf formale Weise zu zeigen, welche Nachrichten und welches initiale Wissen notwendig und hinreichend sind, damit ein Zertifikatsempfänger durch Signaturprüfungen, Kompetenzzusicherungen und die eingeführten Interpretationsregeln eine Zusicherung als Tatsache akzeptieren kann.

Durch Einführung der grammatikbasierten Zertifikate wurde ein Werkzeug mit formalem Rahmen geschaffen, das die Durchführung von Sicherheitsauditierungen umfassend unterstützen kann. Die grammatikbasierten Zertifikate erweitern darüber hinaus die bestehenden Zertifizierungs- und Autorisierungsmethoden um einen neuen flexiblen Mechanismus, der den technischen Zertifizierungsbegriff verallgemeinert und damit neue Anwendungsfelder erschließt.

Abstract

Grammar-based Certificates Applied to the Example of Security Audits of Distributed IT Infrastructures

This work presents and evaluates a new method to issue signed assertions, the so-called grammar-based certificates (GC).

The underlying field of application is the process of security audits of IT infrastructures, performed in a timely and spatially distributed manner. In these audits much information is collected about the status of IT systems including their supporting infrastructure and the applied processes. Transmitted under integrity protection the audit results are brought together for a —preferably automated— evaluation. The applied evaluation criteria may stem from different sources like the *IT-Grundschutz* of the *Bundesamt für Sicherheit in der Informationstechnik* (BSI), corporate security policies, and target state definitions for specific systems.

The GC and the related mechanisms provide appropriate data formats and data processing functionality to achieve a secure, binding transmission and evaluation of the audit data. The application field implies several requirements on the provided new functionality. The expressiveness of GC must be flexible enough to support the multitude of assertion types for different environments. For the given assertions a high level of commitment or even liability is essential since security relevant decisions are made based on the audit results.

There are two more —apparently contradictory— requirements. On the one hand the data formats used for transmission should enable an automated evaluation. On the other hand the issuer of a signed assertion should know the exact meaning of the transmitted data without knowledge about the type of coding in order to take responsibility for his statements. Therefore, a given assertion should be as intelligible as possible, preferably it should be natural language.

The GC satisfy these requirements by defining context free grammars for assertions in natural language that comply with an arbitrary but fixed formal structure which corresponds to specifications like BSI-Grundschutz. The grammars can be used to transform an assertion in natural language in a bijective way to an abstract syntax tree. This syntax tree can be coded into a GC, signed and processed automatically.

In addition to the abstract syntax tree coded in XML, a GC contains a hash value of the applied grammar as reference and further information specific to certificates. The abstract syntax tree is optimized by omitting all elements that are redundant with respect to the grammar, maintaining the possibility to recover the original text in natural language.

Besides the automated processing, this coding of assertions offers two additional benefits. Firstly, the amount of transmitted data can possibly be reduced considerably for assertions with long static phrases. Secondly, with a suitable selection of the grammar's non-terminal labels and production rules the naming of the XML structures can use intelligible terms of the application case. Thus, the parameterization of the automated processing of assertions can be designed in a user-friendly way.

A further requirement on the GC addressed by this work is the possibility to delegate the right to issue assertions. Based on the GC mechanism a method has been developed that makes use of simple path expressions. These expressions are based on the XPath standard and can be build with the names of the nodes of the abstract syntax tree. They can be used to declare requirements on assertions and restrictions when the right to make assertions is delegated in chains.

As the knowledge of the underlying grammar is essential for coding and recovery and as grammars might be confidential by themselves, a grammar distribution service has been designed. This service can handle encrypted grammars and can make them available without the necessity to be trustworthy regarding the confidentiality of the grammars. Moreover, the service can guarantee a segregation of users based on cryptographic authentication.

The three mechanisms presented can support the whole chain of tasks of an audit. Starting with laying down the possible wordings for an assertion type by defining a suitable grammar, specific requirements of security policies and the targets for an audit can be expressed. In combination with a public key infrastructure all necessary rights to issue assertions can be granted or passed on. The used grammars can be distributed securely. The requested assertions can be created with the possibility to be presented in a binding version in natural language to all participants. The assertion can be transmitted loss-free as structured data coded in XML. Finally, the authenticity, the presented assertions of jurisdiction and the compliance with the initial requirements can be evaluated automatically.

The process has been implemented as a prototype. The implementation comprises the specification of a meta-grammar for the design of an application specific grammar (grammar-based certificate grammar, GCG), a simple method for transforming example assertions into a suitable grammar for the underlying application field, and generators that translate a GCG into grammars for the existing ANTLR parser generator.

For every grammar a pair of parsers is generated. The first parser builds the certificate's abstract syntax tree from the natural language text. The second parser is a so-called tree parser that recovers the original assertion from the optimized syntax tree. Moreover, validation editors can be provided, that support the input of grammar-compliant natural language assertions by offering predefined text blocks and a continuous syntax check.

The grammar distribution service was designed as a web service using a separated database in the back-end network. A suitable client program has been designed that realizes the service access on the command line.

The implementation confirmed the operational capability of the mechanisms and the applicability of the approach. Additionally, the implementation provides insight into the limitations of the process concerning it's possible use cases.

The grammar-based certificates and the delegation mechanism are founded on a set of agreements on the interpretation of the data, that lay down the way how data structures must be transformed and how they shall be evaluated. Since this information is security critical it is of decisive importance that these interpretations are given a precise formal semantics. It has to be shown that even in the presence of an attacker the correct conclusions can be drawn from the grammar-based certificates.

Therefore, the new mechanisms for certification and delegation were formally analyzed by means of the SVO logic, a belief logic by Syverson and van Oorshoot. To suit the problem, this logic has been extended, thus it can capture the assertion of timeliness of a message in a general way. New specific deduction rules were established that model and analyze the processing and the interpretation of grammar-based certificates.

These rules set up a unique semantics in terms of the SVO logic. Thus, it was possible to show formally which messages and initial knowledge is necessary and sufficient that a certificate's recipient can derive enough knowledge by means of signature checks, jurisdiction statements and the provided interpretation rules to accept an assertion as a fact.

By the introduction of the grammar-based certificates a tool with a formal framework has been created that can support security audits in a comprehensive way. Moreover, the grammar-based certificates extend the existing methods for certification and authorization by a new flexible mechanism that generalizes the technical certification term. Thus, grammar-based certificates help to explore new application fields.

Abbildungsverzeichnis

Tabellenverzeichnis

Symbol- und Abkürzungsverzeichnis

Symbole

$\perp$	ungültig, leer
$\neg$	Negation
$\models$	Semantische Implikation
$\vdash$	Herleitungsbeziehung
$\vee$	Disjunktion
$\wedge$	Konjunktion
$\rightarrow$	Implikation der SVO-Logik
$\Rightarrow$	allgemeine Implikation
$\mapsto$	Übermittlung von Nachrichten
$\rightsquigarrow$	Korrespondenz zwischen Ereignissen
E	Ersteller einer Signatur
F	logisch falsch
Φ	Menge aller Formeln in SVO-Analysen
φ	beliebige Formel aus Φ
Φ_G	Kodierungsfunktion auf Basis der Grammatik G
Φ_G^{-1}	Dekodierungsfunktion auf Basis der Grammatik G
H	Bezeichner für Hash-Funktion
I	Inhalt eines zu signierenden Dokumentes
$\tilde{K}$	korrespondierender Schlüssel

K	Bezeichner für kryptographische Schlüssel
k	Bezeichner für symmetrische und öffentliche Schlüssel
K^{-1}	privater Teil eines asymmetrischen Schlüsselpaares
M	Menge aller Nachrichten (messages) in SVO-Analysen
P	Prinzipal P
PK_δ	Öffentlicher Schlüssel für Schlüsselaushandlungen
PK_φ	Öffentlicher Verschlüsselungsschlüssel
PK_σ	Öffentlicher Signaturschlüssel
ψ	beliebige Formel aus Φ
Q	Prinzipal Q
ρ	Prüffunktion für strukturierte Daten
ρ_g	Prüffunktion für Grammatikregeln
ρ_p	Prüffunktion für Primärtexte
r	Zusammengesetzte Beschränkung
Σ	Menge der Funktionen und Konstanten des Pi-Kalküls
Σ	Vorrat aller erlaubter Zeichen für Textbausteine
S	Signatur
Θ	Menge primitiver Terme der SVO-Logik
T	logisch wahr
$w_{\max}$	Anzahl maximal erlaubter weiterer Delegationsstufen
$\mathcal{X}$	Menge der zulässigen XPath-Ausdrücke
X	Bezeichner für Daten
x	XPath-Ausdruck
Y	Bezeichner für Daten

Akronyme

AIDE	Advanced Intrusion Detection Environment
ANTLR	Another Tool for Language Recognition

AS authentication service

ASN.1 Abstract Syntax Notation Number 1

AST abstract syntax tree

AWK Skript-Sprache von Aho, Weinberger und Kernighan

CA Certification Authority

CCS Calculus of Communicating Systems

CMS Cryptographic Message Syntax

CN Common Name

CRL Certificate Revocation List

CSP Communicating Sequential Processes

DAC Discretionary Access Control

DER Distinguished Encoding Rules

DSA Digital Signature Algorithm

DSS Digital Signature Standard

EAL Evaluation Assurance Level

EBNF Extended Backus Naur Form

ECC Elliptic Curve Cryptography

ECDSA Elliptic Curve Digital Signature Algorithm

ETSI European Telecommunications Standards Institute

FDR Failures-Divergence Refinement

GC Grammar-based Certificate

GCG Grammar-based Certificate Grammar

GCR GC-spezifische Prüffunktion für Restriktionen

GCV Prädikat für GC-spezifische Verifikation

GDC Grammar-based Delegation Certificate

GPG GNU Privacy Guard

GT Prädikat für Greater-than-Relation

HTTP HyperText Transfer Protocol

IETF Internet Engineering Task Force

IP Internet Protocol

IT Informationstechnik

LISP LISt Processing Language

LOTOS Language of Temporal Ordering Specification

LSAT Linux Security Audit Tool

MAC Mandatory Access Control

MIT Massachusetts Institute of Technology

MP Modus Ponens

Nec necessitation

NIST National Institute of Standards and Technology

O Organisation

OASIS Organization for the Advancement of Structured Information Standards

OCSP Online Certificate Status Protocol

OID Object ID

OpenVAS Open Vulnerability Assessment System

PDF Portable Document Format

PDP Policy Decision Point

PERMIS PrivilEge and Role Management Infrastructure Standards validation

PGP Pretty Good Privacy

PK Public Key

PKCS#1 Public Key Cryptography Standard No. 1

PKCS#7 Public Key Cryptography Standard No. 7

PKI Public Key Infrastructure

PMI Privilege Management Infrastructure

PSS Probabilistic Signature Scheme

RBAC Role Based Access Control

RFC Request for Comments

RIPEMD-160 RACE Integrity Primitives Evaluation Message Digest

RSA Signaturschema nach Rivest, Shamir und Adleman

RSASSA RSA Signature Scheme with Appendix

S/MIME Secure / Multipurpose Internet Mail Extensions

SAML Security Assertion Markup Language

SDSI Simple Distributed Security Infrastructure

SE1 1. Schritt zur Signaturerstellung

SE2 2. Schritt zur Signaturerstellung

SecPAL Security Policy Assertion Language

SHA-256 Secure Hash Algorithm Version 256

SOAP Simple Object Access Protocol

SPKI Simple Public Key Infrastructure

SSL Secure Socket Layer

SSO Single Sign On

SV Signaturvalidierung

SVO Belief-Logik nach **S**yverson und **v**an **O**orshoot

TGS ticket granting service

TGT Ticket Granting Ticket

URI Unified Resource Identifier

URL Unified Resource Locator

USB Universal Serial Bus

UTC Coordinated Universal Time

W3C World Wide Web Consortium

XACML eXtensible Access Control Markup Language

XAdES XML Advanced Electronic Signatures

XML eXtensible Markup Language

ZAS zentraler Auditierungsserver

1 Motivation und Zielsetzung

Informationsverarbeitende Systeme durchziehen heute alle Lebensbereiche von Wirtschaft, Politik, Medizin, Verwaltung bis hin in nahezu alle privaten Haushalte. Entsprechend wird auf diesen Systemen eine enorme Menge von Daten aller Art verarbeitet und gespeichert. Aus diesem Grund ist die Sicherheit dieser Systeme vor missbräuchlicher Nutzung, Ausspähung und Manipulation eine Forderung, deren Erfüllung für Firmen, Behörden und Privatpersonen von sehr hoher Bedeutung ist.

In vielen der genannten Bereiche werden die Betreiber dieser Systeme sowohl durch ihre eigenen Interessen als auch durch den Gesetzgeber gezwungen, ihre Systeme gegen drohende Angriffe von außen und innen zu sichern.

Die datenverarbeitenden Systeme müssen sowohl auf der physikalischen Ebene vor unberechtigtem Zugriff als auch auf der logischen Ebene, d. h. an den Schnittstellen zum Austausch von Kommunikationsdaten, vor Belauschen und Manipulation ihrer Daten geschützt werden. Speziell bei großen Systemen, die sowohl logisch als auch räumlich verteilt sind, spielen dabei neben einem Systemdesign, das Sicherheitsanforderungen berücksichtigt, vor allem die Parametrisierung und organisatorische Randbedingungen wie z. B. Rollenkonzepte eine erhebliche Rolle.

Für Standardanwendungen wie z. B. Web-basierte Client-Server-Dienste, aber auch für spezielle Anwendungsfälle, wird oftmals anhand von Kriterien- und Maßnahmenkatalogen die Korrektheit der Parametrisierung der Systeme geprüft. Neben diesen Katalogen sind häufig auch unternehmensspezifische Sicherheitsrichtlinien, Arbeits- und Verfahrensanweisungen und einzelvertraglich geregelte Anforderungen von Kunden zu berücksichtigen.

Eine umfassende Prüfung einer Infrastruktur umfasst teilweise hunderte von Details. Die Feststellung der dafür benötigten Sachverhalte erfolgt oftmals nicht nur durch eine Person, sondern durch Befragung einer ganzen Reihe von Personen, die jeweils für Teilbereiche des zu sichernden Systems verantwortlich sind. Diese haben die einzelnen Maßnahmen umzusetzen und für deren Korrektheit zu garantieren. Hierzu sichern die Befragten beispielsweise die vollständige Durchführung von Schutzmaßnahmen oder die Korrektheit einer sicherheitsrelevanten Parametrisierung zu.

Es ist die Aufgabe desjenigen, der eine erstmalige Sicherheitsanalyse oder wiederholte Auditierung eines großen Systems durchführt, die Zusicherungen aller Beteiligten zusammenzuführen und zu prüfen, ob damit die Gesamtheit der sicherheitsrelevanten Vorgaben erfüllt ist.

Die Einholung der Zusicherungen kann durch verschiedene Methoden erfolgen. Neben dem klassischen Interview mit ggf. gegengezeichnetem Protokoll besteht auch die Möglichkeit zur Nutzung elektronischer Kommunikation. Werden Zusicherungen jedoch über diesen Weg ausgetauscht, müssen diese sowohl in Hinblick auf Authentizität und Integrität und u. U. auch bezüglich ihrer Vertraulichkeit geschützt werden.

Die vorliegende Arbeit stellt einen Ansatz vor, bei dem Zusicherungen in elektronischer Form unter Verwendung von neu eingeführten speziellen Zertifikaten – den sog. grammatikbasierten Zertifikaten – ausgetauscht werden.

Im Kern bieten die grammatikbasierten Zertifikate die Möglichkeit, Zusicherungen sowohl in verständlicher Darstellung als auch in einer strukturierten Form für die automatische Weiterverarbeitung zu kodieren, sowie die Möglichkeit, beide Formen eindeutig in die jeweils andere Form zu überführen.

In diesem Kapitel wird zunächst ein Überblick über die Abläufe und Zusammenhänge bei Sicherheitsauditierungen und -evaluierungen gegeben. Im Anschluss werden Anforderungen an elektronische Zusicherungen aufgestellt, die zur Unterstützung eines Auditierungsprozesses erfüllt werden sollen.

1.1 Sicherheitsanalysen

Das Ziel einer Sicherheitsanalyse ist es, für ein zu untersuchendes System zu bestimmen, ob es geeignet ist, eines oder mehrere vorgegebene Schutzziele gegen einen Angreifer einer bestimmten Stärke zu schützen. Es gibt unterschiedliche Vorgehensweisen bei der Durchführung einer solchen Analyse. Ein bekanntes Verfahren ist die Vorgehensweise nach IT-Grundschutz des Bundesamtes für Sicherheit in der Informationstechnik (BSI) [Bun08b]. Dieses Verfahren wird i. d. R. auf IT-Infrastrukturen und ihr organisatorisches und physikalisches Umfeld angewendet.

Daneben gibt es andere Verfahren, die für Sicherheitsanalysen bei einzelnen Protokollen [BAN89, Kem89], Diensten und Anwendungen [Gro03] oder ganzen Architekturen [Vin02] angewendet werden.

Bei der Vorgehensweise nach IT-Grundschutz werden zunächst für das System eine Strukturanalyse und eine Schutzbedarfsfeststellung durchgeführt. Die Strukturanalyse erhebt die zu einer zu prüfenden Infrastruktur gehörenden Komponenten. Ausgehend von den zu schützenden Geschäftsprozessen und den damit verbundenen Daten werden die beteiligten Komponenten ermittelt. Dabei werden sowohl logische als auch physikalische Elemente bis hin zu den genutzten Räumen erfasst.

Die Schutzbedarfsfeststellung legt fest, für welche Systemelemente welche Schutzziele hinsichtlich Vertraulichkeit, Integrität und Verfügbarkeit bestehen. Durch die Bedrohungsanalyse werden Szenarien festgelegt, die eine Gefährdung für die identifizierten Schutzziele darstellen.

Aus den Ergebnissen dieser Analysen wird eine Liste von zu erfüllenden Systemeigenschaften und Maßnahmen erstellt, mit denen die Bedrohungen ausgeschaltet oder zumindest minimiert

werden können. Die nicht vollständig ausgeschlossenen Bedrohungen müssen in einer Risikoanalyse bewertet werden.

Als Unterstützung können Programme wie das Grundschutz-Tool des BSI eingesetzt werden [Bun08a]. Dieses Werkzeug ermöglicht die hierarchische Modellierung einer IT-Infrastruktur anhand eines Baustein-Kataloges. Den Elementen des Modells sind Fragen- und Maßnahmenkataloge zugeordnet, die eine systematische Abarbeitung von sicherheitsrelevanten Aspekten der Infrastruktur ermöglichen. Der Datenbestand wird in Form einer Datenbank gehalten, die auch in Teilen ex- und importiert werden kann und somit die Erfassung der Informationen durch unterschiedliche Anwender ermöglicht.

Um sicherzustellen, dass das entworfene System auch in der späteren Realisierung die Anforderungen erfüllt, wird in einer Umsetzungsprüfung die korrekte Ausführung und die Funktionsfähigkeit der identifizierten Maßnahmen verifiziert.

Wird die Prüfung bestanden, so können je nach verwendetem Prüfkatalog verschiedene Sicherheitszertifizierungen erteilt werden. Beispiele hierfür sind der Evaluation Assurance Level (EAL) gemäß Common Criteria [Com12], die Erklärung der Konformität mit bestimmten Anforderungen des BSI-Grundschutzkataloges [Bun11] oder die Erfüllung der ETSI-Norm der Anforderungen an die Zertifizierungsrichtlinien einer Zertifizierungsstelle (Certification Authority, CA) für qualifizierte Zertifikate [Eur07].

1.2 Auditierungen

Um die Systemsicherheit auch nachhaltig garantieren zu können, müssen die notwendigen Prüfschritte auch für wiederholte Prüfungen (Sicherheitsauditierungen) aufbereitet und z. B. in Listenform als sog. Prüfkataloge für die Verantwortlichen zur Verfügung gestellt werden.

Für eine kontinuierliche Sicherstellung eines Sollzustandes durch Audits ist es daher notwendig, die Prüflisten regelmäßig abzuarbeiten und die Prüfergebnisse aufzubereiten. Die Verantwortung hierfür liegt in der Regel beim Sicherheitsmanager eines Unternehmens, der die Audits anstoßen und die Ergebnisse aus den Fachbereichen entgegennehmen und bewerten muss.

Bei IT-Systemen mit speziellen Anwendungen gibt es neben den anwendungsspezifischen Sicherheitsmaßnahmen, deren Notwendigkeit durch die Sicherheitsanalyse festgestellt wurde, auch eine große Anzahl von standardisierten Maßnahmen, die in Abhängigkeit von den eingesetzten Betriebssystemen und den darauf eingesetzten Diensten immer wieder angewendet werden können. Für diese Basisfunktionalitäten gibt es eine Reihe von standardisierten Schutzmechanismen und zugehörigen Prüfaufgaben.

Beispiele für Prüfaufgaben sind

- Die Vorgabe für die Maske der Standard-Dateirechte (umask) ist gemäß Sicherheitsrichtlinie gesetzt.

- ❑ Die minimale Passwortlänge ist größer oder gleich acht Stellen.
- ❑ Die USB-Buchsen des Servers sind versiegelt.
- ❑ Die Notentriegelung einer Sicherheitstür führt zu einem Alarm.
- ❑ Für alle Änderungen an Firewall-Regeln liegen Änderungsanforderungen (Change Requests) vor.

Regelmäßig durchgeführte Audits müssen sowohl die standardisierten als auch die speziellen Schutzmechanismen abdecken.

Bei verteilt arbeitenden Produktivsystemen müssen Informationen über die Eigenschaften diverser Rechnersysteme, Kommunikationsstrecken, physikalische Eigenschaften der Umgebung und die Qualität der Ausbildung und Verlässlichkeit des Bedienpersonals durch den Prüfer (oder Evaluator) eingeholt werden.

Die große Menge an Informationen bringt es mit sich, dass für die Auskünfte während der Planungs-, Realisierungs- und Prüfungsphasen i. d. R. auch eine größere Zahl von Personen zu den jeweils ihnen übertragenen Verantwortungsbereichen befragt wird.

Zu diesem Personenkreis gehören im Wesentlichen Netzwerkadministratoren, Sicherheitsmanager, Datenschutzbeauftragte und Anwendungsadministratoren. Diese *Informationsgeber* müssen in Interviews oder schriftlichen Äußerungen zu den jeweils aus Sicherheitssicht relevanten Fragen (z. B. aus dem IT-Grundschutz) Stellung beziehen.

Neben diesen grundsätzlichen Eigenschaften sind zur Beschreibung eines Prüfbedarfs vor allem die *Häufigkeit* der Prüfungen, die geforderte *Dokumentation* der Prüfergebnisse und das Management der Prüfergebnisse beim Abgleich mit den *Vorgaben* relevant. Diese Aspekte werden in den folgenden Abschnitten betrachtet.

1.2.1 Häufigkeit

In einer initialen Umsetzungsprüfung werden die beim Entwurf der Systemarchitektur und den Sicherheitsanalysen getroffenen Annahmen, die abgeleiteten Maßnahmen und deren korrekte Umsetzung auditiert. Oftmals werden Umsetzungsprüfungen durch externe Prüfstellen ausgeführt, die das Ergebnis wiederum durch Ausstellung entsprechender Zertifizierungen öffentlich belegen.

Bei den periodisch – teilweise im mehrjährigen Rhythmus – durchgeführten Wiederholungsprüfungen werden auch die Grundkonzepte und -annahmen einem Review unterzogen und die Korrektheit der abgeleiteten Schutzmaßnahmen hinterfragt. Abbildung 1.1 zeigt die Häufigkeit verschiedener Prüfungsarten periodischer und zufallsabhängiger Audits.

Für die ebenfalls durchzuführenden Auditierungen gibt es diverse zeitliche Schemata. Teilweise werden nach einem engmaschigen periodischen Zeitplan sicherheitskritische Eigenschaften und

Parametrisierungen der Infrastruktur auditiert. Dies ist z. B. bei der täglichen automatisierten Integritätsprüfung der Fall. Daneben werden aber auch periodische Prüfungen mit längeren Abständen sowie unangekündigte oder anlassbezogene Prüfungen, die beiden letztgenannten ohne festes Zeitschema, durchgeführt.

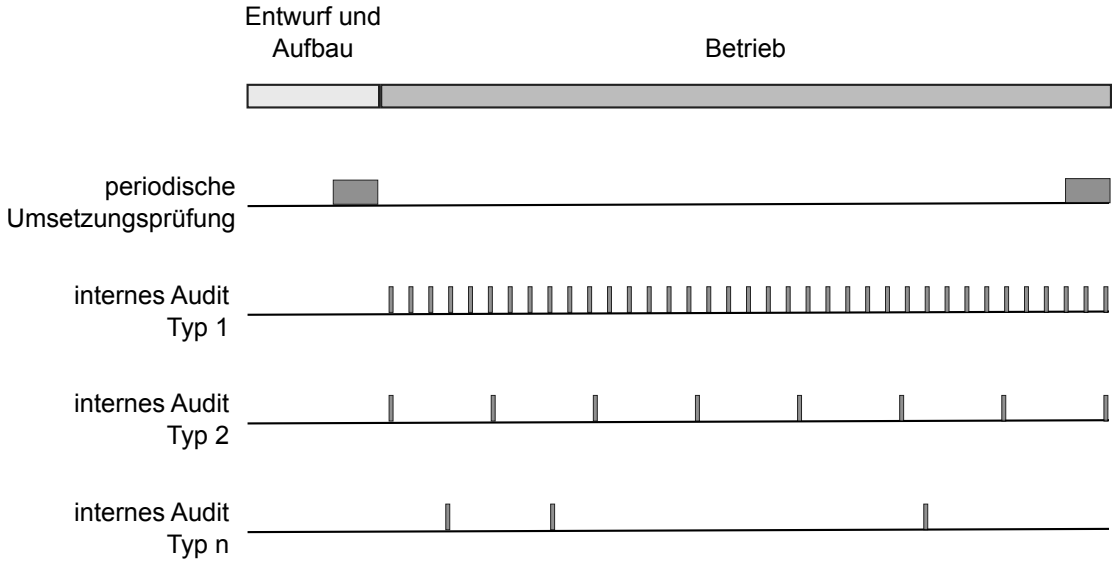


Abbildung 1.1: Prüfhäufigkeit für Annahmen und Vorgaben

Die häufigen Prüfungen sind notwendig, da i. A. ein System aus verteilten IT-Komponenten nicht unverändert bleibt, wie dies bei einer einmal ausgelieferten unveränderbaren Chipkarte der Fall ist. Neben Software-Aktualisierungen werden Komponenten gegen leistungsfähigere Modelle ausgetauscht, Dienste werden weiter entwickelt und in die Systemumgebung eingebracht. Durch diese Arbeiten kann es zu – möglicherweise ungewollten – Veränderungen an den Systemen bzw. ihrer Parametrisierung kommen. Daneben kann nicht mit völliger Sicherheit ausgeschlossen werden, dass Dienste oder Infrastrukturkomponenten Schwachstellen aufweisen, über die es einem Angreifer gelingen kann, Systemeigenschaften so zu verändern, dass die Schutzziele nicht eingehalten werden können.

Inhalte periodischer Prüfungen sind beispielsweise die Sichtung der auf einem Serversystem angelegten Accounts für Mitarbeiter mit administrativem Zugriff, die Kontrolle der Einstellungen für Passwortregeln, die tägliche Log-Analyse etc.

Neben den manuellen Prüfungen gibt es auch automatische Werkzeug-gestützte Auditierungen. Dabei kommen diverse Programme zum Einsatz. Beispiele für frei verfügbare Programmunterstützung sind das Linux Security Audit Tool (LSAT) [Min08] oder das Open Vulnerability Assessment System (OpenVAS) [DAdB$^+$11]; ein führendes kommerzielles Werkzeug ist der Schwachstellen-Scanner Nessus [Ten11]. Zur Überwachung der Integrität des Dateisystems eines Servers kommt beispielsweise das Advanced Intrusion Detection Environment (AIDE) zum Einsatz [vH11].

Automatisierte Prüfungen können teilweise in sehr engem Zeitraster (z. B. stündlich) erfolgen. Demzufolge fällt eine große Anzahl von Ergebnissen an, die im Idealfall ebenfalls automatisch weiterverarbeitet und bewertet werden.

1.2.2 Dokumentation

Prüfergebnisse werden häufig in Form von Listen mit Prüfschritt und Ergebnis oder auch durch frei formulierte Texte dargestellt. Bei automatisierter Prüfung können die Ergebnisse noch kompakter sein. Beispielsweise erzeugt die Prüfung der Datei-Integrität mit AIDE bei Abwesenheit von Abweichungen eine Meldungszeile
`### All files match AIDE database. Looks okay!`
Diese Log-Zeile kann dann ggf. durch ein Monitoring-System wie nagios noch in einen OK-Status eines sog. Probes umgesetzt werden.

Während bei Ausgaben von Prüfwerkzeugen teilweise eine automatische Verarbeitbarkeit gegeben ist, wird dies bei Verwendung von freien Formulierungen (z. B. in einer E-Mail eines Systemadministrators an das Sicherheitsmanagement) i. d. R. nicht der Fall sein. Hier ist somit sehr oft die manuelle Auswertung und abschließende Bewertung notwendig.

Die Ergebnisse einer Auditierung oder Sicherheitsprüfung sind darüber hinaus selbst schützenswert. Gelänge es einem Angreifer, die Meldungen über ein kompromittiertes System, welche an das Monitoringsystem gesendet werden, ebenfalls zu manipulieren, so würde die Sicherheit des Systems falsch bewertet.

Es ist daher wichtig, auch die Dokumentation von Prüfschritten in ihrer Integrität und Authentizität zu schützen. Dies erfolgt teilweise durch den Einsatz digitaler Signaturen (z.B. bei E-Mails), welche die Quelle und Korrektheit einer Nachricht für den Empfänger nachvollziehbar machen.

1.2.3 Vorgaben

Durch das Sicherheitskonzept, Betriebsführungshandbücher, Programmdokumentationen und Anforderungen an die Einsatzumgebung aus Produktevaluierungen ergibt sich ein Soll-Zustand für die IT-Infrastruktur. Bei einer Prüfung wird der Ist-Zustand erfasst, mit dem Soll-Zustand verglichen und einer Bewertung unterzogen. Dabei ist es grundsätzlich frei wählbar, wo die Vergleichsoperation und die Bewertung stattfindet.

Am Integritätsprüfprogramm AIDE lässt sich dieses Prinzip gut verdeutlichen. Nach einer frischen Systeminstallation oder nach einer autorisierten Veränderung an einem integritätsüberwachten System wird der Zustand aller statischen Dateien (Hash-Werte des Inhalts, Zugriffsrechte, Eigentümer, etc.) erfasst und in einer Referenzdatenbank abgelegt. Jede anschließende Prüfung ermittelt erneut den Ist-Zustand und vergleicht die gefundenen Eigenschaften mit der Referenzdatenbank. Es werden dann die Abweichungen zum Soll-Zustand an eine bewertende Stelle übermittelt.

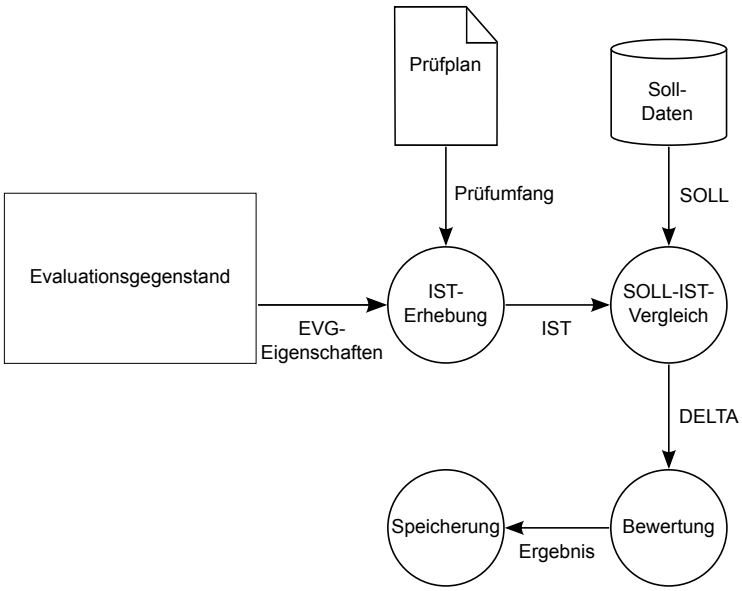


Abbildung 1.2: Allgemeines Prüfverfahren

Diese Vorgehensweise lässt sich auf beliebige manuelle Prüfungen durch Auditoren übertragen. In Abbildung 1.2 wird das Verfahren in allgemeiner Form gezeigt.

Wenn bei der Übermittlung von Informationen von einem Bereich zum anderen eine Übertragung der Daten über unsichere Kommunikationskanäle erfolgt, ist ein Schutz der Daten vor Veränderung und die Sicherstellung der Authentizität zwingend notwendig, da anderenfalls durch entsprechende Manipulation der Daten die Bewertung zu einem falschen Ergebnis gelangt.

Die Vorgaben bestehen somit aus einem Prüfplan, der die Prüfschritte und Prüfgegenstände festlegt, sowie einer Vorgabe des Soll-Zustandes. Abweichungen vom Soll-Zustand werden bewertet und das Ergebnis mit anderen Prüfungen zusammengeführt. Gegebenenfalls werden aufgrund der getroffenen Bewertungen auch unmittelbar Maßnahmen zur Wiederherstellung des Sollzustandes abgeleitet.

Die einzelnen Schritte können dabei von unterschiedlichen Rolleninhabern wahrgenommen werden. So kann beispielsweise der Systemadministrator die Ist-Erhebung durchführen, das Sicherheitsmanagement vergleicht danach die Ergebnisse mit den Vorgaben und bewertet die Abweichungen.

1.3 Ziele

Ein wesentlicher Aufgabenschwerpunkt bei der Realisierung und beim Betrieb sicherheitskritischer IT-Infrastrukturen ist eine kontinuierliche Sammlung von Informationen über das zu sichernde System.

Während des Betriebs der Infrastruktur kommen zu Sicherheitsprüfungen durch Personen auch Informationen aus automatisch arbeitenden Systemen hinzu, die den ordnungsgemäßen Zustand des oder der Systeme in Hinblick auf Integrität und Verfügbarkeit kontinuierlich überwachen. Hierzu gehören beispielsweise Integritätsprüfprogramme wie AIDE [vH11] oder allgemeine Monitoring-Programme wie nagios [NE11].

Auf Basis der Aussagen der Informationsgeber kann dann die Evaluation oder Auditierung durchgeführt werden und eine Sicherheitseinschätzung für das untersuchte System durch einen Prüfer abgegeben werden. Bei sehr umfangreichen Untersuchungen können die Informationen auch hierarchisch eingeholt werden, d. h. ein für einen Bereich Verantwortlicher befragt seinerseits Untergebene oder Kollegen und führt die dabei erhaltenen Informationen entweder selbst zusammen oder reicht diese entsprechend an die bewertende Stelle weiter.

Ein solcher sicherheitskritischer Prozess soll durch den gezielten Einsatz von digitalen Signaturen, geeigneten Zertifikatsformaten und automatisierten Kommunikations- und Auswertungsmechanismen verbessert werden.

Wie bereits erwähnt, ermöglichen Programme wie das BSI-Grundschutz-Tool die elektronische Erfassung und Auswertung der gesammelten Informationen. Es ist teilweise ebenfalls möglich, die Informationen verteilt zu erfassen und durch Export- bzw. Importfunktionen zur Auswertung zusammenzuführen. Hierbei sind Zugriffskontrollmechanismen auf Betriebssystem- oder Datenbankebene wirksam, so dass Berechtigungen für die Eingabe bestimmter Informationen an die jeweils Verantwortlichen vergeben und durchgesetzt werden können.

Diese Mechanismen ermöglichen jedoch nur eine Authentisierung der Daten, nicht aber eine rechtliche Bindung des Informationsgebers an die Aussage. Des Weiteren ist die verteilte Informationssammlung nur mit Instanzen eines Programms möglich, wenn keine standardisierten Schnittstellen zum Datenaustausch bestehen. Eine Plattform- und Programm-übergreifende Erfassung der Daten ist daher oft nicht möglich.

Ziel der in diesem Kapitel vorgestellten Vorgehensweise auf Basis neu einzuführender grammatikbasierter Zertifikate ist es, die Prozesse der Erfassung sicherheitsrelevanter Informationen für Sicherheitsaudits hinsichtlich der Verbindlichkeit durch den Einsatz von digitalen Signaturen zu verbessern. Gleichzeitig sollen jedoch auch im Sinne der mehrseitigen Sicherheit die Interessen der Informationsgeber berücksichtigt werden.

1.3.1 Flexibilität

Das vorgestellte Anwendungsfeld erfordert Zusicherungen zu sehr verschiedenen Themen, die von technischen Details aus Systemkonfigurationen über konkrete Systemzustände bis zur Erfüllung organisatorischer Vorgaben reichen.

Dadurch werden für die Zusicherungen entsprechend vielfältige Ausdrucksmöglichkeiten benötigt, die durch die bereitzustellenden Mechanismen nicht an sich beschränkt sein dürfen. Es sollen daher prinzipiell beliebige Aussagen darstellbar sein.

Werden auf dieser Basis für einen Anwendungsfall bestimmte Aussageformen gewählt, so sollen diese wiederum flexibel im Sinne einer Parametrisierung gestaltbar sein. Dadurch sollen im Rahmen einer konkreten Auditierung für gleichartige Zusicherungen Bausteine bereitgestellt werden, die z. B. durch Einsetzen der jeweils untersuchten Systeme für den vorliegenden Untersuchungsgegenstand ausgeprägt werden oder unterschiedliche Prüfmethoden und -ergebnisse darstellen können.

1.3.2 Erhöhung der Verbindlichkeit

Aus Sicht eines Prüfers ist es wünschenswert, dass die durch die Informationsgeber erteilten Auskünfte so verlässlich wie möglich sind, da eine Sicherheitsanalyse nur so gut sein kann wie die Qualität der Ausgangsinformationen, auf denen sie beruht. Im Idealfall übernimmt der jeweilige Informationsgeber die Verantwortung für die Richtigkeit seiner Auskünfte. Diese Verantwortungsübernahme ist jedoch nur dann aus den in elektronischer Form vorliegenden Daten abzulesen, wenn diese sowohl eindeutig einem Aussteller zuzuordnen sind als auch eine Zusicherung der Verantwortungsübernahme existiert. Diese Zusicherung kann durch organisatorische Rahmenbedingungen wie Verfahrensanweisungen gegeben oder explizit in den Daten enthalten sein. Die höchste Verbindlichkeit wird dabei durch den Einsatz einer qualifizierten digitalen Signatur erreicht.

Durch eine geeignete Vereinbarung der Interpretation der Aussagen bzw. eine entsprechende Formulierung der signierten Inhalte kann damit der Informationsgeber für die Korrektheit der Information die Haftung übernehmen oder die Bereitschaft zur Aktualisierung der Aussagen im Bedarfsfall zusichern.

1.3.3 Verständlichkeit

Wird einerseits die Bindung eines Informationsgebers an seine Aussagen durch den Einsatz von Signaturen verstärkt, so muss andererseits sein Interesse an überschaubaren und damit bewertbaren Konsequenzen der verbindlichen Abgabe seiner Aussage berücksichtigt werden, da sonst mit einer zu geringen Akzeptanz für diesen Mechanismus zu rechnen ist.

Zum einen muss sichergestellt werden, dass sowohl Informationsgeber als auch Informationsanwender die gemachten Aussagen in gleicher Weise interpretieren, bzw. der Interpretationsspielraum dieser Aussagen muss soweit verkleinert werden, dass die Aussagen für beide Seiten in der gleichen, natürlichsprachlichen Form vorliegen. Diese Darstellungsform ist dann zwar ihrerseits z. B. im Fall der Verwendung einer Fachsprache interpretationsbedürftig, jedoch umfasst die notwendige Interpretation nicht mehr die Dekodierung der signierten Daten. Die technische Aufbereitung von Aussagen in Form signierter Zertifikate soll also keinen weiteren Interpretationsspielraum hinzufügen.

Zum anderen muss dem Informationsgeber die Möglichkeit gegeben werden, die Gültigkeit seiner Zusicherung zeitlich einzuschränken, da er die Verantwortung für den Zustand eines Systembereichs nicht auf beliebige Zeit übernehmen will oder kann. Grundsätzlich sollte es dem Informationsgeber daher möglich sein, eine beliebige zeitliche Beschränkung an seine Aussagen zu binden, wobei organisatorische Vorgaben durch den technischen Mechanismus unberührt bleiben müssen. Alternativ bzw. ergänzend können Mechanismen zum Widerruf einer gegebenen Zusicherung sinnvoll sein, wenn dem Informationsgeber bekannt wird, dass die Grundlage seiner Zusicherung nicht mehr gültig ist.

1.3.4 Automatisierte Auswertung

Die Kodierung der erstellten Zertifikate muss so gestaltet sein, dass sie eine automatisierte Auswertung ermöglicht. Beispiele für eine solche Auswertung sind der Vergleich von gegebenen Zusicherungen gegen einen Prüfkatalog zur Feststellung der Vollständigkeit oder die Feststellung der Erfüllung von Vorgaben aus Sicherheitsrichtlinien.

Die automatisierte Verarbeitung fordert daher eine wohldefinierte Form (Syntax) der Zertifikate. Die verwendete Kodierung sollte einerseits flexibel, andererseits auch etabliert sein, um in einfacher Weise die notwendigen Verarbeitungsmechanismen implementieren zu können.

Überdies sollte im Zertifikat die Datenmenge möglichst gering gehalten werden, um die Systemressourcen zu schonen. Hierzu muss die Kodierung so gestaltet werden, dass lediglich der notwendige Informationsgehalt in das Zertifikat aufgenommen wird, nicht jedoch die dahinter stehende natürlichsprachliche Darstellung.

1.3.5 Delegation

Um im Fall der Untersuchung großer Systeme mit hierarchisch organisierten Verantwortungsstrukturen den Abstimmungsaufwand über Verantwortungsbereiche und mögliche Informationsgeber für den Prüfer zu vereinfachen, sollte es auch möglich sein, die eingehenden Informationen nicht nur auf ihre Authentizität zu prüfen, sondern auch die jeweilige Berechtigung des Informationsgebers automatisiert zu validieren.

Hierzu müssen Möglichkeiten geschaffen werden, die es erlauben, die Berechtigung eines Informationsgebers hinsichtlich bestimmter Aussagetypen oder Systemkomponenten z. B. in Berechtigungszertifikaten auszudrücken, so dass bei Eintreffen einer signierten Zusicherung diese gegen die aus den Berechtigungszertifikaten abgeleiteten Rechte validiert werden kann. Die Delegationszertifikate müssen geeignet sein, hierarchische Verantwortungsstrukturen abzubilden, so dass es beispielsweise genügt, in einem Programm zur Auswertung von zertifizierten Sicherheitsaussagen lediglich wenige Vertrauensanker zu hinterlegen. Dieses Vertrauen wird dann bei der Auswertung von Zertifikatsketten transitiv auf Delegierte übertragen. Andererseits darf die Weitergabe von Rechten nicht über zu viele Stationen erfolgen, da es bei Sicherheitsaudits i. d. R. Absprachen zwischen den Auditoren und den Informationsgebern geben muss.

1.4 Überblick über die Arbeit

Der Anspruch der Arbeit ist die Bereitstellung und Bewertung eines neuen Zertifizierungsmechanismus, der den im Abschnitt 1.3 aufgestellten Zielen gerecht wird. Diese sog. grammatikbasierten Zertifikate stellen eine Möglichkeit zur integritätsgeschützten, verbindlichen Übermittlung von kompakt kodierten Zusicherungen im Rahmen von Auditierungen bereit. Sie bieten dabei flexible Ausdrucksmöglichkeiten und unterstützen beweisbar gleichwertige Darstellungsformen. Dies sind einerseits die für den Anwender verständliche Form in natürlicher Sprache und andererseits strukturierte XML-Daten, die automatisch verarbeitet werden können.

In Kapitel 2 werden zwei wesentliche Grundbausteine gesicherter elektronischer Kommunikation – Verschlüsselung und Signaturen – eingeführt. Es werden Zertifikate am Beispiel von Public-Key-Infrastrukturen auf Basis der X.509-Spezifikation vorgestellt und verschiedene Verfahren zur Übermittlung von signierten Zusicherungen wie Berechtigungen u. ä. betrachtet. Eine Bewertung der vorgestellten Zusicherungs- und Zertifizierungsverfahren mit den in diesem Kapitel aufgestellten Anforderungen motiviert den nachfolgenden Entwurf neuer Mechanismen.

Anschließend werden in Kapitel 3 Methoden zur Analyse von Sicherheitseigenschaften in Kommunikationssystemen vorgestellt, die im weiteren Verlauf der Arbeit angewendet werden. Dieses sind zum einen die Belief-Logik SVO von Syverson und van Oorshoot, sowie der Pi-Kalkül und das darauf aufsetzende Model-Checking-Werkzeug *proverif*. Diese Methoden ermöglichen die präzise Modellierung von Kommunikationsprotokollen und bieten die Möglichkeit, die Modelle hinsichtlich der erreichbaren Zusicherungen und der Einhaltung von Vertraulichkeits- und Integritätszielen zu evaluieren.

In Kapitel 4 wird das neue Konzept der *grammatikbasierten Zertifikate* auf Basis kontextfreier Grammatiken vorgestellt. Auf Basis dieses neuen Zertifikatstyps werden weitere Mechanismen entworfen, die dazu dienen, die in diesem Kapitel für den Anwendungsfall der Sicherheitsprüfungen geforderten Eigenschaften bereitzustellen. Diese Mechanismen umfassen die Möglichkeit zur präzisen Definition von Anforderungen für Sicherheitsprüfungen, die Beschreibung und ggf. Einschränkung von Rechten zur Erstellung von Zusicherungen im Falle der Delegation und die ggf. vertrauliche Verteilung kontextfreier Grammatiken durch einen zentralen Dienst.

Schließlich wird in Kapitel 5 die Implementierung der neuen grammatikbasierten Zertifikate und der zugehörigen Mechanismen vorgestellt und die zugrunde liegenden Konzepte einer formalen Analyse mittels der in Kapitel 3 eingeführten Werkzeuge unterzogen. Dabei wird eine formale Semantik für die grammatikbasierten Zertifikate und die Delegationsmechanismen angegeben und die Wirksamkeit der Mechanismen mittels der SVO-Logik gezeigt.

Eine Zusammenfassung der Ergebnisse und ein Ausblick schließen die Arbeit ab.

2 Zertifizierungssysteme

Die schriftliche Bestätigung von Sachverhalten ist eine sehr alte Vorgehensweise. So wurden bereits im Mittelalter Urkunden für z. B. Stadtrechtsverleihungen, Kaufverträge, usw. gefertigt. Sie wurden genutzt, um Ansprüche oder Sachverhalte auch in Abwesenheit des Ausstellers gegenüber Dritten zu beliebigen Zeitpunkten an beliebigen Orten belegen zu können.

Die bereits früh entwickelten Formen von Urkunden mit formalem Aufbau haben sich bis heute erhalten und Verträge und Zertifizierungen werden auch heute noch nach festen Kriterien mit spezifischen Verifikationsmerkmalen aufgebaut, die eine Prüfung durch Dritte ermöglichen.

Bei der Digitalisierung solcher Dokumente werden jedoch neue Anforderungen gestellt. Die Dokumente existieren nicht mehr in Form physischer Entitäten wie Papier mit Unterschrift und Siegel sondern sind zunächst ein beliebig kopier- und manipulierbarer Datenstrom.

Die Verfügbarkeit der asymmetrischen Kryptographie hat es jedoch möglich gemacht, Nachweise des Ursprungs und damit auch Zusicherungen von Inhalten elektronisch mit digitalen Signaturen nachzubilden.

Es werden in diesem Kapitel Grundverfahren von digitalen Signaturen und Zusicherungen (Zertifikaten) dargestellt und spezielle Formen von Zusicherungen aus den Bereichen Schlüsselverteilung und Autorisierung vorgestellt. Daneben werden teilweise bereits Schreibweisen eingeführt, die der im nachfolgenden Kapitel eingeführten, nach ihren Autoren Syverson und van Oorschot benannten SVO-Logik entnommen sind.

Die dargestellten Verfahren stellen den Stand der Technik dar und werden abschließend qualitativ hinsichtlich ihrer Eignung bewertet, die in Abschnitt 1.3 aufgestellten Ziele zu erreichen.

2.1 Grundbegriffe kryptographischer Kommunikationssicherung

Zur Absicherung von Kommunikationsvorgängen über elektronische Medien werden i. d. R. kryptographische Verfahren eingesetzt. An dieser Stelle werden die in dieser Arbeit zur Benennung eingesetzter Schlüssel und Verfahrensschritte verwendeten Begriffe zusammengestellt.

Kommunikationsvorgänge können hinsichtlich verschiedener Ziele schützenswert sein. Für diese Arbeit sind folgende Ziele relevant:

Vertraulichkeit Beim Erhalt der Vertraulichkeit werden die Kommunikationsinhalte vor unautorisierter Kenntnisnahme durch Dritte geschützt. Lediglich der Sender und die durch den Absender vorgesehenen Empfänger einer Kommunikation sollen auf die Inhalte zugreifen können.

Integrität Bei der Absicherung der Integrität werden die Kommunikationsinhalte vor unautorisierter Veränderung geschützt bzw. eine Veränderung zuverlässig erkennbar gemacht. Integritätsgeschützte Inhalte erreichen die Empfänger nur in der durch den Absender erzeugten Form.

Authentizität Ein Kommunikationsinhalt gilt als authentisch, wenn er eindeutig und korrekt durch einen Empfänger einer Quelle zugeordnet werden kann.

Nicht-Abstreitbarkeit Der Urheber eines Kommunikationsinhaltes kann seine Urheberschaft nicht verleugnen. Diese Eigenschaft wird auch als *Zurechenbarkeit* bezeichnet.

Die geschützte Kommunikation kann dabei über Distanzen (Nachrichtenübertragung), über die Zeit (Speicherung von Information) oder beides sein.

Der gewünschte Schutz wird dabei i. d. R. durch den Einsatz von kryptographischen Verfahren erreicht. Es werden im Folgenden die bekannten Standardverfahren *Verschlüsselung* und *Signatur* betrachtet.

Beim Einsatz von Verschlüsselung werden die zu schützenden Daten als *Klartext* bezeichnet. Durch den Einsatz eines geeigneten Rechenverfahrens, der *Verschlüsselung* (auch Chiffrierung) wird der Klartext mit Hilfe eines durch einen *Schlüssel* parametrisierten Algorithmus in das sog. *Chiffrat* überführt. Dieses kann nur durch den Einsatz der *Entschlüsselung* (auch Dechiffrierung) in den ursprünglichen Klartext zurück gewandelt werden.

Beim Einsatz von Signaturen wird zu einem Text, dessen Integrität zu schützen und dessen Authentizität zu belegen sind, durch die Operation der *Signaturerstellung* mittels eines *Signaturschlüssels* eine *Signatur* (auch digitale Unterschrift) erstellt. Diese kann genutzt werden, um im Rahmen der *Signaturprüfung* unter Verwendung des *Signaturprüfschlüssels* die Authentizität des signierten Textes nachzuweisen.

Abbildung 2.1 veranschaulicht graphisch diese Zusammenhänge.

2.2 Signaturen

2.2.1 Signaturbegriff

Der Begriff Signatur (von lat. *signatura* von lat. *signare* = unterzeichnen, mit einem Zeichen versehen) steht für ein eindeutig einer Person zugeordnetes Namens- oder sonstiges Zeichen, das Bestandteil eines Dokumentes ist. Im Idealfall ermöglicht eine Eindeutigkeit des Namenszeichens die eindeutige Zuordnung des Zeichens (z. B. einer eigenhändigen Unterschrift) zu einer

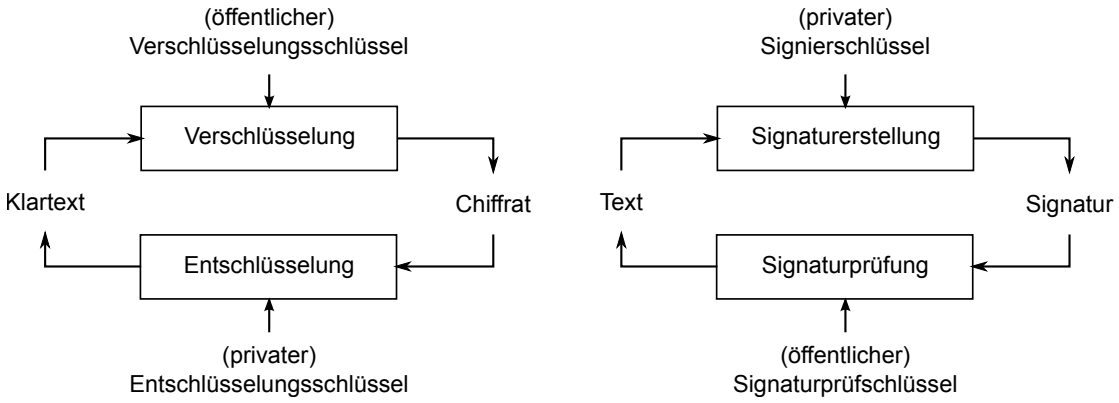


Abbildung 2.1: Übersicht über verwendete kryptographische Grundbegriffe

Person. Ziel einer Signatur ist i. d. R. die Sicherung der Authentizität eines Dokumentes. Darüber hinaus kann die Signatur jedoch auch als Nachweis der Kenntnis oder des Einverständnisses mit dem signierten Inhalt eines Dokumentes eingesetzt werden. Im Folgenden werden die Begriffe Signatur und Unterschrift synonym verwendet.

Der Zusammenhang zwischen signiertem Inhalt und Signatur wird in Abbildung 2.2 skizziert.

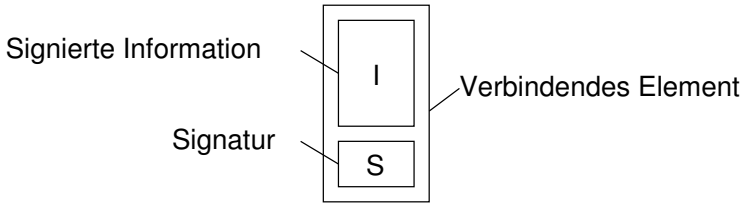


Abbildung 2.2: Zusammenhang zwischen Inhalt und Signatur

Das verbindende Element fügt Inhalt I und Signatur S eindeutig zusammen, so dass der Bezug zweifelsfrei erkennbar ist. Bei herkömmlichen Dokumenten ist dieses verbindende Element die physische Einheit des Trägermediums (z. B. das Papier), auf das Inhalt und Signatur aufgebracht wurden. Bei elektronischen Signaturen übernimmt diese Funktion ein Rechenverfahren (Verschlüsselung des Dokumentes oder eines Hashwertes $\mathrm{H}(I)$ des Dokumentes mit dem privatem Schlüssel K^{-1}) bzw. der durch Anwendung des Verfahrens erzeugte funktionale Zusammenhang zwischen Signatur, signiertem Text und Signaturschlüssel.

2.2.2 Signaturarten

2.2.2.1 Grundprinzip der Signaturerstellung

Die prinzipielle Vorgehensweise bei der Erstellung einer Signatur wird in Abbildung 2.3 dargestellt.

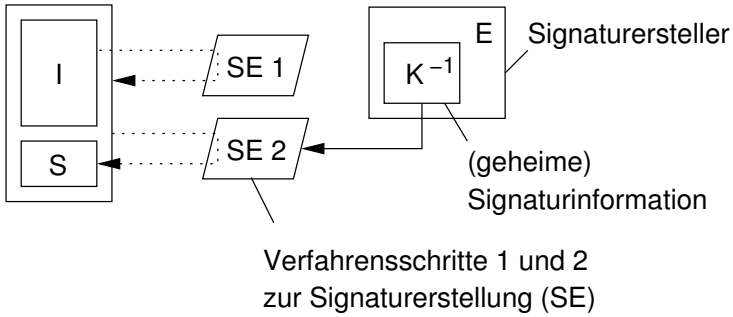


Abbildung 2.3: Signaturerstellung

2.2.2.2 Manuelles Verfahren

In einem ersten Schritt wird die zu signierende Information I erstellt und das verbindende Element bereitgestellt. Dies kann beispielsweise durch Aufschreiben der zu signierenden Information auf ein Blatt Papier erfolgen. Dieser erste Schritt muss nicht vom Signaturersteller ausgeführt werden, sondern kann zeit- und ortsunabhängig durch einen Dritten erfolgen.

Der Signaturersteller erhält die zu signierende Information mit dem verbindenden Element (Text in gedruckter Form) und wendet ein Verfahren zur Erstellung der Signatur an, wobei eine nur ihm bekannte Information oder eine für ihn spezifische Eigenschaft angewendet wird. Bei der Unterschrift wird ein nur vom Signaturersteller erzeugbarer Schriftzug mit dokumentenechten Schreibgeräten auf das Dokument aufgebracht. Dieser Schriftzug kann i. d. R. nur mit dem Wissen (eingeübter Bewegungsablauf) und der Handgeometrie des Signaturerstellers erzeugt werden. Eine alternative Methode ist das Aufbringen eines Fingerabdrucks, bei dem eine eindeutige Eigenschaft des Signaturerstellers (Formgebung der Papillarleisten) zum Einsatz kommt. In modernen biometrischen Zugangskontrollsystemen wird mit daktyloskopischen Verfahren, z. B. die Infrarot-Abtastung, diese Eigenschaft zur Authentifizierung von Personen benutzt.

2.2.2.3 Elektronische Signatur

Wird das Prinzip der manuellen Signatur auf ein elektronisches Dokument übertragen, so kann beispielsweise als verbindendes Element eine Datei dienen, in der die zu signierende Information in Form einer Zeichenfolge gespeichert wird. Zusätzlich können in diese Datei Bilddaten eingefügt werden, die eine Digitalisierung der eigenhändigen Unterschrift des Unterzeichnenden repräsentieren. Diese einfache Form einer digitalen Unterschrift hat jedoch den Nachteil, dass die Bindung zwischen Dokumentinhalt und Unterschrift nur logischer Natur ist, also auf Vereinbarungen beruht (z. B. ist im Dateisystem eines Rechners abgelegt, welche gespeicherten Zeichen zu einer Datei gehören). Auch wenn die logische Zuordnung innerhalb von Systemen geschützt werden kann, so ist der Schutz bei Übertragung über nicht vertrauenswürdige Bereiche (z. B. Versenden per E-Mail) nicht mehr aufrecht zu halten. Ein Angreifer kann problemlos Inhalt oder Signatur austauschen. Darüber hinaus ist es auch möglich, das digitalisierte Abbild der Signatur

beliebig oft zu kopieren und anderweitig zu verwenden. Auf Basis einer solchen elektronischen Signatur ist es daher nicht möglich, die Authentizität einer in einem elektronischen Dokument enthaltenen Aussage zu beweisen.

2.2.2.4 Kryptographische Signatur

Eine Verbesserung der Signatureigenschaften bei elektronischen Dokumenten kann durch Übergang von einer logischen Verknüpfung zwischen Dokumentinhalt und Signatur auf eine kryptographische Verbindung erreicht werden. Damit ermöglichen digitale Signaturen mit Hilfe kryptographischer Mittel, den Absender einer Nachricht eindeutig zu bestimmen. Eine solche kryptographische Signatur wird auch *technische Signatur* genannt.

Nach [MvOV96] ist eine kryptographische Signatur ganz allgemein ein Datum, das eine digitale Nachricht einer Nachrichtenquelle überprüfbar zuordnet. Die Basis für dieses Verfahren ist ein digitales Signaturschema, das aus einem Algorithmus zur Generierung und einem Algorithmus zur Überprüfung von Signaturen besteht. Generell lassen sich nach [PMH96] dabei *Signaturverfahren mit Nachrichtenrückgewinnung* von *Signaturverfahren mit Anhang* unterscheiden. Erstgenannte ermöglichen die Berechnung des signierten Dokumentes aus der Signatur während letztere zusätzlich das signierte Dokument zur Überprüfung der Signatur benötigen. Im Folgenden werden überwiegend Signaturverfahren mit Anhang betrachtet, da diese effizienter bei langen Nachrichten sind.

Bekannte Vertreter der Signaturschemata sind das Signaturschema nach Rivest, Shamir und Adleman (RSA) [RSA78] sowie der Digitale Signatur Algorithmus (Digital Signature Algorithm, DSA) des National Institute of Standards and Technology (NIST)[Nat94], der wiederum auf dem Signaturschema von El Gamal [Gam85] basiert. Daneben haben in den letzten Jahren die Verfahren zur Erzeugung von Signaturen auf Basis von elliptischen Kurven (Elliptic Curve Cryptography, ECC)[KMV00] an Bedeutung gewonnen, die wiederum für die Definition des Elliptic Curve Digital Signature Algorithm (ECDSA) verwendet werden [JMV01].

Ein digitales Dokument kann durch Bereitstellen einer verschlüsselten Zusammenfassung des Dokumentes digital signiert werden. Dabei muss die Zusammenfassung eindeutig dem Dokument zuordenbar sein und die Verschlüsselung muss so gestaltet sein, dass nur genau eine Person oder Instanz diese erzeugt haben kann.

Eine Dokumentzusammenfassung in diesem Sinne wird durch sog. Einweg-Hash-Funktionen bereitgestellt, die einen z. B. 160 oder 256 Bit langen Wert aus dem Inhalt des Dokumentes berechnen. Beispiele für Hash-Algorithmen sind RACE Integrity Primitives Evaluation Message Digest (RIPEMD-160)[ISO04] und Secure Hash Algorithm Version 256 (SHA-256)[Nat02]. Letzterer wird derzeit zur Anwendung empfohlen. Es existieren jedoch auch stärkere Verfahren mit längeren Hashwerten (SHA-384 und SHA-512 [Nat02, Nat04]), die voraussichtlich in den kommenden Jahren die derzeitige Empfehlung in der technischen Richtline 02102 des Bundesamtes für Sicherheit in der Informationstechnik [Bun08c], dem sog. Algorithmenkatalog, ablösen werden. Darüber hinaus wurde der Keccak-Algorithmus [BDPA09] durch das NIST als SHA3-Algorithmus ausgewählt [jCPB+12].

Das Verfahren zur Referenzierung von Dokumenten durch Hash-Werte lässt sich ebenfalls mit der Darstellung in Abbildung 2.3 erläutern. Das zu signierende Dokument entspricht der Information I. Diese wird durch Anwendung der Einweg-Hash-Funktion im Schritt Signaturerstellung 1 (SE1) auf einen Wert $\mathrm{H}(I)$ abgebildet. Dieser Wert ist eindeutig dem Inhalt I zuzuordnen. Durch Verschlüsselung des Wertes $\mathrm{H}(I)$ mit dem privaten Schlüssel K^{-1} des Signaturerstellers E entsteht im Schritt SE2 die eigentliche Signatur

$$S = \{\mathrm{H}(I)\}_{K^{-1}}$$

(zu lesen als: der Hashwert von I verschlüsselt unter dem privaten Schlüssel K^{-1}). Es handelt sich damit um eine Signatur ohne Nachrichtenrückgewinnung. Daher müssen die signierten Daten und die Signatur gemeinsam als (I, S) zur Durchführung einer Signaturprüfung vorliegen. Hierfür wird die kürzere Schreibweise der SVO-Logik verwendet:

$$\lfloor I \rfloor_{K^{-1}} = (I, S) = (I, \{\mathrm{H}(I)\}_{K^{-1}})$$

Die elektronische Signatur kann in ihrer Qualität durch das Einhalten von technischen und organisatorischen Maßnahmen in ihrer Aussagekraft maßgeblich beeinflusst werden. Spezielle Anforderungen an Eigenschaften werden für die *fortgeschrittene* und die *qualifizierte* Signatur gestellt.

2.2.2.5 Fortgeschrittene Signatur

Eine *fortgeschrittene Signatur* ist nach [Bun01a] eine elektronische Signatur, die es ermöglicht, den Signaturschlüsselinhaber eindeutig zu identifizieren. Dies wird durch die Forderung der alleinigen Kontrolle einer Person oder Instanz (Signaturschlüssel-Inhaber) über die Mittel zur Erstellung der Signatur und einen durch die Signatur bedingten Integritätsschutz ermöglicht.

2.2.2.6 Qualifizierte Signatur

Unter einer *qualifizierten elektronischen Signatur* wird nach [Bun01a] eine Signatur verstanden, die alle Eigenschaften einer fortgeschrittenen Signatur aufweist und die zusätzlich auf Basis eines asymmetrischen Schlüsselpaares erzeugt wurde, für dessen öffentlichen Schlüssel zum Zeitpunkt der Erstellung der Signatur ein qualifiziertes Zertifikat vorlag und die mit Hilfe einer als sicher eingestuften technischen Umgebung erzeugt wurde.

Die „Qualifizierung" der Zertifikate wird dabei durch eine Reihe von Anforderungen technischer und organisatorischer Art erreicht, die das Sicherheitsniveau betreffen. Als Beispiele können hier die Speicherung der privaten Schlüssel und die Signaturerstellung in evaluierten Umgebungen (z. Z. lediglich durch spezielle Chipkarten erfüllt), Anforderung an das Schlüsselmanagement wie geteilte Geheimnisse, Protokollierung und Auditierung, aber auch die organisatorischen Abläufe bei der Registrierung usw. genannt werden.

2.2.3 Verwendung von Signaturen

Signaturen werden auf unterschiedlichen Ebenen innerhalb von Kommunikationssystemen angewendet. Auf Sitzungsebenen können Signaturen zur Identifikation und Authentisierung von berechtigten Systembenutzern an Stelle von Passwortmechanismen oder anderen geteilten Geheimnissen eingesetzt werden. So lassen sich Signaturschlüssel z. B. beim Aufbau von SSL-Verbindungen zwischen Browser und Webserver einsetzen.

Auf der Anwendungsebene können Signaturen zum Schutz von Dokumenten wie beispielsweise PDF-Dateien oder zur Authentisierung von E-Mail-Kommunikation eingesetzt werden.

Die Kernaufgabe der Signatur ist es, die Urheberschaft eines signierten Datums eindeutig belegbar zu machen.

Hierzu spielen zwei Eigenschaften zusammen: zum einen wird durch die Anwendung des Signaturverfahrens ein Integritätsschutz über die signierten Daten gelegt. Zum anderen kann durch eine exklusive Zuordnung des asymmetrischen Schlüsselpaares zu einer ausstellenden Instanz eine Aussage über den Urheber der signierten Daten gemacht werden. Beide Eigenschaften werden bei der Signaturprüfung eingesetzt.

Im Fall einer Signatur ohne Nachrichtenrückgewinnung liegt beim Prüfenden die Information $(I', \{\mathrm{H}(I)\}_{K^{-1}})$ vor. Ist er im Besitz des öffentlichen Prüfschlüssels K, so kann er diesen zur Entschlüsselung des Hashwertes und damit zur Signaturprüfung verwenden. Das erhaltene Datum wird als I' bezeichnet, da die Integrität, also die Übereinstimmung mit dem ursprünglich gesendeten Datum I, noch nicht nachgewiesen ist. Zur Prüfung werden zwei Signaturprüfschritte (SP) durchgeführt: Berechnung des Hashwertes über I' (SP1) und Entschlüsselung des verschlüsselten Hashwertes über I (SP2). Auf diese Terme wird folgender Zusammenhang angewendet:

$$\mathrm{H}(I') = \{\{\mathrm{H}(I)\}_{K^{-1}}\}_K \quad \Leftrightarrow \quad I = I' \tag{2.1}$$

Ist diese Identität gegeben, so liegt mit I' das gleiche Datum vor, dass zur Bildung von $\mathrm{H}(I)$ verwendet wurde, damit stellt das vorliegende Werte-Paar eine korrekte angehängte Signatur von I dar:

$$(I', \{\mathrm{H}(I)\}_{K^{-1}}) = \lfloor I \rfloor_{K^{-1}}$$

Aus der Tatsache, dass die Prüfoperation (2.1) mit diesem asymmetrischen öffentlichen Schlüssel durchgeführt wurde, lässt sich aufgrund der Grundeigenschaft der asymmetrischen Verschlüsselung ableiten, dass die Signatur unter Anwendung des korrespondierenden privaten Schlüssels K^{-1} gebildet wurde.

Ist der Prüfschlüssel (Public Key, PK) für Signaturen (Index σ) der exklusiven Verwendung durch einen Ersteller E zugeordnet, so wird dies durch das Prädikat

$$\mathrm{PK}_\sigma(E, K)$$

ausgedrückt. Das Prädikat $\mathrm{PK}_\sigma(E, K)$ impliziert dabei die Annahme der alleinigen Kontrolle von E über K, wie sie in der Definition der fortgeschrittenen Signatur (vgl. Abschnitt 2.2.2.5) gefordert wird.

Mit einer solchen Zuordnung kann die Signatur über I nur vom Inhaber E des zu K gehörenden privaten Schlüssels stammen und somit wird der Ursprung des Datums E zugeordnet. Somit gilt I als authentisch in Bezug auf E.

Abbildung 2.4 veranschaulicht die beschriebenen Prüfschritte.

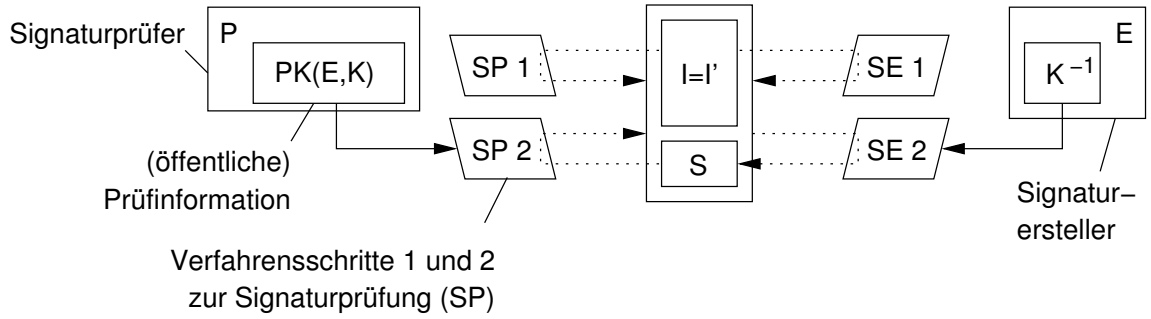


Abbildung 2.4: Signaturprüfung

Diese Signaturvalidierung (SV) kann für angehängte Signaturen auch durch ein entsprechendes dreistelliges Prädikat $\mathrm{SV}(X, K, Y)$ ausgedrückt werden. Dieses wird gelesen als „X geht aus Y durch Anwendung des zu K gehörenden privaten Schlüssels hervor“. Somit ist z. B. $\mathrm{SV}(\lfloor I \rfloor_{K^{-1}}, K, I)$ wahr.

2.2.3.1 Technische Interpretation von Signaturen

Wurde eine Signatur mit Rückgewinnung über ein Dokument erfolgreich geprüft, so kann daraus abgeleitet werden, dass es dem Unterzeichnenden vorlag, er also zumindest temporär im Besitz des Dokumentes war.

Im engeren technischen Sinn ist bei Signaturerstellung mit Hashwert zwar lediglich abzuleiten, dass dem Signaturersteller der Hashwert und nicht zwangsläufig das Dokument vorlag, in der Umsetzung guter Signaturprogramme wird jedoch der Hashwert erst bei der Signaturerstellung gebildet, so dass das zu signierende Dokument vorliegen muss.

Im Allgemeinen wird daher angenommen, dass der Unterzeichnende den Inhalt des Dokumentes, das er unterschrieben hat, auch kennt. Hierbei ist jedoch zwischen menschlichen Anwendern und technischen Programmen zu unterscheiden.

Im Fall der Nutzung von Signaturen zur Authentisierung gegenüber Web-Anwendungen oder Servern werden bei den eingesetzten Protokollen i. d. R. Zufallswerte (Nonces) eingesetzt, die

durch eine Signatur des zu Authentisierenden genutzt werden, um beispielsweise einen Session-Kontext dem Eigentümer des verwendeten Signaturschlüssels zuzuordnen. In einem solchen Fall kennt der menschliche Nutzer die signierten Daten nicht, da diese vom Client-Programm üblicherweise nicht zur Anzeige gebracht werden. Das ist auch der Grund, warum oftmals für Verschlüsselung und Signaturerstellung getrennte Schlüssel verwendet werden. Bei doppelter Nutzung eines Schlüssels für beide Anwendungsfälle könnte ein Angreifer durch Betrieb einer bösartigen Webseite durch geeignete Wahl des zu signierenden Wertes eine Signatur erschleichen.

Daher wird für die weiteren Betrachtungen die Verwendung eines Signaturschlüssels exklusiv für die Erstellung von Signaturen festgelegt.

Die technische Interpretation besagt somit, dass die erfolgreiche Signaturprüfung eines Datums die Beteiligung des Zertifikatsinhabers an der Erstellung der Signatur nachweist. Damit ist ebenfalls ein schwacher Integritätsschutz gegeben (Änderungen an den signierten Daten sind erkennbar).

Die digitale Signatur schafft somit einen kryptographischen Kanal für die signierten Daten mit einem Eingangspunkt beim Signaturersteller und vielen Endpunkten bei den jeweiligen Empfängern der signierten Nachricht, der vor Manipulation durch Dritte schützt (vgl. Abbildung 2.5).

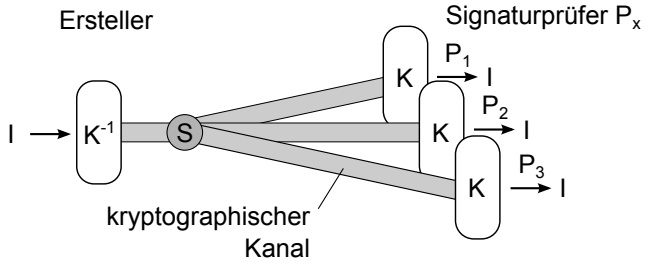


Abbildung 2.5: Verteilung signierter Information

2.2.3.2 Weitergehende Bedeutung von Signaturen

Aus dem Vorhandensein einer Signatur zu einem Datum lässt sich außer der Authentizität zunächst keine weitere Aussage ableiten. Erst durch Betrachtung der signierten Inhalte oder nicht technischer Randbedingungen können weitere Aussagen zur Bedeutung einer Signatur getroffen werden. Daher muss stets zwischen der eigentlichen (technischen) Signatur mit ihrer technischen Funktion und einer darüber hinausgehenden Interpretation (speziell mit Rechtsfolgen) unterschieden werden.

Handelt es sich um eine Erklärung (i. S. einer Willenserklärung), so wird angenommen, dass diese im Sinne des Unterzeichnenden ist. Hat das Dokument einen bestätigenden Charakter oder geht aus der Art der Anbringung der Signatur der Wille zur Bestätigung eines Sachverhalts hervor, so wird angenommen, dass der Signierende von der Richtigkeit des Sachverhalts überzeugt ist.

Werden die Signaturen jedoch auch im Rahmen von Rechtsgeschäften eingesetzt, so ergeben sich weitere Aspekte. Die Form der zu signierenden Daten erhält ein höheres Gewicht und erweiterte organisatorische Randbedingungen müssen eingehalten werden.

Bei der Verwendung der Signaturen zur Erreichung von Schutzzielen der Nicht-Abstreitbarkeit (engl. non-repudiation), wie dies bei Abgabe einer Willenserklärung z. B. bei einem Vertragsschluss oder einer Antragstellung der Fall ist, müssen organisatorische Randbedingungen beachtet werden. In Deutschland sind hier die Festlegungen des Signaturgesetzes [Bun01a] und der zugehörigen Signaturverordnung [Bun01b] relevant, die ihrerseits die Direktive der Europäischen Union zum Rahmenwerk für elektronische Signaturen [EU-00] berücksichtigen.

Diese legen technische und organisatorische Maßnahmen fest, die auf ein möglichst hohes Maß an Sicherheit hinsichtlich der Nicht-Abstreitbarkeit von qualifiziert signierten Dokumenten abzielen. So sind die Verwendung einer Public Key Infrastruktur mit zertifikatsbasierter Verteilung des öffentlichen Schlüssels, die sichere Speicherung des privaten Schlüssels auf einer Chipkarte und die persönliche Registrierung des Schlüsselinhabers bei einer Registrierungsstelle vorgeschrieben. Die verwendeten Schlüsselzertifikate müssen ebenfalls Formvorgaben entsprechen und die ausgebende Zertifizierungsstelle muss bestimmten Sicherheitsanforderungen entsprechen.

Diese Rahmenbedingungen werden teilweise von Standardisierungsgremien aufgegriffen und in den entsprechenden Standards berücksichtigt [SNP04, TT09].

Neben der Aufgabe der Sicherung der Authentizität eines Dokumentes verändert das Vorhandensein einer Signatur ggf. die juristische Form eines Dokumentes. So geht eine Willenserklärung in *Textform* (eine in lesbaren Schriftzeichen fixierte Erklärung oder Mitteilung) erst durch Hinzufügen des Namens des Ausstellers und seiner Unterschrift in die vom Gesetz geforderte *Schriftform* bzw. im Falle eines elektronischen Dokumentes durch Hinzufügen der qualifizierten Signatur in die *elektronische Form* über (§126a BGB und §3 Satz 2 SigG).

Die Schriftform für Erklärungen ist im Bürgerlichen Gesetzbuch [Bun02] für verschiedene Arten von Rechtsgeschäften vorgeschrieben, z. B. Miet- und Pachtverträge auf bestimmte Zeit (§550, §585a), Zeugnisse (§630) und Bürgschaften (§766). Durch §126 BGB ist die elektronische Form der Schriftform grundsätzlich gleichgestellt, sofern durch das Gesetz nicht eine Ausnahme bestimmt ist. Beispiele für solche Ausnahmen sind Bürgschaften (§766), Testamente (§2247) und Darlehensverträge (§492).

Es ist allerdings festzustellen, dass entweder aus den signierten Daten selbst oder den Umständen der Signaturerstellung (z. B. Art der Anwendung) der Charakter der Willenserklärung oder Bestätigung hervorgehen muss, damit diese juristisch motivierten Überlegungen angewendet werden können.

2.3 Standardisierte Signaturformate

Bei den im Weiteren beschriebenen standardisierten Signaturformaten besteht die eigentliche Signatur in der bereits oben beschriebenen kryptographischen Operation mit Hilfe eines asym-

metrischen Schlüssels über einem Hashwert der zu signierenden Daten. Der Unterschied liegt jedoch jeweils in der Einbettung eines solchen Signaturwertes in zusätzliche Daten, die Informationen über die Art und Weise der Erstellung der Signatur und damit ggf. über ihre Semantik enthalten. Ein weiteres Unterscheidungsmerkmal sind zusätzlich organisatorische Randbedingungen, die von den Ausstellern solcher Signaturen und den Zertifizierungsstellen für die öffentlichen Schlüssel eingehalten werden müssen. Teilweise enthalten die Daten, die die Signatur ergänzen, Hinweise auf diese Randbedingungen und verändern damit die Aussage oder Aussagestärke einer digitalen Signatur. Es wird mit PKCS#1 zunächst ein Grundverfahren beschrieben, das teilweise wiederum in anderen Formaten verwendet wird. Bei den darauf aufsetzenden Signaturverfahren wird neben einer kurzen Formatbeschreibung auch auf die jeweiligen Festlegungen zur Bedeutung der Signaturen eingegangen.

2.3.1 PKCS#1

Der Public Key Cryptography Standard No. 1 (PKCS#1) wurde in seiner jüngsten Fassung 2012 durch die RSA Laboratories veröffentlicht [RSA12]. Er legt Verfahren zum Umgang mit RSA-Schlüsseln fest, die auf dem im Jahr 1977 von Ron **Ri**vest, Adi **S**hamir und Leonard **A**dleman entwickelten asymmetrischen Verschlüsselungsverfahren auf Basis großer Primzahlen und des Faktorisierungsproblems beruhen [RSA78].

Dieser Standard definiert Verfahren zur Erstellung einer angehängten Signatur (d. h. es ist keine Rückgewinnung des signierten Textes aus der Signatur möglich). Sie sind die Grundlage vieler Anwendungen, die in den folgenden Teilkapiteln erläutert werden.

Das als *RSASSA-PSS* bezeichnete Signaturschema erzeugt aus der zu signierenden Nachricht zunächst einen Hashwert, der als Oktettfolge vorliegt. Nach Umwandlung dieser Oktettfolge in eine Ganzzahl wird auf diese die Verschlüsselungsoperation mit dem privaten Schlüssel des Signaturerstellers angewendet. Das Ergebnis wird von einer Ganzzahl wieder in eine Oktettfolge umgewandelt und stellt die Signatur S dar.

Zur Prüfung wird aus der Signatur wieder mit Formatwandlungen und Anwendung des aus Modulus n und Exponent e bestehenden öffentlichen Schlüssels (n, e) [RSA78] der Hashwert zurückgewonnen und mit einem neu berechneten Hashwert über die zu prüfende Nachricht verglichen.

2.3.2 PKCS#7

Ebenfalls von den RSA Laboratories wurde der Public Key Cryptography Standard No. 7 (PKCS#7) mit dem Titel *Cryptographic Message Syntax Standard* [RSA93] im Jahr 1993 veröffentlicht.

Dieser Standard legt Kodierungsschemata für signierte und verschlüsselte Nachrichten auf Basis der ASN.1-Kodierung fest. Die damit definierten Objekt-IDs decken die verschiedenen Anwendungsfälle ab: *signedData, envelopedData, signedAndEnvelopedData, digestedData, encryptedData*. Das Prinzip des Umschlags (envelope) verknüpft verschlüsselte Inhalte mit dem für einen

oder mehrere Empfänger asymmetrisch verschlüsselten symmetrischen Schlüssel, der zur Sicherung des Inhalts verwendet wurde.

Gemeinsam mit den signierten Daten können auch Metainformationen wie die verwendeten Algorithmen und zusätzliche Prüfinformationen wie Zertifikate und Rückruflisten von Zertifikaten zusammen mit den signierten Daten und der Signatur kodiert werden.

Der Standard war insbesondere entworfen worden, um das ebenfalls 1993 bei der IETF standardisierte Verfahren zur Bereitstellung gesicherter E-Mail-Kommunikation *Privacy Enhancement for Internet Electronic Mail* [Lin93, Ken93, Bal93, Kal93] zu unterstützen. Dieses wurde später vom S/MIME-Verfahren abgelöst (vgl. Abschnitt 2.3.6), in das auch die Weiterentwicklung der Festlegungen des PKCS#7 übernommen wurden.

2.3.3 XML basierte Signaturen

Das World Wide Web Consortium (W3C), das für viele Standardisierungen von Datenformaten und Protokollen im Anwendungsumfeld des Internets verantwortlich ist, hat im Rahmen seiner Spezifikationen der eXtensible Markup Language (XML) auch Festlegungen für digitale Signaturen getroffen. Dies sind u. a. die Empfehlung XML-Signature Syntax and Processing [BBF+08] sowie die Empfehlung XML Advanced Electronic Signatures (XAdES)[CKPR03].

Das in [BBF+08] entworfene Signaturschema bettet digitale Signaturen in ein XML-Dokument ein, das Angaben über die verwendeten Signaturalgorithmen, Schlüssel u. a. enthält.

Der prinzipielle Aufbau einer XML-kodierten Signatur hat die Form

```
<Signature ID?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI? >
       (<Transforms>)?
       <DigestMethod>
       <DigestValue>
     </Reference>)+
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

Das Referenzelement, das beliebig oft verwendet werden kann, ist ein Verweis auf die eigentlichen Daten, aus denen mit dem angegebenen Verfahren (DigestMethod) ein Hashwert (DigestValue) abgeleitet wurde. Der signierte Bereich des XML-Dokumentes (SignedInfo) wird vor der Berechnung eines Hashwertes noch auf eine vereinheitlichte Darstellung mit dem referenzierten Verfahren (CanonicalizationMethod) gebracht. Schließlich wird dieser Bereich mit den angegebenen Hash- und Signaturalgorithmen (SignatureMethod) signiert. Eine Referenz auf

den zugehörigen Prüfschlüssel kann in KeyInfo angegeben werden. Innerhalb des Object-Tags können Erweiterungen angegeben werden, die XMLSIG zur Ergänzung der Signatur um ein SignatureProperties-Objekt nutzt. Diese Signature Properties können beispielsweise den Zeitpunkt der Signaturerstellung oder eine Gerätenummer der Signatur-Hardware enthalten.

Eine Besonderheit an XML-Signaturen ist das Verwenden der Referenzen auf die eigentlich zu signierenden Daten. Diese werden durch das URI-Attribut adressiert und können entweder externe Dateien oder Teile eines externen oder des aktuellen XML-Dokumentes referenzieren.

Die XML-Signaturen ermöglichen darüber hinaus die Aufnahme einer Angabe zur Semantik innerhalb eines zusätzlichen Objektes. Im nachstehenden Beispiel aus [BBF+08] wird ein Zeitstempel als zur Signatur gehörendes Metadatum aufgenommen, das den Zeitpunkt der Signatur nachvollziehbar macht.

```
<Object>
 <SignatureProperties>
   <SignatureProperty Id="AMadeUpTimeStamp" Target="#MySecondSignature">
      <timestamp xmlns="http://www.ietf.org/rfcXXXX.txt">
        <date>19990914</date>
        <time>14:34:34:34</time>
      </timestamp>
   </SignatureProperty>
 </SignatureProperties>
</Object>
```

2.3.4 Pretty Good Privacy

Pretty Good Privacy (PGP) ist ein ursprünglich von Phil Zimmermann Anfang der Neunziger Jahre entwickeltes Programm, dessen Ziel die Bereitstellung von Kryptographie für jedermann war. Zur Signaturerstellung verwendet das Programm den Digital Signature Algorithm des Digital Signature Standard (DSS) [Nat94]. Weitere Festlegungen zur Kodierung der Nachrichtenformate zum Austausch von PGP-Signaturen sind in den OpenPGP-Festlegungen [CDF+07] zu finden.

Diese definieren ein eigenes binäres Datenformat, das aus Datentypen sog. Pakete zusammensetzt. Diese können signierte Daten, Schlüssel usw. enthalten. Die Signaturen werden wie im Fall von PKCS#1 ebenfalls durch Verschlüsselung von Hashwerten erzeugt.

Am Beispiel von OpenPGP lässt sich der fließende Übergang zwischen Signatur und Zertifikat aufzeigen. Wird eine Signatur über einen öffentlichen Signaturschlüssel erstellt, so kann dieser Signatur in einer zusätzlichen Kennung (Typ-Feld) eine entsprechende Zertifikatssemantik zugeordnet werden.

```
There are a number of possible meanings for a signature, which are
specified in a signature type octet in any given signature.
[...]
These meanings are as follows:
```

```
0x00: Signature of a binary document.
          This means the signer owns it, created it, or certifies that it
          has not been modified.
[...]
0x11: Persona certification of a User ID and Public Key packet.
          The issuer of this certification has not done any verification
          of the claim that the owner of this key is the user ID
          specified.
[...]
0x13: Positive certification of a User ID and Public Key packet.
          The issuer of this certification has done substantial
          verification of the claim of identity.
```

Neben den beiden zur Authentisierung von Dateien (einschließlich Information zur Kodierung) vorgesehenen Typen 0x00 und 0x01 sind einige Typen aufgeführt, die die Signaturen im Zusammenhang mit der Weitergabe von Schlüsselmaterial Dritter verwenden. Hierbei wird der Signatur nicht nur eine spezielle Bedeutung zugewiesen, sondern auch die Qualität der bestätigten Aussage klassifiziert.

Als Hash-Verfahren werden durch OpenPGP u. a. SHA-1 sowie die SHA-2-Familie unterstützt. Als PGP-Schlüssel werden DSA-Schlüssel mit einer Länge von bis zu 3072 Bit oder ein ElGamal-Schlüssel verwendet.

2.3.5 Portable Document Format

Das von der Firma Adobe entwickelte Portable Document Format (PDF), das im Jahr 2008 durch die ISO standardisiert wurde [Int08], unterstützt seit der Version 1.3 digitale Signaturen innerhalb von PDF-Dokumenten. Die Signaturen können sich dabei auf das gesamte Dokument (ohne den Signaturwert selbst) oder auf Teile des Dokumentes beziehen.

Die zur Aufnahme der Signaturen vorgesehenen Datenstrukturen enthalten neben den notwendigen kryptographischen Elementen und Zeitangaben auch die Möglichkeit zur Angabe zusätzlicher Informationen. Diese sind als Freitext gestaltet und können z. B. die Kontaktadresse des Signaturerstellers enthalten. Ein Feld dient der Angabe einer Signaturbedeutung. Hier lassen sich Inhalte wie „Ich stimme dem Inhalt zu", „Ich habe den Inhalt zur Kenntnis genommen" oder „Ich genehmige den Inhalt" hinterlegen. Dieses als Signaturgrund (Reason) bezeichnete Feld kann dann durch die auswertende Applikation angezeigt werden und der Signatur damit eine Bedeutung zuordnen.

In der Version 1.5 wurde die PDF-Funktionalität noch um Signaturen zur gesicherten Festlegung von Anwendungsrechten erweitert. Dabei werden die Rechte zum Drucken, Exportieren, Verändern etc. in einem durch eine Signatur geschützten Bereich festgelegt.

In der ETSI-Standard-Reihe PAdES [Eur09] werden Verfahren zur Erstellung von Signaturen über PDF-Dokumente, Signaturen über Formulardaten in PDF-Dokumenten, der Einsatz von Zeitstempeln, Mehrfach-Signaturen und Signaturen zur Langzeitarchivierung von PDF-Dokumenten beschrieben.

2.3.6 S/MIME

Secure / Multipurpose Internet Mail Extensions (S/MIME) ist ein Standard zum Austausch digital signierter und/oder verschlüsselter E-Mail. Er wurde ursprünglich von RSA Data Security Inc. entwickelt (damals als PKCS#7) veröffentlicht und in RFC 2315 aufgegriffen [Kal98]. Seit 2004 werden die Formate zum Austausch verschlüsselter und/oder signierter E-Mail durch die Cryptographic Message Syntax (CMS) gemäß RFC 5652 [Hou09] und RFC 4853 [Hou07] festgelegt. Es wird durch eine große Zahl von E-Mail-Programmen bereitgestellt und ist im kommerziellen Umfeld die wichtigste Form der gesicherten elektronischen Kommunikation über E-Mail.

Die CMS bietet die Möglichkeit, die Signatur des eigentlichen Dokumentes noch durch sog. signierte Attribute zu ergänzen. Diese werden dadurch fest in die Signatur integriert und können dadurch zur Interpretation der Signatur beitragen. Beispiele für signierte Attribute sind der Signaturzeitpunkt, Verweise auf Signaturerstellerzertifikate, Verweise auf Sicherheitsrichtlinien und Sicherheitslabel, mit denen z. B. die Vertraulichkeitsstufe einer Information festgelegt werden kann [Hof99, Sch07, ets12].

2.4 Digitale Zertifikate

2.4.1 Zertifikatsbegriff

Ein *Zertifikat* (von lat. *certificatum* von lat. *certus* = sicher, gewiss, zuverlässig und facere = machen, errichten) ist ein Dokument, das die Zusicherung der Korrektheit eines Sachverhalts enthält. Die Zusicherung wird vom Aussteller des Dokumentes unterschrieben und damit deren Echtheit nachgewiesen.

Ein Zertifikat hat die Aufgabe, dem Anwender die Prüfung des im Zertifikat genannten Sachverhalts zu ersparen, indem dieser durch einen für den Anwender vertrauenswürdigen Dritten bestätigt wird.

Im Allgemeinen weisen Zertifikate gewisse Form- und Inhaltsmerkmale auf, die eine Erkennung des Bestätigungscharakters des Dokumentes ermöglichen. So sind beispielsweise Urkunden bei der Ausstellung von Zertifizierungen nach ISO 9001 durch entsprechende Überschriften und Formulierungen so gestaltet, dass die Bestätigungsabsicht der erklärenden Stelle (z. B. Dekra) klar erkennbar ist. Einleitende Sätze im textlichen Inhalt wie „Hiermit bestätigen wir ...” oder „Der Unterzeichnende garantiert die Richtigkeit ...” können ebenfalls ein Erkennungsmerkmal für die Bestätigungsabsicht sein.

Da die Bestätigung durch einen Dritten nur dann sinnvoll einsetzbar ist, wenn der Dritte bekannt und bzgl. des bestätigten Sachverhalts als kompetent und vertrauenswürdig angesehen wird, muss er zweifelsfrei identifiziert und die von ihm gemachte Aussage auch authentifiziert werden. Um dies zu ermöglichen, enthalten Zertifikate i. d. R. Angaben zum Bestätiger, also dem Aussteller des Zertifikates, und entsprechende Echtheitsnachweise wie Siegel und Signaturen. Teilweise kann eine Ausstellerangabe entfallen, wenn aus dem Echtheitsnachweis der Aussteller

hervorgeht. Die Ausstellerangabe ist jedoch auch dann sinnvoll, wenn sie lediglich zur Unterstützung der Auswahl der korrekten Prüfinformation dient (z. B. passende Unterschriften- oder Stempelprobe).

2.4.2 Zertifikatsarten

Zertifikate lassen sich nach der Art der enthaltenen Zusicherungen klassifizieren. In der Informationstechnik haben vor allem Schlüssel- und Attributszertifikate eine herausragende Bedeutung. Ein weiteres Anwendungsfeld von Zusicherungen sind Verfahren zur (verteilten) Verwaltung von Vertrauensbeziehungen und der Zugriffssteuerung. Diese im Englischen als Trust Management bzw. Access Control bezeichneten Verfahren verfügen über ihre eigenen Zusicherungsarten und umfassen daher im o. g. Sinne ebenfalls eine Art Zertifizierung.

2.4.2.1 Schlüsselzertifikate

An erster Stelle sind die *Schlüsselzertifikate* (public key certificates) zu nennen, die auch als *Identitätszertifikate* bezeichnet werden. Diese haben die Aufgabe, einer Identität (Person, Institution oder Kommunikationssystem wie Server u. ä.) einen öffentlichen Schlüssel zuzuordnen. Dies geschieht durch Aufnahme eines möglichst eindeutigen Bezeichners der Identität des Schlüsselinhabers und des öffentlichen Schlüssels in die Bestätigung. Ein solches Zertifikat soll aussagen, dass das im Zertifikat genannte Subjekt alleinige Kontrolle über den privaten Schlüssel hat, der zu dem im Zertifikat genannten öffentlichen Schlüssel gehört. Damit weist ein Zertifikat das Vorliegen der Voraussetzungen zur Erstellung von fortgeschrittenen oder qualifizierten Signaturen entsprechend der Abschnitte 2.2.2.5 bzw. 2.2.2.6 nach.

Die Kernaussage eines Schlüsselzertifikates lässt sich durch ein zweistelliges Prädikat darstellen, das über eine Relation über Identitäten und Schlüssel definiert ist. Dabei wird der jeweilige Einsatzzweck des Schlüssels unterschieden:

$\mathrm{PK}_\sigma(E, K_1)$	K_1 ist ein Signaturschlüssel von E
$\mathrm{PK}_\varphi(E, K_2)$	K_2 ist ein Verschlüsselungsschlüssel von E
$\mathrm{PK}_\delta(E, K_3)$	K_3 ist ein Schlüsselaushandlungsschlüssel von E

Daneben enthalten Schlüsselzertifikate eine Reihe weiterer Metadaten, auf die anhand eines Beispiels in Abschnitt 2.5 eingegangen wird.

Eine Abwandlung von Schlüsselzertifikaten enthält statt der Identität des Schlüsselinhabers lediglich ein Pseudonym. Die Zuordnung zwischen Pseudonym und wahrer Identität ist i. d. R. nur dem Zertifikatsaussteller bekannt. Ein Anwendungsfall eines Pseudonymzertifikates ist z. B. ein Zertifikat über einen Signaturschlüssel, der für die Rechnungserstellung im Namen eines Unternehmens verwendet werden soll. Die Identität des Mitarbeiters, der den Schlüssel kontrolliert, wird durch ein Pseudonym wie z. B. *CN=Rechnungsstelle, O=Firma XYZ* ersetzt.

2.4.2.2 Attributszertifikate

Die zweite wichtige Gruppe wird durch die *Attributszertifikate* gebildet. Diese ordnen einer Identität statt eines Schlüssels eine Eigenschaft (Attribut) zu. Das können zum einen Bestätigungen von mit dem Subjekt fest verbundenen Eigenschaften sein, wie Zulassungen zu einer Berufsgruppe, Volljährigkeit, etc. Zum anderen können in einem Zertifikat dem Subjekt auch Rechte und/oder Rollen zugeordnet werden. Ein typischer Anwendungsfall für Attribute ist das Verknüpfen von Privilegien wie Zugriffsrechte u. ä. mit einem berechtigten Subjekt. Diese spezielle Form der Attributszertifikate wird als *Berechtigungs-* oder *Delegationszertifikat* bezeichnet.

Das Subjekt kann direkt durch Angaben eines Namens oder, im Fall von digitalen Subjekten (z. B. Programmcode), durch einen Objekt-Hashwert bestimmt werden. Eine zweite Möglichkeit besteht in der Referenzierung eines Schlüsselzertifikates, so dass die Attribute für den Halter des Schlüsselzertifikates gelten. Ein spezieller Fall eines Objekt-Hashwertes ist der Hash eines öffentlichen Schlüssels, bei dem dann wiederum dem Besitzer des Schlüssels die Attribute zugeordnet werden.

Der RFC 5755 [FHT10] spezifiziert Attributszertifikate, die auf Basis des X.509-Standards bzw. des RFC 5280 [CSF+08] die Zusicherung von beispielsweise Rollen, Gruppenzugehörigkeiten, Sicherheitsfreigaben u. a. ermöglichen.

2.4.2.3 Mischformen

Neben der Reinform der oben genannten Zertifikatsarten existieren auch *Mischformen*. Hierbei werden in Schlüsselzertifikate zusätzlich Attributsangaben in Form von Erweiterungen (engl. extensions) aufgenommen. Der Common-PKI-Standard [TT09] definiert beispielsweise Erweiterungen wie Berufszulassung (admission) und Vertretungsberechtigungen (procuration), die wahlweise als Attributszertifikat oder als Extension im Schlüsselzertifikat verwendet werden können.

2.5 Schlüsselzertifikate am Beispiel des X.509-Standards

Die International Telecommunication Union (ITU-T) hat im Rahmen ihrer X.500-Serie, die Formate und Mechanismen für Verzeichnisdienste bereitstellt, auch einen Standard X.509 für Zertifikate erstellt. X.509-Zertifikate sind derzeit die bedeutendsten Zertifikate, da mit ihnen sehr viele Sicherheitsmechanismen parametrisiert werden. Wichtige Vertreter sind z. B. die Verschlüsselungsmechanismen der WWW-Browser (Secure Socket Layer) und Secure/Multipurpose Internet Mail Extensions [DHR+98, DHRW98], die beide auf X.509-Zertifikaten aufsetzen und damit die beiden wichtigsten Kommunikationsanwendungen WWW und E-Mail abdecken.

Das Ziel der X.509-Festlegungen ist es, ein Rahmenwerk für den sicheren Austausch von Informationen über die Zuordnung öffentlicher Schlüssel oder Attributen zu Personen oder Institutionen bereitzustellen und so Authentisierungs- und Zugriffssteuermechanismen zu unterstützen.

Feldbezeichner	Bedeutung
Version	gibt den Typ des Zertifikates und damit die möglicherweise enthaltenen Felder an
Seriennummer	Eindeutige Nummer innerhalb aller vom Aussteller erstellten Zertifikate
Algorithmenkennung	Signatur- und Hashalgorithmen, die für die Erstellung der digitalen Unterschrift des Ausstellers verwendet wurden
Ausstellerkennung	Name der ausstellenden Instanz
Subjektkennung	Name des Zertifikatshalters (Subjekt)
Öffentlicher Schlüssel	der dem Subjekt zugeordnete öffentliche Schlüssel
Gültigkeitsintervall	Zeitangaben *nicht vor* und *nicht nach*, die die Gültigkeitsdauer des Zertifikates angeben
Erweiterungen	Felder für zusätzliche Informationen

Tabelle 2.1: Aufbau eines Schlüsselzertifikates nach X.509

2.5.1 Zertifikatsinhalt

X.509-Zertifikate enthalten in der Version 3 die in Tabelle 2.1 aufgeführten Angaben.

Neben Subjekt- und Schlüsselfeldern, die den eigentlichen Kern der Zusicherung des ID-Zertifikates ausmachen, wird der Aussteller genannt. Diese Angabe dient einerseits zur Bewertung der Glaubwürdigkeit des Zertifikates und zum Auffinden des benötigten Prüfschlüssels für das Zertifikat.

Die Algorithmenkennung wird für die technische Durchführung der Signaturprüfung benötigt. Sie beschreibt sowohl die Art des Schlüssels (z. B. RSA oder ECC) und den verwendeten Hash-Algorithmus (z. B. SHA-256).

Das Gültigkeitsintervall ist eine Meta-Information, die die Dauer der Verwendbarkeit der Aussage über die Zuordnung von Identität und Schlüssel einschränkt.

Zusätzlich können noch optionale eindeutige Kennungen für Aussteller und Subjekt in Form von Bitstrings aufgenommen werden, falls die entsprechenden Angaben der Pflichtfelder diese Eindeutigkeit nicht gewährleisten.

2.5.2 Erweiterungsfelder

Der X.509-Standard sieht eine Reihe von Erweiterungen vor, die dazu dienen, die Semantik eines Zertifikates genauer festzulegen. Daneben ist es auch möglich, beliebige anwendungsspezifische Erweiterungen zu definieren.

Die standardisierten Erweiterungen enthalten unter anderem

- Informationen über die Zertifizierungsrichtlinie
- Einschränkungen zur möglichen Verwendung des Schlüssels
- Bezeichner für den Schlüssel
- zusätzliche Information über Aussteller und Subjekt (z. B. alternative Namensformen, E-Mail-Adressen etc.)
- Einschränkungen der Zertifikatsketten
- Angaben über Ausgabestellen für Revokationslisten

Erweiterungsfelder können als kritisch markiert werden. In diesem Fall muss eine Anwendung den Feldinhalt bei der Auswertung des Zertifikats interpretieren können, anderenfalls muss sie das Zertifikat als ungültig verwerfen.

2.5.3 Kodierung der X.509-Zertifikate

Bei der Spezifikation von X.509-Zertifikaten wurde u. a. großer Wert auf eine platzsparende Kodierung gelegt. Daher sind die Datenfelder gemäß der Distinguished Encoding Rules (DER) entsprechend ihrer in ASN.1 vorliegenden Definition kodiert.

Diese Kodierung ist redundanzarm und so enthalten X.509-Zertifikate neben den eigentlichen Feldwerten kaum Informationen, die direkt etwas über die Bedeutung der Felder aussagen. Lediglich die Feldtypen sind eindeutig kodiert und können unter Verwendung des Standards interpretiert werden. Es wird bei ASN.1 eine sog. Object ID (OID) angegeben, um den nachfolgenden Datentyp zu charakterisieren. Für eine Adresse zum Abruf von Sperrlisten ist dies beispielsweise die OID id-ce-CRLDistributionPoints mit dem Wert 2.5.29.31.

Bei der Dekodierung für menschliche Anwender kann die Bedeutung demzufolge nicht unmittelbar aus dem Zertifikat entnommen werden. Sie ist fest in den Dekodier- und Anzeigeprogrammen hinterlegt, da sie lediglich als allgemeiner beschreibender Text im Standard vorliegt. Beispielsweise heißt es für das Feld *CRL distribution points*:

> This field identifies the CRL distribution point or points to which a certificate user should refer to ascertain if the certificate has been revoked. A certificate user can obtain a CRL from an applicable distribution point or it may be able to obtain a current complete CRL from the authority directory entry.

Mittels eines Programms wie openssl wird der Feldinhalt mit entsprechendem Label angezeigt.

```
X509v3 CRL Distribution Points:
 Full Name:
  URI:ldap://ldap.nrca-ds.de:389/CN=CRL,
    O=Bundesnetzagentur,C=DE,dc=ldap,dc=nrca-ds,
    dc=de?certificateRevocationList;
    binary?base?objectClass=cRLDistributionPoint
```

Alle weiteren Informationen über die Bedeutung müssen dem Standard oder der Sekundärliteratur entnommen werden.

2.5.4 Zertifizierungsrichtlinien

Eine weitere Festlegung zur Bedeutung der Zertifikatsinhalte sind sog. Zertifizierungsrichtlinien (certificate policies). Sie definieren Verpflichtungen hinsichtlich der eingesetzten Verfahrensweisen, die Aussteller, Inhaber und Anwender von Zertifikaten eingehen, wenn ein Zertifikat durch eine CA ausgestellt wird, die sich auf diese Richtlinien beruft bzw. ein solches Zertifikat eingesetzt oder akzeptiert wird. Der X.509-Standard und die zugehörigen RFCs gehen nicht auf den Inhalt von Richtlinien ein und benennen lediglich den Mechanismus der Bindung: durch das Akzeptieren des öffentlichen Schlüssels der Root-CA als Vertrauensanker oder durch Verwendung eines Zertifikates mit expliziter Policy-Referenz akzeptieren die Anwender die jeweiligen Zertifizierungsrichtlinien [CFS+03].

Die Richtlinien können sowohl den Inhabern von Zertifikaten Verpflichtungen auferlegen, als auch die Anwendungsbereiche der Zertifikate bzw. der darin enthaltenen Schlüssel einschränken. Darüber hinaus kann auch der Kreis der möglichen Zertifikatsanwender vorgegeben werden. Hiervon wird insbesondere bei qualifizierten Zertifikaten Gebrauch gemacht, wenn Zertifikatsinhaber vertraglich an besondere Sorgfaltspflichten beim Umgang mit dem Schlüsselmaterial verpflichtet werden.

2.5.5 Gültigkeitsprüfung

Bei der Zertifizierung öffentlicher Schlüssel werden die Zertifikate der Endbenutzer (Anwenderzertifikate) durch einen Aussteller (CA) bereitgestellt. Der Schlüssel des Ausstellers wird i. d. R. wiederum durch mindestens eine weitere CA bestätigt. Die letzte Ausstellerinstanz in der Kette von Ausstellerzertifikaten wird als *Wurzel* bezeichnet und durch ein selbstausgestelltes Zertifikat gebildet, bei dem Aussteller und Subjekt identisch sind und das selbstsigniert ist. Bei selbstsignierten Zertifikaten gehört der öffentliche Schlüssel im Zertifikat zum privaten Schlüssel, mit dem das Zertifikat signiert wurde. Ein selbstsigniertes Zertifikat kann nicht mit einem weiteren geprüft werden und das Vertrauen in dieses Zertifikat muss über einen anderen Weg etabliert werden.

Die Wurzelinstanz stellt häufig auch zusätzliche Zertifikate für Schlüssel aus, die zur Signatur von Sperrlisten oder Online-Auskünften zum Zertifikatsstatus benötigt werden (vgl. Abbildung 2.6).

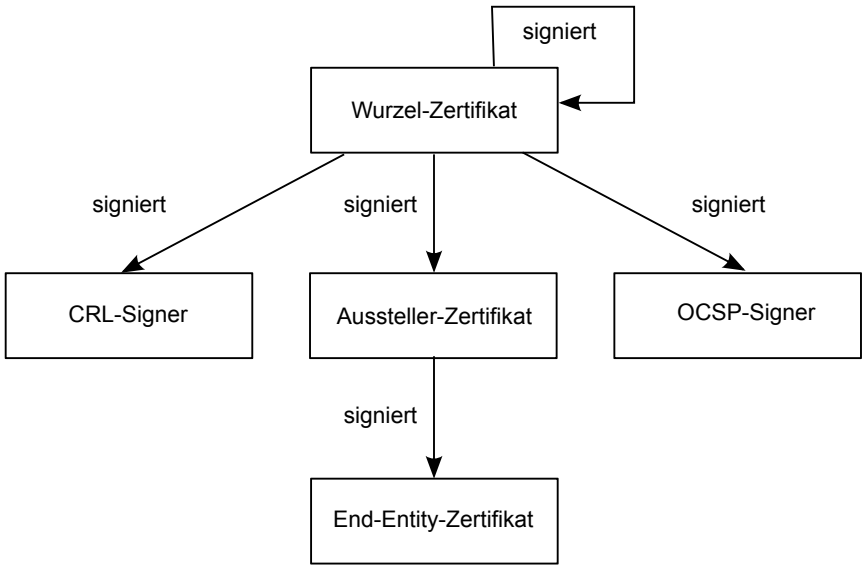


Abbildung 2.6: Beispiel einer Zertifikatshierarchie

Ein Zertifikat eines Ausstellers für einen anderen Aussteller wird als Kreuzzertifikat (cross certificate) bezeichnet. Diese können sowohl hierarchisch als auch auf gleicher Ebene zur Anerkennung einer parallelen CA-Kette verwendet werden [Tur00, CSF+08].

Um ein Anwenderzertifikat zu überprüfen, muss eine Kette von Zertifikaten vorliegen, bei der jeweils eine Ebene die nächste zertifiziert, d. h. für jede CA außer der obersten Root-CA liegt ein von einer hierarchisch übergeordneten CA ausgestelltes Zertifikat vor.

Bei der Überprüfung wird zunächst die zeitliche Gültigkeit der Zertifikate und die korrekte Verkettung von Aussteller und Subjektnamen überprüft. Gegebenenfalls werden die Vorgaben zu den Zertifizierungsrichtlinien auf Eignung für den jeweiligen Anwendungsfall geprüft. Die jeweiligen Signaturen müssen durch die von den übergeordneten Instanzen zertifizierten öffentlichen Schlüsseln prüfbar sein. Es werden die Berechtigungen zur Ausstellung von Zertifikaten anhand der erlaubten und ausgeschlossenen Namensräume bestimmt. Eine detaillierte Darstellung zur Bildung von Zertifikatspfaden zur Prüfung gibt [CDH+05].

Für die verwendeten Zertifikate können online zusätzlich noch aktuelle Sperrlisten geholt oder eine Abfrage an einen im Zertifikat angegebenen Validierungsdienst gesendet werden, um zu prüfen, ob das Zertifikat nicht zwischenzeitlich gesperrt wurde.

Das letzte Zertifikat einer Kette ist nicht mehr über den beschriebenen Mechanismus zu prüfen. Es wird daher auch als Vertrauensanker bezeichnet, d. h. der Anwender muss bei der Zertifikatsprüfung auf die Richtigkeit dieses Wurzelzertifikates vertrauen.

Da sich der Prüfprozess letztlich auf die Korrektheit des letzten Zertifikates einer Kette abstützt, kommt diesem Vertrauensanker eine besondere Bedeutung zu. Es ist daher beispielsweise nicht

möglich, den Vertrauensanker über den gleichen (ungeschützten) Kommunikationsweg zur Verfügung zu stellen, über den die signierte Information übertragen werden kann. Ein übliches Verfahren zur Verteilung ist die Auslieferung innerhalb von Betriebssystemen oder Progammpaketen. Dabei muss der Anwender dem Hersteller oder Distributor seines Betriebssystems bzw. Programmpaketes vertrauen oder einen Vergleich eines Hashwertes (sog. Fingerprint-Vergleich) mit dem Herausgeber des Root-Zertifikates durchführen.

2.5.6 Rückrufmechanismen

Ein Grundproblem der Ausgabe von Zusicherungen in Form von Zertifikaten ist die nachhaltige Sicherstellung der Gültigkeit der zugesicherten Information. Ein Zertifikat sichert einen bestimmten Sachverhalt für die im Zertifikat enthaltene Gültigkeitsdauer zu. Teilweise ist mit der Zusicherung auch eine gewisse Haftung des Ausstellers verbunden (z. B. Warranty Certificate Extension nach [LPS05]).

2.5.6.1 Zertifikatssperrung

Ändert sich der zugesicherte Sachverhalt vor Ablauf der Gültigkeitsdauer oder wird ein kryptographischer Schutzmechanismus unterlaufen (z. B. durch Aufdeckung des privaten Signaturschlüssels), so muss die Zusicherung zurückgenommen werden. Es liegt in der Regel sowohl im Interesse des Ausstellers als auch des Anwenders, dass die Gültigkeit eines Zertifikates zum Zeitpunkt der Anwendung sichergestellt werden kann.

Zu den Gründen für das Ungültigwerden eines Schlüsselzertifikates zählen u. a. die Änderung des Inhalts von Zertifikatsbestandteilen (Namen, Schlüssel, Attribute usw.), der Wegfall des Zertifizierungsgrundes (z. B. Arbeitgeberwechsel des Zertifikatshalters) oder der Verlust des Schlüssels.

Wird eine Zertifikatsaussage ungültig, so muss dieser Umstand möglichst schnell, effizient und zuverlässig an die potenziellen Anwender des Zertifikates übermittelt oder ihnen diese Information zumindest zugänglich gemacht werden. Der Vorgang der Zurücknahme ungültiger Zertifikate wird als *Zertfikatssperrung* oder als *Zertifikatsrückruf* (von engl. revocation) bezeichnet. Hierzu gibt es eine Reihe von Mechanismen.

In [Mye98] werden vier Klassen von Mechanismen unterschieden: Rückruflisten (Certificate Revocation List, CRL), Online-Abfragen, vertrauenswürdige Verzeichnisdienste und Zertifikate mit kurzer Lebensdauer. Im Bereich der X.509-Zertifikate sind vor allem die beiden erstgenannten Mechanismen von Bedeutung.

2.5.6.2 Rückruflisten

Das Grundprinzip der Rückrufliste (Certificate Revocation List, CRL) beruht auf der Veröffentlichung einer Liste von Zertifikatsseriennummern einer CA, die vor dem Ablauf ihrer Gültigkeit

gesperrt wurden. Die Liste selbst enthält einen Zeitstempel, da sie i. A. lediglich eine Momentaufnahme ist, und wird vom Aussteller digital signiert.

Zu jedem gesperrten Zertifikat wird in der Liste der Sperrzeitpunkt und optional auch der bei der Sperrung genannte Sperrgrund bekannt gegeben.

2.5.6.3 Direkte Abfrage des Zertifikatsstatus

Durch die IETF wurde in [MAM+99] ein Mechanismus spezifiziert, der die direkte Abfrage des Status eines oder mehrerer explizit genannter Zertifikate ermöglicht. Im Gegensatz zu den Lösungen auf Basis von CRLs wird die signierte Auskunft erst zum Zeitpunkt der Abfrage erstellt und kann damit eine Auskunft mit der maximal möglichen Aktualität bereitstellen.

Dieses Online Certificate Status Protocol (OCSP) besteht aus einer einfachen Anfrage (request), in der der Zertifikatsanwender dem sog. OCSP-Responder eine oder mehrere Zertifikatsidentitäten (bestehend aus einem Hashwert über je den CA-Namen und den CA-Schlüssel sowie die Seriennummern der Zertifikate) und optionale Anfrageparameter wie einen Zufallswert zum Schutz vor Angriffen durch Wiedereinspielen alter Nachrichten oder die gewünschten Antworttypen übermittelt.

Im OCSP-Responder sind in Form einer Datenbank alle Zustände der Zertifikate einer zugeordneten CA hinterlegt, so dass dieser eine Antwort generieren kann. Die Antwort ist in der Regel digital signiert. Der dafür verwendete Schlüssel muss entweder der zugeordneten CA, einem allgemein vertrauenswürdigen OCSP-Dienst oder einer durch die CA mit dem OCSP-Dienst beauftragten Instanz gehören. Zu jedem angefragten Zertifikat enthält die Antwort eine Statusinformation aus den drei Möglichkeiten „good", „revoked" und „unknown". Bei Vorliegen von Sperrinformationen kann es sich um eine vorübergehende Sperrung (on-hold) oder den endgültigen Widerruf des Zertifikates handeln.

2.6 Standardisierte Zusicherungen und Zertifikate

Im vorangehenden Abschnitt wurde das Prinzip einer kryptographisch geschützten Zusicherung am Beispiel der Zertifikate nach X.509 ausführlich vorgestellt. X.509-Zertifikate decken jedoch nur einen speziellen Bereich der Zusicherungen ab. Neben diesen Zertifikaten gibt es noch weitere Formen von Zertifizierungen und Zusicherungen, auf die im Folgenden eingegangen wird.

Neben weiteren Formen von Schlüsselzertifikaten werden insbesondere Mechanismen aus dem Bereich Attributszertifikate vorgestellt, die im Bereich von Autorisierungs- und sog. Trust-Management-Systemen verwendet werden. Bei diesen Systemen sind naturgemäß Zusicherungen (assertions) und teilweise die Weitergabe von Rechten (delegation) ein wichtiger Bestandteil. Diese kommen den in Kapitel 1 geforderten Möglichkeiten für die Bestätigung von Sachverhalten durch ihr breiteres Anwendungsfeld näher als die reinen Schlüsselzertifikate.

2.6.1 Pretty Good Privacy

In PGP-Zertifikaten werden Kennungen in Form von E-Mail-Adressen öffentlichen Schlüsseln (i. d. R. RSA) zugeordnet. Im Gegensatz zu der in X.509 vorgesehenen zentralen Key-Management-Infrastruktur mit Zertifizierungsstellen (CAs) wird in der „Pretty Good Privacy"-Welt auf die Zertifizierung durch andere Schlüsselinhaber vertraut. Auf diese Weise entsteht ein so genanntes Web of Trust, also ein Netzwerk sich gegenseitig vertrauender Benutzer, die wiederum andere in diese Gemeinschaft einbinden können.

2.6.2 SPKI/SDSI

Im RFC 2693 der IETF [EFL+99] stellt eine Autorengruppe um C. Ellison eine spezielle Variante der Anwendung von Zertifikaten unter der Bezeichnung (Simple Public Key Infrastructure, SPKI) vor. Kerngedanke dabei ist die Verwendung der öffentlichen Schlüssel anstelle von Benutzernamen als Kennungen. Die zugehörige Theorie wird in [HK00] formal dargestellt.

Während die in Abschnitt 2.4.2.2 vorgestellten Attributszertifikate einer Identität ein Attribut zuordnen und über die Identität und ein Schlüsselzertifikat wiederum ein für die Authentisierung geeigneter Schlüssel ermittelt werden kann, wird dieser Ansatz von SPKI umgekehrt.

SPKI verfolgt einen schlüsselzentrischen Ansatz. Hierbei werden Attribute unmittelbar kryptographischen Schlüsseln zugeordnet. Dadurch vereinfacht sich die auf Attributen beruhende Autorisierungsprüfung im Vergleich zu dem oben beschriebenen Ansatz.

Bei Überprüfung der berechtigten Nutzung wird nur die Tatsache geprüft, ob die das Zertifikat vorweisende Instanz den zugehörigen privaten Schlüssel besitzt. Die Kenntnis des Identitätszertifikates ist an dieser Stelle nicht mehr notwendig. Sogar die Kenntnis der zum öffentlichen Schlüssel gehörenden Identität ist für den Prüfenden nicht mehr notwendig. Voraussetzung ist hierbei jedoch die Annahme der korrekten Attributierung durch den Aussteller des Attributszertifikates. Zur Kodierung der Zertifikate werden die von LISP bekannten S-Expressions verwendet. SPKI umfasst ein Delegationskonzept, bei dem jede beteiligte Instanz Rechte delegieren kann, sofern in einem Berechtigungszertifikat das *propagate*-Feld enthalten ist.

2.6.3 Kerberos

Kerberos ist ein am Massachusetts Institute of Technology (MIT) entwickeltes Protokoll sowie eine zugehörige Dienstarchitektur auf Basis eines zentralen Servers. Kerberos stellt einen Single-Sign-On-Dienst für vertrauenswürdige Systeme mit unsicheren Kommunikationsverbindungen bereit.

Der zentrale Server authentifiziert einen Dienstnutzer und stellt für ihn ein sog. Ticket für die Nutzung eines Dienstes aus. Der Vermittler agiert als vertrauenswürdiger Dritter aus Sicht des Dienstes und des Nutzers. Zum Einsatz kommt das Kerberos-Protokoll, das in seiner aktuellen Version 5 im RFC 4120 [NYHR05] definiert ist und auf [NT94, KN93] beruht. Es ermöglicht

den kryptographischen Schutz der ausgetauschten Authentisierungsdaten auf Basis symmetrischer Schlüssel und kann durch zusätzliche Maßnahmen auf asymmetrische Verfahren erweitert werden [Dow03, ZT06]. Die Instanz des Vermittlers verteilt sich auf zwei Dienste, den Authentisierungsdienst (engl. authentication service (AS)) und den Ticket-Erstellungsdienst (engl. ticket granting service (TGS)).

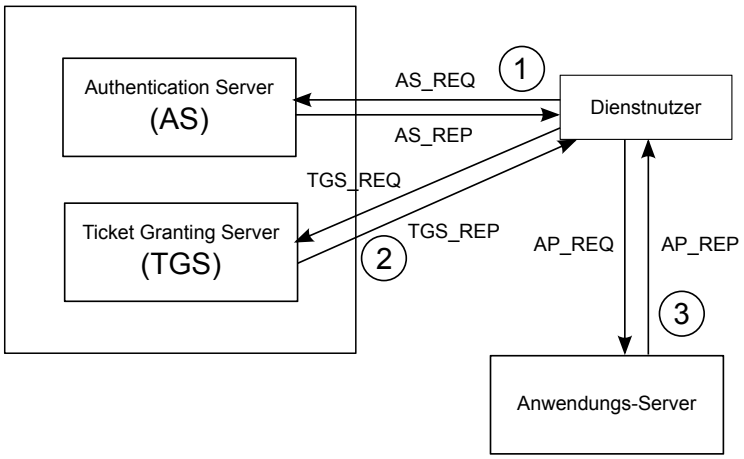


Abbildung 2.7: 3-Phasen-Architektur von Kerberos

Phase 1 Der Anwender leitet mit AS_REQ eine Authentisierung ein, die mit einer AS_REP-Nachricht beantwortet wird. Die Antwort enthält ein sog. Ticket Granting Ticket (TGT).

Phase 2 Mittels des TGT kann der Anwender vom TGS weitere Tickets zur Nutzung mit den Anwendungsservern anfordern (Nachrichten TGS_REQ und TGS_REP).

Phase 3 Das erhaltene und für den Zielserver verschlüsselte Ticket übermittelt der Anwender an den gewünschten Dienst und erhält damit Zugriff auf diesen.

Die Tickets die in Phase 2 ausgestellt und in 3 angewendet werden, sind Zusicherungen, mit denen der TGS dem zu nutzenden Dienst bestätigt, dass der Nutzer, für den dieses Ticket erstellt wird, über die notwendigen Berechtigungen verfügt. Es enthält einen Sitzungsschlüssel, der für die Sitzung zwischen Nutzer und Dienst verwendet wird. Das Ticket wird auch als Zertifikat bezeichnet [NT94] und enthält neben den Kennungen des Anwenders und des gewünschten Dienstes auch Angaben von welchen IP-Adressen aus das Ticket genutzt werden darf, Beginn der Gültigkeit und Lebensdauer sowie einen Sitzungsschlüssel. Das Ticket gilt als ausreichende Autorisierung zur Dienstnutzung. Im Feld *authorization-data* können zusätzlich Einschränkungen aufgenommen werden, die die Autorisierung weiter einschränken.

Die Bestätigung über die Client-Autorisierung, die der Server erhält, ist lediglich implizit und beruht auf der Verteilung des Sitzungsschlüssels. Eine Interpretation des Ticket-Inhaltes im Sinne einer Aussage wird nicht vorgenommen.

2.6.4 SAML

Die Security Assertion Markup Language (SAML) ist ein XML-Rahmenwerk, das es ermöglicht, Aussagen für die Authentisierung und Autorisierung von Kommunikationspartnern auszudrücken. Der SAML-Standard [CKPE05] wurde vom Security Services Technical Committee der Organization for the Advancement of Structured Information Standards (OASIS) herausgegeben, die von einer großen Zahl von Industrieunternehmen unterstützt wird.

Ziel des Standards ist die Unterstützung von Web-basierten Single-Sign-On-Lösungen. Dazu werden geeignete Kodierungsformate und Protokolle definiert, mit denen verschiedene Arten von Zusicherungen dargestellt und kommuniziert werden können. Dabei werden unterschieden

- *Authentication Statements*, die eine Aussage über die Art und den Ort der Authentisierung eines Subjektes bereitstellen
- *Attribute Statements*, die erweiterte Informationen zu einem Subjekt angeben
- *Authorization Decision Statements* zur Zuordnung von Rechten zu einem Subjekt

Es besteht darüber hinaus im Rahmen von SAML die Möglichkeit, zusätzliche Erweiterungen zu definieren, mit denen andere Informationen ausgedrückt werden.

Der Standard definiert ein Challenge-Response-Protokoll, mit dem Anfragen in SAML übertragen werden können. Über sog. *Bindings* wird die Abbildung der Kommunikationsabläufe auf Kommunikationsprotokolle wie HTTP oder SOAP festgelegt.

Zusicherungen (assertion-Tag) in SAML müssen eine Reihe von Informationen wie Versionsnummer, Ausstellername und Ausstellungszeitpunkt enthalten. Als weitere Informationen können zusätzliche Bedingungen und Hinweise in der Zusicherung enthalten sein. Die Bedingungen (`Condition`) müssen bei der Auswertung berücksichtigt werden und können Einschränkungen zeitlicher Art oder der möglichen Empfänger enthalten. Weitere Hinweise z. B. auf Belege für die Zusicherungen oder auf Verteilpunkte von erneuerten Zusicherungen können in einem Advice-Element angegeben werden, dessen Auswertung optional ist.

Die Art der eigentlichen Zusicherung kann durch Auswahl eines oder mehrerer Elemente aus den Erweiterungen bzw. bereits standardisierten Zusicherungen ausgedrückt werden. Die vordefinierten Typen sind die bereits oben aufgeführten Aussagen zu Authentisierung, Attributen und Autorisierungsentscheidungen. Als Erweiterungen stehen Aussage (`Statement`) und subjektbezogene Aussage (`SubjectStatement`) zur Verfügung.

Die drei vordefinierten Zusicherungen des SAML-Standards sind vom `SubjectStatement`-Typ abgeleitet und sind daher Subjekt-zentrisch.

Mit einer *Authentisierungszusicherung* wird durch die ausstellende Instanz bestätigt, dass das referenzierte Subjekt zum angegebenen Zeitpunkt mit der genannten Methode authentifiziert wurde. Es ist möglich, die IP-Adresse und den Domain-Namen sowie eine Instanz anzugeben, die weitere Informationen über das Subjekt bereitstellen kann.

Bei der *Zusicherung eines Attributes* werden Attribute in Form einer Zeichenkette benannt, die dem Subjekt zugeordnet werden können. Zusätzlich werden ein Namensraum und ein darin enthaltener Attributsname angegeben und damit implizit die Semantik des Attributs festgelegt.

Die *Zusicherung einer Autorisierungsentscheidung* gibt an, ob einem Subjekt der Zugriff auf eine bestimmte Ressource gewährt wird. Die Ressource wird durch einen URI und die gewünschte Aktion durch eine Zeichenkette und einen relevanten Namensraum spezifiziert. Es können darüber hinaus in einem `Evidence`-Feld die der Entscheidung zugrunde liegenden Zusicherungen angegeben oder referenziert werden.

2.6.5 XACML

Die ebenfalls vom OASIS-Konsortium entworfene eXtensible Access Control Markup Language (XACML) in der Version 2.0 definiert eine auf XML basierende Syntax zur Beschreibung von Zugriffsrichtlinien (access control policies) sowie ein Schema für die Abfrage von Zugriffsrichtlinien [Mos05].

Neben vordefinierten Datentypen und Vergleichsoperationen gibt es auch die Möglichkeit, die Sprache durch Erweiterungen zu ergänzen. XACML definiert mit Hilfe von XML Baumstrukturen, bei denen die Bäume für Zugriffsziele einer Richtlinie stehen und die Blätter Regeln für Zugriffsentscheidungen aus der Menge „Deny“, „Permit“, „NotApplicable“, „Indeterminate“ repräsentieren. Richtlinien und Mengen von Richtlinien werden grundsätzlich für ein Zugriffsziel definiert.

In der aktuell in Bearbeitung befindlichen Version 3.0 [PL10] werden zusätzlich noch Delegationsmechanismen eingeführt. Hierbei werden die Rechte zur Delegation streng von den zu delegierenden Zugriffsrechten getrennt und in einer eigenen Policy ausgedrückt.

Die Ausdrucksmöglichkeiten von XACML sind wesentlich umfassender als die von SAML. Die zentralen Zusicherungen in XACML sind die Zugriffsrichtlinien, mit denen ein Policy Decision Point (PDP) parametrisiert werden kann, um den Zugriff auf eine Ressource zu steuern. In den Policies können verschiedene Angaben zum Anfragenden, zum eingesetzten Authentifizierungsverfahren, zu verschiedenen Ereignis-Zeiten usw. mit anderen Daten als Parameter für Funktionen dienen. Beispiele für solche Funktionen sind *string-equal*, *integer-add*, *double-divide*, *date-add-year MonthDuration* und *string-is-in*.

Dadurch lassen sich innerhalb der Policies Algorithmen zur Bestimmung der Zugriffsentscheidung definieren.

2.6.6 PERMIS

PrivilEge and Role Management Infrastructure Standards validation (PERMIS) ist ein System zur Generierung, Verwaltung und Prüfung von Benutzerberechtigungen auf Basis des Modells der rollenbasierten Zugriffssteuerung (Role Based Access Control, RBAC), wie sie von Ferraiolo und Kuhn beschrieben wurde [FK92]. Die zugrunde liegende Architektur wird in [COB03,

CO04] beschrieben und setzt auf der Privilege Management Infrastructure (PMI) des X.509-Standards [IT05] auf. Die dort eingeführten Mechanismen werden durch Permis mit einer XML-basierten Policy-Sprache erweitert, die zur Definition und Vergabe von Berechtigungen eingesetzt wird.

Die XML-Strukturen ermöglichen die Definitionen von Rollen, denen wiederum erlaubte Aktionen in Bezug auf Zielobjekte zugeordnet werden können.

2.6.7 SecPAL

Security Policy Assertion Language (SecPAL) ist eine auf einfachen Konstrukten basierende Beschreibungssprache für Sicherheitsrichtlinien und Autorisierungsabfragen, die nahe an einer natürlichen Sprachdarstellung liegt [BFG07, BFG10]. Die Sicherheitsrichtlinien können mit Variablen entworfen werden. Im Rahmen von Autorisierungsbestätigungen werden diese Variablen entsprechend vorliegender Autorisierungen mit konkreten Werten belegt. Beim Entwurf von SecPAL wurde Wert auf eine verständliche Darstellung gelegt, so dass es neben der in XML kodierten Form auch noch eine Darstellung nahe an natürlicher Sprache gibt. Ein Beispiel einer Autorisierung in Anlehnung an [BFG07] lautet

FileServer says **STS** can say$_\infty$ x can execute y from $t1$ till $t2$ if $t2 - t1 \leq 8$ hours.

Dieser Ausdruck besagt, dass die Instanz FileServer der Instanz STS das weiter delegierbare Recht vergibt, beliebigen weiteren Instanzen (dargestellt durch x) das Recht zur Ausführung der Ressource y für Zeitintervalle kleiner als 8 Stunden einzuräumen.

Zusicherungen in SecPal können mit einer zeitlichen Beschränkung ihrer Gültigkeit versehen werden und es können Widerrufe von Zusicherungen formuliert werden.

Es besteht die Möglichkeit, Rechte zu delegieren. Dabei können die Prinzipien der diskreten Zugriffssteuerung (Discretionary Access Control, DAC), der erzwungenen Zugriffssteuerung (Mandatory Access Control, MAC), von hierarchischen Rollenmodellen u. a. berücksichtigt werden. Wie auch die Zusicherungen können Delegationen mit zeitlichen und weiteren Beschränkungen versehen werden.

Die Ausdrucksstärke von SecPAL ist auf den Anwendungsfall der (delegierten) Zugriffsrechte beschränkt, kann jedoch bezüglich der zugesicherten Rechte z. B. durch Definition weiterer parametrisierter Ausdrücke erweitert werden.

2.6.8 Ponder

Ponder [DDLS00] ist eine deklarative Sprache zur Formulierung von Sicherheitsrichtlinien in verteilten Systemen und ermöglicht neben der Definition von Zugriffsregeln und Autorisierungen auch die Beschreibung von Verpflichtungen (z. B. die Forderung der Anwendung einer bestimmten Dienstfunktion auf Zielobjekte) und Verbote (i. S. von negativen Autorisierungen). Es ist in

Ponder möglich, Regeln zu delegieren und durch zusätzliche Beschränkungen wie beispielsweise Zeiten einzuschränken. Desweiteren bietet Ponder die Möglichkeit, Regeln in Gruppen zusammenzufassen sowie die Verträglichkeit von Regeln zueinander zu definieren und bestimmte Kombinationen auszuschließen.

Die Sprache ist über EBNF-Syntax definiert und verwendet teilweise Elemente der Object Constraint Language version 3 (OCL) syntax der Open Management Group. Die Sprachkonstrukte sind objektorientiert.

In Ponder können Aktionen, die als Funktionsbezeichner ggf. mit Parametern dargestellt werden, delegiert werden.

2.6.9 PolicyMaker

PolicyMaker [BFL96] ist ein Rahmenwerk für Kryptografie-basierte Dienste zum Schutz von Vertraulichkeit und Authentizität mit expliziter Beschreibung von Vertrauensaspekten auf Basis einer einfachen Programmiersprache, die vertrauenswürdige Aktionen, Autorisierungen und Berechtigungsnachweise als Prädikate spezifiziert. Der Kern von PolicyMaker ist eine Prüffunktion, der eine Anfrage zur Ausführung einer Aktion und dazu passende Autorisierungen und Zertifikate übergeben werden. Die Prüffunktion entscheidet, ob die Anfrage angenommen werden kann, d. h. die Aktion ausgeführt werden darf, oder ob sie verworfen werden muss. Ergänzend können zusätzliche Einschränkungen zurückgegeben werden, mit denen eine abzulehnende Anfrage angenommen werden könnte.

Eine Besonderheit von PolicyMaker ist, dass die Aktionen als beliebige anwendungsspezifische Zeichenketten formuliert werden können. Berechtigungen werden in Form von Prüfungen durch reguläre Ausdrücke oder Programmcode für einen speziellen AWK-Interpreter spezifiziert, die bei der Prüfung der Anfragen auf die Zeichenketten im Sinne eines Filters angewendet werden.

Auf diese Weise ermöglicht PolicyMaker das Vertrauensmanagement für beliebige Anwendungsbereiche nutzbar zu machen.

2.6.10 Keynote

Keynote ist ein Trustmanagement-System und baut auf dem PolicyMaker-Projekt auf.

KeyNote legt in [BFIK99] eine spezielle Sprache für Berechtigungen und Anfragen fest. Diese ist vergleichbar mit Label-Wert-Paaren eines E-Mail-Headers. Die Autoren verfolgen dabei das erklärte Ziel, die Zusicherungen auch für den menschlichen Benutzer in lesbarer Form zu kodieren.

Die Autoren definieren als Elemente eines Trustmanagement-Systems fünf Punkte. Diese umfassen

- eine Sprache zur Beschreibung von Aktionen, die für die Sicherheit eines Systems relevant sind (z. B. Lesen einer Datei)

- Möglichkeiten, um Akteure innerhalb des Systems identifizieren zu können
- eine Sprache zur Formulierung von Anwendungsrichtlinien zur Verwaltung der im System möglichen Aktionen, zu denen Akteure ermächtigt werden können
- eine Sprache zur Spezifikation von Berechtigungsnachweisen, die es Akteuren ermöglicht, an andere Akteure Berechtigungen zu delegieren
- eine Prüfinstanz, die die Erfüllung von Richtlinien auf Basis einer Menge von Berechtigungsnachweisen verifiziert

Der RFC spezifiziert sprachliche und algorithmische Mittel, um diese fünf Elemente bereitzustellen. Aktionen werden dabei in Form einfacher Prädikate ausgedrückt und als Bezeichner-Werte-Paare kodiert, die frei durch den Applikationskontext bestimmt werden können. Als Akteure können Personen, Systeme und – wie bei SPKI – auch kryptographische Schlüssel verwendet werden.

Prüfausdrücke können mit Hilfe der Feldnamen, Vergleichsoperationen, boolschen Verknüpfungen und regulären Ausdrücken erstellt werden und einen Wert der möglichen Freigabestufen (z. B. „approve“, „reject“ oder „Kein Zugriff“, „Lesezugriff“, „Vollzugriff“) enthalten.

Für die Validierung einer Aktion kann eine Applikation eine Anfrage an die Prüfinstanz (Compliance Checker) stellen, der sie die Identität des Akteurs, die gewünschten Aktionen und einen Satz von Berechtigungen schickt. Der Compliance Checker wertet die übergebene Anfrage mit Hilfe der in den Berechtigungen enthalten Prüfausdrücke aus und gibt die Freigabestufe zurück.

Die Semantik der Felder einer Anfrage überlässt Keynote vollständig der Applikation und betrachtet lediglich die Auswertung der Bedingungen.

2.7 Bewertung der bestehenden Mechanismen

Die in diesem Kapitel vorgestellten Mechanismen ermöglichen es, unterschiedliche Arten von Nachweisen der Authentizität bzw. von Zusicherungen zu geben. Um jedoch die in Kapitel 1 beschriebenen Aussagen darstellen und dabei die in Abschnitt 1.3 vorgestellten Ziele erfüllen zu können sind sie jeweils nur teilweise geeignet.

Die *Flexibilität* bei der Gestaltung von Zusicherungen ist naturgemäß bei reinen Signaturverfahren gegeben, da diese keine Festlegung der jeweils signierten Inhalte vornehmen. Die Trust-Management-Verfahren haben überwiegend einen begrenzten Anwendungsbereich.

Bei den Zertifikaten nach X.509, SPKI und PGP ist der Informationsgehalt eng auf die Anforderungen der sicheren Schlüsselverteilung zugeschnitten.

Bei den Verfahren Kerberos, SAML, XACML, PERMIS, Ponder und SecPAL steht die Erteilung von Autorisierungen von Dienstnutzungsberechtigungen im Mittelpunkt. Die sprachlichen Mittel sind daher darauf spezialisiert. PolicyMaker nimmt eine Sonderstellung ein, da hier keine

Vorgaben zum Inhalt gemacht werden. Es ist somit prinzipiell geeignet, beliebige Zusicherungen zu unterstützen. Das darauf aufsetzende KeyNote hat wiederum einen Anwendungskontext für Autorisierungen.

Zur Erhöhung der *Verbindlichkeit* tragen alle Verfahren bei, da sie über Signaturen abgesichert sind. Diese Signaturen sind teilweise fester Bestandteil oder sogar Gegenstand des Verfahrens (X.509, PGP, S/MIME, ...). Bei anderen müssen die Signaturen gesondert betrachtet werden (Ponder, PolicyMaker, Keynote, ...). Bei allen Verfahren lassen sich durch den Einsatz von Signaturen die Integrität und Authentizität der gegebenen Zusicherungen oder Aussagen belegen.

Die gewünschte *Verständlichkeit* ist bei den vorgestellten Zusicherungen nur durch zusätzliche Maßnahmen wie spezielle Anzeigeprogramme erreichbar. Wie bei X.509 aufgezeigt, wird eine für den Anwender verständliche Bedeutung durch Definitionen in den Standards festgelegt und nicht fest, d. h. per Kodierung, an die Zertifikate gekoppelt.

Bei Verfahren wie SPKI/SDSI ist durch die Verwendung sprechender Labels in den S-Expressions die Bedeutung in lesbarer Form vorhanden. Jedoch muss der Zusammenhang zwischen den einzelnen Elementen ebenfalls mit Kenntnis der Standards abgeleitet werden.

Eine etwas bessere Form bietet hierbei SecPAL, das für seinen engen Anwendungsbereich eine Sprache verwendet, deren Darstellung Sätzen der natürlichen Sprache sehr nahe kommt.

Signaturverfahren, die lediglich ein Dokument signieren (PDF, S/MIME) können eine sehr hohe Verständlichkeit der zugesicherten Daten erreichen, da in diesem Fall direkt mit einer natürlichsprachlichen Form gearbeitet werden kann.

Hinsichtlich der *Automatischen Auswertung* sind alle Verfahren zur Zertifikats- und Zusicherungsverarbeitung sehr gut geeignet, da diese Eigenschaft ein Entwurfsziel solcher Zertifizierungssysteme ist. Bei den reinen Signaturen z. B. über natürlichsprachliche Texte gibt es keine vorgegebene Möglichkeit zur automatisierten Verarbeitung der signierten Daten.

PDF-Dokumente nehmen bei der Automatisierbarkeit eine Mittelstellung ein, da es hier möglich ist, Formulare zu definieren, die sowohl signiert als auch automatisch verarbeitet werden können. Diese können dabei neben den Formularfeldern natürlichsprachlichen erläuternden Text enthalten.

In Abbildung 2.8 wird ein Einordnung der Verfahren hinsichtlich Automatisierbarkeit und Flexibilität graphisch veranschaulicht.

Die Bereitstellung von *Delegationsmechanismen* ist ebenfalls unterschiedlich stark ausgeprägt. SPKI/SDSI unterstützt die Weitergabe von Berechtigungen durch die Berechtigten. Kerberos unterstützt Delegation eingeschränkt durch Weitergabe von Berechtigungen eines Anwenders an einen Server. Ponder ermöglicht komplexe, in der Länge beschränkbare Delegationsketten mit diversen Einschränkungsmöglichkeiten. PERMIS unterstützt ebenfalls mehrstufige Delegation einschließlich Delegationsrichtlinien [CZO+08]. SAML kann ebenfalls um Mechanismen der Delegation erweitert werden [WVH05]. PolicyMaker definiert selbst keine Delegationsmechanismen. Erst im darauf aufsetzenden KeyNote wird ein entsprechender Mechanismus zur Weitergabe von Autorisierungen unterstützt. Die Zertifizierungsmechanismen unterstützen nur eine

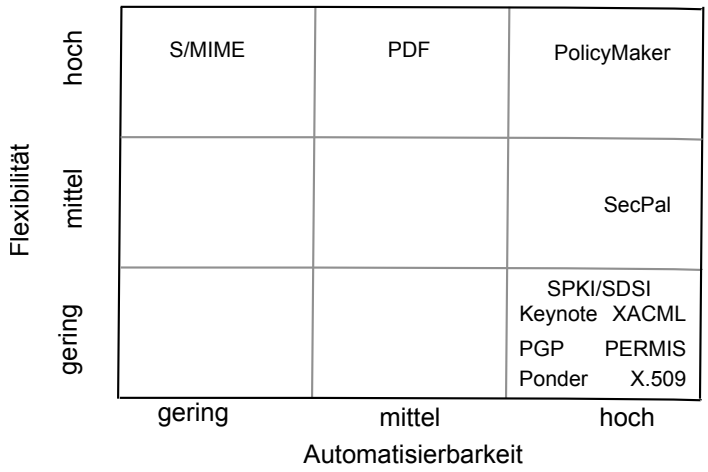


Abbildung 2.8: Qualitativer Vergleich der Zielerfüllung für Automatisierbarkeit und Flexibilität

spezielle Art der Delegation. Bei X.509 erfolgt Delegation implizit durch Erstellen von Ausstellerzertifikaten, die zur Erstellung von Zertifikaten auf der nächsten hierarchischen Ebene berechtigen. Bei PGP kann durch Signaturen Dritter das mögliche Vertrauen in einen Schlüsselinhaber angehoben und dieser dadurch in eine verbesserte Bestätigerrolle gebracht werden.

Zusammenfassend muss festgestellt werden, dass keines der untersuchten Verfahren alle Anforderungen erfüllen kann. Dieser Umstand wird sowohl durch die hohe Spezialisierung der Funktionalität wie z. B. bei X.509 als auch durch die allgemeine Verwendbarkeit wie bei den Signaturverfahren wie S/MIME bedingt.

Es wird deshalb mit dem Ansatz der grammatikbasierten Zertifikate in Kapitel 4 eine Möglichkeit geschaffen, allen Anforderungen in ausgewogenerer Weise gerecht zu werden.

3 Methoden zur Sicherheitsanalyse

In diesem Kapitel werden zwei Verfahren zur formalen Verifikation von Sicherheitseigenschaften in Protokollen bzw. kommunizierenden Systemen vorgestellt, die in den weiteren Teilen der Arbeit zur Anwendung kommen. Der Fokus liegt dabei auf den Eigenschaften Vertraulichkeit und Authentizität sowie der Bewertung der jeweils notwendigen Voraussetzungen.

Das erste Verfahren ist die sog. SVO-Logik, eine Belief-Logik, mit der Überzeugungen und Beobachtungen bei Anwendung von Kommunikationsprotokollen modelliert werden können. Anhand des Modells und einer Menge von Ableitungsregeln können aus Sicht der Kommunikationspartner die jeweils gültigen Schlussfolgerungen im Hinblick auf Sicherheitseigenschaften abgeleitet werden.

Das zweite Verfahren ist der angewandte Pi-Kalkül und *Proverif*, eine Implementierung dieses Kalküls zur Durchführung von Modellchecking-Analysen der Sicherheitseigenschaften in kommunizierenden verteilten Systemen. Die Modellierung umfasst dabei sowohl die ausgetauschten Nachrichten als auch die (sicherheitsrelevanten) Verarbeitungsschritte innerhalb der beteiligten Instanzen.

3.1 SVO-Logik

Um eine Bewertung von Kommunikationsprotokollen im Hinblick auf die jeweils für die Kommunikationspartner nachweislich erreichbaren Überzeugungen zu ermöglichen, wird im Folgenden mit der SVO-Logik eine sog. Belief-Logik vorgestellt.

Im Jahr 1996 entwickelten Syverson und van Oorschot die nach ihnen benannte SVO-Logik [SO96]. Diese ist eine Weiterentwicklung der durch die BAN-Logik begründeten Reihe von Logiken und vereint neben den Fähigkeiten der von Burrows, Abadi und Needham entwickelten BAN-Logik [BAN89] auch noch die Aussagefähigkeiten der GNY-Logik [GNY90] von Gong, Needham und Yahalom sowie der ebenfalls auf BAN aufsetzenden Logiken AT [AT91] von Abadi und Tuttle und VO [vO93] von van Oorschot.

Wesentlicher Beitrag der SVO-Logik ist die Definition einer formalen Semantik der Logik. Im Folgenden wird die später in [SC01] eingeführte Schreibweise verwendet.

Bei einer Belief-Logik handelt es sich um eine Sammlung von formalisierten Ableitungsregeln, die den Austausch von Information z. B. in Kommunikationsprotokollen aus der beobachtenden

Sicht eines an der Kommunikation Beteiligten beschreiben. Beteiligte können Personen oder Prozesse sein, die eigenständig handeln (im Weiteren auch als Prinzipale bezeichnet). Aus Informationen, die dieser sieht, sowie seinem vorhandenen Vorwissen, kann er durch Anwendung der Ableitungsregeln Schlussfolgerungen ziehen.

3.1.1 Vorgehensweise bei SVO-Analysen

Eine Analyse eines Protokolls auf Basis einer Modellierung in SVO-Logik wird in mehreren Schritten durchgeführt.

Zieldefinition Es wird festgelegt, welche Überzeugungen bei einem oder mehreren Kommunikationspartnern erzielt werden müssen. Ein Beispiel ist die Etablierung eines geteilten Geheimnisses zwischen den Beteiligten A und B. Eines der entsprechenden Ziele lautet dann „A glaubt, dass der Schlüssel k ein geteiltes Geheimnis zwischen A und B ist“. In SVO-Schreibweise ist dies „A believes $A \overset{k}{\longleftrightarrow} B$“.

Prämissen – Initiales Wissen Die für die Untersuchung notwendigen Prämissen werden aufgestellt. In diesen Prämissen werden alle Annahmen bzgl. eines initialen Kenntnisstandes der Beteiligten festgehalten. Hierzu gehören u. a. welche Schlüssel sie besitzen und wem diese ggf. zugeordnet werden, welche Zufallswerte als „frisch“ in Bezug auf den aktuellen Protokolllauf angesehen werden, usw.

Prämissen – Protokollbeschreibung Die empfangenen Nachrichten werden aus der subjektiven Sicht des oder der Beteiligten modelliert. Hierbei wird neben dem Verständnis der Nachrichten (welche Daten sind bekannt, welche sind neu) auch die Interpretation der Nachrichten durch den Empfänger beschrieben (z. B. die Übermittlung eines bestimmten Wertes k bedeutet im Rahmen des Protokolls, dass dieser Wert k als Sitzungsschlüssel zwischen den Beteiligten verwendet werden soll). Die dabei entstehenden Ausdrücke erweitern die im vorhergehenden Schritt aufgestellten Prämissen.

Ableitungen Durch Anwendung der Ableitungsregeln des Kalküls wird versucht, die Formeln abzuleiten, die durch die festgelegten Ziele vorgegeben sind. Hierbei kommen die axiomatischen Ableitungsregeln des Kalküls und die aufgestellten Prämissen zur Anwendung.

Bewertung Abschließend wird bewertet, ob einerseits das bzw. die Ziele erreicht werden konnten und welche Prämissen hierzu nötig waren. Werden unterschiedliche Protokolle miteinander verglichen, so können bei gleichem Ziel die jeweils anzunehmenden Prämissen zur Bewertung herangezogen werden.

3.1.2 Notation der SVO-Logik

Die hier vorgestellte Schreibweise der Ausdrücke der SVO-Logik basiert auf den Darstellungen in [SO96] und [SC01].

Formel[1]	Semantik
$P \overset{k}{\longleftrightarrow} Q$	k ist ein zwischen P und Q geteiltes Geheimnis
$\mathrm{PK}_\sigma(P, k)$	k ist ein öffentlicher Signaturschlüssel von P
$\mathrm{PK}_\varphi(P, k)$	k ist ein öffentlicher Verschlüsselungsschlüssel von P
$\mathrm{PK}_\delta(P, k)$	k ist ein öffentlicher Schlüssel von P zur Schlüsselaushandlung
$\mathrm{SV}(X, k, Y)$	Die Formel ist wahr, wenn mit k geprüft werden kann, dass X aus Y durch Verwendung des zu k gehörenden Signaturschlüssel hervorgeht.
P received X	P hat die Nachricht X empfangen.
P has X	P ist in Besitz der Nachricht X.
P said X	P hat (irgendwann) in der Vergangenheit die (Teil-)Nachricht X gesendet.
P says X	P hat im aktuellen Protokolllauf die (Teil-)Nachricht X gesendet.
fresh(X)	X war in keiner Nachricht enthalten, die vor dem aktuellen Protokolllauf gesendet wurde.
P believes φ	P kann sich so verhalten, als ob φ wahr ist.
P controls φ	P hat die Kompetenz (Jurisdiktion) über φ. Mit anderen Worten: wenn P φ sagt, so gilt φ als wahr.

[1] mit P, Q: Prinzipale, X, Y: Nachrichten, k: Schlüssel und φ: Formel

Tabelle 3.1: Formelausdrücke der SVO-Logik

Grundbausteine der Logik sind primitive Terme, die für Konstanten ($L, M, N, \ldots$), Identitäten ($A, B, P, Q, \ldots$) sowie symmetrische, private und öffentliche Schlüssel ($K, k, k^{-1}, K_A, \ldots$) verwendet werden. Diese primitiven Terme werden in der Menge Θ zusammengefasst.

Ausgehend von Θ definiert SVO zwei verschiedene Sprachen. Zum einen werden Formeln ausgedrückt, denen ein Wahrheitswert (wahr: $\mathbf{T}$, falsch: $\mathbf{F}$) zugeordnet werden kann, zum anderen wird eine Schreibweise für Nachrichten festgelegt.

Die Elemente der Menge aller Nachrichten M sind wie folgt definiert: Alle Elemente aus Θ sind Nachrichten. Jede Funktion, deren Argumente Nachrichten sind, ist eine Nachricht. Hierzu gehört insbesondere auch die Tupel-Bildung der Form $(x_1, \ldots, x_n)$ mit $x_\nu \in \Theta$ sowie die unten angegebenen Ausdrücke für verschlüsselte und signierte Nachrichten. Jede Formel ist eine Nachricht.

Alle Ausdrücke in Tabelle 3.1 sind Formeln und bilden durch Konjunktion ($\wedge$) und Negation ($\neg$) wiederum Formeln.

Für verschlüsselte und signierte Nachrichten wurden folgende Schreibweisen festgelegt:

$\lfloor X \rfloor_K$ beschreibt eine mit dem Schlüssel K ausgeführte Signatur über X. Die Schreibweise steht für das Paar aus Signatur nach einem Schema ohne Rückgewinnung der signierten Nachricht und den Daten selbst. Zur Prüfung der Signatur wird der korrespondierende Schlüssel $\tilde{K}$ zu K benötigt. Ist $K = k^{-1}$, so ist $\tilde{K} = k$. Für Verschlüsselung wird die Schreibweise $\{X\}_K$ verwendet. Zur Entschlüsselung wird wieder $\tilde{K}$ verwendet. Für asymmetrische Verfahren gilt: $K = k$ und $\tilde{K} = k^{-1}$. Im symmetrischen Fall ist $K = \tilde{K} = k$.

Zusätzlich wurden folgende Annotationen definiert: $\langle X \rangle_{*P}$ bedeutet „X in der Wahrnehmung von Prinzipal P“. Diese spezielle Form der Darstellung ist notwendig, da P eine empfangene Nachricht nicht immer vollständig interpretieren kann. Beispielsweise kann der Empfänger einer Nachricht $H(Y)$ diese nur dann als Hashwert über Y erkennen, wenn ihm auch Y vorliegt. Dennoch ist der empfangene Bitstring nicht beliebig, da der Empfänger diesen – auch ohne korrekte Interpretation – wiedererkennen kann. Die Schreibweise X from Q bezeichnet ein Datum X, das von Q stammt.

3.1.3 Schlussregeln der SVO-Logik

Die SVO-Logik besteht in der modifizierten Form aus [SC01] aus 22 als Axiome bezeichneten Schlussregeln. Diese setzen die Prämissen $\varphi_1 \ldots \varphi_n$ mit den Folgerungen $\psi_1 \ldots \psi_m$ durch die Schreibweise

$$\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow (\psi_1 \wedge \ldots \wedge \psi_m) \qquad \varphi_i, \psi_j \in \Phi$$

in Bezug. Dabei steht das Zeichen „$\rightarrow$“ für die logische Implikation.

Zur Durchführung von Ableitungen kommen zwei Schlussregeln zum Einsatz. Die erste Regel ist *Modus Ponens (MP)*:

$$\frac{\varphi, \quad \varphi \rightarrow \psi}{\psi} \tag{MP}$$

Aus den Prämissen φ und $\varphi \rightarrow \psi$ lässt sich die Schlussfolgerung (Conclusio) ψ ziehen.

Die zweite Regel ist die Ableitung notwendiger Prämissen, engl. necessitation (Nec).

$$\frac{\vdash \varphi}{\vdash P \text{ believes } \varphi} \tag{Nec}$$

Jede Aussage φ, die aus den Axiomen abgeleitet werden kann, wird von P als richtig angesehen (geglaubt). Das Symbol „$\vdash$“ steht für die Herleitungsbeziehung.

Im Folgenden werden einige der Axiome anhand eines Beispiels eingeführt. Die vollständige Auflistung aller Axiome findet sich in Anhang A.

3.1.4 Subjektive Sichtweise

Die besondere Eigenschaft der Belief-Logik ist die Modellierung der *subjektiven* Sicht eines oder mehrerer an der Kommunikation Beteiligter. Es ist also bei der Betrachtung nicht von Bedeutung, wer an wen welche Informationen sendet, sondern welche Nachrichten der jeweils Betroffene beobachtet bzw. empfängt und welche Schlüsse er – ggf. unter Einbeziehung von bereits vorhandenem Wissen – daraus ziehen kann.

Als Einführung wird zunächst ein einfaches Beispiel betrachtet. Ein Anwender A eines Dienstes B kann an den Dienst einfache Kommandos C_x schicken, die vom Dienst ausgeführt werden.

Wird in einem ungeschützten Protokoll die Identität A des Nutzers, das Kommando C_1 und zur Legitimation ein Passwort PW an B übertragen, so führt dies nur dann zu einem korrekten Verhalten des Servers, wenn die Nachricht wirklich von A geschickt wurde. Die übliche Notation solcher Übertragungen, die Sender und Empfänger in der Form „A sendet an B“ darstellen, führt häufig zu einer Vernachlässigung der Betrachtung von Bedrohungen, da die Darstellung bereits A als Sender impliziert (die *sendet an*-Beziehung wird durch das Zeichen $\mapsto$ dargestellt):

$$A \mapsto B : A, PW, C_1$$

Die zugehörige subjektive Darstellung ist

$$B \text{ received } (A, PW, C_1)$$

Es liegt also in der Schreibweise keine Information über die (tatsächliche) Quelle vor.

Dieser Perspektivenwechsel ist ein wesentlicher Beitrag, den Belief-Logiken leisten, um eine fehlerhafte Interpretation zu vermeiden. Identische Nachrichten führen in der SVO-Schreibweise auch auf eine identische Wahrnehmung beim Empfänger.

$$\begin{aligned} A \mapsto B : A, PW, C_1 \quad &\rightsquigarrow \quad B \text{ received } A, PW, C_1 \\ E \mapsto B : A, PW, C_1 \quad &\rightsquigarrow \quad B \text{ received } A, PW, C_1 \end{aligned}$$

Hiermit ist für den Empfänger nicht unterscheidbar, ob die Nachrichten von einem legitimierten Anwender A (Alice) oder von einem Angreifer E (Evil) stammen.

Die SVO-Logik bietet eine Reihe von Regeln an, die es ermöglichen, aus vorhandenen Daten, Überzeugungen und Beobachtungen korrekte Schlussfolgerungen zu Protokollabläufen zu ziehen. Diese Regeln sind in Anhang A aufgeführt.

Entsprechend der oben beschriebenen Vorgehensweise werden die Prämissen aufgestellt. Neben der Modellierung des initialen Wissens (Zuordnung von öffentlichen Schlüsseln, Frische von Zufallswerten, Auswertungen von Signaturen, ...) werden bei der Modellierung der Nachrichten des Protokolls folgende Aspekte berücksichtigt:

- ❑ Empfangene Nachrichten gemäß der Protokollbeschreibung. Diese Darstellung dient zur Dokumentation und als Ausgangspunkt zur Festlegung der weiteren protokollbezogenen Prämissen.
- ❑ Verständnis der empfangenen Nachrichten zur Unterscheidung der bekannten und der neuen Elemente. Neue Elemente X aus Sicht von A werden als $\langle X \rangle_{*A}$ geschrieben.
- ❑ Interpretation der Nachrichten durch den Empfänger. Hierbei können u. a. Symbole durch Formeln ersetzt werden (z. B. kann die durch den Schlüssel von B signierte Übertragung der Kennung von B und einer für A neuen Bitfolge $\langle K_{AB} \rangle_{*A}$ als Aussage interpretiert werden, dass diese als neuer symmetrischer Schlüssel zwischen A und B fungieren soll: es wird daher $\lfloor B, \langle K_{AB} \rangle_{*A} \rfloor_{k_B}$ durch $\lfloor A \overset{\langle K_{AB} \rangle_{*A}}{\longleftrightarrow} B \rfloor_{k_B}$) ersetzt.

Das oben eingeführte Beispiel wird nun um die Signatur des Befehls durch den Anwender erweitert. Hierbei ist (K_A, K_A^{-1}) das Schlüsselpaar von A. Damit sich der Server auch von der Aktualität der Anfrage überzeugen kann, werden zwei zusätzliche Protokollschritte eingeführt. A sendet zunächst eine Initialnachricht mit seiner Kennung an den Server und erhält als Antwort einen durch den Server bestimmten frischen Zufallswert N.

Damit sieht das erweiterte Protokoll wie folgt aus:

$$A \mapsto B : A$$
$$B \mapsto A : N$$
$$A \mapsto B : \lfloor C_1, N \rfloor_{K_A^{-1}}$$

Das zu zeigende Ziel ist die Überzeugung von Server B, dass die empfangene Nachricht aktuell von A gesendet wurde: B believes A says C_1 .

Die Prämissen, die das initiale Wissen von B beschreiben, sind:

$$B \text{ believes } \mathrm{PK}_\sigma(A, K_A) \tag{P1}$$
$$B \text{ believes } \mathrm{fresh}(N) \tag{P2}$$
$$B \text{ believes } \mathrm{SV}(\lfloor C_1, N \rfloor_{K_A^{-1}}, K_A, (C_1, N)) \tag{P3}$$

Die beiden ersten Nachrichten des Protokolls werden nicht modelliert, da sie ungeschützt übertragen werden und beliebig manipulierbar sind und somit nicht zur Ableitung beitragen können. Die empfangene signierte Nachricht wird geschrieben als:

$$B \text{ received } \lfloor C_1, N \rfloor_{K_A^{-1}} \tag{P4}$$

Da die Nachricht keine für B neuen Terme enthält, ist das Termverständnis identisch zur empfangenen Nachricht.

$$B \text{ believes } B \text{ received } \lfloor C_1, N \rfloor_{K_A^{-1}} \tag{P5}$$

Diese Prämissen können nun für Ableitungen (als C1, C2, ... nummeriert) eingesetzt werden. Mit Axiom (Ax6)

$$(\mathrm{PK}_\sigma(Q, K) \wedge R \text{ received } X \wedge \mathrm{SV}(X, K, Y)) \rightarrow Q \text{ said } Y$$

sowie (P1), (P3) und (P4) kann B ableiten, dass die Information authentisch ist.

$$B \text{ believes } A \text{ said } (C_1, N) \tag{C1}$$

Mit (Ax19) ist aus der Frische von N (P2) die Frische von (C_1, N) ableitbar.

$$B \text{ believes } \mathrm{fresh}(C_1, N) \tag{C2}$$

Schließlich kann mit (Ax21) die Aktualität der Nachricht gezeigt werden:

$$B \text{ believes } A \text{ says } (C_1, N) \tag{C3}$$

Durch (Ax17) kann schließlich das Ziel erreicht werden

$$B \text{ believes } A \text{ says } C_1 \quad \square \tag{C4}$$

3.1.5 Erweiterung der SVO-Logik

Ob eine Nachricht bzw. eine Aussage in einem aktuellen Protokolllauf gültig ist, kann in der SVO-Logik lediglich über den Frische-Begriff abgeleitet werden. Bei Zertifikaten wird jedoch kein frischer Nonce-Wert verwendet, sondern eine Statusauskunft z. B. von einem OCSP-Dienst eingeholt. Die Aussage des Dienstes besagt, dass ein zuvor ausgestelltes Zertifikat als gültig anzusehen ist, oder anders ausgedrückt, dass der Ersteller der Aussage diese nach wie vor aufrecht erhält und sie deshalb so betrachtet werden kann, als wäre sie im aktuellen Protokolllauf geäußert worden. Dieser Mechanismus soll durch ein neues Prädikat *current* im Rahmen der SVO-Logik beschrieben werden.

Zur genauen Definition des Prädikats wird das in [SO96] eingeführte Zustandsmodell verwendet: Die Beteiligten eines Protokolllaufs sind $P_1, \ldots, P_n$ sowie ein Angreifer P_e. Zu jedem P_i wird ein lokaler Zustand s_i definiert, der durch eine Menge von Überzeugungen der Form P_i believes ... dargestellt werden kann. Durch Ereignisse wie die Generierung neuer Werte und den Empfang von Nachrichten ändert sich jeweils zu einem diskreten Zeitpunkt ein lokaler Zustand. Der Initialzustand der aktuellen Authentisierung (hier auch der aktuellen Auditanfrage) liegt bei $t = 0$. Die lokalen Zustände s_i zum Zeitpunkt t können zu einem globalen Zustand $r(t)$ zusammengefasst werden, der auch als (r, t) geschrieben wird.

Die Schreibweise $(r, t) \vDash \varphi$ bedeutet, dass φ im globalen Zustand r und damit auch in allen lokalen Zuständen zum Zeitpunkt t gültig ist (Semantische Implikation, $\vDash$). So sind z. B. die Wahrheitsbedingungen für die Kompetenz in [SO96] wie folgt definiert: „$(r, t) \vDash P$ controls φ“ g.d.w. die Aussage „$(r, t) \vDash P$ says φ“ die Aussage „$(r, t') \vDash \varphi$ für alle $t' \geq 0$“ impliziert.

In diesem Modell kann für das neue Prädikat *current*(Q, φ) formal definiert werden:

$$
\begin{aligned}
&(r, t) \vDash \text{current}(Q, \varphi) \\
&\quad \text{g.d.w. die Aussage } Q \text{ said } \varphi \text{ die Aussage } Q \text{ says } \varphi \text{ impliziert.}
\end{aligned}
\tag{3.1}
$$

Zur Anwendung des Prädikates wird folgende Ableitungsregel ergänzt:

$$
Q \text{ said } \varphi \wedge \text{current}(Q, \varphi) \rightarrow Q \text{ says } \varphi \tag{Ax23}
$$

Die Korrektheit dieser Implikation ist offensichtlich, da sie den Festlegungen der Wahrheitsbedingungen in (3.1) entspricht. Das *current*-Prädikat ähnelt dem *fresh*-Prädikat hinsichtlich der Wirkung auf eine vormals gesagte Nachricht (Ax21), ist insgesamt jedoch deutlich schwächer, da keine Übertragung der Eigenschaft wie bei Frische durch (Ax19) und (Ax20) erfolgen kann.

In Anhang A wird in Abschnitt B.1 ein ausführliches Beispiel zur Anwendung der SVO-Logik bei der Prüfung signierter Daten unter Einbeziehung von PKI-Diensten gezeigt, das dieses neue Prädikat verwendet.

3.2 Angewandter Pi-Kalkül

Die in den vorausgehenden Abschnitten vorgestellte Methode der SVO-Logik beschränkt sich bei der Beschreibung auf die subjektive Wahrnehmung von Nachrichten eines oder mehrerer Beteiligter. Neben dieser Methode gibt es jedoch auch Verfahren, die die Kommunikation zwischen allen Beteiligten untersuchen und neben den ausgetauschten Nachrichten auch das jeweilige interne Verhalten der Kommunikationspartner zwischen dem Austausch von Nachrichten modellieren.

3.2.1 Prozesskalküle

Eine solche Möglichkeit zur Analyse von kryptographisch gesicherten Protokollen ist eine formale Beschreibung mittels eines geeigneten Prozesskalküls, der das relevante Verhalten aller am Protokolllauf Beteiligten einschließlich eines Angreifers modelliert.

In der Literatur finden sich zu Prozesskalkülen verschiedene Ansätze. Hierzu zählen u. a. die Prozessalgebra Communicating Sequential Processes (CSP) von Hoare [Hoa85], die in Form des Model-Checkers Failures-Divergence Refinement (FDR) [Oxf10] z. B. in [Low96] zur Analyse von sicheren Protokollen eingesetzt wird, der von Milner entwickelte Kalkül kommunizierender Systeme (Calculus of Communicating Systems, CCS) [Mil80] und der darauf aufsetzende Pi-Kalkül, der in einer Variante, dem sog. *angewandten Pi-Kalkül*, im Folgenden vorgestellt und eingesetzt wird. Ein weiterer Vertreter für Prozesskalküle ist die Language of Temporal Ordering Specification (LOTOS) [ISO89], eine für die Modellierung von Protokollen und Diensten in der Telekommunikation geschaffene Beschreibungsform, die z. B. zur Analyse parallel arbeitender Kommunikationsdienste eingesetzt werden kann [Kec00].

3.2.2 Der Pi-Kalkül

Der Pi-Kalkül ist ein Prozesskalkül, der 1992 von Robin Milner, Joachim Parrow und David Walker zur Modellierung kommunizierender Prozesse entwickelt wurde [MPW92].

Die Elemente, aus denen sich der Pi-Kalkül zusammensetzt, sind Systemmodelle aus parallel ausgeführten Prozessen und Nachrichten, die auf benannten Kanälen zwischen diesen Prozessen ausgetauscht werden. Dabei wird der Austausch von Nachrichten lediglich durch die in den Prozessen vorkommenden Sende- und Empfangsschritte modelliert. Die Prozesse enthalten Namen und Variablen. Mit den Namen werden sowohl Konstanten wie kryptographische Schlüssel als auch Kanäle zwischen den Prozessen bezeichnet.

Die über benannte Kanäle ausgetauschten Nachrichten bilden die einzige Interaktion zwischen Prozessen. Die Kanalnamen können ihrerseits als Nachrichten verschickt werden. Auf diese Weise können im Verlauf eines Protokolls neue Kommunikationsbeziehungen zwischen Prozessen geschaffen werden, indem diese durch Kenntnisnahme zuvor unbekannter Kanalnamen die dadurch bezeichneten Kanäle zur Kommunikation nutzen können.

3.2.3 Erweiterung zum angewandten Pi-Kalkül

Abadi und Gordon entwickelten 1999 auf Basis des Pi-Kalküls zunächst den Spi-Kalkül [AG99], welcher den Pi-Kalkül um Funktionen zur Beschreibung von verschlüsselter Kommunikation erweitert. In 2001 entwickelten Abadi und Fournet daraus den *angewandten Pi-Kalkül* [AF01], der später noch gemeinsam mit Bruno Blanchet [AB05, BAF08] modifiziert wurde.

Der angewandte Pi-Kalkül erweitert den Ansatz von Milner um Funktionen und Gleichungen, die wiederum zur Beschreibung kryptographischer Funktionen wie Hashwertberechnung, symmetrische und asymmetrische Verschlüsselung und Bildung von Zufallswerten dienen. Schlüssel und Zufallswerte werden dabei durch gebundene Namen repräsentiert.

Die Notation verwendet eine Menge Σ von Funktionen und Konstanten, Terme $(L, M, N, \ldots)$, Namen $(a, b, c, k, n, s, \ldots)$, Variablen $(x, y, z, \ldots)$ und angewendete Funktionen wie z. B. $f(M_1, \ldots, M_l)$.

Namen werden einmalig im Modell festgelegt und repräsentieren im gesamten Protokolllauf eine unveränderliche Größe. Variablen werden im Verlauf des Protokolls durch eine Ersetzung dynamisch mit einem Wert verknüpft.

Für die Beschreibung von Prozessen $(P, Q, R, \ldots)$ können die in Tabelle 3.2 aufgeführten Schreibweisen eingesetzt werden.

Parallel laufende Prozesse	$P \mid Q$
Empfang einer Nachricht	$a(x).P$
Versenden einer Nachricht	$\overline{a}\langle M \rangle.P$
bedingte Ausführung von Prozessen	if $M = N$ then P else Q
Termauswertung	let $x = g(M_1, \ldots, M_n)$ in P else Q
beliebig häufig instantiierte Prozesse	$!P$
Erzeugung eines neuen gebundenen Symbols	$(\nu n).P$
leerer Prozess (Null-Prozess)	$\mathbf{0}$

Tabelle 3.2: Schreibweisen des angewandten Pi-Kalküls

Prozesse werden durch Konkatenation dieser Elemente geschrieben, die jeweils durch Punkte verbunden werden. Ein Beispiel für einen solchen Prozess ist

$$a(x).b(y).\text{if } y = s \text{ then } \overline{x}\langle z \rangle \text{ else } \mathbf{0} \quad .$$

Auf dem Kanal a wird ein Wert empfangen und im Weiteren die Variable x daran gebunden. Ein zweiter Wert, an den y in der Folge gebunden wird, wird auf Kanal b empfangen. Wenn y gleich s ist, so wird x als Kanalname interpretiert und über diesen Kanal ein Name z ausgegeben. Liegt keine Gleichheit vor, so wird keine Operation ausgeführt (dargestellt durch den Null-Prozess).

Wenn Variablen oder Namen neu erzeugt werden, so ist ihre Sichtbarkeit durch Bindung an den nachfolgenden (Teil-)Prozess beschränkt. Die Schreibweise ist $(\nu x)P$. Dieser Bindungsmechanismus wird z. B. für Nonce-Werte oder symmetrische Schlüssel verwendet, die zu Beginn oder während des ganzen Protokollablaufs nur innerhalb eines Teilprozesses bekannt sein dürfen und insbesondere vor einem Angreifer geschützt werden müssen.

Darüber hinaus können auf Prozesse auch sog. aktive Ersetzungen angewendet werden. Diese werden als $\{M/x\}$ geschrieben und erzeugen eine Ersetzung der Variable x durch den Term M. Die Notation $\{M/x\}$ entspricht der Form „let $x = M$ in $\ldots$“. Abadi und Fournet definieren diese Form der Ersetzung als „gleitend“, d. h. sie wirkt sich auf alle Prozesse aus, die sich in der Prozessmenge befinden. Somit ist beispielsweise $\{M/x\}|P|Q$ gleichbedeutend mit $\{M/x\}|P\{M/x\}|Q\{M/x\}$.

Die Wirkung einer Ersetzung kann auch auf einen oder mehrere Prozesse durch Definition einer neuen lokal gebundenen Variable beschränkt werden: $(\nu x)(\{M/x\}|P)|Q$. Da x auf die Ersetzung und den Prozess P beschränkt ist, kann die Ersetzung auch nur innerhalb von P wirksam werden; Q bleibt von der Ersetzung unberührt, da es per Definition kein x enthalten darf (dies wird ggf. durch Umbenennung sichergestellt).

Das Grundmodell zweier über einen Kanal c kommunizierender Prozesse P und Q ist in Abbildung 3.1 zu sehen.

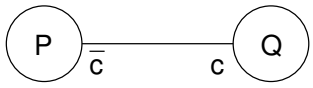


Abbildung 3.1: Direkte Prozesskommunikation

Der Prozess P verfügt über eine „Ausgangsschnittstelle“ $\overline{c}$, über die Informationen an Q übertragen werden können. Q nimmt die Informationen an der „Eingangsschnittstelle“ c entgegen. Der Kanal ist bidirektional, die Zuordnung von Ein- und Ausgangsschnittstelle zeigt lediglich die durch das Protokoll intendierte Nutzung. Für Kanäle, die nicht öffentlich sind, werden die Kanalbezeichner innerhalb der Kreise der Prozesse notiert.

In der Notation des angewandten Pi-Kalküls können die Prozesse wie folgt geschrieben werden:

$$P = \overline{c}\langle d\rangle.P'$$
$$Q = c(x).Q'$$

Dabei ist P ein Prozess, der als ersten Verarbeitungsschritt über seine Ausgangsschnittstelle $\overline{c}\langle\rangle$ zum Kanal c ein Datum d in diesen Kanal sendet. Danach werden alle Schritte des (Sub-) Prozesses P' ausgeführt. Die Punktschreibweise verkettet neben einzelnen Prozessschritten auch Subprozesse.

Der Prozess Q wartet auf eine Nachricht auf Kanal c, um diese der Variable x zuzuweisen, die im folgenden Subprozess Q' dann mit dem empfangenen Wert belegt wird. Die Präfixnotation

$$A \equiv A|\mathbf{0} \qquad \text{(PAR-0)}$$
$$A|(B|C) \equiv (A|B)|C \qquad \text{(PAR-A)}$$
$$A|B \equiv B|A \qquad \text{(PAR-C)}$$
$$!P \equiv P|!P \qquad \text{(REPL)}$$
$$(\nu n).\mathbf{0} \equiv \mathbf{0} \qquad \text{(NEW-0)}$$
$$(\nu u).(\nu v).A \equiv (\nu v).(\nu u).A \qquad \text{(NEW-C)}$$
$$A|(\nu u).B \equiv (\nu u).(A|B) \qquad \text{(NEW-PAR)}$$
$$\text{falls} \quad u \notin \mathrm{fv}(A) \cup \mathrm{fn}(A)$$
$$(\nu x).\{M/x\} \equiv \mathbf{0} \qquad \text{(ALIAS)}$$
$$\{M/x\}|A \equiv \{M/x\}|A\{M/x\} \qquad \text{(SUBST)}$$
$$\{M/x\} \equiv \{N/x\} \quad \text{falls} \quad \Sigma \vdash M = N \qquad \text{(REWRITE)}$$

Tabelle 3.3: Äquivalenzen des Pi-Kalküls

für den Empfang von Informationen bewirkt auch eine Bindung von x an Q'. Außerhalb von Q' ist x nicht bekannt.

Die im Rahmen des angewandten Pi-Kalküls definierten Prozesse liegen parallel zueinander in einem Prozessraum. Diese Nebenläufigkeit von Prozessen kann durch das $|$-Symbol dargestellt werden. Somit lassen sich die Beispielprozesse auch folgendermaßen schreiben:

$$P|Q = \overline{c}\langle d\rangle.P'|c(x).Q'$$

Das Fortschreiten der Kommunikation zwischen den Prozessen kann durch Umformungen nach den Regeln des Kalküls beschrieben werden. Diese Umformungen werden als *Reduktionen* bezeichnet und in der Form $M \to N$ geschrieben. Diese sind nach [AF01]

$$\overline{a}\langle x\rangle.P|a(x).Q \to P|Q \qquad \text{(COMM)}$$
$$\text{if } M = N \text{ then } P \text{ else } Q \to P \qquad \text{(THEN)}$$
$$\text{für alle Aussagen } M, N, \text{ wenn } \Sigma \vdash M = N$$
$$\text{if } M = N \text{ then } P \text{ else } Q \to Q \qquad \text{(ELSE)}$$
$$\text{für alle Aussagen } M, N, \text{ wenn } \Sigma \nvdash M = N$$

Die (COMM)-Regel besagt, dass ein auf dem Kanal a gesendetes Datum durch einen auf Kanal a hörenden Prozess aufgenommen wird. Die Vergleichsoperation $M = N$ bewirkt eine fallabhängige Fortsetzung der Prozessbearbeitung (THEN, ELSE).

Neben der Anwendung der Reduktion werden auch sog. *strukturelle Äquivalenzen* ($\equiv$) verwendet. Eine strukturelle Äquivalenz besteht zwischen zwei Prozessen, wenn sie bis auf eine mögliche Umbenennung von gebundenen Namen gleichwertig in ihrem Verhalten sind. Diese Äquivalenzen nach [AF01] sind in Tabelle 3.3 dargestellt.

Die freien Variablen und Namen eines Prozesses A werden durch die Ausdrücke $\mathrm{fv}(A)$ bzw. $\mathrm{fn}(A)$ bezeichnet. Die entsprechende Notation für gebundene Variablen und Namen ist $\mathrm{bv}(A)$ bzw. $\mathrm{bn}(A)$.

Mittels der Reduktionen und Äquivalenzen können die kommunizierenden Prozesse durch eine entsprechende Substitution wie folgt geschrieben werden:

$$\begin{aligned} P|R &= \overline{c}\langle y\rangle.P'|c(x)R' \\ &\rightarrow P'|R'\{y/x\} \quad \text{mit} \quad x \notin \mathrm{fv}(P') \end{aligned}$$

Alle Vorkommen von x in R' werden somit durch den Wert y ersetzt.

Die Eigenschaft des Kalküls, auch Kanalbezeichner übertragen zu können, ermöglicht die Etablierung neuer Kommunikationsbeziehungen, wie im Folgenden, aus [MPW92] entnommenen, Beispiel dargestellt wird.

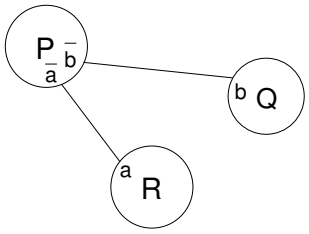


Abbildung 3.2: Beispiel kommunizierender Prozesse

In Abbildung 3.2 hat Prozess P einen Kanal a zum Prozess R und einen Kanal b zum Prozess Q. Nun soll Q von P einen Kanalnamen zur Kommunikation mit R und eine numerische Konstante erhalten und diese über den neu übermittelten Kanal übertragen.

Die zugehörige Prozessdefinition lautet:

$$\begin{aligned} P &\equiv \overline{b}\langle a\rangle.\overline{b}\langle 5\rangle.P' \\ Q &\equiv b(y).b(z).\overline{y}\langle z\rangle.\mathbf{0} \\ R &\equiv a(x).R' \end{aligned}$$

Damit ist die Kommunikation mit gebundenen Variablen zwischen P, Q und R :

$$\begin{aligned} (\nu a)(\nu b)(P|Q|R) &= (\nu a)(\nu b)(\overline{b}\langle a\rangle.\overline{b}\langle 5\rangle.P'|b(y).b(z).\overline{y}\langle z\rangle.\mathbf{0}|a(x).R') \\ &\rightarrow (\nu a)(\nu b)(\overline{b}\langle 5\rangle.P'|b(z).\overline{y}\langle z\rangle.\mathbf{0}\{a/y\}|a(x).R') \\ &\rightarrow (\nu a)(\nu b)(P'|\overline{y}\langle z\rangle.\mathbf{0}\{a/y\}\{5/z\}|a(x).R') \\ &\equiv (\nu a)(\nu b)(P'|\overline{a}\langle 5\rangle.\mathbf{0}|a(x).R') \\ &\rightarrow (\nu a)(\nu b)(P'|\mathbf{0}|R'\{5/x\}) \\ &\equiv (\nu a)(\nu b)(P'|R'\{5/x\}) \\ &\equiv P'|R'\{5/x\} \quad \text{für} \quad a, b \notin \mathrm{fv}(P') \cup \mathrm{fv}(R') \end{aligned}$$

3.2.4 Funktionen

Im angewandten Pi-Kalkül können beliebige Funktionen definiert werden, die Terme aus Σ miteinander verknüpfen. So ist beispielsweise die Definition von mehrstelligen Termen durch entsprechende Funktionen möglich. So kann ein 3-Tupel durch eine Funktion $DreiTupel(x, y, z)$ beschrieben werden. Der Zugriff auf die einzelnen Elemente des Tupels erfolgt durch entsprechende Stellenfunktionen:

$$\begin{aligned} erstes(DreiTupel(x, y, z)) &= x \\ zweites(DreiTupel(x, y, z)) &= y \\ drittes(DreiTupel(x, y, z)) &= z \end{aligned}$$

Nach [Bla09] können solche Funktionen als Konstruktoren und Destruktoren verwendet werden. Hierbei werden Konstruktoren für Terme der Form $f(M_1, \ldots, M_n)$ definiert. Daneben werden Destruktoren der Form $g(M_1, \ldots, M_n)$ definiert. Diese Destruktoren werden nicht selbst in Termen verwendet, sondern dienen zur Verarbeitung von Termen innerhalb der Prozesse. Hierzu liegt für $g(M_1, \ldots, M_n)$ eine Menge von Reduktionsregeln der Form $g(M'_1, \ldots, M'_n) \rightarrow M'$ vor, die häufig Konstruktoren und Destruktoren paarweise in Beziehung zueinander setzen. Ein Beispiel ist

Konstruktor:	$f(x_1, x_2)$
Destruktor:	$g(f(x_1, x_2), x_2) \rightarrow x_1$

Zur Definition der symmetrischen Verschlüsselung werden zwei parametrisierte Funktionsbezeichner für Verschlüsselung (*enc*ryption) und Entschlüsselung (*dec*ryption) eingeführt.

$$\begin{aligned} y &= \mathrm{enc_s}(x, k) \\ \mathrm{dec_s}(y, k) &= \mathrm{dec_s}(\mathrm{enc_s}(x, k), k) \rightarrow x \end{aligned}$$

Hierbei sind x der Klartext, y das Chiffrat und k ein symmetrischer Schlüssel. Die Definition des Destruktors erfolgt dabei wie bereits eingeführt in Form einer Reduktionsregel.

In gleicher Weise lässt sich die asymmetrische Kryptographie beschreiben. Hierbei ist jedoch zunächst die Beziehung zwischen öffentlichem und privatem Schlüsselteil zu modellieren. Dies kann durch eine generierende Funktion geschehen, die einem vorhandenen privaten Schlüssel K^{-1} sein öffentliches Pendant K zuordnet.

Schlüsselgenerierung	$K = \mathrm{pk}(K^{-1})$
Asymmetrische Verschlüsselung	$\mathrm{enc_a}(x, \mathrm{pk}(K^{-1}))$
Asymmetrische Entschlüsselung	$\mathrm{dec_a}(\mathrm{enc_a}(x, \mathrm{pk}(K^{-1})), K^{-1}) \rightarrow x$

Regeln für Signaturverfahren *mit* Rückgewinnung der Nachricht:

Schlüsselgenerierung	$K = \mathrm{pk}(K^{-1})$
Signatur von p mit Schlüssel K^{-1}	$\mathrm{sign}(p, K^{-1})$
Signaturprüfung	$\mathrm{checksign}(\mathrm{sign}(p, K^{-1}), \mathrm{pk}(K^{-1})) \rightarrow p$
Abspaltung der Nachricht	$\mathrm{getmessage}(\mathrm{sign}(p, K^{-1})) \rightarrow p$

Für Signaturverfahren *ohne* Rückgewinnung (or) der Nachricht:

Schlüsselgenerierung	$K = \mathrm{pk}(K^{-1})$
Signatur von p mit Schlüssel K^{-1}	$\mathrm{sign}_{\mathrm{or}}(p, K^{-1})$
Signaturprüfung	$\mathrm{checksign}_{\mathrm{or}}(\mathrm{sign}_{\mathrm{or}}(p, K^{-1}), \mathrm{pk}(K^{-1}), p) \rightarrow true$

Nicht-umkehrbare Hash-Funktionen werden nur durch einen Konstruktor dargestellt.

Hash-Funktion	$h(x)$

Ohne Destruktor ist daher auch kein Rückschluss auf das Argument der Hashfunktion möglich, außer dieses ist zuvor bekannt: $h(x) = h(x') \Leftrightarrow x = x'$.

3.3 Automatisierte Protokollanalysen mit Proverif

Für Analysen bei Protokollmodellierungen im angewandten Pi-Kalkül wurde durch Bruno Blanchet in [Bla01] eine auf Prolog basierende Verifikationsmethodik vorgestellt, die in Form des Modell-Checkers Proverif implementiert wurde [Bla11].

Proverif stellt eine Beschreibungssprache bereit, mit der Prozesse im Modell des angewandten Pi-Kalküls beschrieben werden können. Proverif unterstützt dabei eine untypisierte und eine typisierte Variante. Die Modellierung der Prozesse kann durch Proverif automatisiert in ein Horn-Klausel-System überführt werden und durch einen Theorembeweiser auf Basis des Resolutionsverfahrens hinsichtlich vorgegebener Beweisziele überprüft werden [Bla09].

Mit Hilfe von Proverif können sowohl die Eigenschaft der Wahrung der Vertraulichkeit bestimmter Informationen als auch die Garantie der Authentizität von Nachrichten für das modellierte Protokoll nachgewiesen werden.

3.3.1 Notation und Semantik

Alle im angewandten Pi-Kalkül definierten Funktionen können in Proverif mit einer etwas modifizierten Notation dargestellt werden. Tabelle 3.4 setzt die beiden Notationen in Beziehung.

Es können in Proverif eigene Datentypen definiert werden und Formeln für Konstruktoren und Destruktoren angegeben werden. Der folgende Ausschnitt zeigt die Funktionsdefinition für die Generierung eines öffentlichen Schlüssels (Typ: pkey) aus einem privaten Schlüssel (Typ: skey), die Definition einer zweistelligen asymmetrischen Verschlüsselungsfunktion und die zugehörige Reduktion für die Entschlüsselung.

angewandter Pi-Kalkül	Proverif mit Datentypen
$P \vert Q$	$P \mid Q$
$a(x).P$	$\mathbf{in}(\mathsf{a}, x\text{: bitstring}); P$
$\overline{a}\langle M\rangle.P$	$\mathbf{out}(\mathsf{a}, M); P$
if $M = N$ then P else Q	$\mathbf{if}\ M = N\ \mathbf{then}\ P\ \mathbf{else}\ Q$
$(\nu n).P$	$\mathbf{new}\ n\text{: bitstring};\ P$
$!P$	$!P$
$\mathbf{0}$	0
$\{M/x\}$	$\mathbf{let}\ x\text{: bitstring} = M\ \mathbf{in}\ \ldots$

Tabelle 3.4: Gegenüberstellung der Schreibweisen von Pi-Kalkül und Proverif

```
type skey.
type pkey.
fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall m: bitstring, sk: skey; adec(aenc(m, pk(sk)), sk) = m.
```

Der Datentyp *bitstring* ist dabei ein bereits von Proverif vordefinierter Typ.
Die gezeigte Definition entspricht den oben eingeführten Ausdrücken $\mathrm{enc_a}(x, \mathrm{pk}(K^{-1}))$ und $\mathrm{dec_a}(\mathrm{enc_a}(x, \mathrm{pk}(K^{-1})), K^{-1}) \rightarrow x$.

Übertragungskanäle zwischen den Prozessen werden durch die Definition einer freien Variablen mit dem *channel*-Typ durch „**free** c:channel.“ definiert. Alle freien Variablen, Konstanten und Funktionen werden als öffentlich bekannt betrachtet, sofern sie nicht durch den Zusatz *private* als nicht öffentlich gekennzeichnet sind.

Neue, gebundene Namen sind per Definition nicht öffentlich und müssen ggf. explizit an Prozesse übergeben werden. Bei der Erzeugung eines asymmetrischen Schlüsselpaares für einen Prozess *userB* kann durch einen Parameter in der Prozessdefinition der private Schlüssel skB vom Typ *secret signing key (sskey)* dem Prozess bekannt gemacht werden:

```
let userB(skB:sskey, ...) =
  new d:bitstring;
...
.

process
  ...
  new skB:sskey;
  let pkB = pk(skB) in
  ...
  (... | (!userB(skB, ...)) | ...)
```

Um Eigenschaftsprüfungen modellieren zu können, wird in Proverif ein Destruktor $g(M_1, \ldots, M_n) = M$ innerhalb einer bedingten Zuweisung verwendet:

$$\text{let } x = g(M_1, \ldots, M_n) \text{ in } P \,\text{else}\, Q$$

Da ein Destruktor eine partielle Funktion ist, wird die bedingte Zuweisung nach [AB05] wie folgt definiert: Gibt es eine Substitution σ so, dass für jedes M'_i gilt $\sigma M'_i = M_i$ und $\sigma M' = M$, so wird $P\{M/x\}$ fortgeführt, anderenfalls Q.

Der folgende Beispielprozess zeigt eine symmetrische Ver- und Entschlüsselung mit Definition eines Konstruktors (Zeile 6), der zugehörigen Reduktion (Zeile 7), der Anwendung des Konstruktors (Zeile 12) und des Destruktors (Zeile 14).

```
01  (* öffentlicher Kommunikationskanal *)
02  free net:channel.
03
04  (* Verschlüsselung mit symmetrischen Schlüsseln *)
05  type ekey.
06  fun senc(bitstring, ekey): bitstring.
07  reduc forall m: bitstring, k: ekey; sdec(senc(m, k), k) = m.
08
09  process
10          new K:ekey;
11          new d:bitstring;
12          let encdata = senc(d, K) in
13            ...
14            let decdata = sdec(encdata, K) in
15                    out(net, decdata)
16            else 0
```

In diesem Prozess ist die Zuweisung an die Variable $decdata$ nur dann möglich, wenn der Destruktor sdec für die gegebene Variablenbelegung definiert ist. In diesem Fall ist

$$g(M_1, M_2) = M \quad \equiv \quad \mathsf{sdec}(\overbrace{\mathsf{senc}(d, K)}^{M_1}, \overbrace{K}^{M_2}) = \overbrace{d}^{M}$$

$$g(M'_1, M'_2) = M' \quad \equiv \quad \mathsf{sdec}(\overbrace{\mathsf{senc}(m, k)}^{M'_1}, \overbrace{k}^{M'_2}) = \overbrace{m}^{M'}$$

Somit kann die Substitution $\sigma : \{d/m, K/k\}$ angewendet werden. Damit ist $g(\sigma M'_1, \sigma M'_2) = g(M_1, M_2)$ und $g(M_1, M_2) = \sigma M'$ und der Prozessschritt $\mathbf{out}(\mathsf{net}, decdata)$ kann mit $\{d/decdata\}$ ausgeführt werden und somit wird das Datum d über den Kommunikationskanal net ausgegeben.

3.3.2 Nachweis von Vertraulichkeit

Liegt eine Prozessmodellierung in Proverif vor, so können in der Prozessbeschreibung Beweisziele definiert werden. Hierzu werden Abfragen (*queries*) formuliert, die durch den Theorembeweiser ausgewertet werden sollen.

Dabei wird das Angreifer-Modell von Dolev und Yao [DY83] zugrunde gelegt. Dieses Modell beschreibt einen Angreifer, der vollständige Gewalt über alle Nachrichten hat, die über öffentliche Kanäle übermittelt werden. Das bedeutet, dass er Nachrichten mitlesen, verzögern, löschen, verändern oder nach Belieben neu erzeugen kann. Verschlüsselte Nachrichten können gelesen werden, deren Bedeutung bleibt jedoch verborgen, solange der geheime Schlüssel nicht zur Verfügung steht. Das initiale Wissen des Angreifers umfasst zudem alle Funktionen und Namen, die nicht als privat markiert sind. Jede Nachricht, die übertragen wird, ergänzt die Wissensmenge des Angreifers. Um die Vertraulichkeit eines Datums nachzuweisen, muss dieses zunächst als gebundener Name erzeugt werden, wie z. B. der geheime Signaturschlüssel von *userB*. Die zu untersuchende Fragestellung wird dann formuliert als $\mathbf{query}\ attacker(\mathbf{new}\ skB)$.

Proverif prüft dann, ob die Eigenschaft „$\mathbf{not}\ attacker\ (\mathbf{new}\ skA)$." gegeben ist, also ob der Schlüssel *skA* nicht zur Wissensmenge des Angreifers gehört. Dabei werden das Initialwissen, alle Nachrichten, die über öffentliche Kanäle übertragen werden und alle daraus durch Funktionen ableitbaren Terme berücksichtigt.

Gelingt der Nachweis, so wird das Beweisziel als wahr ausgegeben. Anderenfalls wird es als falsch gekennzeichnet und ein möglicher Ableitungsweg angegeben, mit dem es dem Angreifer gelingt, das betrachtete Datum herzuleiten.

Zur Anwendung kommt dieses Verfahren beispielsweise bei der Analyse von Sicherheitsprotokollen in [AB05].

3.3.3 Nachweis von Authentizität

Blanchet erweitert den angewandten Pi-Kalkül um die syntaktische Struktur $\mathrm{event}(M).P$, die das Eintreten eines Ereignisses anzeigt [Bla09]. M ist dabei ein Satz von Parametern.

Um Authentizität nachzuweisen wird zunächst festgelegt, dass ein Ereignis E bei einem Empfänger B, das einem Sender A zugeordnet wird, dann als authentisch in Bezug auf A gewertet wird, wenn zwingend ein anderes Ereignis E_0 bei A vorausgegangen sein muss. Diesen Zusammenhang bezeichnet Blanchet als *Korrespondenz*:

$$E \rightsquigarrow E_0$$

Statt eines Ereignisses E_0 können auch verbundene Ereignisse mit dem betrachteten Ereignis korrespondieren:

$$E \rightsquigarrow E_1 \wedge E_2 \wedge \ldots \wedge E_n$$

In Abfragen in Proverif wird diese Abhängigkeit durch die Notation ==> geschrieben.

Ein Beispiel für den Nachweis von Authentizität ist die Übermittlung eines signierten Datums von A an B.

```
query d:bitstring; event(received(d)) ==> event(newdata(d)).

(* Definition Teilprozess User A *)
let userA(skA:sskey) =
  new d:bitstring;
  event newdata(d);
  out(net, sign(d, skA)).

(* Definition Teilprozess User B *)
let userB(pkA:pskey) =
  in(net, signedData:bitstring);
  let d' = checksign(signedData, pkA) in
    event received(d').

process
  new skA:sskey;
  (!userA(skA) |!userB(pk(skA)))
```

Im Teilprozess *userA* wird ein neues Datum d erzeugt, signiert und über den Kanal net übertragen. *userB* empfängt das Datum und führt eine Signaturprüfung mit dem öffentlichen Schlüssel $\mathsf{pk}(skA)$ von *userA* aus.

Für jedes erzeugte Datum d wird ein Ereignis *newdata* und nach erfolgreicher Signaturprüfung entsprechend ein *received*-Ereignis registriert. Da die Signaturprüfung nur dann erfolgreich ist, wenn das Datum von *userA* signiert wurde, so muss einem *received*-Ereignis ein *newdata*-Ereignis vorausgehen und die in der Abfrage geprüfte Korrespondenz

$$received(d) \rightsquigarrow newdata(d)$$

ist erfüllt.

Im Beispielprozess ist es für einen Angreifer möglich, ein signiertes Datum beliebig oft wiederholt an *userB* zu senden. Zum Nachweis von frischen, authentischen Nachrichten wurde die *injektive Korrespondenz* definiert:

$$E \rightsquigarrow [inj]E_0$$

Diese Korrespondenz ist nur dann erfüllt, wenn zu jedem Ereignis E genau ein Ereignis E_0 ausgeführt wird.

Ein ausführliches Anwendungsbeispiel zur Anwendung der Proverif-Modellierung findet sich in B.2.

4 Grammatikbasierte Zertifikate

Die im ersten Kapitel formulierten Ziele spannen einen breiten Anforderungskorridor auf, der nicht durch die in Kapitel 2 vorgestellten Verfahren abgedeckt werden kann. Ein wesentliches Problem der vorliegenden Aufgabenstellung liegt im vordergründigen Widerspruch der Anforderungen Verständlichkeit und automatisierte Auswertung bei gleichzeitig geforderter Verbindlichkeit durch Signaturen und Delegierbarkeit.

Wird eine Zusicherung als Textdokument erstellt, in der der Informationsgeber den genauen Wortlaut seiner Erklärung niederlegt und anschließend signiert, so ist zwar die Verständlichkeit gegeben, jedoch eine automatische Weiterverarbeitung nicht einfach möglich.

Werden hingegen die wesentlichen Informationselemente einer Zusicherung strukturiert für die automatische Weiterverarbeitung aufbereitet und signiert, so ist zwar den Anforderungen der maschinellen Verarbeitung Genüge getan, ein rechtssicheres Verständnis beim Informationsgeber oder bei Interpretation der Daten durch einen menschlichen Anwender ist jedoch schwierig oder nicht möglich.

In diesem Kapitel wird ein neues Konzept *grammatikbasierter Zertifikate* vorgestellt, das es ermöglicht, die vorgegebenen Ziele zu erfüllen.

4.1 Darstellungsformen

Die Darstellung von Zusicherungen kann auf unterschiedliche Art und Weise erfolgen. Im Folgenden werden zwei Darstellungsformen für Zusicherungen betrachtet, die jeweils einen Teil der Anforderungen abdecken können. Diese Darstellungsformen sind die *textuelle* und die *strukturierte* Form. Die beiden Formen werden durch ein Beispiel veranschaulicht.

4.1.1 Textuelle Darstellung

Bei der *textuellen Darstellung* handelt es sich um einen natürlichsprachlichen Text, der dem Anwender direkt ohne Hinzunahme von Auswertungswerkzeugen oder Interpretationsfestlegungen verständlich ist.

Diese Vorgabe schließt jedoch explizit keine Allgemeinverständlichkeit ein. Es kann sich bei den Texten – insbesondere im Fall der Zusicherung von Sicherheitseigenschaften – um eine

Fachsprache für ein spezielles Anwendungsfeld handeln. Im Rahmen dieser Arbeit wird davon ausgegangen, dass es sich bei den Texten um Aussagesätze handelt, deren Wahr-sein durch die Erstellung einer Zusicherung bestätigt wird.

Als Beispiel zur Einführung der unterschiedlichen Darstellungsformen wird eine Auditierung des Einsatzes von Kryptoservern mit Hochsicherheitsmodulen (HSM) betrachtet. Im Rahmen einer solchen Prüfung wird häufig nachvollzogen, nach welchen Prüfverfahren wie FIPS oder Common Criteria ein Gerät zertifiziert wurde. Darüber hinaus wird geprüft, ob das Gerät logisch und physikalisch noch in unverändertem Zustand ist und somit auch die Prüfgrundlage für die erteilten Zertifizierungen noch gegeben ist. Alternativ können für durchgeführte Änderungen durch Herstellererklärungen wiederum neue Firmware-Module legitimiert werden.

Ein entsprechendes (verkürztes) natürlichsprachliches Protokoll einer solchen Prüfung könnte beispielsweise folgende Formen haben:

> Es wurde der Kryptoserver cryptsrv1 in Serverschrank S-7009 auditiert.
> Das HSM hat folgende Zertifizierungen:
> – FIPS 140-2 Level 3
> – ZKA-Zulassung
> Die Firmwaremodule entsprechen dem Evaluierungszustand.
> Die Siegel an den Schnittstellen sind ungebrochen.

oder auch

> Es wurde das Kryptomodul nciphercrypt1 im Hostsystem lcryptsrv in Serverschrank S-8011 auditiert.
> Das HSM hat folgende Zertifizierungen:
> – FIPS 140-2 Level 3
> Die Firmwaremodule entsprechen nicht dem Evaluierungszustand.
> Folgende Module wurden veraendert bzw. ergaenzt:
> – pin-1.0.2.mod
> – basefunc-1.3.4.mod
> Die Siegel an den Schnittstellen sind ungebrochen.

Die Aussagen sind lesbar und im Anwendungsbereich Sicherheitsprüfungen direkt durch den Anwender interpretierbar. Im Weiteren wird die natürlichsprachliche textuelle Form als *primäre Form* oder auch *Primärtext* p bezeichnet.

4.1.2 Strukturierte Darstellung

Zur einfachen automatisierten Weiterverarbeitung der gegebenen Informationen ist eine strukturierte Darstellung der Informationen unerlässlich.

Eine mögliche Strukturierung besteht darin, jede Information, die in der textuellen Form enthalten ist, durch ein (ggf. hierarchisch angeordnetes) Strukturelement darzustellen. Diesem Strukturelement wird außerhalb der strukturierten Daten die Bedeutung zugeordnet, die aus dem Satz in der textuellen Darstellung hervorgeht. Im allgemeinen Fall stehen einzelne Informationselemente in ebenfalls hierarchischen Beziehungen zueinander, so dass dadurch baumartige Strukturen entstehen, die ermöglichen, den in den Blättern angeordneten Angaben und den Bezeichnern von Knoten jeweils durch ihre Position im Baum eine Bedeutung zuzuordnen. Dabei lassen sich aus den jeweils übergeordneten Knoten des Baumes die Zusammenhänge von Teilbäumen in einem übergeordneten Kontext ableiten.

Der Informationsgehalt der Auditierung aus dem zweiten Beispiel im vorigen Abschnitt ist in Abbildung 4.1 auf diese Weise graphisch als Baumstruktur dargestellt. Es sind gegenüber der textuellen Darstellung zusätzliche Metainformationen enthalten (Typ *Name*) und nicht jede Kontextinformation ist ablesbar (aus dem Teilbaum *Firmware* ist beispielsweise nicht erkennbar, dass die Nennung der darunter liegenden Firmware-Module eine Abweichung von der Standard-Firmware impliziert).

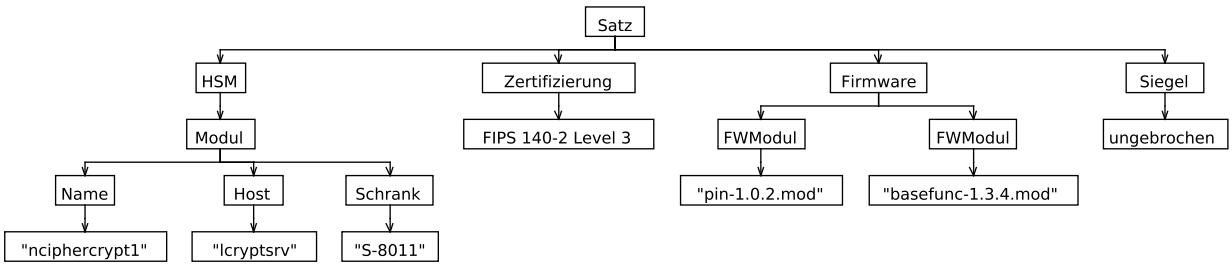


Abbildung 4.1: Strukturierte Darstellung des Informationsgehalts einer Zusicherung

Die analoge Darstellung in maschinenlesbarer Form kann beispielsweise durch das XML-Format geleistet werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<Satz>
   <HSMSystem>
      <Modul>
         <Name>nciphercrypt1</Name>
         <Host>lcryptsrv</Host>
         <Schrank>S-8011</Schrank>
      </Modul>
   </HSMSystem>
   <Zertifizierung>FIPS 140-2 Level 3</Zertifizierung>
   <Firmware>
      <FWModul>pin-1.0.2.mod</FWModul>
      <FWModul>basefunc-1.3.4.mod</FWModul>
   </Firmware>
   <Siegel>ungebrochen</Siegel>
</Satz>
```

Bei den gewählten Darstellungsformen handelt es sich um sog. *semi-strukturierte Daten*, da sie zwar eine Struktur aufweisen, diese jedoch im Gegensatz zu starren Strukturen wie in Datenbanktabellen variieren kann. Im weiteren wird jedoch lediglich der Begriff *strukturierte Daten* hierfür verwendet. Diese Darstellungsform wird auch als *sekundäre Form* bezeichnet. Mit diesem Begriff wird impliziert, dass diese Form aus der primären, textuellen Form unter noch zu betrachtenden Randbedingungen ableitbar ist.

Auf diese Weise strukturierte Daten können einfach weiterverarbeitet werden. Für eine korrekte Interpretation werden jedoch zusätzliche Informationen benötigt.

4.2 Zusicherungen als Textbausteine

4.2.1 Formerfordernis

Der Einsatz von Zusicherungen mit der Möglichkeit zur automatisierten Auswertung ist in erster Linie da sinnvoll, wo wiederholt gleichartige oder ähnliche Zusicherungen benötigt werden.

Bei Sicherheitsanalysen nach Kapitel 1 werden i. d. R. eine große Zahl vorgegebener Prüfschritte abgearbeitet und die zugehörigen Ergebnisse dokumentiert. Werden die Prüfungen periodisch durchgeführt, so ist neben der genauen Vorgabe auch eine festgeschriebene Form der Dokumentation wie Formulare oder Textbausteine sinnvoll. Durch Vorgabe solcher Bausteine kann sowohl sichergestellt werden, dass die durch den Auditor getroffene Aussage den inhaltlichen und formalen Anforderungen an eine Sicherheitsüberprüfung genügt als auch die Vergleichbarkeit verschiedener Durchführungen der gleichen Prüfung gewährleistet wird.

Es stellt daher keine Einschränkung des Anwendungsfalles dar, für die textuelle Darstellung einer Zusicherung zu fordern, dass sie zunächst beliebig gewählt werden kann, dann jedoch hinsichtlich der verwendeten Formulierungen fest bleibt.

Zusicherungen, die zum gleichen Anwendungsfall wie beispielsweise der Auditierung der HSMs im vorigen Abschnitt gehören, sollen daher grundsätzlich aus gleichartigen Bestandteilen (Textelementen, Phrasen, ...) bestehen und ggf. variable Elemente wie z. B. Namen enthalten. Es wird daher angenommen, dass es für einen konkreten Anwendungsfall eine Beschreibung gibt, aus der sich ableiten lässt, wie die Formulierung einer Zusicherung gestaltet sein muss. Die Festlegung soll sich dabei auf die vollständige Ausgestaltung des Textes beziehen, d. h. alle Freiheitsgrade bei der Erstellung von Zusicherungen sollen definiert sein. Es handelt sich also bei den Zusicherungen nicht um frei wählbare Aussagen, die bestimmte Strukturen aufweisen. Diese Texte sind somit Elemente einer *Domänen-spezifischen Sprache*.

4.2.2 Vorgabe von Textbausteinen

Unter einem Textbaustein werden vorgefertigte Textelemente verstanden, die ggf. miteinander kombiniert werden können und Platzhalter für individuelle Ergänzungen vorsehen. Im ersten Schritt werden statische Textelemente definiert.

Ein *statischer Textbaustein* wird definiert als eine festgelegte Folge von Zeichen aus einem Zeichenvorrat Σ. Der Zeichenvorrat umfasst dabei die Buchstaben der natürlichen Sprache, Ziffern sowie Leer- und Satzzeichen. Ein Beispiel eines Zeichenvorrats für deutsche Zusicherungen ist

$$\Sigma = \{\text{'a'}, \ldots, \text{'z'}, \text{'A'}, \ldots, \text{'Z'}, \text{'ä'}, \text{'ö'}, \text{'ü'}, \text{'Ä'}, \text{'Ö'}, \text{'Ü'}, \text{'ß'}\} \cup \{\text{'0'}, \ldots, \text{'9'}\} \cup \{\text{','}, \text{'.'}, \text{':'}, \text{';'}, \text{'\_'}, \text{'-'}, \text{' '}\}$$

Für statische Textbausteine T wird festgelegt, dass diese nicht leer sein dürfen, also $T \in \Sigma^+$. Die Notation Σ^+ beschreibt die Menge aller möglichen Folgen von Zeichen aus Σ mit Mindestlänge eins. Ein Beispiel für einen statischen Textbaustein ist

> Alle Personen mit Zugang zum geprüften System wurden sicherheitsüberprüft. Es liegen für alle Personen Führungszeugnisse vor, die innerhalb der letzten zwei Jahre ausgestellt wurden.

Zusicherungen können aus mehreren Textbausteinen zusammengesetzt werden. Diese werden durch Aneinanderreihung gebildet. Eine natürlichsprachliche Darstellung p kann damit als $p = T_1T_2T_3 \ldots T_n$ mit $T_1, \ldots, T_n \in \Sigma^+$ geschrieben werden. Die Bezeichner für die Textbausteine können beliebig gewählt werden.

4.2.3 Variable Bestandteile

Da es für Zusicherungen i. d. R. verschiedene Varianten gibt und Zusicherungen z. B. für ein konkretes System individualisiert werden sollen, ist es notwendig, variable Bestandteile vorzusehen, die durch den Bestätiger ausgewählt oder eingesetzt werden können.

4.2.3.1 Bausteinebene

Werden für eine Zusicherung Varianten vorgesehen, die sich jeweils durch einen eigenen Textbaustein darstellen lassen, so kann dies durch die Notation $T_x|T_y$ dargestellt werden. Es können dabei wahlweise die Textbausteine T_x oder T_y eingesetzt werden. Zur Anwendung kommt eine solche Auswahlmöglichkeit für Alternativen wie z. B. *Die Eignung der Komponente wurde durch eine Evaluationsurkunde belegt.* und *Für die Komponente wurde als Eignungsnachweis eine Herstellererklärung abgegeben.* Damit lassen sich Mengen von darstellbaren Primärtexten z. B. in der Form $T_1(T_x|T_y)T_2$ ausdrücken.

Auswahlmöglichkeiten lassen sich durch eigene Bezeichner kennzeichnen und in einer Menge der *variablen Textbausteine* zusammenfassen. Jeder dieser variablen Textbausteine legt fest, durch welche anderen statischen oder variablen Textbausteine er ersetzt werden kann. Ein variabler Textbaustein stellt somit eine *Ersetzungsregel* dar.

Ein variabler Textbaustein kann als Anfangspunkt festgelegt werden. Durch fortgesetzte Anwendung von Ersetzungsregeln können solange variable Textbausteine ersetzt werden, bis lediglich eine Verkettung von statischen Textbausteinen übrig bleibt.

Ein einfaches Beispiel soll diese Zusammenhänge verdeutlichen. Für Aussagen der Form

> Das geprüfte System wurde nach BSI-Grundschutz gehärtet. Für alle individuellen Benutzerkonten wurden genehmigte Anträge vorgelegt.
>
> Das geprüfte System ist entsprechend der Arbeitsanweisung AA-2010-237 gehärtet. Es gibt keine individuellen Benutzerkonten.

können folgende Textbausteine festgelegt werden.

Statische Bausteine
H_1 ‘Das geprüfte System wurde nach BSI-Grundschutz gehärtet. ’
H_2 ‘Das geprüfte System ist entsprechend der Arbeitsanweisung AA-2010-237 gehärtet. ’
B_1 ‘Für alle individuellen Benutzerkonten wurden genehmigte Anträge vorgelegt. ’
B_2 ‘Es gibt keine individuellen Benutzerkonten. ’

Variable Bausteine
V_H $H_1|H_2$
V_B $B_1|B_2$
Z $V_H V_B$

Eine Zusicherung (Baustein Z) besteht aus einem variablen Baustein V_H für die Härtungsoptionen und einem variablen Baustein V_B für die Angaben zu den Benutzerkonten. Für V_H und V_B stehen jeweils zwei Alternativen zur Verfügung, so dass insgesamt vier Varianten für die Zusicherung möglich sind.

4.2.3.2 Feldebene

Zusicherungen können auch durch frei oder in Grenzen wählbare Angaben individualisiert werden. Ein Beispiel für eine solche Angabe sind Servernamen, die nicht in den Textbausteinen enthalten sind oder Zahlwerte wie Port-Nummern, IP-Adressen, usw. Für diese Angaben werden üblicherweise Formatvorgaben in Form von Datentypen wie *Integer*, *String* oder *Timestamp* gemacht. Häufig werden zu diesen Datentypen auch Prüfausdrücke in Form von regulären Ausdrücken definiert.

Reguläre Ausdrücke sind formalisierte Vorlagen zur Durchführung eines Mustervergleichs mit Zeichenketten. Dabei enthalten diese Vergleichsmuster u. a.

- ❑ normale Zeichen, die in der Zeichenkette enthalten sein müssen
- ❑ Platzhalter für beliebige Zeichen: ‘.’
- ❑ Platzhalter für bestimmte Zeichenmengen: [A-Z], [0-9], . . .
- ❑ Quantoren: *, +, ?
- ❑ Klammern zur Gruppierung: (abc)+
- ❑ Alternativen: (abc | cde)

Die Quantoren geben dabei die mögliche Häufigkeit des davor stehenden Elementes an (*: keinmal oder beliebig oft, +: einmal oder beliebig oft, ?: keinmal oder einmal).

Ein Integer-Wert ohne führende Nullen wird beispielsweise durch den regulären Ausdruck `[1-9][0-9]+` beschrieben. Eine ausführliche Darstellung regulärer Ausdrücke findet sich in [Fri06].

Solche gestaltbaren Felder können durch Zuordnung eines variablen Textbausteins zur Bezeichnung eines Datentyps und damit mittelbar zu einem Prüfausdruck realisiert werden.

Beispielsweise kann ein Servername anstelle des Ausdrucks „Das geprüfte System ...“ angegeben werden. Damit wird das oben angegebene Beispiel wie folgt modifiziert:

Statische Bausteine
H_0 ‘Der Server ’
H_1 ‘ wurde nach BSI-Grundschutz gehärtet. ’
H_2 ‘ ist entsprechend der Arbeitsanweisung AA-2010-237 gehärtet. ’
B_1 ‘Für alle individuellen Benutzerkonten wurden genehmigte Anträge vorgelegt. ’
B_2 ‘Es gibt keine individuellen Benutzerkonten. ’

Variable Bausteine
V_H $H_1|H_2$
V_B $B_1|B_2$
S_1 *String*
Z $H_0 S_1 V_H V_B$

Frei wählbare Zeichenketten werden zur Unterscheidung vom festgelegten umgebenden Text durch Anführungszeichen abgetrennt. Eine konkrete Ausprägung für den Anfang der oben definierten Zusicherung kann damit beispielsweise heißen: *Der Server “mailsrv1” wurde nach BSI-Grundschutz gehärtet.*

4.3 Nutzung kontextfreier Grammatiken

Ob eine Zusicherung in Form einer Zeichenkette die Anforderungen an die Strukturvorgaben eines bestimmten Anwendungsfalles erfüllt und damit ein geeigneter Primärtext ist, soll mittels einer Prüffunktion entschieden werden.

Diese Anforderung lässt sich durch die Vorgabe einer sog. *kontextfreien Grammatik* für die innerhalb eines Anwendungsfalles zulässigen Zusicherungen umsetzen. Die Prüfung der syntaktischen Korrektheit einer Zusicherung kann durch einen für diese Grammatik entworfenen Parser erfolgen.

Durch die Vorgabe der Grammatik erkennt der Parser die Strukturen und Formate der Eingangsdaten, kann diese in ihre Bestandteile zerlegen und ggf. fehlerhafte Eingangsdaten zurückweisen (validierender Parser). Damit ist es auch möglich, durch geeignete Gestaltung der kontextfreien Grammatik die hierarchisch aufeinander aufbauenden Informationsbestandteile, die für eine strukturierte Darstellung benötigt werden, zu extrahieren.

Kontextfreie Grammatiken kommen i. d. R. zur Beschreibung von Programmiersprachen und ähnlich syntaktisch konkret festgelegten Textdarstellungen zum Einsatz. Die Elemente zur Definition einer Grammatik entsprechen den bereits eingeführten Elementen zur Beschreibung der Zusicherungen.

4.3.1 Prinzip der kontextfreien Grammatik

Die statischen Textbausteine einer Zusicherung werden als *Terminale* und die variablen Bausteine als *Nicht-Terminale* bezeichnet. Entsprechend sind die Mengen $\mathcal{T}$ als Menge der Terminale und $\mathcal{N}$ als Menge der Nicht-Terminale definiert. Die Ersetzungsregeln für Nicht-Terminale werden als Produktionen bezeichnet und in einer Menge $\mathcal{P}$ zusammengefasst. Zusätzlich wird ein Startsymbol S definiert.

Zusicherungen lassen sich somit durch eine anwendungsfallspezifische formale Grammatik

$$G =< \mathcal{N}, \mathcal{T}, \mathcal{P}, S >$$

beschreiben.

Die Grammatik soll kontextfrei sein, d. h. die Produktionsregeln sind jeweils für genau ein Nicht-Terminal auf der linken Seite definiert: $\mathcal{P} \subset \mathcal{N} \times (\mathcal{N} \cup \mathcal{T})^+$. In der Chomsky-Hierarchie von Grammatiken entspricht dies dem Typ 2 [Cho59].

Ein Element p aus $\mathcal{P}$ wird geschrieben als $p = N \rightarrow \alpha$. Das Nicht-Terminal N kann bei Anwendung der Produktionsregel p durch die Symbolfolge α ersetzt werden.

Soll eine Zeichenfolge, die den Regeln der Grammatik G entspricht, konstruiert werden, so werden ausgehend von S die Produktionsregeln solange angewendet, bis durch Ersetzungen kein Nicht-Terminal mehr in der Symbolfolge enthalten ist.

$$S \rightarrow_G \alpha_1 \rightarrow_G \ldots \rightarrow_G \alpha_2 \rightarrow_G w$$

$w \in \mathcal{T}^+$ wird als Ableitung von S (oder *Wort*) bezeichnet. Die Folge von Anwendungen der Produktionsregeln kann auch verkürzt als $S \rightarrow_G^* w$ geschrieben werden.

Die Menge aller dieser aus S ableitbaren Symbolfolgen (oder auch Worte w) wird Sprache von G genannt (Schreibweise $\mathscr{L}(G)$):

$$\mathscr{L}(G) = \{w | \exists S \rightarrow_G^* w, w \in \mathcal{T}^+\}$$

In der Menge $\mathcal{T}$ kann auch das Symbol ϵ enthalten sein, das das sog. „leere Wort“ repräsentiert.

Neben der Kontextfreiheit wird für den vorliegenden Anwendungsfall gefordert, dass die Grammatik so gestaltet sein soll, dass keine Ableitung das leere Wort ergibt ($\epsilon \notin \mathscr{L}(G)$).

4.3.2 Beispiel einer Grammatik

Als Beispiel wird für die folgenden drei Varianten einer Zusicherung eine mögliche Grammatik angegeben.

```
Auf dem Server cntsrv1 wurden alle nicht benötigten Dienste abgeschaltet.
Die Korrektheit der Dienstkonfiguration wurde mittels netstat-Befehl geprüft.
Die aktiven Dienste sind:
ssh auf Port 22
http auf Port 80
https auf Port 443

Auf dem Server cntsrv2 wurden alle nicht benötigten Dienste abgeschaltet.
Die Korrektheit der Dienstkonfiguration wurde mittels Portscan geprüft.
Die aktiven Dienste sind:
ssh auf Port 22
http auf Port 80
https auf Port 443

Auf dem Server mailsrv wurden alle nicht benötigten Dienste abgeschaltet.
Die Korrektheit der Dienstkonfiguration wurde mittels netstat-Befehl geprüft.
Die aktiven Dienste sind:
ssh auf Port 22
smtp auf Port 25
```

Im festen Grundaufbau der Aussage variieren Servername, Prüfmethode und die Liste mit den jeweiligen Diensten und Ports.

Eine entsprechende Grammatik fasst konstante Teile in Terminalsymbole zusammen. Dies können auch Teilsätze wie „Auf dem Server“ sein. Die Bezeichner der Nicht-Terminale werden dabei so gewählt, dass sie dem semantischen Gehalt des jeweils durch die Ersetzungsregel abgedeckten Teilstrings gerecht werden.

Der durch die Grammatik eingeräumte Freiheitsgrad ist gestaltbar. So könnte anstelle einer Liste von Servernamen auch ein Literal vom Typ STRING angegeben werden. In diesem Fall kann der Ersteller der Zusicherung einen beliebigen (in Hochkomma gefassten) Servernamen eintragen.

Das eingeführte Beispiel kann durch die in Tabelle 4.1 dargestellte Grammatik erzeugt werden.

Die Verwendung des $|$-Symbols dient der verkürzten Darstellung von Alternativen:

$$A \to B|C|D \equiv \begin{cases} A \to B \\ A \to C \\ A \to D \end{cases}$$

Die Quantoren $?$,$+$ und $*$ sind identisch zu denen der regulären Ausdrücke in Abschnitt 4.2.3.2 definiert. Die Zeichenfolge '\n' steht für einen Zeilenumbruch.

Somit stellt die Grammatik G eine Beschreibung für die erlaubten Zusicherungsformen dar. Alle erlaubten Zusicherungsformen bilden gemeinsam eine *Klasse von Zusicherungen*.

$$
\mathcal{P} = \left\{ \begin{array}{rcl}
\textit{Satz} & \rightarrow & \textit{Mn5\_72} \\
\textit{Mn5\_72} & \rightarrow & \text{'Auf dem Server '} \textit{Server} \text{ ' wurden alle nicht benötigten Dienste abgeschaltet.\textbackslash n'} \textit{Pruefung}? \text{ 'Die aktiven Dienste sind:\textbackslash n'} \textit{Dienste}? \\
\textit{Dienste} & \rightarrow & \textit{Dienst}+ \\
\textit{Dienst} & \rightarrow & \textit{Dienstname} \text{ ' auf Port '} \textit{Portnummer} \text{ '\textbackslash n'} \\
\textit{Portnummer} & \rightarrow & \textit{INT} \\
\textit{Dienstname} & \rightarrow & \text{'ssh' | 'dns' | 'smtp' | 'http' | 'https' | 'ntp'} \\
\textit{Pruefung} & \rightarrow & \text{'Die Korrektheit der Dienstkonfiguration wurde mittels '} \textit{Methode} \text{ ' geprüft.\textbackslash n'} \\
\textit{Methode} & \rightarrow & \text{'Portscan' | 'nmap-Befehl' | 'netstat-Befehl'} \\
\textit{Server} & \rightarrow & \text{'cntsrv1' | 'cntsrv2' | 'cntsrv3' | 'mailsrv'} \\
\textit{INT} & \rightarrow & \text{('0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9')+}
\end{array} \right\}
$$

$$
\mathcal{N} = \left\{ \begin{array}{l} \textit{Dienst, Dienste, Dienstname, INT, Mn5\_72, Methode, Portnummer, Pruefung,} \\ \textit{Satz, Server} \end{array} \right\}
$$

$$
\mathcal{T} = \left\{ \begin{array}{l} \text{'Auf dem Server ', 'Die Korrektheit der Dienstkonfiguration wurde mittels ',} \\ \text{'Die aktiven Dienste sind:\textbackslash n', 'Portscan', '\textbackslash n', ' auf Port ', 'cntsrv1', 'cntsrv2',} \\ \text{'cntsrv3', 'dns', ' geprüft.\textbackslash n', 'http', 'https', 'mailsrv', 'netstat-Befehl',} \\ \text{'nmap-Befehl', 'smtp', 'ssh', ' wurden alle nicht benötigten Dienste} \\ \text{abgeschaltet.\textbackslash n'} \end{array} \right\}
$$

$$
S = \textit{Satz}
$$

Tabelle 4.1: Kontextfreie Grammatik für Zusicherungen nach Grundschutz-Maßnahme M 5.72

4.3.3 Das Wortproblem

Das *Wortproblem* ist das Entscheidungsproblem, ob ein Wort w – i. S. einer Ableitung – zur Sprache $\mathscr{L}(G)$ einer Grammatik G gehört. Für die Aufgabenstellung Zusicherungen entsprechend einer vorgegebenen Syntax auszustellen, stellt sich dieses Entscheidungsproblem, wenn geprüft werden soll, ob eine gemachte Zusicherung aus dem durch die Grammatik vorgegebenen Sprachumfang stammt.

Um diese Prüfung durchzuführen, werden i. d. R. validierende Parser eingesetzt. Hierbei wird die zu prüfende Zeichenfolge (z. B. die gegebene Zusicherung) symbolweise auf Basis der Produktionsregeln der Grammatik analysiert.

Für die Ausführung des Parsings gibt es eine Reihe von Parsing-Algorithmen. Diese werden im Allgemeinen nach ihrer Arbeitsweise klassifiziert. Es werden die grundsätzlichen Verfahren Top-Down-Parser und Bottom-Up-Parser unterschieden. Typische Vertreter dieser Kategorien sind u. a. die LL- bzw. die LR-Parser. Die LL-Parser nach dem Top-Down-Prinzip parsen von *L*inks nach Rechts und führen dann eine *L*inksableitung durch, bei der die Abarbeitung der

Nicht-Terminale in den Produktionsregeln von der linken Seite her erfolgt. Die LR-Parser führen als Bottom-Up-Parser eine *R*echtsreduktion durch, d. h. sie arbeiten die Produktionsregeln von der rechten Seite her ab und ersetzen Symbolsequenzen durch Nicht-Terminale bis hin zum Startsymbol [GJ90].

Auf die konkrete Auswahl eines Parsing-Verfahrens und die daraus resultierenden Anforderungen an die Grammatik wird in Kapitel 5 eingegangen.

Bei der Prüfung der Zugehörigkeit einer Zeichenfolge w zum Sprachumfang $\mathscr{L}(G)$ einer Grammatik G werden, sofern $w \in \mathscr{L}(G)$ gilt, die Entscheidungen des Erstellers der Zeichenfolge nachvollzogen. Die jeweils angewendeten Produktionsregeln können dann in einer Baumstruktur dokumentiert werden, dem sog. Syntaxbaum.

Für die einfache Grammatik

$$\begin{array}{lcl} S & \rightarrow & X \mid bY \\ X & \rightarrow & aXY \mid aY \\ Y & \rightarrow & bY \mid c \end{array}$$

ergibt sich für die Eingangsfolge *aabcbbc* der Syntaxbaum nach Abbildung 4.2.

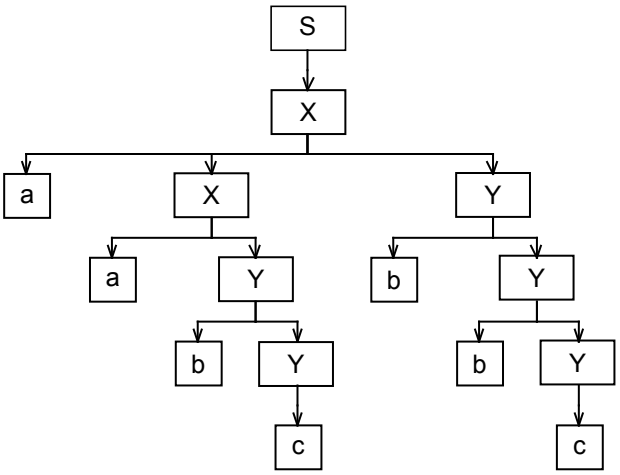


Abbildung 4.2: Beispiel eines einfachen Syntaxbaumes

Durch eine Tiefensuche, die die Blätter des Baumes von links nach rechts durchläuft, kann die erzeugende Eingangsfolge abgelesen werden. Ein Syntaxbaum, der direkt aus dem Parse-Vorgang gewonnen wird und alle dabei genutzten Token bzw. Zeichen enthält, wird als *konkreter Syntaxbaum* bezeichnet.

Wird eine Grammatik G zur Beschreibung einer Klasse von Zusicherungen verwendet, so sollen die Elemente $w \in \mathscr{L}(G)$ als *gültige Zusicherungen über G* bezeichnet werden.

4.3.4 Syntaxbaum als strukturierte Darstellung

Ein konkreter Syntaxbaum entspricht – bei geeigneter Wahl der Grammatik – der in Abschnitt 4.1.2 vorgeschlagenen strukturierten Darstellung einer Zusicherung.

Ein möglicher Syntaxbaum für die bereits auf Seite 72 dargestellte Grammatik kann Abbildung 4.3 entnommen werden.

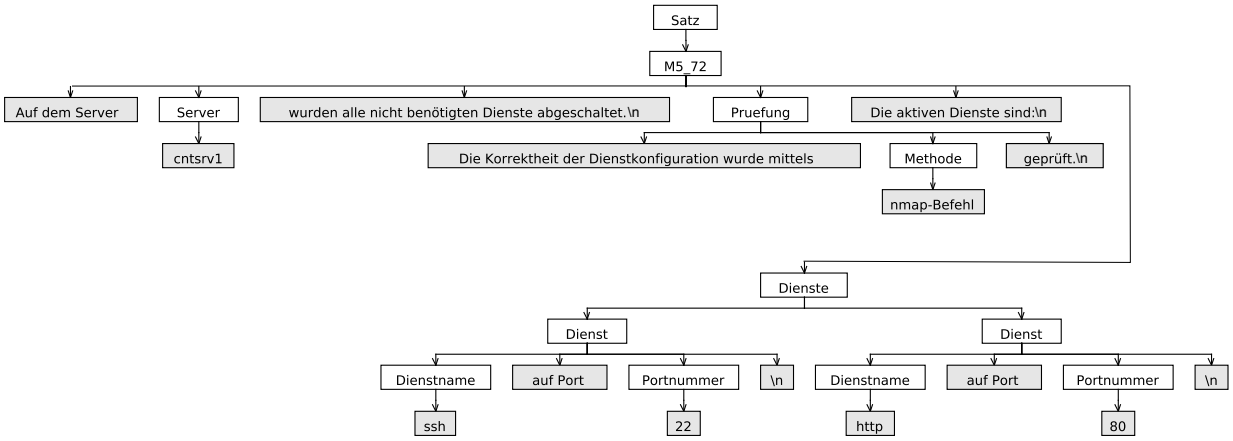


Abbildung 4.3: Konkreter Syntaxbaum

Das Nicht-Terminal *INT* war bei der Erstellung des Syntaxbaumes nach Abbildung 4.3 als sog. *Lexem* deklariert. Dadurch wird kein *INT*-Knoten eingefügt, sondern die durch die *INT*-Produktionsregel abgedeckten Zeichen werden als eine zusammengefasste Zeichenkette in den Baum aufgenommen.

Der Informationsgehalt des Syntaxbaumes umfasst somit zum einen den ursprünglichen Text der Zusicherung. Zusätzlich ist diese Information noch durch die Zuordnung zu den einzelnen Knoten strukturiert. Aus der Strukturinformation der Knoten lässt sich den verschiedenen Teilen der Zusicherung (die jeweils durch die Teilbäume gebildet werden) eine entsprechende Knotenbeschriftung und damit ggf. eine Bedeutung zuordnen. Die zur Kodierung des Syntaxbaumes benötigte Datenmenge ist jedoch auch entsprechend größer als der ursprüngliche Primärtext.

Bei der automatischen Verarbeitung eines solchen Syntaxbaumes kann, ausgehend vom Wurzelsymbol, ein Teilbaum extrahiert werden. Soll beispielsweise die Prüfmethode extrahiert werden, so wird der Teilbaum *Satz/Mn5_72-Pruefung/Methode* betrachtet.

4.3.5 Optimierung durch abstrakte Syntaxbäume

Der konkrete Syntaxbaum enthält alle Terminale, die in den bei der Analyse angewendeten Produktionsregeln enthalten sind. Liegen jedoch sowohl der Syntaxbaum als auch die zugehörige

Grammatik vor, so ist die Angabe etlicher Terminale redundant. Ein Beispiel einer redundanten Angabe ist das Terminal „ auf Port “ im Knoten bzw. der Produktionsregel *Dienst*. Aus der Grammatik geht hervor, dass dieser Bestandteil immer an dieser Stelle enthalten sein muss.

Diese bei Programmiersprachen häufig auch als „syntaktischer Zucker“ bezeichneten Teile sind zwar in der primären Darstellungsform für die Lesbarkeit und die Verständlichkeit notwendig, werden jedoch in der sekundären Form für eine Auswertung der Daten und für eine Rückgewinnung des Textes nicht zwangsläufig gebraucht.

Daher können alle Terminale, für die keine Wahlmöglichkeiten vorgesehen sind, aus dem Syntaxbaum entfernt werden, da sie keine Information (bei vorhandener Grammatik) tragen. Der entsprechend optimierte sog. *abstrakte Syntaxbaum* (engl. abstract syntax tree (AST)), der dem Beispiel aus Abbildung 4.3 entspricht, ist in Abbildung 4.4 zu sehen.

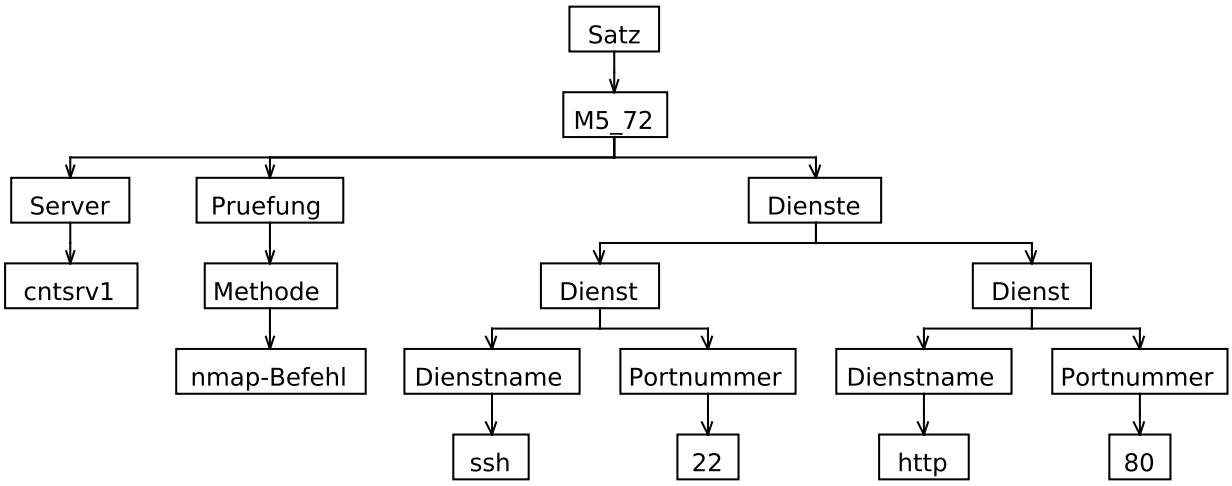


Abbildung 4.4: Abstrakter Syntaxbaum

Um genau bestimmen zu können, was unter einem redundanten Terminal zu verstehen ist, sollen zunächst einige Festlegungen getroffen werden.

Für die betrachteten Grammatiken sollen folgende Eigenschaften gegeben sein:

- ❑ Die Grammatik liegt ohne Klammern vor.
- ❑ Quantoren $(?, +, *)$ werden nur auf Nicht-Terminale angewendet.
- ❑ Die Grammatik G ist eindeutig.

Da Klammern beim Analysevorgang zu implizit definierten Knoten im Syntaxbaum führen, ist die Entfernung durch Ersetzen jedes geklammerten Ausdrucks durch ein neues Nicht-Terminal sinnvoll. Dadurch erhalten die zugehörigen Knoten im Syntaxbaum einen explizit gewählten Bezeichner und können in späteren Verarbeitungsschritten besser referenziert werden.

Werden Quantoren $(?, +, *)$ auf Terminale angewendet, so kann ein solches Terminal nicht aus dem Syntaxbaum entfernt werden, da sein Vorhandensein einen Informationsgehalt hinsichtlich der Wahlmöglichkeiten darstellt. Wird ein quantisiertes Terminal benötigt, so kann ein zusätzliches mit einem Quantor versehenes Nicht-Terminal definiert werden, das auf seiner rechten Seite lediglich aus dem zu quantisierenden Terminal besteht.

Die Eigenschaft der Eindeutigkeit einer Grammatik besagt, dass es zu jedem mit der Grammatik erzeugbaren Wort genau eine Ableitung gibt. Diese Eigenschaft ist für das korrekte Funktionieren eines Parsers notwendig und kann durch Tests nachgewiesen werden [Sch01].

Bei der Rückgewinnung müssen in jedem Knoten des AST genügend Informationen enthalten sein, um eindeutig die Produktionsregel zu identifizieren, aus der der Knoten des AST abgeleitet wurde.

Da der Knotenname dem Nicht-Terminal der linken Seite der verwendeten Produktionsregel entspricht, sind die unterhalb des Knotens liegenden Knoten bzw. Blätter nur für die Differenzierung innerhalb der Produktionsregeln *eines* Nicht-Terminals notwendig.

Die Größe des Entscheidungsraumes eines AST-Knotens entspricht der Anzahl der zugehörigen Produktionsregeln (hierbei werden über „|“ getrennte Regelalternativen als eigenständige Regeln gezählt). Für eine Grammatik $G = \langle \mathcal{N}, \mathcal{T}, \mathcal{P}, S \rangle$ und ein Nicht-Terminal $N \in \mathcal{N}$ ist die Menge der zugehörigen Produktionsregeln

$$\mathcal{P}_N = \{P | P \in \mathcal{P} \wedge \mathrm{LS}(P) = N\}$$

Die Funktion LS liefert das Nicht-Terminal der linken Seite der Produktion. Die Größe des zugehörigen Entscheidungsraumes ist $|\mathcal{P}_N|$. Alle Produktionsregeln aus $\mathcal{P}_N$ haben paarweise disjunkte rechte Seiten. Werden nun aus den Produktionsregeln aus $\mathcal{P}_N$ alle Terminale entfernt, so entsteht eine neue Menge $\mathcal{P}'_N$.

Wenn nach dem Entfernen der Terminale zwei oder mehr der neu entstandenen Produktionsregeln die gleiche rechte Seite aufweisen, so sind diese Regeln nicht mehr unterscheidbar. Somit ist $|\mathcal{P}'_N| < |\mathcal{P}_N|$. Ist jedoch $|\mathcal{P}'_N| = |\mathcal{P}_N|$, so ist der Entscheidungsraum auch nach Streichung der Terminale gleich groß. Die Speicherung der Terminale im AST ist somit redundant und sie können daher aus dem AST entfernt werden.

4.3.6 Rückgewinnung

Die Transformation, die einen Primärtext p mittels einer Grammatik G in einen konkreten und durch Vereinfachung in einen abstrakten Syntaxbaum X transformiert, wird im Folgenden durch Φ_G beschrieben. Dabei steht das Symbol $\perp$ für „falsch“ oder „ungültig“.

$$\Phi_G(p) = \begin{cases} X & \text{wenn } p \in \mathscr{L}(G) \\ \perp & \text{sonst} \end{cases} \tag{4.1}$$

$$\Phi_G^{-1}(X) = \begin{cases} p & \text{wenn } X = \Phi_G(p), X \neq \perp \\ \perp & \text{sonst} \end{cases} \tag{4.2}$$

Damit übernimmt Φ_G sowohl die Prüfung des Wortproblems als auch die Kodierung der strukturierten Daten. Die Umkehrfunktion Φ_G^{-1} soll als *Rückgewinnung* des Primärtextes bezeichnet werden.

Bei der Rückgewinnung kann wie bei der Tiefensuche zur Ermittlung des ursprünglichen Textes auf dem konkreten Syntaxbaum vorgegangen werden. Allerdings ist hierzu ein sog. Baumparser notwendig, der bei der Bearbeitung eines Knotens jedes aufgrund der Redundanz zuvor entfernte Terminal wieder an der richtigen Stelle einfügt.

Wird z. B. einer der „Dienst“-Knoten aus Abbildung 4.4 bearbeitet, so hat dieser lediglich die Kindknoten „Dienstname“ und „Portnummer“. Ein Vergleich mit Abbildung 4.3 bzw. der zugehörigen Produktionsregel zeigt, dass nach Auswertung des ersten Kindknotens „Dienstname“ das Terminal „ auf Port “ und nach dem letzten Knoten das Terminal „\n“ ergänzt werden muss.

Der Vergleich des konkreten Syntaxbaumes in Abbildung 4.3 mit der abstrakten Variante in Abbildung 4.4 zeigt außerdem, dass die Abstraktion in Anwendungsfällen mit Terminalen, die ganze Textpassagen enthalten, zu einer erheblich kürzeren Kodierung führen kann.

Bei größeren Mengen zu speichernder Zusicherungen ist damit die Kodierung einer Zusicherung in der sekundären Form auch dann sinnvoll, wenn keine Verarbeitung der strukturierten Daten erfolgt, sondern ggf. nur der Primärtext zurückgewonnen werden soll.

4.4 Entwurf grammatikbasierter Zertifikate

Für die geschützte, vertrauenswürdige Übertragung von Zusicherungen existieren bereits etliche Verfahren für spezielle Anwendungsfälle (vgl. Kapitel 2). Daher ist hier kein neuer Entwurf notwendig und es wird für das Grundgerüst der grammatikbasierten Zertifikate (Grammar-based Certificates, GC) auf den bewährten Mechanismus der X.509-Zertifikate zurückgegriffen. Damit können Verarbeitungsschritte wie Signaturprüfung, Abruf von Sperrinformation und die Etablierung öffentlicher Schlüssel als gegeben vorausgesetzt werden.

Im Weiteren wird grundsätzlich angenommen, dass Bestätiger von Zusicherungen über geeignetes asymmetrisches Schlüsselmaterial und zugehörige gültige Schlüsselzertifikate (z. B. nach X.509) verfügen. Ebenso sollen bei den Empfängern von signierten Dokumenten die jeweiligen Zertifikate der Bestätiger bekannt und die zugehörige CA-Hierarchie nach Abschnitt 2.5.5 als vertrauenswürdig eingestuft sein.

4.4.1 Aufbau grammatikbasierter Zertifikate

Das GC erhält die gleiche Grundstruktur wie die X.509-Zertifikate, die jedoch noch um eine Kennung *gcertIdentifier* für grammatikbasierte Zertifikate erweitert wird. Tabelle 4.2 zeigt die verwendeten Datenstrukturen und ihre Bezeichner, die denen eines Schlüsselzertifikates nach [CSF+08] entsprechen.

Tabelle 4.3 zeigt den Vergleich der Felder von X.509-Zertifikat und GC.

X.509	GC
—	gcertIdentifier
tbsCertificate	tbsGCertificate
signatureAlgorithm	signatureAlgorithm
signatureValue	signatureValue

Tabelle 4.2: Grundstruktur X.509-Zertifikate und GC

X.509	GC
version	Version
serialNumber	SerialNumber
signature	Signature
issuer	Issuer
validity	Validity
subject	–
subjectPublicKeyInfo	–
–	GrammarReference
–	Assertion
	ParseTree
	PrimarySignature
–	Nonce
issuerUniqueID	–
subjectUniqueID	–
extensions	Extensions
authorityKeyIdentifier	AuthorityKeyIdentifier
x509CRLDistributionPoints	CRLDistributionPoint
authorityInformationAccess	AuthorityInformationAccess

Tabelle 4.3: Felder X.509-Zertifikate und GC

Das eigentliche Nutzdatum der X.509-Zertifikate, die Zuordnung eines Subjektes zu einem öffentlichen Schlüssel, entfällt; damit entfallen auch die Felder *subject* und *subjectPublicKeyInfo*. Dafür erhält das GC die Zusicherung (*Assertion*) mit dem abstrakten Syntaxbaum (*ParseTree*), also die kodierte Form der Zusicherung, sowie eine zur Interpretation des Syntaxbaumes notwendige Referenz auf die zugrunde liegende Grammatik (*GrammarReference*).

Die Grammatikreferenz besteht aus einer Kennung des verwendeten Hashalgorithmus, dem eigentlichen Hashwert über die Grammatikdatei und einem optionalen URI, der den Ort und das Zugriffsprotokoll zum Abruf der Grammatikdatei bezeichnet. Alternativ zum URI bzw. zusätzlich kann die Adresse eines speziellen Grammatikservers angegeben werden, über den die Grammatik ebenfalls abgerufen werden kann.

Grammatikbasierte Zertifikate können bzw. müssen wie X.509-Zertifikate durch Erweiterungen (Extensions) ergänzt werden. Wird dem Aussteller einer Zusicherung bei Sicherheitsüberprüfungen eine geeignete Infrastruktur zur Verfügung gestellt, so kann dieser – wie eine CA – die von

ihm gemachten Zusicherungen ggf. entsprechend der Prinzipien aus Abschnitt 2.5.6 widerrufen.

Für den betrachteten Anwendungsfall der Sicherheitsauditierungen wird ein zentraler Auditierungsserver (ZAS) vorgesehen, auf dem alle Zusicherungen und ggf. ihre Rückrufe zusammengeführt werden.

In den Erweiterungen des GC werden daher, neben der Schlüsselkennung des Ausstellers (authority key identifier), auch ein optionaler Verteilpunkt für Sperrlisten *CRL Distribution Point* und ein optionaler Verweis auf einen Statusauskunftsdienst *Authority Information Access* für GC vorgesehen.

Der ebenfalls neu eingeführte optionale Zufallswert *Nonce* kann zur Überprüfung der Frische der Zusicherung eingesetzt werden. Zusätzlich wird zu einer Zusicherung eine Struktur zur Aufnahme einer Signatur über den Primärtext vorgesehen (*PrimarySignature*).

Diese im GC enthaltenen Informationen, deren Kodierung als vollständiges Zertifikat noch in Kapitel 5 beschrieben wird, haben das Ziel, den Akzeptor von der Korrektheit der Zertifikatsaussage zu überzeugen. Hierbei liefern die Signaturen bereits die *Authentizität* der Nachricht. Darüber hinaus müssen aber auch die *Aktualität* der Zusicherung und die Erfüllung der Formvorgaben betrachtet werden. Diese Aspekte und die verschiedenen Varianten, die Informationen Primärtext und strukturierte Daten und ihre jeweiligen Signaturen zu nutzen, werden in den folgenden Abschnitten betrachtet.

4.4.2 Aktualität der Zusicherung

Die *Aktualität der Zusicherung* soll der in Abschnitt 3.1.5 im Rahmen der Erweiterung der SVO-Logik gegebenen Definition entsprechen. Zur Prüfung dieser Eigenschaft können drei Mechanismen eingesetzt werden.

Zum Ersten ist die Verwendung des optionalen Nonce-Parameters möglich, der dem Ersteller im Rahmen einer Anforderung einer Zusicherung mitgeteilt und von diesem im erstellten GC aufgenommen wird.

Zum Zweiten besteht die Möglichkeit, wie im Fall der X.509-Zertifikate einen Statusauskunftsdienst oder Rückruflisten einzusetzen. Ein solcher Dienst sichert die Aktualität i. S. der eingeführten *current*-Aussage in Bezug auf einen Aussteller und die Seriennummer eines GC zu und ermöglicht, ein einmal ausgestelltes Zertifikat als frisch zu akzeptieren.

Die dritte Möglichkeit besteht darin, im Gültigkeitsintervall eine so kurze Gültigkeit zuzusichern, dass weder ein Wiedereinspielen kritisch ist noch die zugesicherte Information innerhalb des Intervalls widerrufen werden müsste.

4.4.3 Prüfung von Formerfüllung

Eine Zusicherung im Sinne grammatikbasierter Zertifikate erfüllt definitionsgemäß nach (4.1) und (4.2) die Formerfordernis, wenn der Primärtext $p \in \mathscr{L}(G)$ erfüllt und somit ein strukturiertes Datum $X = \Phi_G(p)$ existiert.

Durch Ausführen von $\Phi_G(p)$ oder $\Phi_G^{-1}(X)$ lassen sich der Primärtext p bzw. das strukturierte Datum X im Hinblick auf die Erfüllung der korrekten Form validieren. Daher muss beim Anwender auch im Fall der Signatur und Übertragung des Primärtextes die Grammatik G zur Durchführung der Formprüfung mittels $\Phi_G(p)$ vorliegen.

4.4.4 Signaturvarianten

Die GC enthalten neben dem Feld zur Signatur der Zertifikatsdaten, die den AST enthalten, auch die Möglichkeit eine Signatur des Primärtextes zu übertragen. Damit können verschiedene Signaturvarianten unterschieden werden. Die Signaturfunktion kann auf Primärtexte, auf die daraus abgeleiteten strukturierten Daten oder auf beide als *kombiniertes Verfahren* angewendet werden. Die Signatur des Primärtextes soll als *Primärsignatur*, die der strukturierten Daten als *Sekundärsignatur* bezeichnet werden.

Da Primärtext und strukturierte Daten voneinander abgeleitet werden können, können statt eines umfangreichen signierten Primärtextes auch die strukturierten Daten gemeinsam mit der Primärsignatur übertragen werden. Dieses Vorgehen soll als *hybrides Verfahren* bezeichnet werden.

In dem folgenden Teilkapitel werden die verschiedenen Verfahren erläutert. Eine formale Beschreibung der Mechanismen und ein bewertender Vergleich erfolgt in Kapitel 5.3.

4.4.4.1 Signierter Primärtext

Formuliert ein Bestätiger B eine Zusicherung, so kann diese in ihrer primären Form p mit seinem privaten Schlüssel K_B^{-1} signiert und die entstehende Primärsignatur an den Anwender A übertragen werden.

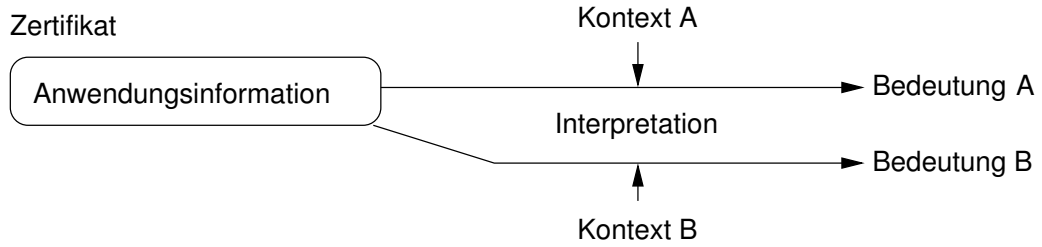
$$B \mapsto A : \lfloor p \rfloor_{K_B^{-1}}$$

Dies stellt die einfachste Form der Zusicherung dar und entspricht z. B. der Vorgehensweise bei der Signatur von E-Mails mit Zusicherungen. Die Betrachtung der allein stehenden Primärsignatur wird der Vollständigkeit halber durchgeführt. Ihre zusätzliche Bedeutung erhält die Primärsignatur mit dem unten beschriebenen hybriden Verfahren.

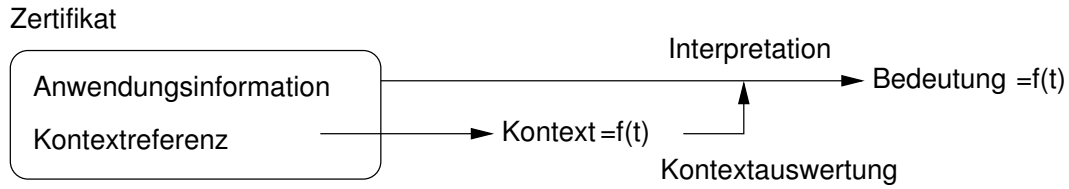
4.4.4.2 Signierte strukturierte Daten

Diese Variante ist die Hauptform der grammatikbasierten Zertifikate. Hierbei wird statt des Primärtextes die strukturierte Kodierungsform, also der abstrakte Syntaxbaum, signiert übertragen. So müssen zwei Aspekte genauer betrachtet werden.

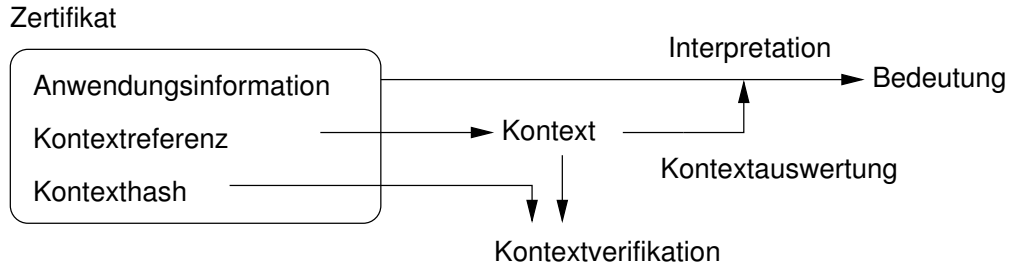
Erstens hängt die Interpretation der strukturierten Daten von der für die Rückgewinnung verwendeten Grammatik ab. Die Grammatik bildet den *Interpretationskontext* für die im Zertifikat enthaltene Zusicherung. Somit genügt es nicht, lediglich den URI für die Grammatik zu hinterlegen, da die dadurch implizierte Bedeutungsinterpretation durch Veränderung der hinterlegten Grammatik unbemerkt verändert werden könnte. Das Problem nicht festgelegter Kontextinformation wird in Abbildung 4.5(a) verdeutlicht.



(a) Zertifikat ohne Kontextvorgabe



(b) Zertifikat mit Kontextreferenz



(c) Zertifikat mit Referenz und Hashwert

Abbildung 4.5: Unterschiedlich starke Kontextbindung

Auch eine konkrete Adressierung einer Kontextinformation ist nicht ausreichend, sofern diese nicht über die Zeit konstant sondern variabel ist, da in diesem Fall die Interpretation ebenfalls einer zeitlichen Variabilität unterworfen sein kann. Diesen Effekt verdeutlicht Abbildung 4.5(b).

Um diese Mehrdeutigkeit zu vermeiden, muss die zur Interpretation des Zertifikatsinhalts benötigte Grammatik innerhalb der signierten Daten eindeutig referenziert werden (vgl. Abbildung 4.5(c)). Dies kann durch die Einbettung des Hashwertes der Grammatik in die Signatur erfolgen.

Der Kern einer Zusicherung p gemäß Grammatik G auf Basis der strukturierten Daten X sieht daher wie folgt aus:

$$B \mapsto A : \lfloor X, \mathrm{H}(G), \ldots \rfloor_{K_B^{-1}}$$

wobei gelten muss, dass $X = \Phi_G(p)$ ist.

Der zweite wichtige Aspekt ist die Festlegung der grundsätzlichen Interpretation eines Syntaxbaumes bei gleichzeitiger Angabe des Hashwertes einer Grammatik. Hierzu wird eine Definition aufgestellt, deren Akzeptanz Grundlage für die Nutzung grammatikbasierter Zertifikate ist.

Wird ein strukturiertes Datum X im Kontext eines grammatikbasierten Zertifikats zusammen mit einem Hashwert einer Grammatik G, mit der aus einem Primärtext p das strukturierte Datum X erzeugt werden kann, geäußert, so ist dies gleichbedeutend mit dem Äußern von p.

Nur in Anwendungsfällen, in denen dieser Zusammenhang durch einen Akzeptor eines GC als gültig erachtet wird, kann er durch das GC von der Korrektheit des Primärtextes überzeugt werden. Die Akzeptanz kann entfallen, wenn die signierten strukturierten Daten als Bestätigung ausreichen. In einem solchen Fall kann der Zusammenhang mit einem Primärtext über $p = \Phi_G^{-1}(X)$ dennoch für eine Berechtigungsprüfung genutzt werden und bei der Erstellung der Zusicherung zur Erhöhung des Verständnisses beim Ersteller dienen.

4.4.4.3 Hybrides Verfahren

Es ist auch möglich, bei Erstellung einer Signatur über den Primärtext (der ggf. auch die Nonce enthält) statt des signierten Primärtextes die strukturierten Daten zu übertragen, um beispielsweise die Datenmenge zu reduzieren.

$$B \mapsto A : \lfloor X, \ldots \rfloor_{K_B^{-1}}, \mathrm{H}(G)$$

In diesem Fall muss ebenfalls eine Kennung der Grammatik übertragen werden. Vor der Signaturprüfung ist die Rückgewinnung der Nachricht durchzuführen. Erst danach kann die eigentliche Signaturprüfung erfolgen, in die dann der rückgewonnene Primärtext eingeht.

Der Hashwert der Grammatik muss bei diesem Verfahren nicht übersigniert werden, da im Fall einer manipulierten Grammatik zwar ein anderer Primärtext erzeugt werden könnte, dieser jedoch beim Vergleich der Hashwerte mit der Primärsignatur auffällt.

4.4.4.4 Kombiniertes Verfahren

Beim kombinierten Verfahren werden sowohl die Primärsignatur als auch die Sekundärsignatur übertragen.

$$B \mapsto A : \lfloor p, \ldots \rfloor_{K_B^{-1}}, \lfloor X, \mathrm{H}(G), \ldots \rfloor_{K_B^{-1}}$$

Dieser Modus ist z. B. dann sinnvoll, wenn beispielsweise der Primärtext mit einer qualifizierten Signatur versehen wird, um Rechtsverbindlichkeit nach Signaturgesetz zu erreichen, die Absicherung der strukturierten Daten jedoch mittels einer fortgeschrittenen Signatur erfolgt.

4.5 Gesamtarchitektur

Um auf Basis der definierten grammatikbasierten Zertifikate die funktionalen Anforderungen aus Kapitel 1 realisieren zu können, werden weitere Mechanismen benötigt, deren Zusammenspiel in Abbildung 4.6 gezeigt wird.

Wenn durch eine geplante Auditierung Bedarf an Zusicherungen entsteht, werden – sofern noch nicht vorhanden – eine oder mehrere entsprechende Grammatiken erstellt, die i. d. R. einen größeren Sprachumfang als den konkret benötigten beschreiben.

Da die Grammatiken sowohl bei der Erstellung als auch bei der Prüfung an unterschiedlichen Orten und zu unterschiedlichen Zeitpunkten benötigt werden und auch für spätere Anwendungsfälle erneut genutzt werden können, wird zur Speicherung und Verteilung der Grammatiken ein zentraler Server vorgesehen, über den die Grammatik durch Indizierung über den Grammatik-Hashwert abgerufen werden kann.

In den meisten Fällen, in denen Zusicherungen gemäß einer Grammatik angefordert und automatisch ausgewertet werden, umfassen die tatsächlich benötigten Zusicherungen lediglich eine Teilmenge des Sprachumfangs der Grammatik. Entsprechend müssen diese Teilmengen als Anforderungen spezifiziert und dem Aussteller und Prüfer der Zusicherungen bekannt gemacht werden. Die so kodierten Anforderungen stellen somit die elektronische Entsprechung zu einem Prüfkatalog gemäß Kapitel 1.3.4 dar.

Ein weiterer Mechanismus, bei dem die Definition von Teilmengen des Sprachumfangs einer Grammatik benötigt wird, ist die Vergabe von Berechtigungen zur Ausstellung einer Zusicherung (vgl. Abschnitt 1.3.5). Dabei kann ein Berechtigter die erhaltene Berechtigung ggf. an weitere Personen delegieren, so dass eine Delegationskette entstehen kann. Die Nachweise über diese Delegationsschritte werden in Form von Delegationszertifikaten prüfbar dokumentiert und müssen bei der Analyse eingehender Zusicherungen ebenfalls ausgewertet werden. Daher müssen sie an die prüfende Instanz (hier der ZAS) übermittelt werden.

Schließlich werden die Zusicherungen auf Basis der Grammatik erstellt und als grammatikbasiertes Zertifikat kodiert und signiert. Nach der Übermittlung findet zunächst eine Prüfung des Zertifikats (Signatur, Gültigkeitsdauer, Sperrinformationen) statt. Danach kann mit Hilfe der

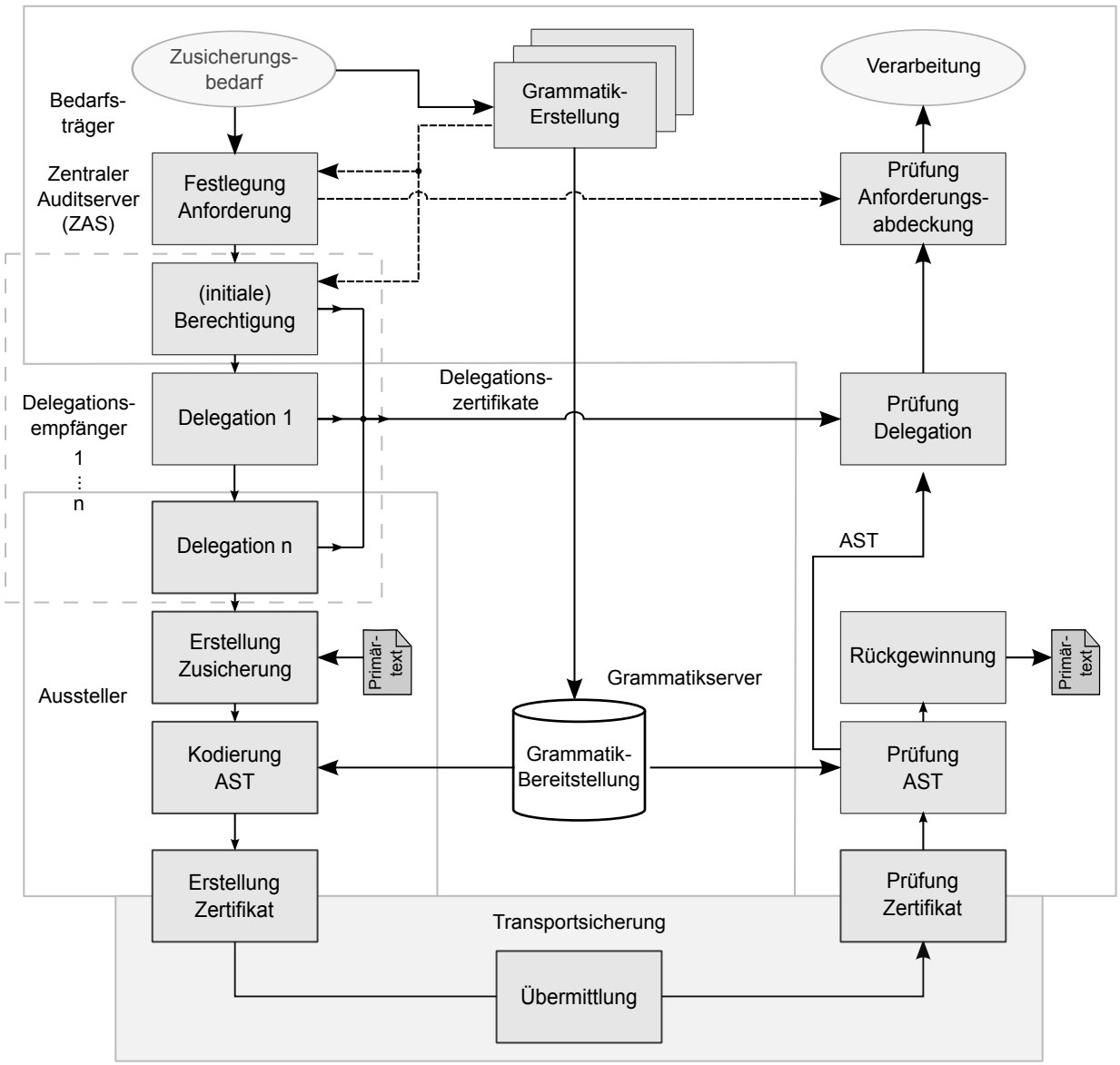


Abbildung 4.6: Architektur

Grammatik geprüft werden, ob der übertragene AST korrekt ist. Durch diese Prüfung kann auch der Primärtext zurückgewonnen werden.

Der AST bildet auch die Grundlage für die Prüfung der Berechtigung auf Basis der durch Delegationszertifikate belegten Delegationskette, die ggf. noch Beschränkungen der Berechtigungen enthalten kann.

Den Abschluss bildet die Prüfung gegen die anfangs festgelegten Anforderungen. Werden diese vollständig durch die eingegangenen Zusicherungen abgedeckt, so ist der Bedarf des Audits gedeckt.

4.6 Festlegung von Anforderungen

Im Prüfkatalog eines Audits sind die Anforderungen durchzuführende Prüfungen, deren (i. d. R. positives) Ergebnis durch den Prüfer zugesichert werden soll. Diese Prüfvorschrift wird häufig schriftlich vorgehalten. Ein Beispiel, das eine Prüfung der physikalischen Sicherheit von Kryptomodulen nach der Maßnahme M-4.87 des BSI-Grundschutzkataloges etwas verkürzt[1] beschreibt, lautet:

> Für die Prüfung von Hochsicherheitsmodulen sind folgende Prüfschritte durchzuführen:
>
> Der HSM-Typ (Kryptoserver oder Modul), der Host-Name, sowie der Aufstellungsort (Serverschrank) sind zu dokumentieren.
>
> Die für das HSM geltenden Zulassungen sind zu erheben und der Fortbestand der Gültigkeit ist zu prüfen. Relevante Zulassungen sind
>
> - ❑ FIPS 140-2 Level 3
> - ❑ FIPS 140-2 Level 4 Phys. Security
> - ❑ Common Criteria EAL 4+
> - ❑ ZKA-Zulassung
>
> Der Versionsstand der Firmware-Module ist zu erheben, zu dokumentieren und mit dem in den Zulassungen vorgegebenen Stand zu vergleichen und die Übereinstimmung festzustellen.
>
> Bei Abweichungen der Firmware-Module (Art oder Versionsstand) ist dies sowie ggf. das Vorliegen einer passenden Herstellererklärung zu dokumentieren.
>
> Die Unversehrtheit der Siegel an nicht genutzten Schnittstellen ist zu prüfen und zu dokumentieren.

Zwei Zusicherungen entsprechend dieser Vorschrift wurden in Abschnitt 4.1.1 auf Seite 64 gezeigt. Für die zweite Zusicherung ist auf Seite 65 die zugehörige XML-Darstellung der strukturierten Daten angeben.

Um den konkreten Prüfbedarf auszudrücken, sei eine Prüfung nach oben angegebener Vorschrift mit dem Hinweis auf das zu prüfende System „nciphercrypt1“ angefordert.

Zur automatisierten Prüfung der Anforderungen ist es notwendig, den gewünschten Zusicherungsgehalt formalisiert auszudrücken. Für das genannte Beispiel wären dies z. B. der Name des HSM-Moduls, die erforderliche ZKA-Zulassung und ungebrochene Siegel. Bezüglich der

[1] Zur besseren Übersichtlichkeit wurde u. a. auf Typenbezeichnungen, Seriennummern, etc. verzichtet und Formulierungen gekürzt.

Firmware-Module sei wahlweise ein unverändertes System oder das Vorliegen einer Herstellererklärung zulässig.

Bei entsprechend weit gefassten $\mathscr{L}(G)$ lassen sich Zusicherungen abweichend von dieser Anforderung abgeben. So weisen die gezeigten XML-Daten auf Seite 65 zwar veränderte Firmware-Module, jedoch nicht das Vorliegen einer Herstellererklärung aus. Diese Abweichung von den geforderten Eigenschaften soll automatisiert erkannt werden können.

Um Beschränkungen auszudrücken, kann an allen drei wesentlichen Bestandteilen der Mechanismen der grammatikbasierten Zertifikate angesetzt werden: Primärtext, Grammatik und strukturierte Daten. Die beiden erstgenannten Verfahren werden nur kurz vorgestellt, da die Beschränkung der strukturierten Daten die Möglichkeiten dieser beiden Verfahren mit abdecken kann und daher als Basis für die benötigten Beschränkungs- und Delegationsmechanismen ausgewählt wurde.

4.6.1 Beschränkung des Primärtextes

Eine über den durch die Grammatik bestimmten Rahmen hinausgehende Beschränkung eines Primärtextes p ist – wie auch die Prüfung, ob $p \in \mathscr{L}(G)$ – eine Eigenschaftsprüfung und ist daher parallel zu der bereits eingeführten Beschränkung durch die Grammatik zu sehen.

Somit kann zunächst eine allgemeine auf den Primärtext bezogene Eigenschaftsprüfung ρ_p definiert werden.

Die Prüffunktion ρ_p liefert als Ergebnis einen Wahrheitswert aus $(\mathbf{T}, \mathbf{F})$ und hat zwei Argumente: den Primärtext p und einen Ausdruck r, der eine Eigenschaft beschreibt, die p erfüllen muss.

Somit gilt

$$\rho_p(p, r) = \mathbf{T}, \text{ wenn } p \text{ die Eigenschaft } r \text{ erfüllt.}$$
$$\rho_p(p, r) = \mathbf{F}, \text{ wenn } p \text{ die Eigenschaft } r \textit{ nicht} \text{ erfüllt.}$$

Die Realisierung einer solchen Prüffunktion kann u. a. beliebige Grammatiken wie z. B. reguläre Ausdrücke einsetzen. Das bereits eingeführte Prädikat $p \in \mathscr{L}(G)$ zur Prüfung von p gegen die Grammatik ist eine solche Prüffunktion. Da dieser Mechanismus jedoch nicht zweifach angewendet werden soll, wird dieser Ansatz nicht weiter betrachtet.

4.6.2 Beschränkung der Grammatik

Eine Alternative zur Prüfung des Primärtextes ist die Beschreibung einer angepassten Grammatik G' durch Definition von Filterregeln ρ_g, die auf die Produktionsregeln der Basisgrammatik G angewendet werden.

Hierbei können für Nicht-Terminale mit $n > 1$ Produktionen bis zu $n - 1$ Produktionen gestrichen werden. Desweiteren können nach vorhergehenden Streichungen alle Regeln für die Nicht-Terminale gestrichen werden, die in keiner Regel auf der rechten Seite enthalten sind.

Als Eigenschaftsprüfung ρ_g werden Prädikate über die Produktionsregeln definiert, die in Analogie zu den Regeln für Primärtexte alle erfüllt sein müssen.

$$\mathcal{P}' = \{P | P \in \mathcal{P} \wedge (\bigwedge_{\nu=1}^{n} \rho_{g,\nu}(P, r_\nu))\} \tag{4.3}$$

Beispiel: Aus der einfachen Grammatik G

$$\begin{array}{lcl} S & \rightarrow & X \mid bY \\ X & \rightarrow & aXY \mid aY \mid daZ \\ Y & \rightarrow & bY \mid c \\ Z & \rightarrow & bd \mid dd \end{array}$$

soll die Regel $X \rightarrow daZ$ gestrichen werden. Dann können auch die Regeln $Z \rightarrow bd$ und $Z \rightarrow dd$ gestrichen werden, da diese nicht mehr erreichbar sind.

Die zugehörigen Beschränkungen können z. B. als

$$\rho_{g,1} : \mathrm{RS}(P) \neq daZ$$
$$\rho_{g,2} : \mathrm{LS}(P) \neq Z$$

geschrieben werden.

Die Sprache der daraus resultierenden Grammatik G' ist eine Untermenge der Sprache von G, also $\mathscr{L}(G') \subseteq \mathscr{L}(G)$, da $\mathcal{P}_{G'} \subset \mathcal{P}_G$:

Für jede Ableitung $S \overset{P_a}{\Rightarrow}_{G'} w_0 \overset{P_b}{\Rightarrow}_{G'} \ldots \overset{P_x}{\Rightarrow}_{G'} w$ mit $P_a, P_b, \ldots, P_x \in \mathcal{P}'$ existiert eine identische Ableitung $S \overset{P_a}{\Rightarrow}_{G} w_0 \overset{P_b}{\Rightarrow}_{G} \ldots \overset{P_x}{\Rightarrow}_{G} w$, da alle $P_\nu \in \mathcal{P}'$ auch in $\mathcal{P}$ enthalten sind. Damit ist auch $\forall w' \exists w (w' \in \mathscr{L}(G') \wedge w \in \mathscr{L}(G) \wedge w' = w)$ erfüllt und damit schließlich $\mathscr{L}(G') \subseteq \mathscr{L}(G)$.

4.6.3 Beschränkung der strukturierten Daten

Der dritte Ansatzpunkt für die Definition von Beschränkungen ist der abstrakte Syntaxbaum. Wie bei Beschränkungen auf Basis der Grammatik sind Beschränkungen der beim Parsing-Vorgang entstehenden strukturierten Daten direkt mit den semantisch motivierten Knoten formulierbar.

4.6.3.1 XML-Kodierung der strukturierten Daten

Um konkrete Beschränkungsmöglichkeiten darstellen zu können, wird zunächst eine geeignete Repräsentationsform für strukturierte Daten festgelegt.

- Jede genutzte Produktionsregel, die auf der rechten Seite auch Nicht-Terminale enthält, wird durch einen mit dem linksseitigen Nicht-Terminal beschrifteten Knoten dargestellt, der als Subknoten die Knoten der auf der rechten Seite enthaltenen Nicht-Terminale enthält.

- Jede genutzte Produktionsregel, die auf der rechten Seite nur Terminalsymbole enthält, wird durch einen mit dem linksseitigen Nicht-Terminal beschrifteten Knoten dargestellt, der als Subknoten die nicht-redundanten Terminale enthält.
- Für Blätter des Baumes, die aus Lexemen gebildet werden, wird der Primärtext des jeweiligen Lexems direkt in den strukturierten Daten abgelegt.

Als textuelle Darstellungsform für die AST-Strukturen wird eine Darstellung in XML festgelegt. Hierbei ist jeder innere Knoten ein XML-Element, dessen Start- und Ende-Tag mit dem Nicht-Terminal als Tag-Namen versehen werden, wie dies bereits in den weiter oben gezeigten Beispielen vorgestellt wurde.

Bei Blättern des Baumes (Terminale und Lexeme) erfolgt die Darstellung durch den speziellen Elementtyp GCELEMENT. Dieser enthält ein Attribut *type*, das für Terminale den Wert *TOKEN* und für Lexeme den Lexem-Bezeichner (z. B. *STRING*, *INT*, *REAL*, usw.) enthält.

```
<Zertifizierung>
   <GCELEMENT type="TOKEN">FIPS 140-2 Level 3</GCELEMENT>
</Zertifizierung>
<Firmware>
   <FWModul>
      <GCELEMENT type="STRING">pin-1.0.2.mod</GCELEMENT>
   </FWModul>
   ...
</Firmware>
```

Durch die Attributierung können die als Text-Element gespeicherten Terminale und Lexeme wahlweise als Zeichenkette oder nach Typ-spezifischer Umwandlung z. B. als Ganzzahl verarbeitet werden.

4.6.3.2 XPath-Ausdrücke

Durch die Verwendung von XML kann für die weiteren Schritte der bewährte Mechanismus der XPath-Ausdrücke [BBC+07] zum Einsatz kommen. Mit XPath-Ausdrücken lassen sich unterschiedliche Teile eines XML-Dokumentes adressieren und durch die Verwendung von Funktionen auch Eigenschaftsprüfungen durchführen.

XPath-Ausdrücke sind den Pfadangaben unter Unix ähnlich. Die Basis der Ausdrücke sind durch Schrägstrich getrennte Bezeichner. Statt Verzeichnisse benennen die Bezeichner jedoch Knoten der XML-Struktur. Der Ausdruck /Satz/HSMSystem/Modul bezeichnet alle Modul-Elemente, die Unterknoten des Knotens HSMSystem sind, der wiederum ein Knoten unter dem obersten Knoten *Satz* ist. Positionsangaben können auch relativ sein. So steht die Verwendung eines doppelten Schrägstriches an erster Stelle für einen Knoten an einem beliebigen Punkt unterhalb des Startknotens.

Daneben können auch Eigenschaften von Knoten angegeben werden. Soll z. B. geprüft werden, ob für die Prüfung das nmap-Verfahren zugesichert wurde, so kann dies durch den Prüfausdruck
`//Pruefung/Methode/GCELEMENT[text()='nmap-Befehl']`
erfolgen. Der Ausdruck wird als „Die XML-Struktur enthält einen Knoten 'Pruefung' mit einem Unterknoten 'Methode' mit einem Unterknoten 'GCELEMENT', dessen Inhalt 'nmap-Befehl' ist." gelesen.

Ein anderes Beispiel ist die Prüfung, ob ein ssh-Dienst auch auf Port 22 konfiguriert ist:
`//Dienst[Dienstname/GCELEMENT[text()='ssh']]/Portnummer/GCELEMENT[text()=22]`

Für den vergleichenden Zugriff auf die Textteile in den Blättern des AST wird eine Kurzschreibweise eingeführt, mit der sich die XPath-Ausdrücke kompakter schreiben lassen.

Folgt nach einem XPath-Ausdruck eine in einfachen Hochkommata stehende Zeichenkette, so wird diese durch einen GCELEMENT-Knoten ersetzt, dessen Textinhalt mit der in Hochkommata stehenden Zeichenkette verglichen wird.

Beispiel: Der Ausdruck `//Dienst[Dienstname/'ssh']/Portnummer/'22'` ist die verkürzte Darstellung des oben angegebenen Beispiels zur Prüfung auf einen an Port 22 gebundenen ssh-Dienst. Durch diese Schreibweise wird eine verbesserte Lesbarkeit der Prüfausdrücke erzielt.

4.6.3.3 Prüffunktion für abstrakte Syntaxbäume

Analog zur Prüffunktion für den Primärtext wird eine Prüffunktion zur Auswertung von XPath-Ausdrücken gegen eine als strukturiertes Datum vorliegende Zusicherung definiert. Die verwendbaren XPath-Ausdrücke werden dabei auf die Menge $\mathcal{X}$ der Ausdrücke eingeschränkt, deren Ergebnis eine (ggf. leere) Menge von Elementen aus dem XML-Dokument der Zusicherung sind (im Gegensatz zu Abfragen, die z. B. eine Anzahl von Treffern zurück liefern).

Die XML-kodierte AST-Form einer Zusicherung sei X. $x \in \mathcal{X}$ ist ein beliebiger XPath-Ausdruck. Die Anwendung des XPath-Ausdrucks auf X wird als Funktion $\chi(X, x)$ geschrieben, deren Rückgabewert eine Menge von Elementen aus X ist.

In einem ersten Schritt wird die Prüffunktion $\rho(X, x)$ für abstrakte Syntaxbäume auf Basis eines AST X gemäß einer Grammatik G und eines XPath-Ausdrucks $x \in \mathcal{X}$ definiert:

$$\rho(X, x) = \begin{cases} \textbf{T} \text{ genau dann, wenn } \chi(X, x) \neq \emptyset \\ \textbf{F} \text{ genau dann, wenn } \chi(X, x) = \emptyset \end{cases}$$

Für die Verwendung in SVO-Logiken wird eine alternative Schreibweise $\mathrm{GCR}(X, x)$ eingeführt.

Um die gleichzeitige Prüfung mehrerer Eigenschaften ausdrücken zu können, werden zusammengesetzte Beschränkungen (restrictions) r definiert, die aus XPath-Ausdrücken bestehen, die über *and*- und *or*-Junktoren verknüpft und ggf. auch über die *not*-Funktion invertiert werden

können. Die Bedeutung dieser in Prefix-Notation geschriebenen Verknüpfungen ist intuitiv erschließbar:

$$\begin{aligned}\mathrm{and}(r_1, r_2) &: \text{Es müssen } r_1 \text{ und } r_2 \text{ erfüllt sein.}\\ \mathrm{or}(r_1, r_2) &: \text{Es müssen } r_1 \text{ oder } r_2 \text{ oder beide erfüllt sein.}\\ \mathrm{not}(r_1) &: r_1 \text{ darf nicht erfüllt sein.}\end{aligned}$$

Zusammengesetzte Beschränkungen r können nun rekursiv mit Hilfe der XPath-Ausdrücke und der Junktoren definiert werden:

$$\begin{aligned}&\textit{Beschraenkung} \rightarrow \textit{XPath}\\ &\textit{Beschraenkung} \rightarrow \mathrm{not}(\textit{Beschraenkung})\\ &\textit{Beschraenkung} \rightarrow \mathrm{and}(\textit{Beschraenkung}, \textit{Beschraenkung})\\ &\textit{Beschraenkung} \rightarrow \mathrm{or}(\textit{Beschraenkung}, \textit{Beschraenkung})\end{aligned}$$

Um eine zusammengesetzte Beschränkung auszuwerten, wird die Definition der Prüffunktion $\rho(X, r)$ erweitert.

wenn $r \in \mathcal{X}$

$$\rho(X, r) = \begin{cases} \mathbf{T} \text{ genau dann, wenn } \chi(X, r) \neq \emptyset \\ \mathbf{F} \text{ genau dann, wenn } \chi(X, r) = \emptyset \end{cases} \tag{4.4}$$

sonst

$$\rho(X, \mathrm{not}(r)) = \neg\rho(X, r) \tag{4.5}$$
$$\rho(X, \mathrm{and}(r_1, r_2)) = \rho(X, r_1) \wedge \rho(X, r_2) \tag{4.6}$$
$$\rho(X, \mathrm{or}(r_1, r_2)) = \rho(X, r_1) \vee \rho(X, r_2) \tag{4.7}$$

Damit ist auch

$$\rho(X, \mathrm{and}(r_1, r_2)) = \rho(X, \mathrm{and}(r_2, r_1)) \tag{4.8}$$
$$\rho(X, \mathrm{or}(r_1, r_2)) = \rho(X, \mathrm{or}(r_2, r_1)) \tag{4.9}$$

Eine leere (also nicht wirksame) Beschränkung wird als $r = \bot$ geschrieben. Sie kann durch einen XPath der Form „/*“ kodiert werden. Damit ist

$$\rho(X, \bot) = \mathbf{T} \qquad \text{für beliebige, nicht-leere } X \quad . \tag{4.10}$$

Es gilt

$$\rho(X, \mathrm{and}(r, \bot)) = \rho(X, r) \wedge \rho(X, \bot) = \rho(X, r) \wedge \mathbf{T} = \rho(X, r) \tag{4.11}$$
$$\rho(X, \mathrm{or}(r, \bot)) = \rho(X, r) \vee \rho(X, \bot) = \rho(X, r) \vee \mathbf{T} = \mathbf{T} \tag{4.12}$$
$$\rho(X, \mathrm{not}(\bot)) = \neg\rho(X, \bot) = \neg\mathbf{T} = \mathbf{F} \tag{4.13}$$

Damit sind nun zusammengesetzte Prüfausdrücke möglich, die ihrerseits wieder mittels einer Grammatik kodiert werden können. Die Grammatik ist in Anhang C.3 enthalten.

In Analogie zu (4.3) können auch im Fall der Beschränkung auf Basis der zusammengesetzten Prüfausdrücke mehrere Anforderungen in Form von XPath-Ausdrücken $x_1, \ldots, x_n$ gestellt werden. Diese werden jedoch statt auf Produktionsregeln mittelbar über die strukturierte Darstellung der Primärtexte auf den Sprachumfang angewendet. Die strukturierten Daten X können dabei auch durch die Kodierungsfunktion $X = \Phi_G(p)$ dargestellt werden. Es gilt für den eingeschränkten Sprachumfang $\mathscr{L}(G, r)$:

$$\mathscr{L}(G, r) = \{p | p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), r)\} \tag{4.14}$$

da alle x_ν zu einer Beschränkung $r = \mathrm{and}(x_1, \mathrm{and}(x_2, \ldots))$ zusammengefasst werden können.

Für $r = \bot$ ist $\mathscr{L}(G, r) = \mathscr{L}(G)$, da mit (4.10) gilt:

$$\begin{aligned} \mathscr{L}(G, \bot) &= \{p | p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), \bot)\} \\ &= \{p | p \in \mathscr{L}(G) \wedge \mathbf{T}\} \\ &= \{p | p \in \mathscr{L}(G)\} \\ &= \mathscr{L}(G) \end{aligned} \tag{4.15}$$

Durch Und-Verknüpfung einer Einschränkung r_0 mit einer Einschränkung r_1 ergibt sich eine Teilmenge des Sprachumfangs, da es sich um eine Schnittmengenbildung handelt:

$$\begin{aligned} \mathscr{L}(G, \mathrm{and}(r_0, r_1)) &= \{p | p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), \mathrm{and}(r_0, r_1))\} \\ &= \{p | p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), r_0) \wedge \rho(\Phi_G(p), r_1)\} \\ &= \{p | p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), r_0) \wedge p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), r_1)\} \\ &= \{p | p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), r_0)\} \cap \{p | p \in \mathscr{L}(G) \wedge \rho(\Phi_G(p), r_1)\} \\ &= \mathscr{L}(G, r_0) \cap \mathscr{L}(G, r_1) \end{aligned}$$

4.7 Delegation

Eine wichtige Anforderung an grammatikbasierte Zertifikate ist die Möglichkeit, die Berechtigung, Zusicherungen zu erstellen, an Dritte zu delegieren. Dabei ist es auch notwendig, dass die delegierten Aussagenmengen in ihrem Umfang eingeschränkt werden können.

In diesem Abschnitt werden zunächst einige Grundbegriffe der Delegation erläutert und ein Delegationsmechanismus für GC-Zusicherungen beschrieben. Danach werden der Informationsgehalt für Delegationszertifikate und die notwendigen Prüfschritte für diese Zertifikate betrachtet.

4.7.1 Begriffe der Delegation

Bei der Delegation überträgt eine Person, die bereits das Recht hat, Aussagen innerhalb eines bestimmten Kontextes zu erstellen, dieses Recht auf eine andere Person. Dabei kann unterschieden werden, ob der Übergang des Rechtes die erneute Weitergabe (Unterdelegation) umfasst oder ob dies nicht möglich ist.

Ausgangspunkt für die Delegation ist das Vorliegen einer Berechtigung. Diese wird im Zusammenhang mit einer späteren Weitergabe (*Delegation*) als *initiale Berechtigung* bezeichnet.

Der Inhaber eines Rechtes, der dieses Recht weitergeben will, wird als *Delegierender* bezeichnet.

Die über eine Delegation weitergereichte Berechtigung wird als *delegiertes Recht* und der Empfänger des delegierten Rechtes als *Delegationsempfänger* oder als *Delegierter* bezeichnet.

Die Delegation eines Rechtes kann auf unterschiedliche Art und Weise erfolgen. Wird von einem Vorgesetzten eine Aufgabe an einen Untergebenen delegiert und die Arbeitsergebnisse wieder durch den Vorgesetzten entgegengenommen, so reicht für die Delegation die Absprache zwischen Vorgesetztem und Mitarbeiter.

Umfasst die Delegation mehrere Stufen oder die Kommunikation der auf Basis der delegierten Rechte gemachten Aussagen gegenüber Dritten, so müssen auch diese weiteren Beteiligten von der Delegation in Kenntnis gesetzt werden. Die dabei übermittelte Information wird im Folgenden als *Delegationsnachweis* bezeichnet.

Eine mehrstufige Delegation soll als *Delegationskette* bezeichnet werden. Der Beginn der Delegationskette liegt beim *Initiator* der Delegation.

4.7.2 Delegationsvorgang

Für die vorliegende Arbeit ist der Inhalt einer Delegation auf das Recht beschränkt, Aussagen i. S. von Zusicherungen entsprechend einer vorgegebenen Grammatik zu machen. Die zugrunde liegende Grammatik soll dabei wie eingeführt durch die Angabe eines Hashwertes über eine Grammatikdatei bestimmt werden. Die durch die Grammatik G definierte Sprache $\mathscr{L}(G)$ soll im Rahmen einer Delegation als *Basisberechtigung* bezeichnet werden.

Wie bei der Festlegung der Anforderungen in Kapitel 4.6 ist auch bei der Delegation von Zusicherungen ein Mechanismus zur Einschränkung der möglichen Aussagen sinnvoll. Beispiele hierfür sind die Beschränkung von Auditierungen auf Systeme eines Zuständigkeitsbereichs oder die Einschränkung der für eine Auditierung zugelassenen Methoden u. ä.

Im Rahmen jeder Delegationsstufe kann daher eine zusammengesetzte Beschränkung in Analogie zu Kapitel 4.6.3.3 definiert werden, die erfüllt werden muss, damit ein Zertifikat akzeptiert wird. Die Beschränkungen aller Stufen werden vor der Prüfung Und-verknüpft. Die Auswertung der so verknüpften Beschränkungen kann analog zur Auswertung der Anforderungen an Zusicherungen erfolgen.

Die Delegation wird i. d. R. durch Ausstellen eines Delegationszertifikates durchgeführt. Die wesentlichen Bestandteile der Aussage eines solchen Zertifikates sind die Kennung des Delegationsempfängers B, die Basisberechtigung in Form eines Hashwertes einer Grammatik G, eine optionale Datenstruktur R, die die bei dieser Delegation anzuwendenden Beschränkungen beschreibt und die Angabe einer Ganzzahl $w \geq 0$, die beschreibt, wie viele weitere Delegationsstufen nach der aktuellen noch erlaubt sind.

Das Vorliegen eines solchen Delegationszertifikates bedeutet, dass B Zusicherungen mit der Basisberechtigung $\mathscr{L}(G)$ erstellen darf. Die Zusicherungen sind dabei durch R beschränkt. Zusätzlich sind alle Beschränkungen wirksam, die für den Aussteller des Zertifikates gelten, da dieser nicht mehr Rechte delegieren können soll, als ihm selbst gewährt wurden. Ist $w > 0$, so soll B seinerseits dieses Recht weitergeben dürfen.

Damit eine auswertende Stelle diesen Delegationsnachweis akzeptiert, muss der Akzeptor vom Recht des Ausstellers zur Delegation überzeugt sein. Dieses Recht, weitere Beteiligte mit Zusicherungsrechten zu versorgen, kann z. B. im ZAS durch eine tabellarische Zuordnung realisiert werden. Dabei wird einer Ausstellerkennung Q eine Basisberechtigung $\mathrm{H}(G)$ mit Beschränkungen R und der Anzahl w_{max} maximal erlaubter weiterer Delegationsstufen zugeordnet. Die ggf. eingeschränkte Basisberechtigung bildet die Initialberechtigung für Delegationsketten, die von Q ausgehen.

Ist $w_{\text{max}} = 0$, so ist für Q keine weitere Delegation möglich. Bei $w_{\text{max}} = 1$ kann Q nur das Recht zur Ausstellung von Zusicherungen gemäß der hinterlegten Initialberechtigung $\mathscr{L}(G, R)$ durch ein Delegationszertifikat mit $w = 0$ weitergeben. Für den Fall $w_{\text{max}} > 1$ ist Q berechtigt, Delegationszertifikate mit $0 < w \leq w_{\text{max}}$ zu erstellen. Damit kann Q Delegationsketten mit einer Länge von bis zu $w_{\text{max}} - 1$ Stufen beginnen.

Ergänzend zu dieser nicht-formalen Darstellung wird für das Delegationsverfahren in Kapitel 5.5.5 eine Semantik auf Basis der SVO-Logik angegeben.

4.7.3 Delegationszertifikat

Ein Delegationszertifikat (engl. Grammar-based Delegation Certificate (GDC)) ist eine Zusicherung mit vergleichbarem Informationsinhalt wie ein grammatikbasiertes Zertifikat. Eine Gegenüberstellung der Inhalte zeigt Tabelle 4.4.

Das Delegationszertifikat enthält – wie ein X.509-Zertifikat – einen *Subject*-Eintrag für den Delegationsempfänger, dementsprechend enthält das *Issuer*-Feld den Delegierenden. Die Grammatikreferenz (Hashwert von G) erzeugt die Basisberechtigung und begründet das Recht, Zusicherungen aus $\mathscr{L}(G)$ abzugeben.

Die Aufnahme eines AuthorityInformationAccess- oder eines CRLDistributionPoint-Elementes ist notwendig, wenn Delegationen längerfristig gelten sollen, da in diesem Fall der Mechanismus der Nonce nicht eingesetzt werden kann.

Da keine der zusätzlichen Informationen der GDC ein bestehendes Feld der GC umdeutet, können die neuen Felder Subject, Restriction und PathLength auch zu einer Aussage zusammengefasst und als GC kodiert werden. Da die Grammatikreferenz des GC in diesem Fall für eine Grammatik zur Beschreibung der GDC-Zusicherung verwendet werden muss, muss auch die Grammatikreferenz für die Basisberechtigung in die Zusicherung aufgenommen werden.

Es wird daher eine spezielle Grammatik G_D zur Erstellung von Delegationsaussagen festgelegt. Durch Einsetzen dieser Grammatik in ein GC wird ein Delegationszertifikat erstellt. Eine konkrete Umsetzung wird im Rahmen der Implementierung in Kapitel 5 vorgestellt.

GC	GDC
Version	Version
SerialNumber	SerialNumber
Signature	Signature
Issuer	Issuer
Validity	Validity
–	Subject
GrammarReference	GrammarReference
Assertion	–
–	Restriction
Nonce	–
–	PathLength

Tabelle 4.4: Inhalt von Delegationszertifikaten im Vergleich zu GC

Um zirkuläre Abhängigkeiten und Überlagerung von Mechanismen zu vermeiden, darf die Grammatik G_D nicht selbst Gegenstand einer Delegation sein. Damit muss auch festgelegt werden, dass jeder Beteiligte berechtigt ist, Aussagen nach $\mathscr{L}(G_D)$ zu erstellen.

4.7.4 Delegationsprüfung

Jeder Akzeptor einer Aussage benötigt zur Prüfung der Legitimation der Aussage eine kryptographisch verkettete Liste von Delegationszertifikaten oder anders gesicherter Delegationsaussagen. Es ist dabei zum einen möglich, die relevanten Delegationszertifikate an die als GC kodierte Aussage anzuhängen und gemeinsam an den Akzeptor zu übermitteln. Die andere Möglichkeit besteht darin, die Delegationszertifikate sowohl dem Delegierten als auch dem Ursprung der Delegationskette zum Zeitpunkt der Ausstellung der Delegation zukommen lassen.

Die Delegationsprüfung erfolgt dann in mehreren Schritten.

- ❑ Kontextprüfung (Aussage und Delegation beziehen sich auf die gleiche Grammatik)
- ❑ Vollständigkeitsprüfung der Zertifikatskette
- ❑ Gültigkeitsprüfung aller verwendeten Zertifikate (Zeitraum und ggf. Sperrinformation)
- ❑ Prüfung der Signaturen der Delegationszertifikate
- ❑ Prüfung der Zusicherung gegen die Basisberechtigung (Grammatik)
- ❑ Prüfung der ggf. vorhandenen Beschränkungen

Eine Alternative zur jeweiligen Prüfung der vollständigen Kette ist die Verwendung einer Berechtigungsdatenbank im ZAS, in der die Rechte für einzelne Instanzen schritthaltend mit der

Ausstellung von Berechtigungszertifikaten mitgeführt werden. Hierbei muss jeder Aussteller einer Berechtigung diese an den ZAS übertragen. Vor dem Einstellen der Information wird geprüft, ob bereits eine Berechtigung für den Aussteller des Berechtigungsnachweises vorliegt. Nur wenn dies der Fall ist, wird die neue Berechtigung aus dem Delegationszertifikat aufgenommen. Die jeweils anzuwendenden Beschränkungen müssen ebenfalls mitgeführt werden.

4.8 Grammatikverteilung

Eine Grammatikdatenbank und ein darauf aufsetzender Grammatik-Server haben die Aufgabe, Dateien, die die textuelle Darstellung einer Grammatik enthalten, auf Anfrage auszuliefern. Dabei wird der Hashwert $\mathrm{H}(G)$ einer Grammatik, die z. B. in EBNF-Notation vorliegt, als Referenz genutzt. Für den häufigen Vorgang der Auslieferung einer Grammatik soll für den Grammatik-Server ein möglichst geringer Aufwand entstehen. Desweiteren soll das notwendige Vertrauen in einen solchen Verteildienst lediglich auf die Sicherung der Verfügbarkeit beschränkt werden.

Im einfachsten Fall können Grammatiken an den Server übergeben werden. Der Server berechnet den zugehörigen Hashwert und speichert diesen als Index für den Abruf der Grammatik. Falls der Server vor Missbrauch geschützt werden soll, so müssen zusätzlich Autorisierungs- und Authentisierungsfunktionen unterstützt werden.

4.8.1 Vertraulichkeit von Grammatiken

Grammatikdaten sind im Falle von Bestätigungen für Sicherheitsanalysen u. U. selbst hinsichtlich ihrer Vertraulichkeit zu schützen. Daher soll im Folgenden der Anwendungsfall einer verschlüsselten Speicherung in einem Grammatik-Server betrachtet werden.

Die Verwendung öffentlicher Schlüssel zur Vorabverschlüsselung ist nicht geeignet, da in diesem Fall öffentliche Schlüssel jedes möglichen Delegierten und jedes Prüfers einer Zusicherung vorab bekannt sein und die Speicherung verschlüsselter Grammatiken mehrfach erfolgen müsste. Eine Client-spezifische asymmetrische Verschlüsselung bei Abruf der Grammatik durch den Server setzt wieder Vertrauen in die Wahrung der Vertraulichkeit durch den Dienst voraus. Die Notwendigkeit dieser Eigenschaft wurde jedoch bereits ausgeschlossen. Daher wird die Verwendung eines (beliebigen) symmetrischen Verschlüsselungsverfahrens angenommen.

Der Ersteller E einer Grammatik wählt zusätzlich einen symmetrischen Schlüssel S aus, den er zur Verschlüsselung der Grammatikdaten vor der Übertragung an den Grammatikserver verwendet. Die Übertragung an den Grammatikserver erfolgt als Tupel aus Hashwert der Grammatik, der zum Abruf der Daten benötigt wird, und dem Chiffrat der zugehörigen Grammatik.

Neben der Ermittlung des Hashwertes durch den Server muss es auch die Möglichkeit geben, den Hashwert einer verschlüsselten Grammatik G_1 explizit anzugeben. Ohne eine Berechtigungsprüfung kann ein Angreifer eine Nachricht an den Grammatikserver so manipulieren, dass der Hashwert der Grammatik für die weitere Verwendung blockiert ist.

Der Dienst muss zur Sicherung der Verfügbarkeit daher um einen Legitimationsmechanismus erweitert werden, der nur Berechtigten das Einstellen von verschlüsselten Grammatiken ermöglicht. Auf Basis dieser Berechtigungen kann auch das Löschen von Einträgen dem jeweiligen Ersteller der Grammatik gestattet werden.

4.8.2 Daten- und Funktionsmodell des Grammatikservers

Sollen verschlüsselte Grammatiken auf dem Grammatikserver über den Hashwert der unverschlüsselten Grammatik abgerufen werden, so müssen sowohl das Chiffrat als auch der Hashwert gemeinsam eingestellt werden. Das Einstellen darf nur durch einen autorisierten Akteur erfolgen. Hierzu werden Referenzen auf alle berechtigten öffentlichen Schlüssel in einer Autorisierungstabelle, beschrieben durch die Menge $\mathbf{A}_0$, hinterlegt. Eine Schlüsselreferenz wird durch einen Hashwert über den jeweiligen Schlüssel gebildet. Hashwert und Chiffrat werden zusammen mit dem Schlüssel des Einstellers im Hauptdatenbestand **D** als 3-Tupel gespeichert. Wird eine Grammatik im Klartext eingestellt, so ist die Speicherung des öffentlichen Schlüssels optional.

Bei der Definition des Funktionsmodells werden folgende Größen verwendet:

- Client C
- Grammatikersteller E mit öffentlichem Schlüssel K_E
- Server G mit öffentlichem Schlüssel K_G
- Schlüsselreferenz $\widehat{K_\nu}$ auf Schlüssel K_ν
- Grammatiken G_1, G_2 mit den Hashwerten $H_1 = \mathrm{H}(G_1)$ und $H_2 = \mathrm{H}(G_2)$
- Symmetrischer Schlüssel S
- Zufallswerte r_1, r_2 und zufällige Token t_1, t_2
- Variablen X, Y, Z
- das Symbol $\perp$ zur Kennzeichnung ungültiger oder leerer Datenelemente

Für die Kommunikation zwischen Nutzern und dem Server werden Nachrichten der Form (Label, Parameter 1, [Parameter 2], ...) definiert.

Grammatiken können durch beliebige Nutzer mit einer reqG-Nachricht, die den Hashwert der gewünschten Grammatik (z. B. H_1) enthält, abgerufen werden. Die Anfrage wird vom Server mit einer respG-Nachricht beantwortet:

$$C \mapsto G : (\text{reqG}, H_1)$$
$$G \mapsto C : (\text{respG}, H_1, G_1)$$

Die Antwort wird nur dann erteilt, wenn ein Datensatz mit einem entsprechenden Hashwert in der Datenbank enthalten ist, also $(H_1, Y, Z) \in \mathbf{D}$ gilt. Bei unverschlüsselten Grammatiken ist $Y = G_1$ und $Z = \perp$, bei verschlüsselten Grammatiken ist $Y = \{G_1\}_S$ und $Z = \widehat{K_E}$. Der dritte Wert nimmt im Fall von verschlüsselten Grammatiken eine Referenz auf den öffentlichen Schlüssel der Instanz auf, die die Grammatik eingestellt hat. Ist kein passender Hashwert in der Datenbank, so lautet die Antwort $(\text{NULL}, H_1, \perp)$.

Das Einstellen einer unverschlüsselten Grammatik ohne Autorisierung beantwortet der Server mit einer entsprechenden respG-Nachricht, die auch auf eine entsprechende Anfrage nach der Grammatik erfolgt.

$$C \mapsto G : (\text{addG}, G_1) \tag{GS1}$$

$$G \mapsto C : (\text{respG}, \mathrm{H}(G_1), G_1) \tag{GS2}$$

Die Grammatik wird dann eingestellt, wenn nicht bereits ein Datensatz mit diesem Hashwert enthalten ist $(\forall (X, Y, Z) \in \mathbf{D} \,|\, X \neq \mathrm{H}(G_1))$. Für fehlgeschlagene Operationen wird der Nachrichtentyp *ERR* für Meldungen definiert, die den Anfragenden über die Fehlerursache informieren. Diese Nachrichten können in einer konkreten Implementierung festgelegt werden.

Für das Einstellen verschlüsselter Grammatiken und deren Löschung wird eine Autorisierung vorgesehen. Dazu muss zunächst die Schlüsselreferenz des Erstellers in die Menge der berechtigten Schlüssel aufgenommen werden $(\widehat{K_E} \in \mathbf{A}_0)$.

Um eine autorisierte Operation ausführen zu können, muss E zunächst ein Legitimationstoken vom Server anfordern, der zuvor die Bedingung $\widehat{K_E} \in \mathbf{A}_0$ prüft.

$$E \mapsto G : \lfloor \text{reqT}, r_1, \widehat{K_E} \rfloor_{K_E^{-1}} \tag{GS3}$$

$$G \mapsto E : \lfloor \text{respT}, r_1, t_1 \rfloor_{K_G^{-1}} \tag{GS4}$$

Bei der Beantragung in (GS3) wird die Anfrage signiert und enthält einen Zufallswert r_1, der gegen Wiedereinspielen von Server-Antworten schützt.

In der Antwort (GS4) erhält E ein Token t_1, das für eine Einstell- oder Löschoperation verwendet werden kann. Das Token wird auf Serverseite in Verbindung mit $\widehat{K_E}$ gespeichert.

Zur Einstellung einer Grammatik sendet E Hashwert, Chiffrat und Token signiert an G.

$$E \mapsto G : \lfloor \text{addHG}, \mathrm{H}(G_2), \{G_2\}_S, t_1, \widehat{K_E} \rfloor_{K_E^{-1}} \tag{GS5}$$

$$G \mapsto E : (\text{respG}, \mathrm{H}(G_2), \{G_2\}_S) \tag{GS6}$$

Ist bereits eine Grammatik mit diesem Hashwert eingestellt, so antwortet der Server mit dem Nachrichtentyp *DUP*.

Zum Löschen der Grammatik wird analog ein Token beantragt und damit die entsprechende *delG*-Anfrage legitimiert.

$$E \mapsto G : \lfloor \text{delG}, \mathrm{H}(G_2), t_2, \widehat{K_E} \rfloor_{K_E^{-1}} \tag{GS7}$$

$$G \mapsto E : (\text{respG}, \mathrm{H}(G_2), \perp) \tag{GS8}$$

Ist keine Grammatik unter dem genannten Hashwert eingestellt, so antwortet der Server mit dem Nachrichtentyp *NULL*. Falls der Eintrag einem anderen Schlüssel zugeordnet ist, so wird eine Nachricht vom Typ *NOTOWNER* zurück gesendet.

Desweiteren wird die Möglichkeit zur Delegation des Einstellens und Löschens von Grammatiken vorgesehen. Hierzu kann die Anforderung eines Tokens um die Angabe des öffentlichen Schlüssels K_D eines zu berechtigenden weiteren Akteurs ergänzt werden.

$$\begin{aligned} E \mapsto G &: \lfloor \text{reqT}, r_2, K_D, \widehat{K_E} \rfloor_{K_E^{-1}} \\ G \mapsto E &: \lfloor \text{respT}, r_2, t_2 \rfloor_{K_G^{-1}} \end{aligned} \tag{GS9}$$

Der erstellte Token wird im Grammatikserver zusammen mit dem Schlüssel des Delegierten und der zugehörigen Schlüsselreferenz gespeichert.

Zur Unterstützung des Delegationsmechanismus sollen die autorisierten Schlüsselreferenzen attributiert werden. Daher wird die Menge $\mathbf{A}_0$ in eine Menge $\mathbf{A}$ aus Drei-Tupeln der Form $(\widehat{K}, t, d)$ überführt. Für jeden autorisierten Schlüssel K lässt sich durch $t \in \{\mathbf{T}, \mathbf{F}\}$ das Recht, Token anzufordern und durch $d \in \{\mathbf{T}, \mathbf{F}\}$ das Recht zur Delegation ausdrücken. Sofern der Server die Zertifikate nicht gesondert speichert, kann dieses Tupel auch um eine Stelle für den Schlüssel bzw. für ein Zertifikat zu diesem Schlüssel ergänzt werden.

Nur wenn das d-Recht für den Absender der Token-Anforderungen mit Angabe des Schlüssels K_D eines Dritten gesetzt ist, wird dessen Schlüsselreferenz in $\mathbf{A}$ als $(\widehat{K_D}, \mathbf{F}, \mathbf{F})$ aufgenommen und die Zuordnung des Token t_2 zu $\widehat{K_D}$ wird im Dienst gespeichert.

4.8.3 Sicherheitseigenschaften des Grammatikservers

Die Sicherheitseigenschaften, die durch Serverfunktion und Protokoll erbracht werden sollen, sind:

- ❑ Eine Grammatik darf nur dann vom Server akzeptiert werden, wenn diese durch einen vertrauenswürdigen Client eingestellt wird. Dabei wird Vertrauenswürdigkeit durch Eintragung in die Autorisierungstabelle oder durch Delegation begründet.
- ❑ Dem Akzeptieren eines Löschauftrags durch den Server muss die Annahme der zugehörigen Grammatik vom gleichen Client vorausgegangen sein.
- ❑ Dem Akzeptieren eines Löschauftrags durch den Server muss die Erstellung des Löschauftrags durch den vertrauenswürdigen Client vorausgegangen sein.
- ❑ Jede Anforderung zur Einstellung oder Löschung einer verschlüsselten Grammatik darf zu maximal einer Aktion führen.

Diese Eigenschaften werden in Kapitel 5 formal untersucht und nachgewiesen.

5 Implementierung und formale Bewertung

Für die in Kapitel 4 entworfenen grammatikbasierten Zertifikate wurde eine prototypische Realisierung der für ihre Verarbeitung nötigen Mechanismen erstellt, deren Datenformate und Funktionsbausteine die Anforderungen aus Kapitel 1.3 umsetzen.

Der Entwurf wurde darüber hinaus einer formalen Analyse unterzogen. Zum einen wird eine formale Semantik der grammatikbasierten Zertifikate und der Delegationsmechanismen auf Basis der SVO-Logik angegeben, zum anderen wird eine Proverif-Modellierung des Grammatikverteildienstes zum Nachweis seiner Sicherheitseigenschaften eingesetzt.

5.1 Realisierung grammatikbasierter Zertifikate

Die Realisierung setzt das Prinzip der Gesamtarchitektur aus Abschnitt 4.5 mittels verschiedener Komponenten um. Diese wurden teilweise unter Verwendung existierender Werkzeuge entwickelt. Die Hauptbestandteile der Implementierung sind:

- Syntax für Grammatiken zur Beschreibung grammatikbasierter Zertifikate (Grammar-based Certificate Grammar, GCG)
- Makro-gestütztes Verfahren zur Erstellung von GCGs
- Parsergenerator zur Erstellung von Parsern für Zusicherungen
- XML-Schema und entsprechende Kodier- und Signierfunktionen zur Kodierung grammatikbasierter Zertifikate
- Parsergenerator für Baumparser zum Einlesen strukturierter Daten und zur Rückgewinnung des Primärtextes

Eine Übersicht über alle vorgestellten Grammatik- und Parser-Generatoren gibt Abbildung 5.1.

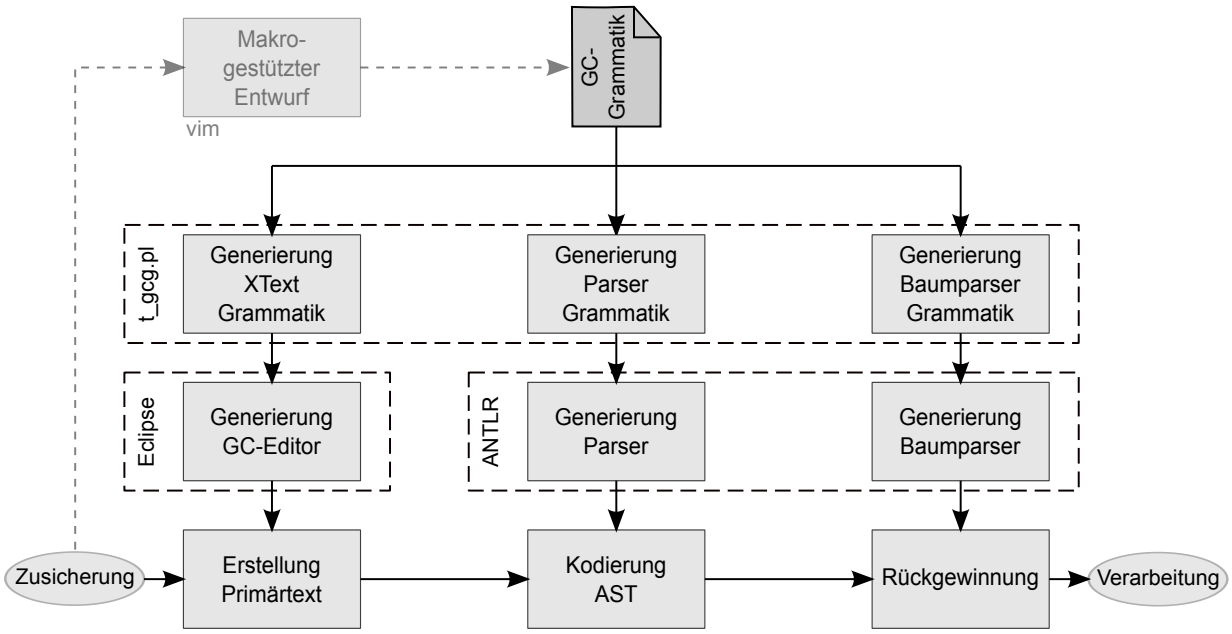


Abbildung 5.1: Automatisierte Editor- und Parsergenerierung

5.1.1 ANTLR-Parser

Als Basis für die prototypische Implementierung wurde der ANTLR-Parsergenerator von Terence Parr gewählt[Par07]. ANTLR steht für Another Tool for Language Recognition und ermöglicht die automatisierte Erstellung von Parsern in Java auf Basis vorgegebener Grammatiken.

Ein besonderer Vorteil von ANTLR ist die Unterstützung von sog. Baumparsern, die es ermöglichen, auch Baumstrukturen, wie sie in Form des AST vorliegen, zu verarbeiten. Darüber hinaus bietet die Syntax der Grammatiken von ANTLR die Möglichkeit, durch die Modifikationszeichen '!' und '^' die Struktur des durch den Parser aufgebauten AST zu steuern.

Mit dem '!'-Zeichen lässt sich die Aufnahme eines Tokens oder Terminals in den AST unterdrücken. Diese Funktion ist essentiell für die Erstellung der optimierten ASTs nach Kapitel 4.3.5, da die Optimierung auf genau dieser Unterdrückung ausgewählter Elemente des AST beruht.

5.1.2 Syntax für Grammatiken

In Kapitel 4.3 wurde die Verwendung kontextfreier Grammatiken zur Analyse von Zusicherungen vorgeschlagen. Für die Implementierung wurde daher die im Folgenden beschriebene konkrete Syntax für Grammatiken festgelegt.

Zunächst sind die grundsätzlichen Sprachelemente zu spezifizieren.

Die Bezeichner (IDENTIFIER) für Nicht-Terminale aus $\mathcal{N}$ beginnen mit einem Großbuchstaben und bestehen aus Groß- und mindestens einem Kleinbuchstaben. Daneben sind Ziffern und

Unterstriche zulässig. Terminale werden durch Zeichenketten in einfachen Hochkommata dargestellt.

Daneben sind die Lexeme INT, REAL, HEX, STRING und DATE definiert, die wie Terminale verarbeitet werden und die jeweiligen frei wählbaren Bestandteile der Primärtexte repräsentieren. Die Lexeme werden in einer Menge $\mathcal{L}$ zusammengefasst.

Eine GCG-Datei besteht aus Regeln der Form

```
IDENTIFIER ':' regelRHS ';'
```

Der Regelkörper *regelRHS* besteht aus einer oder mehreren durch das '|'-Symbol getrennten Varianten für das vorstehende Nicht-Terminal.

Jede Variante des Regelkörpers ist definitionsgemäß aus einem oder mehreren Elementen aus $\mathcal{T} \cup \mathcal{N}$ und Lexemen zusammengesetzt. Alle Elemente können durch ?, * und + quantifiziert werden.

Es wurde bewusst auf die Möglichkeit zur Verwendung von Klammern zur Gruppierung von Elementen verzichtet. Bei der Generierung eines Parsers werden Klammern durch automatisch erzeugte Nicht-Terminale (z. B. PAREN-1) ersetzt. Dieser Vorgang soll jedoch explizit durch den Ersteller der Grammatik durchgeführt werden, indem er für die üblicherweise durch die Klammern gruppierten Terme neue Nicht-Terminale mit entsprechenden Produktionsregeln definiert. Dadurch können sinnvolle Nicht-Terminal-Bezeichner gewählt werden und der gruppierte Bestandteil der Zusicherung wird dadurch im Syntaxbaum durch einen entsprechend benannten Knoten verständlicher repräsentiert.

In Anlehnung an die Syntax der Grammatiken des vorgestellten Parser-Generators ANTLR werden zusätzliche Modifikatoren ('!' und '^') definiert, die die Generierung der AST beeinflussen können.

Enthält eine Regel auf ihrer rechten Seite keine Nicht-Terminale, so soll sie als terminale Regel bezeichnet werden und kann in der einfachsten Form aus einem Terminal oder einem Lexem bestehen. Wird die rechte Seite um weitere Terminale erweitert, so müssen alle Terminale bis auf eines durch ein nachgestelltes '!' als *unsichtbar* markiert werden. Auf diese Weise wird nur eines der Terminale bzw. das Lexem in den AST aufgenommen. Dadurch vereinheitlicht sich die Auswertung des AST.

Umgekehrt kann in einer Regel, die auch Nicht-Terminale enthält und somit einen inneren Knoten des AST beschreibt, ein Terminal durch ein nachgestelltes '^' markiert werden. In diesem Fall wird das Terminal in den AST innerhalb eines inneren Knoten aufgenommen.

Die exakte Syntax der GC-Grammatiken wurde durch eine ANTLR-Grammatik modelliert und ist in Anhang C.1 enthalten.

5.1.3 Parser-Erzeugung

Zur Verarbeitung der Primärtexte und der ASTs werden entsprechende Parser benötigt. Der Primärtext wird durch einen gewöhnlichen auf Textdaten arbeitenden Parser in einen AST umgesetzt. Zur Verarbeitung von ASTs wird ein sogenannter Baumparser benötigt, der auf der hierarchischen Datenstruktur des AST arbeitet und als Ergebnis wieder den ursprünglichen Primärtext liefert.

Beide Parser-Typen können durch ANTLR bereitgestellt werden. Jedoch ist es notwendig, aus einer GC-Grammatik eine ANTLR-Parser-Grammatik und eine ANTLR-Baumparser-Grammatik zu erzeugen, die das gewünschte Ergebnis liefern.

Zu diesem Zweck wurde im Rahmen dieser Arbeit ein Grammatik-Generator entwickelt, der auf Basis einer gegebenen GC-Grammatik die benötigten ANTLR-Grammatiken erzeugt. Der Generator wurde in Perl geschrieben (Programmname: trans_gcertgrammar.pl, kurz: t_gcg.pl) und übersetzt die Grammatiken unter Berücksichtigung der entsprechenden Optimierungen des AST gemäß Abschnitt 4.3.5.

Je nach gewünschter Zielgrammatik wird der Grammatik-Generator in verschiedenen Modi aufgerufen.

5.1.3.1 Generierung der Parser-Grammatik

Im Modus zur Erzeugung einer Parser-Grammatik werden die Produktionsregeln der GC-Grammatik in Regeln der ANTLRv3-Grammatik umgesetzt. Hierbei wird jeder Regel ein spezielles Konstrukt zur Generierung eines Teils des AST mitgegeben. Im Folgenden sollen einige Beispiele dieses Vorgehen veranschaulichen.

Eine einfache terminale Regel, die das Nicht-Terminal *Name* auf eine Zeichenkette abbildet,

```
Name : STRING;
```

wird auf eine entsprechenden ANTLRv3-Regel abgebildet:

```
r_Name : STRING -> ^(Name STRING);
```

Da in ANTLR Nicht-Terminale mit Kleinbuchstaben beginnen müssen, wird dem Regelnamen ein Präfix 'r_' vorangestellt. Der Regelkörper ist um die Bildungsvorschrift für den AST erweitert. Diese beginnt mit dem Pfeil '->' und beschreibt das Einfügen eines Knotens '^(...)' in den AST. Dieser erhält das Label *Name* und enthält die eingelesene Zeichenkette. In der XML-Darstellung entspricht dies dem XML-Knoten

```
<Name>
   <GCELEMENT type="STRING">ncipherkrypt1</GCELEMENT>
</Name>
```

Enthält die GC-Regel weitere durch '!' gekennzeichnete Terminale, so werden diese zwar eingelesen, jedoch nicht im AST berücksichtigt:

```
Zertifizierung :   '- '! 'FIPS 140-2 Level 3' '\n'!
                 [...]
                 |   '- '! 'ZKA-Zulassung' '\n'!;

r_Zertifizierung : '- ' 'FIPS 140-2 Level 3' '\n'
                  ->   ^(Zertifizierung 'FIPS 140-2 Level 3' )
                  [...]
                |  '- ' 'ZKA-Zulassung' '\n'
                  ->   ^(Zertifizierung 'ZKA-Zulassung' );
```

Die Spiegelstriche und Zeilenumbrüche werden lediglich für die bessere Lesbarkeit des Primärtextes benötigt und entfallen daher. Ein entsprechender Abschnitt aus einem XML-codierten AST enthält somit Knoten mit Label *Zertifizierung*. Die GCELEMENT-Knoten vom Typ *TOKEN* werden nicht beim Parsing-Vorgang sondern erst in der abschließenden XML-Kodierung eingefügt.

```
<Zertifizierung>
   <GCELEMENT type="TOKEN">FIPS 140-2 Level 3</GCELEMENT>
</Zertifizierung>
<Zertifizierung>
   <GCELEMENT type="TOKEN">ZKA-Zulassung</GCELEMENT>
</Zertifizierung>
```

Bei GC-Regeln, die auf der rechten Seite Nicht-Terminale enthalten, werden definitionsgemäß alle Terminale nicht in den AST aufgenommen und nur die Nicht-Terminale berücksichtigt.

```
Server   :  'der Kryptoserver ' Servername Schrank;

r_Server :  'der Kryptoserver ' r_Servername r_Schrank
             -> ^(Server r_Servername r_Schrank );
```

Neben der Erzeugung der Produktionsregeln werden zusätzliche, fest definierte Lexeme und notwendige Header erzeugt. Die resultierende Grammatik ist eine kombinierte Parser- und Lexer-Grammatik. Als Parsing-Mechanismus wird die k(*)-Methode (adaptive Vorausschau mit Übergang zu Backtracking) vorgegeben.

5.1.3.2 Generierung der Baumparser-Grammatik

Die Erzeugung der ANTLRv3-Grammatik zur Erstellung eines Baumparsers für ASTs erfolgt nach dem umgekehrten Prinzip wie die Erstellung des Primärtext-Parsers. Im Parsing-Ausdruck werden statt (Nicht-)Terminale Baumausdrücke verwendet. Die dabei auftauchenden Nicht-Terminale oder Lexeme werden mit automatisch generierten Labels der Form $r1, r2, \ldots$ versehen. Diese werden bei der Anwendung des Template-Mechanismus von ANTLR verwendet. Hierbei

wird als abgeleiteter Ausdruck eine Zeichenkette erzeugt, die durch Platzhalter mit den Inhalten der geparsten Baumknoten des AST erweitert werden kann. Dieses Vorgehen wird mittels des bereits im vorangegangenen Abschnitt vorgestellten Beispiels illustriert. Hierbei werden jeweils die Parser- und die Baumparser-Regel gegenübergestellt.

Die einfache Speicherung eines Nicht-Terminals kann mit einem ebenfalls sehr einfachen Template zurückgewonnen werden.

```
r_Name : STRING -> ^(Name STRING);

r_Name :        ^(Name r12=STRING )
        ->        template(p12={$r12.text})
                "<p12>";
```

Die Baumparser-Regeln erhalten die gleichen Namen wie die des Primärtextparsers. Die rechte Seite der Regel ist analog zur Knotengenerierung aufgebaut: ein Knoten mit Label *Name* und einer Zeichenkette. Beim Einlesen der Zeichenkette wird dazu ein internes Objekt mit einem Bezeichner (hier *r12*) angelegt. Für die Befüllung des Templates kann auf die Zeichenkette durch die *text*-Methode des über *$r12* referenzierten Objektes zugegriffen werden. Diese wird der Variablen *p12* zugewiesen. Diese Variable kann – in spitze Klammern gesetzt – in der Template-Zeichenkette verwendet werden.

Wurden bei der Knotengenerierung redundante Terminale verworfen, so müssen diese in der Template-Zeichenkette wieder aufgenommen werden. Die Information der redundanten Zeichenketten wird somit aus der GC-Grammatik in die ANTLR-Grammatik übernommen und ist schließlich im Baumparser enthalten.

```
        r_Zertifizierung
                :        '- ' 'FIPS 140-2 Level 3' '\n'
                ->        ^(Zertifizierung 'FIPS 140-2 Level 3' )
...
                |        '- ' 'ZKA-Zulassung' '\n'
                ->        ^(Zertifizierung 'ZKA-Zulassung' )
                ;

          r_Zertifizierung
                :        ^(Zertifizierung 'FIPS 140-2 Level 3' )
                ->        template()
                        "-<\ >FIPS<\ >140-2<\ >Level<\ >3<\n>"
...
                |        ^(Zertifizierung 'ZKA-Zulassung' )
                ->        template()
                        "-<\ >ZKA-Zulassung<\n>"
```

In diesem Beispiel werden lediglich die eingelesenen Terminale ergänzt und die ausgelassenen Zeichenketten '- ' und '\n' wieder eingefügt (Leerzeichen und Zeilenumbrüche werden ebenfalls in spitze Klammern gesetzt).

Für Regeln mit Nicht-Terminalen wird wie bei den Lexemen der Inhalt durch Objekte aufgenommen, mit denen Platzhalter für die Templates belegt werden können.

```
r_Server :  'der Kryptoserver ' r_Servername r_Schrank
             -> ^(Server r_Servername r_Schrank );

r_Server :  ^(Server r7=r_Servername r8=r_Schrank )
            -> template(p7={$r7.st}, p8={$r8.st})
            "der<\ >Kryptoserver<\ ><p7><p8>";
```

Im Beispiel wird ein *Server*-Knoten des AST eingelesen. Dieser enthält zwei Unterknoten, deren rekursiv geparster Inhalt den beiden Objekten *r7* und *r8* zugewiesen wird. Das Template sorgt für die Rückgewinnung des Terminals 'der Kryptoserver ' und hängt daran den (mittels der *st*-Funktion) zurückgewonnenen Text aus den beiden Unterknoten an.

5.1.4 Erstellung von GC-Grammatiken

Grundsätzlich sind Ersteller von Grammatiken für grammatikbasierte Zertifikate im Rahmen der in Anhang C.1 festgelegten Syntax frei bei deren Gestaltung. Da jedoch beim Entwurf von Grammatiken eine große Zahl von Fehlern gemacht werden kann, wurde ein einfaches Verfahren entwickelt, um aus einem vorgegebenen Beispieltext durch einen kleinen Satz von erlaubten Umformungsschritten eine passende Grammatik zu dem gewählten Beispiel und vergleichbaren Texten im gleichen Anwendungskontext abzuleiten. Zur weiteren Unterstützung wurde exemplarisch ein Satz von Makros für den Editor vim entwickelt, mit denen die Umformungsschritte durch einfaches Markieren und Aufruf des passenden Makros durchgeführt werden können.

Das Verfahren startet mit einem Beispieltext, der als Textdatei im Editor vorliegt. Der Text sollte idealerweise bereits auf eine vorgegebene Breite (z. B. 72 Zeichen) umgebrochen sein und inhaltlich sinnvoll durch Absätze strukturiert sein. So können beispielsweise Aufzählungen als zeilenweise organisierte Listen mit vorangestellten Spiegelstrichen o. ä. gestaltet werden.

In einem ersten Schritt (Makro **gca**) wird die Textdatei in eine Grammatik überführt, indem jede Zeile in ein Terminal umgewandelt wird und die entstehende Folge von Terminalen in eine Regel mit dem Startsymbol S eingefasst wird. Die so entstehende Grammatik hat bei einer n-zeiligen Ausgangsdatei die Form

$$S : T_1 T_2 \ldots T_n;$$

Allgemein wird im Weiteren zur Darstellung der rechten Seite einer Regel die Schreibweise ABC verwendet. Hierbei sind A, B und C Folgen von Elementen, die aus den Mengen der Terminale, Nicht-Terminale und Lexeme bestehen, so dass $A, C \in (\mathcal{N} \cup \mathcal{T} \cup \mathcal{L})^*$ und $B \in (\mathcal{N} \cup \mathcal{T} \cup \mathcal{L})^+$ ist.

Jedes Terminal besteht aus Zeichenfolgen der Form $'\alpha\beta\gamma'$ mit Mindestlänge eins, wobei für die Längen der Teilzeichenketten gelten soll

$$|\alpha| \geq 0 \qquad |\beta| \geq 1 \qquad |\gamma| \geq 0$$

Ausgehend von dieser trivialen Grammatik sind die im Folgenden beschriebenen Umformungen erlaubt. Bei den Umformungen wird jeweils der Aufruf für den zugehörigen Makro-Baustein im Editor in Klammern genannt.

Umwandlung einer Elementfolge in ein Nicht-Terminal (**gctr**): eine Folge B von Elementen der rechten Seite einer Regel für das Nicht-Terminal N_1 wird durch ein neues Nicht-Terminal N_2 ersetzt und eine entsprechende neue Regel angelegt. Die Änderungen in der Grammatik sind dabei

$$N_1 \rightarrow A\ B\ C \qquad \Rightarrow \qquad \begin{cases} N_1 \rightarrow A\ N_2\ C \\ N_2 \rightarrow B \end{cases}$$

Dabei kann N_2 auf der rechten Seite von N_1 durch Quantoren ergänzt werden (*, +, ?). Dem Regelkopf von N_2 kann optional der Knoten-Modifikator '!' zur Unterdrückung der Ausprägung eines Knotens im AST hinzugefügt werden. Diese Modifikatoren werden durch das Makro vom Benutzer abgefragt.

Umwandlung einer Teilzeichenkette innerhalb eines Terminals in ein Nicht-Terminal (**gcsr**): es wird eine Teilzeichenkette β eines Terminals durch ein neu zu definierendes Nicht-Terminal N_2 ersetzt. Dabei entstehen aus den Teilzeichenketten des umgewandelten Terminals $T = '\alpha\beta\gamma'$ neue Terminale $'\alpha'$, $'\beta'$ und $'\gamma'$.

$$N_1 \rightarrow A\ '\alpha\beta\gamma'\ C \qquad \Rightarrow \qquad \begin{cases} N_1 \rightarrow A\ '\alpha'\ N_2\ '\gamma'\ C \\ N_2 \rightarrow '\beta' \end{cases}$$

Die neuen Terminale $'\alpha'$ und $'\gamma'$ werden nur dann eingesetzt, sofern $|\alpha| > 0$ bzw. $|\gamma| > 0$ gilt. Der neue Nicht-Terminal-Bezeichner kann wie bei der Umwandlung von Elementfolgen quantifiziert bzw. modifiziert werden.

Hinzufügen einer (terminalen) Regelalternative (**gcra**): eine bestehende Regel wird um eine ggf. weitere Alternative erweitert.

$$N_1 \rightarrow\ \ldots\ \mid\ A_1\ B_1\ C_1 \qquad \Rightarrow \qquad N_1 \rightarrow\ \ldots\ \mid\ A_1\ B_1\ C_1\ \mid\ '\alpha\beta\gamma'$$

Verkürzung des gespeicherten Terminals innerhalb einer terminalen Regel (**gcsa**): Aus dem im AST gespeicherten Terminal einer terminalen Regel N wird eine Teilzeichenkette β als im AST zu speichernder Teil markiert. Die restlichen Teilzeichenketten werden zu unsichtbaren Terminalen umgewandelt.

$$N \rightarrow A\ '\alpha\beta\gamma'\ C \qquad \Rightarrow \qquad N \rightarrow A\ '\alpha'!\ '\beta'\ '\gamma'!\ C$$

Hierbei dürfen A und C nur aus unsichtbaren Terminalen der Form $'\alpha_\nu\beta_\nu\gamma_\nu'$! bestehen. Die unsichtbaren Terminale werden wiederum nur dann eingesetzt, sofern $|\alpha| > 0$ bzw. $|\gamma| > 0$ gilt.

Ersetzung einer Elementfolge durch ein Element $X \in \mathcal{N} \cup \mathcal{L}$: eine Folge von Elementen kann durch ein Nicht-Terminal ersetzt werden (z. B. wenn dieses bereits die Folge dieser Elemente abdeckt) oder ein Terminal (Elementfolge der Länge 1) kann durch ein passendes Lexem ersetzt werden.

$$N \to A\ B\ C \qquad \Rightarrow \qquad N \to A\ X\ C \qquad \text{mit } X \in \mathcal{N} \cup \mathcal{L}$$

Entfernen einer Regelalternative (**gcda**): In Regeln mit mindestens zwei Alternativen kann eine Alternative gelöscht werden. Als Beispiel soll eine Alternative der Form $A_2\ B_2\ C_2$ aus einer Regel N entfernt werden.

$$N \to A_1\ B_1\ C_1 \quad | \quad A_2\ B_2\ C_2 \quad | \quad A_3\ B_3\ C_3 \qquad \Rightarrow \qquad N \to A_1\ B_1\ C_1 \quad | \quad A_3\ B_3\ C_3$$

Die hier vorgestellten Umformungen ermöglichen es, aus einem Beispieltext einer Zusicherung in einem Top-Down-Verfahren schrittweise eine Grammatik abzuleiten, die eine Klasse von Zusicherungen für den vorliegenden Anwendungsfall abdeckt. Da nach der initialen Erstellung jede Umformung des Textes so ausgeführt wird, dass wiederum eine gültige Grammatik entsteht, können viele Fehler bei der Erstellung vermieden werden.

Ein ausführliches Beispiel, das die Umwandlung eines konkreten Zusicherungstextes durch die oben eingeführten Umwandlungsschritte zeigt, ist in Anhang D.1 enthalten.

Zur Prüfung der entstandenen Grammatik gibt es zwei weitere Werkzeuge. Zum einen wurde ein Parser für die Syntax-Prüfung auf Basis der in Anhang C.1 definierten Meta-Grammatik erstellt, der die syntaktische Korrektheit der erstellten Grammatik prüft. Zum anderen werden bei der Generierung der ANTLR-Parser Mehrdeutigkeiten und Rekursionen erkannt, die dann korrigiert werden können. Diese Fehlerfälle treten jedoch bei der Modellierung der natürlichsprachlichen Zusicherungen selten auf, da speziell die Terminale, die auf Basis ganzer Sätze entstehen, im Gegensatz zu den kompakten Elementen einer Programmiersprache sehr gut voneinander zu differenzieren sind.

5.1.5 Erstellung von Zusicherungen

Bei der Erstellung von Zusicherungen auf Basis einer vorgegebenen GC-Grammatik ist es ebenfalls sinnvoll, den Anwender durch Hilfswerkzeuge zu unterstützen.

Grundsätzlich wäre es zwar möglich, die Zusicherung mittels eines einfachen Editors zu schreiben und im Anschluss einem Parser auf Basis der Grammatik zuzuführen. Allerdings ist dieser Prozess sehr fehleranfällig, da jedes Zeichen einschließlich aller Zeilenumbrüche und Leerzeichen relevant ist und im Rahmen der erlaubten Vorgaben liegen muss.

Daher wurde auf Basis des XText-Projektes [Ecl] der Eclipse Foundation ein Mechanismus zur Bereitstellung spezieller Editoren für die Zusicherungen entworfen. Das XText-Projekt stellt,

ebenfalls unter Verwendung der ANTLR-Bibliotheken, ein Entwicklungsrahmenwerk für anwendungsspezifische Sprachen bereit. Ein wesentliches Element des Rahmenwerkes ist die Generierung von sprachspezifischen Editoren, die es ermöglichen, den Anwender bei der Eingabe grammatikkonformer Texte zu unterstützen.

XText-Grammatiken weisen starke Ähnlichkeit mit den ANTLR-Grammatiken auf, so dass es leicht möglich ist, aus einer vorgegebenen GC-Grammatik eine entsprechende XText-Grammatik abzuleiten. Hierzu wurde der Grammatik-Konverter t_gcg.pl um eine weitere Zielgrammatik erweitert.

Das folgende Beispiel zeigt einen Ausschnitt aus einer XText-Grammatik:

```
Dienst: gc_Dienstname_1=Dienstname l2=' auf Port '
        gc_Portnummer_3=Portnummer l4='\n';

Dienstname: l1='ssh' | l1='dns' | l1='smtp' | l1='http'
          | l1='https' | l1='ntp';

Portnummer:        gc_Portnummer_1=INT;
```

Aus den so entstehenden XText-Grammatiken kann in der prototypischen Implementierung mit Hilfe der Eclipse-Entwicklungsumgebung über einen vordefinierten automatisierten Workflow ein Editor bereitgestellt werden, der an jeder Stelle der Eingabe dem Anwender die erlaubten Textteile zur Auswahl anbietet und nach Auswahl in den Text einfügt. Als korrekt erkannte Textteile (automatisch oder manuell eingefügt) werden deutlich sichtbar durch farbige und fett gedruckte Fonts markiert.

Abbildung 5.2 ist ein Bildschirmfoto, das die Anwendung des Editors zeigt. Der zu Demonstrationszwecken fälschlicherweise klein geschriebene Begriff *portscan* ist durch Wellenlinien markiert. Ein Kontextdialog stellt die möglichen Dienstbezeichner zur Auswahl.

Eine Alternative zur Unterstützung der Erstellung von Zusicherungen ist die Verwendung einer Applikation, die beispielsweise in Dialogform die notwendigen Eingaben erfragt. Diese kann – im Gegensatz zum Texteditor – direkt strukturierte Daten erzeugen. Ein Beispiel für einen solchen Dialog ist in Abbildung 5.3 zu sehen.

Um auch im Fall der Dialog-gestützten Eingabe für das gleiche Verständnis der gegebenen Zusicherung beim Anwender wie bei Erstellung des Primärtextes zu sorgen, kann vor Erstellung der Signatur eine Rückgewinnung mit den strukturierten Daten durchgeführt werden.

Die verschiedenen Arten zur Erstellung eines GC sind in Abbildung 5.4 dargestellt.

5.1.6 Zertifikatskodierung

Für die prototypische Implementierung wurde ein XML-Datenformat gewählt, mit dem die Zertifikatselemente aus Kapitel 4.4.1 umgesetzt werden. Das Datenformat wurde in einer XML-Schema-Datei spezifiziert, die in Anhang C.2 enthalten ist.

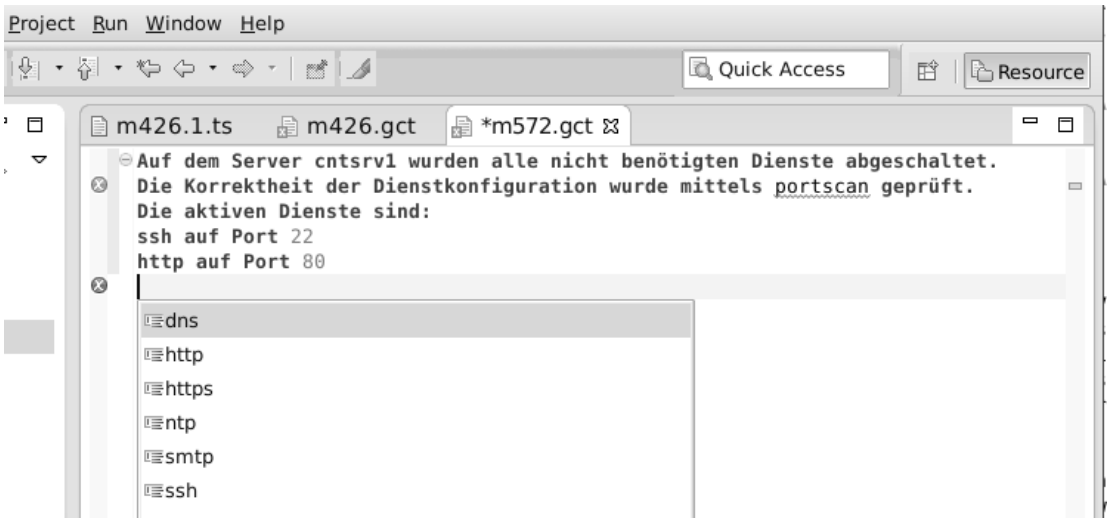


Abbildung 5.2: XText-gesteuerte Eingabe einer Zusicherung

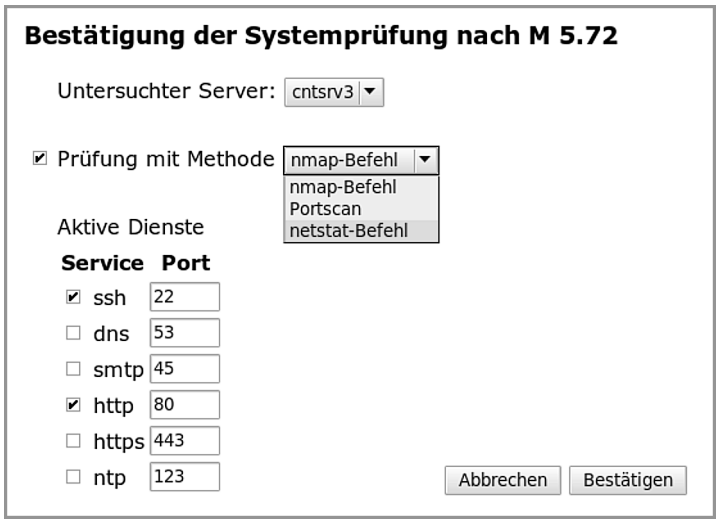


Abbildung 5.3: Eingabemaske für Prüfergebnis

Das gesamte GC wird von einem *GrammarbasedCertificate*-Element eingefasst, das mit einem ID-Attribut mit dem Wert *tbsGCertificate* gekennzeichnet ist. Dieses markiert die gesamte XML-Datenstruktur für die spätere Signatur.

Die Versionsnummer ist auf 1.0 festgelegt. Die Seriennummer eines GC wird durch die Berechnung des md5-Hashwertes über die Konkatenation von inhaltsspezifischen und zufälligen Werten gebildet. Im Prototyp gehen in diesen Hashwert die strukturierten Daten, der öffentliche Schlüssel des Ausstellers der Zusicherung, die aktuelle Zeit und ein Zufallswert ein.

Im *Issuer*-Element wird für jedes Element des *DistinguishedName* des Ausstellers ein Tag mit

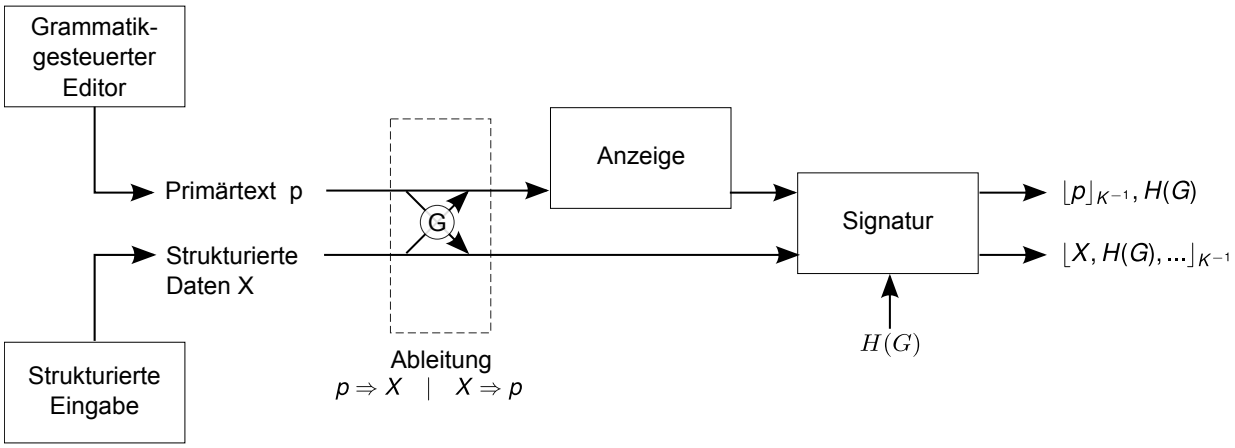


Abbildung 5.4: Verarbeitungsschritte bei der Erstellung grammatikbasierter Zertifikate

dem entsprechenden Inhalt aufgenommen. Die Tags entsprechen dabei in Reihenfolge, Schreibweise und Inhalt der Darstellung nach RFC 2253 [WKH97].

Der Gültigkeitszeitraum wird im *Validity*-Element mit den Elementen *NotBefore* und *NotAfter* angegeben. Diese Angaben sind XML-konforme Zeitangaben in UTC-Zeit. Es ist zulässig, in Anwendungsfällen, in denen ein Sachverhalt lediglich zu einem bestimmten Prüfzeitpunkt festgestellt wird, nur die *NotBefore*-Angabe aufzunehmen.

Die Grammatik-Referenz wird z. B. mittels des SHA256-Hashalgorithmus gebildet und kann neben Hashwert und Algorithmus-Angabe auch die optionalen Elemente *URI* (Ablageort der Grammatik) und *GrammarServer* enthalten. Das letztgenannte weist auf einen Grammatikserver hin, bei dem über den Grammatik-Hash die Grammatik abgerufen werden kann.

```
<GrammarReference>
   <DigestMethod>sha256</DigestMethod>
   <DigestValue>BWGaLRe9975oI ... lwvhSs9XjNBXA=</DigestValue>
   <URI>http://security.example.com/grammars/m572.gcg</URI>
   <GCGrammarServer>http://gcgsrv.example.com/grammarserver</GCGrammarServer>
</GrammarReference>
```

Für die Algorithmen-Bezeichner sind *sha256*, *sha384* und *sha512* zulässig. Grundsätzlich lassen sich hier weitere Verfahren als erlaubt aufnehmen (z. B. die anstehende SHA-3-Familie).

Innerhalb eines *Assertion*-Elementes wird ein *ParseTree*-Element eingefügt, das die XML-kodierte Form des AST der Zusicherung nach Abschnitt 4.6.3.1 enthält.

Das *Assertion*-Element kann auch mehrfach aufgenommen werden. Für diesen Fall kann es optional sowohl ein *Validity*- als auch ein *GrammarReference*-Element enthalten. Diese Angaben übersteuern die *Validity*- und *GrammarReference*-Elemente auf der Ebene des *Assertion*-Elementes bei der Interpretation des AST der vorliegenden Zusicherung.

Als optionales Element kann zusätzlich das *Nonce*-Element aufgenommen werden, das als Frische-Nachweis in Bezug auf eine konkrete Zusicherungsanfrage verwendet werden kann. Das Textelement dieses XML-Knotens enthält die base64-kodierte Form des binären Nonce-Wertes:

```
<Nonce>AoqSG5dNfIcBZ3GVx+P90Q==</Nonce>
```

Die *Assertion*-Elemente können darüber hinaus die Primärsignatur aufnehmen. Diese besteht aus den beiden base64-kodierten Elementen *SignatureAlgorithm* und *SignatureValue* sowie den optionalen Elementen *X509Certificate*, *PrimarySigner* und *PrimaryAuthorityKeyIdentifier*. *PrimarySigner* ist dabei wie das *Issuer*-Element aufgebaut.

Der Einsatz der Primärsignatur ist z. B. dann sinnvoll, wenn der Primärtext rechtsverbindlich mit einer qualifizierten Signatur unterzeichnet wird, das GC-Zertifikat jedoch mit einem fortgeschrittenen Schlüssel transportgesichert wird.

Der Nonce-Wert kann nicht pro *Assertion* angegeben werden, da die Frische der gesamten Nachricht hinsichtlich einer Anfrage nachgewiesen werden soll.

Innerhalb des *Extensions*-Elementes werden Informationen über den Aussteller in Form des optionalen *Authority Key Identifiers* angegeben. Daneben können Informationen zu ggf. erhältlichen Sperrinformationen in einem *AuthorityInformationAccess*- und einem *CRLDistributionPoint*-Element eingefügt werden. Diese Elemente sind in Aufbau und Bedeutung stark an die gleichnamigen Informationen aus RFC 5280 [Day73] angelehnt.

Die gesamten XML-Daten werden nach dem XML Standard Digital Signature signiert. Hierbei wird lediglich der Signaturwert in das signierte Dokument aufgenommen. Die optionalen Schlüsselinformationen nach Abschnitt 2.3.3 (*KeyInfo*) brauchen i. d. R. nicht in das Zertifikat aufgenommen werden, da im *Issuer*-Element und im *Authority Key Identifier* ausreichende Informationen vorliegen, um ein entsprechendes Zertifikat abrufen und für die Prüfung verwenden zu können.

Das XML-Format wurde aus zwei Gründen als Kodierung für die Zertifikate gewählt. Zum einen können zur Festlegung und Prüfung von Beschränkungen die XPath-Werkzeuge verwendet werden. Zum anderen ist das XML-Format durch die Tags selbst erklärend. Abgesehen von der Auswertung per XPath lassen sich aber für Übertragung und Speicherung durchaus auch andere Formate verwenden. Innerhalb des Prototyps wird ein Zwischenformat in Form von S-Expressions verwendet, dessen Informationsgehalt gleich der XML-Darstellung ist. Es wurden entsprechende Datenkonverter für die Umwandlung von S-Expressions in XML und umgekehrt implementiert.

So kann das bereits bekannte XML-Fragment

```
<Dienst>
   <Dienstname>
      <GCELEMENT type="TOKEN">ssh</GCELEMENT>
   </Dienstname>
   <Portnummer>
      <GCELEMENT type="INT">22</GCELEMENT>
   </Portnummer>
</Dienst>
```

gleichwertig durch eine deutlich kompaktere S-Expression dargestellt werden:

```
(Dienst (Dienstname 'ssh') (Portnummer 22))
```

Hierbei ist die Information, dass es sich bei *ssh* um ein Token handelt ist bei dieser Kodierung durch die einfachen Hochkommata enthalten; Lexeme werden wie im Primärtext kodiert.

Bei den kurzen Texten der Beispiele dieser Arbeit erhöht die XML-Kodierung die Zeichenmenge der Zusicherung, mit der alternativen Kodierung lässt sich jedoch die zu übertragende Datenmenge reduzieren. In Tabelle 5.1 wird dies für einige Testfälle dargestellt. Erst wenn die Primärtexte

	Primärtext	XML-Darstellung	S-Expressions
waf.1.tst	309	751	194
m173.1.tst	865	1368	388
m173.3.tst	753	1133	312
m572.2.tst	134	650	134
m487.3.tst	433	822	240
m487.5.tst	461	931	285

Tabelle 5.1: Exemplarischer Vergleich der Kodierungsverfahren hinsichtlich der Datenmenge

eines Anwendungsfalles längere Textpassagen enthalten, kann auch mit der XML-Kodierung eine Reduktion der Datenmenge erzielt werden.

Eine Alternative zu den S-Expressions bietet noch die bei X.509-Zertifikaten übliche ASN.1-Kodierung, mit der ebenfalls sehr kompakte Kodierungen erzielt werden können. Eine Verbesserung aller Verfahren kann darüber hinaus erzielt werden, wenn die Nicht-Terminale einer Grammatik z. B. in alphabetischer Reihenfolge sortiert und nummeriert werden und statt der teilweise langen Bezeichner nur ihre Indizes übertragen werden.

Speicher-reduzierenden Kodierungsverfahren für Zusicherungen können daher im Bedarfsfall eingesetzt werden, werden aber im Rahmen der prototypischen Implementierung nicht weiter betrachtet.

5.1.7 Anwendungsbewertung

Die Anwendung der entwickelten Werkzeuge wurde mit einer ganzen Reihe von Anwendungsfällen geprüft. Hierzu wurden vor allem Formulierungen für Zusicherungen gewählt, die Fragestellungen aus dem IT-Grundschutz und vergleichbaren Auditierungsaufgaben abdecken. Daneben wurden aber auch andere Anwendungsfälle wie die Berichterstattung nach einer periodischen Durchführung von Schwachstellen-Scans (z. B. mit dem Schwachstellenscanner Nessus) und die Automatisierung von Meldungen des aide-Intrusion-Detection-Systems betrachtet.

5.1.7.1 Beispiel einer Prüfung

Als Beispiel für eine Prüfung gemäß IT-Grundschutz wird eine Zusicherung zum Einsatz von Zugangssicherungsverfahren für Rechenzentren gemäß Maßnahmenkatalog M1.73 vorgestellt.

Bei der Auditierung von Rechenzentren wird i. d. R. das Verfahren betrachtet, mit dem der Zugang zu den Rechenzentrumsräumen für Bedienstete und Besucher kontrolliert und dokumentiert wird. Der folgende Text dient als Anfangspunkt zur Erstellung der Grammatik.

> Für das Rechenzentrum in *Haus 2, Raum RZ-01* wurden die Zugangskontrollen auditiert.
> Bei der Autorisierungsprüfung werden die Kriterien
> - Wissen
> - Besitz
> abgefragt.
> Besucher werden eindeutig verantwortlichen Personen zugeordnet.
> Die durchgehende Beaufsichtigung der Besucher ist sichergestellt.
> Die Zutritte zum Rechenzentrum werden für Berechtigte elektronisch protokolliert.
> Die Zutritte zum Rechenzentrum werden für Besucher schriftlich protokolliert.
> Die Austritte aus dem Rechenzentrum werden für Berechtigte elektronisch protokolliert.
> Die Austritte aus dem Rechenzentrum werden für Besucher schriftlich protokolliert.
> Die schriftliche Protokollierung wird durch den Beaufsichtigenden abgezeichnet.
> Beim Zugang zum Rechenzentrum kommt eine Vereinzelungsschleuse zum Einsatz.
> Für die Berechtigungsnachweise wird eine Anti-Passback-Funktion eingesetzt.

Die daraus erstellte Grammatik lautet damit

```
Satz   :       'Für das Rechenzentrum in ' Ort
               ' wurden die Zugangskontrollen auditiert.\n'
               Autorisierung Besucher Protokollierung Vereinzelung
               Passback;

Ort    :       STRING ;

Autorisierung: 'Bei der Autorisierungsprüfung werden die Kriterien\n'
               AutKriterium+ 'abgefragt.\n' ;

AutKriterium: '- '! 'Besitz' '\n'!   |
              '- '! 'Wissen' '\n'!   |
              '- '! 'biometrische Merkmale' '\n'!;

Besucher:     Zuordnung Aufsicht?  ;

Zuordnung:    'Besucher werden eindeutig verantwortlichen Personen '!
              'zugeordnet' '.\n'!  ;

Aufsicht:     'Die durchgehende Beaufsichtigung der Besucher ist '!
              'sichergestellt' '.\n'!  ;

Protokollierung:  Zutritt Austritt ;

Zutritt:      PZ_Berechtigt PZ_Besucher ;

Austritt:     PA_Berechtigt PA_Besucher ;

PZ_Berechtigt: 'Die Zutritte zum Rechenzentrum werden für Berechtigte '
```

```
               ProtArt ' protokolliert.\n' ;

PZ_Besucher:  'Die Zutritte zum Rechenzentrum werden für Besucher '
              ProtArt ' protokolliert.\n' Abzeichnung?  ;

PA_Berechtigt: 'Die Austritte aus dem Rechenzentrum werden für Berechtigte '
               ProtArt ' protokolliert.\n' ;

PA_Besucher:   'Die Austritte aus dem Rechenzentrum werden für Besucher '
               ProtArt ' protokolliert.\n' Abzeichnung?  ;

ProtArt!:      'nicht' | 'elektronisch' | 'schriftlich' ;

Abzeichnung:   'Die schriftliche Protokollierung wird durch den Beaufsichtigenden '!
               'abgezeichnet' '.\n'!  ;

Vereinzelung:  'Beim Zugang zum Rechenzentrum kommt '
               K_eine ' Vereinzelungsschleuse zum Einsatz.\n' ;

K_eine!:       'keine' | 'eine';

Passback:      'Für die Berechtigungsnachweise wird ' K_eine
               ' Anti-Passback-Funktion eingesetzt.\n' ;
```

Das zugehörige XML-kodierte grammatikbasierte Zertifikat wurde in Anhang E aufgenommen.

5.1.7.2 Grammatikerstellung

Bei allen Beispielen zeigte sich, dass die Methode der Erstellung einer Grammatik durch Umformungen von Beispielzusicherungen gut anzuwenden war. Für den Fall optionaler Zusicherungsanteile ist es dabei jedoch wichtig, dass die Formulierungen der Zusicherung dahingehend optimiert sind, dass optionale Teile aus dem Text der Zusicherung ausgeschnitten werden können, ohne die Korrektheit des Textes zu verletzen. Es ist daher ggf. eine entsprechende Anpassung der Formulierungen notwendig.

Ein Beispiel für eine solche Anpassung ist die Nennung zweier jeweils optionaler Eigenschaften in einem Satz *Der Untersuchungsgegenstand erfüllte die Anforderung A und die Vorgabe B*. Durch die Verwendung der Konjunktion ist ein Weglassen einer Teilzusicherung nicht einfach möglich. In solchen Fällen sind Formulierungen mit jeweils eigenständigen Sätzen für Anforderung A und Vorgabe B zu bevorzugen: *Der Untersuchungsgegenstand erfüllte die Anforderung A. Der Untersuchungsgegenstand erfüllte die Vorgabe B.*

Bei Aufzählungen lässt sich eine quasi-tabellarische Form z. B. mit Spiegelstrichen sehr einfach in eine Grammatik überführen. Es lassen sich jedoch auch für aufzählende Formulierungen wie *Für das Beispiel steht Version A, Version B und Version C zur Auswahl*, bei der alle Anteile optional sind, sofern mindestens ein Anteil vorkommt, entsprechende Grammatikstrukturen angeben.

Das Grammatikfragment der Form

```
Optionen: 'Für das Beispiel steht ' Option L1_Option? ' zur Auswahl. \n'

Option!:    'Version A' | 'Version B' | 'Version C' | 'Version D' ;
```

```
L1_Option!: L2_Option* ' und ' Option;

L2_Option!: ', ' Option;
```

erzeugt für das angegebene Beispiel folgende strukturierte Daten:

```
<Optionen>
   <GCELEMENT type="TOKEN">Version A</GCELEMENT>
   <GCELEMENT type="TOKEN">Version B</GCELEMENT>
   <GCELEMENT type="TOKEN">Version C</GCELEMENT>
</Optionen>
```

Wird bei der Erstellung der Grammatik die Eindeutigkeit verletzt, so wird dieses bei der Erstellung der Parser bemerkt und als Fehler angezeigt. Ist beispielsweise die letzte Regel des oben stehenden Beispiels fälschlicherweise als

```
L2_Option!: ', ' Option?;
```

ausgeführt (Nicht-Terminal *Option* wurde fälschlicherweise mit '?' quantifiziert), so werden Fehlermeldungen der Form

```
warning(200): var_r.g:43:7: Decision can match input such as
              "'Version C'" using multiple alternatives: 1, 2
```

erzeugt. Bei Verwendung der Entwicklungsumgebung antlrworks kann dies auch graphisch wie in Abbildung 5.5 angezeigt werden:

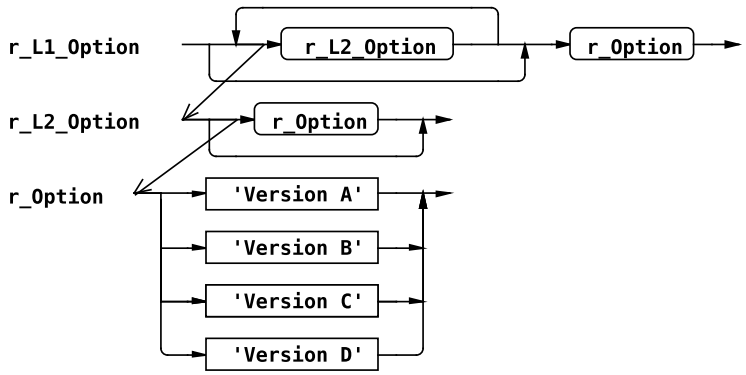


Abbildung 5.5: Graphische Darstellung für uneindeutige Grammatik

Werden in Grammatiken Nicht-Terminale wie im Beispiel gezeigt mehrfach in Regeln verwendet, kann es zu Mehrdeutigkeiten kommen. Diese mehrfache Verwendung geht über die Gestaltungsmöglichkeiten der vorgestellten Top-Down-Methode hinaus und stellt eine Grenze zur Erstellung von Grammatiken für wenig erfahrene Anwender dar.

5.1.7.3 Erstellung von Zusicherungen

Von den drei vorgestellten Eingabeverfahren einfacher Texteditor, Grammatik-gesteuerter Editor und Formular-basierte Eingabe hat sich der einfache Texteditor in den Fällen der Primärsignatur, des hybriden Verfahrens und des kombinierten Verfahrens als weitgehend unpraktikabel erwiesen. Da bei den genannten Verfahren der Primärtext signiert wird, muss zur Prüfung eine zeichengenaue Rückgewinnung erfolgen und somit besteht für den Primärtext kein Freiheitsgrad. Lediglich bei Verzicht auf eine Rückgewinnung (bei ausschließlicher Verwendung der Sekundärsignatur) wäre eine tolerantere Grammatik z. B. durch beliebige Whitespace-Folgen denkbar. Dieser Anwendungsfall erfüllt jedoch nicht die Anforderung an die Erhöhung der Verbindlichkeit und kann daher als nicht relevant eingestuft werden.

Bei der Verwendung der Grammatik-gesteuerten Editoren auf Basis der automatisch generierten XText-Grammatiken ist diese Fehlerquelle ausgeschlossen. Hier zeigte sich bei Tests mit den realisierten Beispielen, dass die Eingabe durch die automatisch vorgegebenen, auswählbaren Textbausteine sehr schnell erfolgen kann. Ebenfalls werden bestimmte Formatierungen wie z. B. die Anführungszeichen bei Eingabe von freien Zeichenketten (Typ *String*) automatisch erzwungen.

Als nachteilig bei dieser Art der Eingabe haben sich Formulierungen in Zusicherungen erwiesen, deren Bedeutung aus dem nächsten vorgegebenen Textelement nicht erfasst werden kann. So werden im o. g. Beispiel nach dem Setzen des Textbausteins *Beim Zugang zum Rechenzentrum kommt* als nächste Auswahl *eine* und *keine* angezeigt, ohne dass das nächste Element *Vereinzelungsschleuse zum Einsatz.* sichtbar ist. Es ist also notwendig, dass der Anwender auch bei der XText-unterstützten Eingabe die jeweils geforderten Formulierungen ungefähr kennt, um die korrekte Auswahl treffen zu können. Damit ist es also nur in den Fällen einsetzbar, in denen dieses Wissen vorausgesetzt werden kann bzw. durch entsprechende Vorgaben in Form von Prüfvorschriften und Formulierungsvorgaben oder -beispielen vermittelt wird.

Für Fälle, in denen dies nicht erfüllt werden kann, müssen die vorgegebenen Formulierungen so gestaltet werden, dass die bei der Eingabe auswählbaren Textbausteine ausreichend selbsterklärend sind.

Bei der Verwendung von Formular-basierten Eingaben mit direkter Generierung der strukturierten Daten kann das notwendige Kontextwissen durch entsprechende Formulierungen bei der Beschriftung der Eingabe- und Auswahlfelder vermittelt werden. Daneben können auch sog. Tool-Tips und Hilfefunktionen verwendet werden, um zusätzliche Informationen zu geben.

Das zur Erhöhung der Verbindlichkeit gewünschte Signieren einer natürlichsprachlichen Erklärung wird bei der Formular-basierten Eingabemethode durch einen nachgelagerten Rückgewinnungsschritt erreicht. In diesem werden die strukturierten Daten in den korrespondierenden Text umgesetzt und dem Anwender im Rahmen einer Signaturerstellung angezeigt.

Das Formular-basierte Verfahren erreicht auf diese Weise sowohl eine einfache Erhebung der Daten als auch die Erstellung einer Primärsignatur.

5.2 Semantik der grammatikbasierten Zertifikate

Die grammatikbasierten Zertifikate können sowohl die normale Zertifikatssignatur als auch die optionale Signatur über die Primärtexte enthalten. Welche Bedeutung diesen jeweiligen Signaturen zukommt, d. h. welche Schlussfolgerungen ein Akzeptor eines solchen Zertifikats aus den Zertifikatsdaten und den referenzierten Grammatiken ziehen kann, wird durch eine Analyse auf Basis der SVO-Logik gezeigt. Die Festlegung einer Semantik erfolgt dabei durch die Definition von zwei neuen Ableitungsregeln als Ergänzung der bestehenden Regeln der SVO-Logik. Diese erweiterten Regeln werden dann dazu verwendet, die Auswertung eines grammatikbasierten Zertifikats formal durchzuführen.

Bei der Modellierung der Zertifikate werden die Mechanismen, die von den standardisierten X.509-Zertifikaten übernommen wurden, nicht weiter betrachtet, da es sich hierbei um etablierte Verfahren handelt. Es wird daher davon ausgegangen, dass alle Personenzertifikate gegen bereits vorhandene Vertrauensanker geprüft werden können und die grammatikbasierten Zertifikate aktuell gültig sind, so dass keine OCSP- oder CRL-Prüfungen betrachtet werden müssen. Damit werden die Zusicherungen auf die charakteristischen Bestandteile der grammatikbasierten Zertifikate reduziert.

Im Kern enthält ein grammatikbasiertes Zertifikat Z den AST X und den Hashwert $\mathrm{H}(G)$ über die zugrunde liegende Grammatik G. Alle weiteren Informationen wie Nonce-Wert N, Aussteller, Seriennummer, Datum, usw. werden in den Meta-Daten $\mathcal{M}$ i. S. einer Menge zusammengefasst. Der zu signierende Teil des Zertifikats wird durch den Funktionsausdruck $\mathrm{GC}(X, \mathrm{H}(G), \mathcal{M})$ dargestellt. Dieser Ausdruck beschreibt die Kodierung der angegebenen Daten als grammatikbasiertes Zertifikat ohne die Signatur. Das gesamte Zertifikat ist damit

$$Z = \lfloor \mathrm{GC}(X, \mathrm{H}(G), \mathcal{M}) \rfloor_{K_B^{-1}}$$

Werden Teile der Meta-Daten benötigt, wie beispielsweise der Nonce-Wert zum Nachweis der Frische, so können diese aus $\mathcal{M}$ herausgezogen werden, so dass z. B. $(N, \mathcal{M}') = \mathcal{M}$ ist:

$$Z = \lfloor \mathrm{GC}(X, \mathrm{H}(G), N, \mathcal{M}') \rfloor_{K_B^{-1}}$$

Zur Auswertung muss der Empfänger den zweiten Wert $\mathrm{H}(G)$ als Hashwert über die Grammatik erkennen können. Dies setzt nach (Ax15) der SVO voraus, dass der Empfänger die Grammatik G besitzt. Mit dieser kann dann mittels Φ_G^{-1} eine Rückgewinnung durchgeführt werden, die aus dem AST X den Primärtext p erstellt.

5.2.1 Axiome zur Interpretation

Zur Darstellung des Zusammenhangs zwischen strukturierten Daten und Primärtext auf Basis der Grammatik G wird das 3-stellige Prädikat *GCV* eingeführt:

$$X = \Phi_G(p) \wedge X \neq \bot \quad \Leftrightarrow \quad p = \Phi_G^{-1}(X) \wedge p \neq \bot \quad \Leftrightarrow \quad \mathrm{GCV}(X, G, p) \tag{5.1}$$

Diese Darstellung ist an das SVO-Prädikat zur Signaturprüfung $\mathrm{SV}(X, K, Y)$ angelehnt. Somit wird das Prädikat *GCV* mit (4.1) und (4.2) genau dann wahr, wenn die Rückgewinnung von p aus X mit $\Phi_G^{-1}(X)$ möglich ist oder umgekehrt aus $X = \Phi_G(p)$ aus p gebildet werden kann. Damit gilt

$$\mathrm{GCV}(X, G, p) \quad \Rightarrow \quad p \in \mathscr{L}(G) \tag{5.2}$$

Definition 1. *Das Äußern eines strukturierten Datums X im Kontext eines grammatikbasierten Zertifikats zusammen mit einem Hashwert einer Grammatik G, mit der aus einem Primärtext p das strukturierte Datum X erzeugt werden kann, ist gleichbedeutend mit dem Äußern von p.*

Die Interpretation eines grammatikbasierten Zertifikates wird festgelegt als:

$$Q \text{ says } \mathrm{GC}(X, \mathrm{H}(G), \mathcal{M}) \wedge \mathrm{GCV}(X, G, p) \rightarrow Q \text{ says } p \tag{GC1}$$

5.2.2 Kompetenz

Zur Übernahme von Aussagen in die eigenen Überzeugungen ist jedoch noch die Betrachtung der Kompetenz notwendig. Der Kompetenzbereich (Jurisdiktion) wird dabei ebenfalls durch die Referenzierung einer Grammatik definiert.

Der Sprachumfang $\mathscr{L}(G)$ einer Grammatik ist in extensionaler Darstellung eine Menge von Worten $\{p_1, p_2, \ldots\}$ (hier auch als Sätze oder Zusicherungen bezeichnet). Daher wird eine entsprechende Kurzschreibweise festgelegt.

Definition 2. *Die Kompetenz (Jurisdiktion) über den Sprachumfang einer Grammatik (Menge von Aussagen) entspricht der Konjunktion der Kompetenz über die einzelnen Aussagen der Menge, geschrieben als*

$$\begin{aligned} P \textit{ controls } \mathscr{L}(G) = &\quad P \textit{ controls } \{p_1, p_2, \ldots\} \\ =_{\mathrm{def}} &\; P \textit{ controls } p_1 \wedge P \textit{ controls } p_2 \wedge \ldots \qquad \textit{für } p_\nu \in \mathscr{L}(G) \end{aligned} \tag{5.3}$$

Für eine eingeschränkte Sprache $\mathscr{L}(G, r)$ nach (4.14) gehört p nur dann zum Sprachumfang, wenn sowohl p dem Sprachumfang der unbeschränkten Grammatik angehört, als auch das korrespondierende X die Anforderungen an die Beschränkungen erfüllt. Daher kann folgende Regel aufgestellt werden:

$$P \text{ controls } \mathscr{L}(G, r) \wedge \mathrm{GCV}(X, G, p) \wedge \mathrm{GCR}(X, r) \rightarrow P \text{ controls } p \tag{GC2}$$

Für $r = \perp$ ergibt sich mit (4.10) und (4.15) eine einfachere Form dieser Implikation:

$$P \text{ controls } \mathscr{L}(G) \wedge \mathrm{GCV}(X, G, p) \rightarrow P \text{ controls } p$$

5.2.3 Korrektheit

Zum Nachweis der Korrektheit von (GC1) wird zunächst die Prämisse als falsch angenommen. Da mit einer falschen Prämisse auf jede Konklusion geschlossen werden kann, ist auch (GC1) für diesen Fall wahr.

Ist die linke Seite wahr, so ist die Regel ebenfalls richtig, da das Prädikat $\mathrm{GCV}(X, G, p)$ nach (5.2) genau den in der Definition 1 geforderten Zusammenhang zwischen den Parametern X und p sicherstellt und somit die Implikation wahr ist. Somit kann aus der Äußerung des Zertifikats die Äußerung von p geschlossen werden. Damit ist (GC1) für alle möglichen Belegungen der Prämissen wahr und damit korrekt. □

Für das Axiom (GC2) gilt: ist die Prämisse falsch, so ist (GC2) wahr. Ist die Prämisse wahr, so ist aus $\mathrm{GCV}(X, G, p)$ nach (5.2) $p \in \mathscr{L}(G)$ und damit $X = \Phi_G(p)$ ableitbar. Da auch $\mathrm{GCR}(X, r)$ wahr ist, folgt über (4.14), dass $p \in \mathscr{L}(G, r)$ ist. Damit ist aber nach (5.3) auch Q controls p wahr. □

5.3 Vergleich der Signaturverfahren

Um die Aussagekraft signierter Zusicherungen in primärer und sekundärer Form und die jeweils notwendigen Prämissen und Ableitungsschritte miteinander vergleichen zu können, werden diese in SVO-Logik modelliert. Allen Modellierungen gemeinsam sind dabei folgende Annahmen:

Der Primärtext p korrespondiert mit den strukturierten Daten X durch eine Kodierung mittels der Grammatik G, so dass $\mathrm{GCV}(X, G, p) = \mathbf{T}$ ist.

Der Bestätiger B signiert die von ihm gegebenen Zusicherungen mit einem Schlüssel K_B^{-1}, dessen öffentliches Pendant K_B beim Akzeptor A bekannt und dem Eigentümer B zugeordnet ist. Desgleichen räumt A dem Bestätiger die Kompetenz über den Sprachraum $\mathscr{L}(G)$ der Grammatik G ohne Beschränkungen ein. Die Kompetenzbestätigung liegt A bereits vor und kann mit der einfachen Form von (GC2) angewendet werden. B hat zuvor von A im Rahmen der Bitte um eine Bestätigung eine Anfragenummer (Nonce-Wert N) erhalten, die er im Rahmen seiner Zusicherung als Frische-Nachweis beifügen kann.

Es werden in jedem Teilkapitel die Prämissen mit P*x* gezählt, die Schlussfolgerungen mit C*x*. Bei allen Schritten der Herleitung werden jeweils die angewandten SVO-Axiome gemäß Anhang A sowie die verwendeten Prämissen und Zwischenergebnisse angegeben.

Das Ziel jeder Herleitung ist die Akzeptanz der Aussage p bzw. der lokalen Sicht $\langle p \rangle_{*A}$ durch A.

5.3.1 Primärtext mit Primärsignatur (p-p)

Der Bestätiger signiert unmittelbar den Primärtext und überträgt diesen samt Signatur.

Die Prämissen sind:

A believes $\mathrm{PK}_\sigma(B, K_B)$	K_B ist für A der öffentliche Schlüssel von *B*.	(P1)
A believes $\mathrm{fresh}(N)$	*A* betrachtet *N* als frisch.	(P2)
A believes B controls $\mathscr{L}(G)$	*B* hat Kompetenz über Aussagen aus Grammatik *G*.	(P3)
A received $\lfloor p \rfloor_{K_B^{-1}}$	*A* empfängt die signierte Aussage *p*.	(P4)
A believes A received $\lfloor \langle p \rangle_{*A} \rfloor_{K_B^{-1}}$	Interpretation von (P4) durch *A*	(P5)
A believes $\mathrm{SV}(\lfloor \langle p \rangle_{*A} \rfloor_{K_B^{-1}}, K_B, \langle p \rangle_{*A})$	A kann die Signatur positiv prüfen.	(P6)
A believes $\mathrm{GCV}(X, G, p)$	A kann prüfen, dass p Element von $\mathscr{L}(G)$ ist.	(P7)

(5.4)

Im ersten Schritt stellt der Akzeptor die Authentizität der empfangenen Nachricht fest

$$A \text{ believes } B \text{ said } (\langle p \rangle_{*A}) \tag{C1}$$

durch Ax6, Nec, P1, P5, P6, Ax1, MP

Somit ist zunächst die Authentizität des Primärtextes nachgewiesen. Jedoch kann keine Aussage zur Frische dieses Datums gemacht werden, es kann sich also auch um eine Wiedereinspielung handeln.

Um die Frische dennoch nachweisen zu können, besteht die Möglichkeit, in der erteilten Zusicherung die Anfragenummer (Nonce) aufzunehmen. Somit ist

$$p = F(p', N) \tag{P8}$$

Da A von der Frische von N überzeugt ist, muss auch eine Zusicherung mit N frisch sein.

$$A \text{ believes } \mathrm{fresh}(\langle F(p', N) \rangle_{*A}) \tag{C2'}$$

durch Ax19, Nec, P2, Ax1, MP

$$A \text{ believes } \mathrm{fresh}(\langle p \rangle_{*A}) \tag{C2}$$

durch P8

Durch die Frische kann die Nachricht der laufenden Anfrage zugeordnet werden (aktueller Protokolllauf)

$$A \text{ believes } B \text{ says } (\langle p \rangle_{*A}) \tag{C3}$$

durch Ax21, Nec, C1, C2, Ax1, MP

Die Ableitung der Kompetenz für die Aussage p erfolgt durch die Grammatikprüfung

$$A \text{ believes } B \text{ controls } \langle p \rangle_{*A} \tag{C4}$$

durch GC2, Nec, P3, P7, Ax1, MP

Abschließend erfolgt die Übernahme der Aussage in die eigene Sicht von A

$$A \text{ believes } \langle p \rangle_{*A} \quad \square \tag{C5}$$

durch Ax18, Nec, C3, C4, Ax1, MP

5.3.2 Sekundärdaten mit Sekundärsignatur (s-s)

Bei diesem Verfahren erstellt der Bestätiger ein grammatikbasiertes Zertifikat durch Signatur der strukturierten Daten $X = \Phi_G(p)$.

Die zugehörigen Prämissen lauten:

$$\begin{aligned}
&A \text{ believes } \mathrm{PK}_\sigma(B, K_B) && \text{(P1)}\\
&A \text{ believes } \mathrm{fresh}(N) && \text{(P2)}\\
&A \text{ believes } B \text{ controls } \mathscr{L}(G) && \text{(P3)}\\
&A \text{ received } \lfloor \mathrm{GC}(X, \mathrm{H}(G), N, \mathcal{M}) \rfloor_{K_B^{-1}} && \text{(P4)}\\
&A \text{ believes } A \text{ received } \lfloor \mathrm{GC}(\langle X \rangle_{*A}, \mathrm{H}(G), N, \langle \mathcal{M} \rangle_{*A}) \rfloor_{K_B^{-1}} && \text{(P5)}\\
&A \text{ believes } \mathrm{SV}(\lfloor \mathrm{GC}(\langle X \rangle_{*A}, \mathrm{H}(G), N, \langle \mathcal{M} \rangle_{*A}) \rfloor_{K_B^{-1}}, K_B, \mathrm{GC}(\langle X \rangle_{*A}, \mathrm{H}(G), N, \langle \mathcal{M} \rangle_{*A}) && \text{(P6)}\\
&A \text{ believes } \mathrm{GCV}(\langle X \rangle_{*A}, G, \langle p \rangle_{*A}) && \text{(P7)}\\
& && (5.5)
\end{aligned}$$

Die Ableitung der Authentizität erfolgt auf Basis der Signatur

$$A \text{ believes } B \text{ said } \mathrm{GC}(\langle X \rangle_{*A}, \mathrm{H}(G), N, \langle \mathcal{M} \rangle_{*A}) \tag{C1}$$

durch Ax6, Nec, P1, P5, P6, Ax1, MP

Für die Ableitung der Frische der Nachricht wurde vom Ersteller der Nonce-Wert N im Zertifikat aufgenommen.

$$A \text{ believes } B \, \mathrm{fresh}(\mathrm{GC}(\langle X \rangle_{*A}, \mathrm{H}(G), N, \langle \mathcal{M} \rangle_{*A})) \tag{C2}$$

durch Ax19, Nec, P2, Ax1, MP

Damit kann die Nachricht zum aktuellen Lauf zugeordnet werden.

$$A \text{ believes } B \text{ says } \mathrm{GC}(\langle X \rangle_{*A}, \mathrm{H}(G), N, \langle \mathcal{M} \rangle_{*A}) \tag{C3}$$

durch Ax21, Nec, C1, C2, Ax1, MP

Zur Ableitung der Aussage in primärer Darstellung wird die Rückgewinnung von p durchgeführt.

$$A \text{ believes } B \text{ says } (\langle p \rangle_{*A}) \tag{C4}$$

durch GC1, Nec, C3, P7, Ax1, MP

Dem Bestätiger kann die Kompetenz für die Aussage p zugeordnet werden.

$$A \text{ believes } B \text{ controls } \langle p \rangle_{*A} \tag{C5}$$
durch GC2, Nec, P3, P7, Ax1, MP

Die relevante Teilaussage ist ebenfalls eine aktuelle Äußerung des Bestätigers.

$$A \text{ believes } B \text{ says } \langle p \rangle_{*A} \tag{C6}$$
durch Ax17, Nec, C4, Ax1, MP

Damit kann die Aussage durch den Akzeptor übernommen werden.

$$A \text{ believes } \langle p \rangle_{*A} \quad \square \tag{C7}$$
durch Ax18, Nec, C5, C6, Ax1, MP

5.3.3 Sekundärdaten mit Primärsignatur (s-p)

Bei diesem Szenario wird durch den Bestätiger der Primärtext signiert, jedoch lediglich die Signaturdaten und die mit dem Primärtext korrespondierenden Sekundärdaten übertragen. Zur Signaturprüfung muss der Empfänger zunächst die Nachricht aus den strukturierten Daten und der Grammatik zurückgewinnen.

Der SVO-Term für signierte Daten $\lfloor X \rfloor_K$ stellt eine komplette signierte Nachricht (Daten und Signatur) dar [SO96]. Um eine getrennte Übertragung von Nachricht und Signatur zu modellieren wird daher eine alternative Darstellung notwendig. Im Folgenden werden nur Nachrichten betrachtet, die Signaturen auf Basis von asymmetrisch verschlüsselten Hashwerten verwenden. Damit lässt sich die SVO-Schreibweise für signierte Nachrichten entsprechend schreiben.

$$\lfloor X \rfloor_{K^{-1}} =_{\text{def}} (X, \{\mathrm{H}(X)\}_{K^{-1}}) \tag{5.6}$$

Zur Beschreibung des Falles s-p verwendeten Prämissen sind:

$$A \text{ believes } \mathrm{PK}_\sigma(B, K_B) \tag{P1}$$
$$A \text{ believes } \mathrm{fresh}(N) \tag{P2}$$
$$A \text{ believes } B \text{ controls } \mathscr{L}(G) \tag{P3}$$
$$A \text{ received } (X, \mathrm{H}(G), \{\mathrm{H}(p)\}_{K_B^{-1}}) \tag{P4}$$
$$A \text{ has } G \tag{P5}$$
$$A \text{ believes } A \text{ received } (\langle X \rangle_{*A}, \mathrm{H}(G), \{\langle \mathrm{H}(p) \rangle_{*A}\}_{K_B^{-1}}) \tag{P6}$$
$$A \text{ believes } \mathrm{SV}(\{\mathrm{H}(\langle p \rangle_{*A})\}_{K_B^{-1}}, K_B, \langle p \rangle_{*A}) \tag{P7}$$
$$A \text{ believes } \mathrm{GCV}(\langle X \rangle_{*A}, G, \langle p \rangle_{*A}) \tag{P8}$$

Zusätzlich wird wie beim p-p-Verfahren als (P9) angenommen, dass die Zusicherung p den Nonce-Wert enthält, also $p = F(p', N)$ ist.

In Prämisse P7 wird der in (5.6) gezeigte Zusammenhang berücksichtigt. Mit dem Prüfschlüssel K_B kann die kryptographische Verknüpfung zwischen dem mit K_B^{-1} signierten Hashwert und dem Primärtext p nachgewiesen werden.

Zur Auswertung der empfangenen Nachricht wird durch A im Gegensatz zu den anderen Fällen zunächst der Primärtext zurückgewonnen, ehe die Authentizität der signierten Daten abgeleitet wird.

$$A \text{ has } \langle X \rangle_{*A} \tag{C1}$$

durch Ax9, Nec, Ax12, Ax1, MP

Mit dieser Information kann A den Primärtext unter Nutzung der Funktion $F = \Phi_G^{-1}(X)$ zurückgewinnen.

$$A \text{ has } \Phi_G^{-1}(\langle X \rangle_{*A}) \quad = \quad A \text{ has } \langle p \rangle_{*A} \tag{C2}$$

durch Ax14, Nec, Ax1, MP

Gleichheit durch (4.2)

Extrahiere einen Teil der empfangenen Nachricht.

$$A \text{ believes } A \text{ received } \{\langle \mathrm{H}(p) \rangle_{*A}\}_{K_B^{-1}} \tag{C3}$$

durch Ax9, Nec, P6, Ax1, MP

Da A nach (C2) den Primärtext kennt, kann A den in der Signatur enthaltenen Wert $\langle \mathrm{H}(p) \rangle_{*A}$ als Hashwert von p erkennen.

$$A \text{ has } \mathrm{H}(\langle p \rangle_{*A}) \tag{C4}$$

durch Ax14, Nec, C2, Ax1, MP

Damit kann (C3) wie folgt geschrieben werden:

$$A \text{ believes } A \text{ received } \{\mathrm{H}(\langle p \rangle_{*A})\}_{K_B^{-1}} \tag{C5}$$

durch C3, C4

Es erfolgt die Auswertung der Signatur.

$$A \text{ believes } B \text{ said } \langle p \rangle_{*A} \tag{C6}$$

durch Ax6, Nec, C5, P1, P7, Ax1, MP

Wie bei der reinen Primärtext-Signatur kann über den eingebetteten Nonce-Wert auf die Frische der Nachricht geschlossen werden.

$$A \text{ believes } \text{fresh}(\langle F(p', N)\rangle_{*A}) \tag{C7'}$$

durch Ax19,P2,Ax1,MP

$$A \text{ believes } \text{fresh}(\langle p\rangle_{*A}) \tag{C7}$$

durch P9

Es erfolgt die Zuordnung der Nachricht zur aktuellen Zusicherungsanfrage.

$$A \text{ believes } B \text{ says } (\langle p\rangle_{*A}) \tag{C8}$$

durch Ax21, Nec, C7, C6, Ax1, MP

Die Berechtigung zur Zusicherung des Primärtextes wird abgeleitet.

$$A \text{ believes } B \text{ controls } \langle p\rangle_{*A} \tag{C9}$$

durch GC2, Nec, P3, P8, Ax1, MP

Die Aussage kann durch A übernommen werden.

$$A \text{ believes } \langle p\rangle_{*A} \quad \square \tag{C10}$$

durch Ax18, Nec, C8, C9, Ax1, MP

5.3.4 Diskussion der Aussagekraft

Durch die starke Kopplung zwischen den Darstellungsformen ist es möglich, in allen untersuchten Verfahren (p-p, s-s, s-p) die Aussage in die Überzeugungen des Akzeptors zu übernehmen.

Während der Fall p-p trivial ist, da die Auswertung des Wortproblems mittels (GC2) lediglich zur Feststellung der Autorisierung von B benötigt wird, sind in den beiden anderen Verfahren zusätzliche Verfahrensschritte notwendig, die über den Anwendungsfall normaler Authentisierung von Daten hinausgehen.

In allen drei Verfahren wird (GC2) angewendet, um den Schluss von der Berechtigung hinsichtlich der Grammatik auf die einzelne Aussage zu ermöglichen.

Wie beim Verfahren p-p kommt das Verfahren s-p ohne das spezielle Axiom (GC1) aus, da die Anwendung der Rückgewinnung ausschließlich zur Verfügbarmachung des Primärtextes dient. Liegt der Primärtext vor, so kann wiederum das Verfahren wie bei p-p angewendet werden. Damit nimmt die Kodierung der Aussage p in Form strukturierter Daten X die Rolle einer speziellen Übertragungskodierung ein und ist für die Sicherheitsbewertung vollständig transparent.

Der Hauptanwendungsfall der grammatikbasierten Zertifikate (s-s) nutzt hingegen das neue Axiom (GC1), um aus der Signatur der strukturierten Daten die Authentizität des Primärtextes abzuleiten.

Die dargestellten Schritte der Ableitung bilden auch eine Vorgabe für die durchzuführenden Prüfungen und Umformungen innerhalb von Implementierungen. Für jedes Verfahren kann daraus abgelesen werden, welche Daten oder Prüfergebnisse vorliegen müssen. So wird z. B. die Prüfung der Sprachzugehörigkeit von p als Prämisse zu $\mathrm{GCV}(X, G, P)$ bereits impliziert. Um diese Prämisse jedoch aufstellen zu können, muss der Anwender im Besitz der Grammatik sein und die Rückgewinnung durchgeführt haben. Die Voraussetzungen zur Aufstellung der Prämissen zur Kompetenz (B controls ...) bzw. deren Ableitung werden bei der Betrachtung der Delegation in Kapitel 5.5.5 betrachtet.

5.4 Grammatikverteildienst

Der in Kapitel 4.8 vorgestellte Grammatikverteildienst wurde als Webdienst implementiert. Basis der Implementierung ist eine MySQL-Datenbank, die die Grammatikdaten und die Autorisierungen speichert. Dazu wurde ein prototypischer Webdienst entwickelt, der Anfragen von Browsern bzw. Kommandozeilen-Werkzeugen verarbeitet.

5.4.1 Algorithmen

Als Sicherheitsmechanismen werden exemplarisch eingesetzt:

- X.509-Zertifikate mit 2048 Bit RSA-Schlüsseln und SHA256-RSA-Verschlüsselung als Signatur
- AES-256-Verschlüsselung für Grammatiken auf Basis von GPG.
- SHA256-Hashwerte für Grammatiken
- 15-stellige Nonce- und Token-Werte

Bei den Zertifikaten können auch andere Schlüssellängen und Hashverfahren eingesetzt werden. Bei den Verschlüsselungsalgorithmen können ebenfalls andere Algorithmen und Kodierungsverfahren gewählt werden. Es bietet sich dabei jedoch an, Werkzeuge einzusetzen, die mit dem Chiffrat auch den verwendeten Algorithmus ablegen (GPG, s/mime, u.a.), da diese Informationen nicht gesondert im Grammatikserver gespeichert werden.

5.4.2 Protokollablauf und Prüfschritte

Die Kommunikation zwischen Client-Programm und Dienst erfolgt dabei über HTTP-Post-Nachrichten. Die notwendigen Signaturen über die Konkatenation der zu signierenden Werte werden vor der URL-Kodierung durchgeführt und als zusätzlicher *signature*-Parameter übermittelt.

Im Folgenden werden die in den verschiedenen Funktionsbausteinen des Dienstes realisierten Sicherheitsoperationen im Rahmen von Sequenzdiagrammen dargestellt.

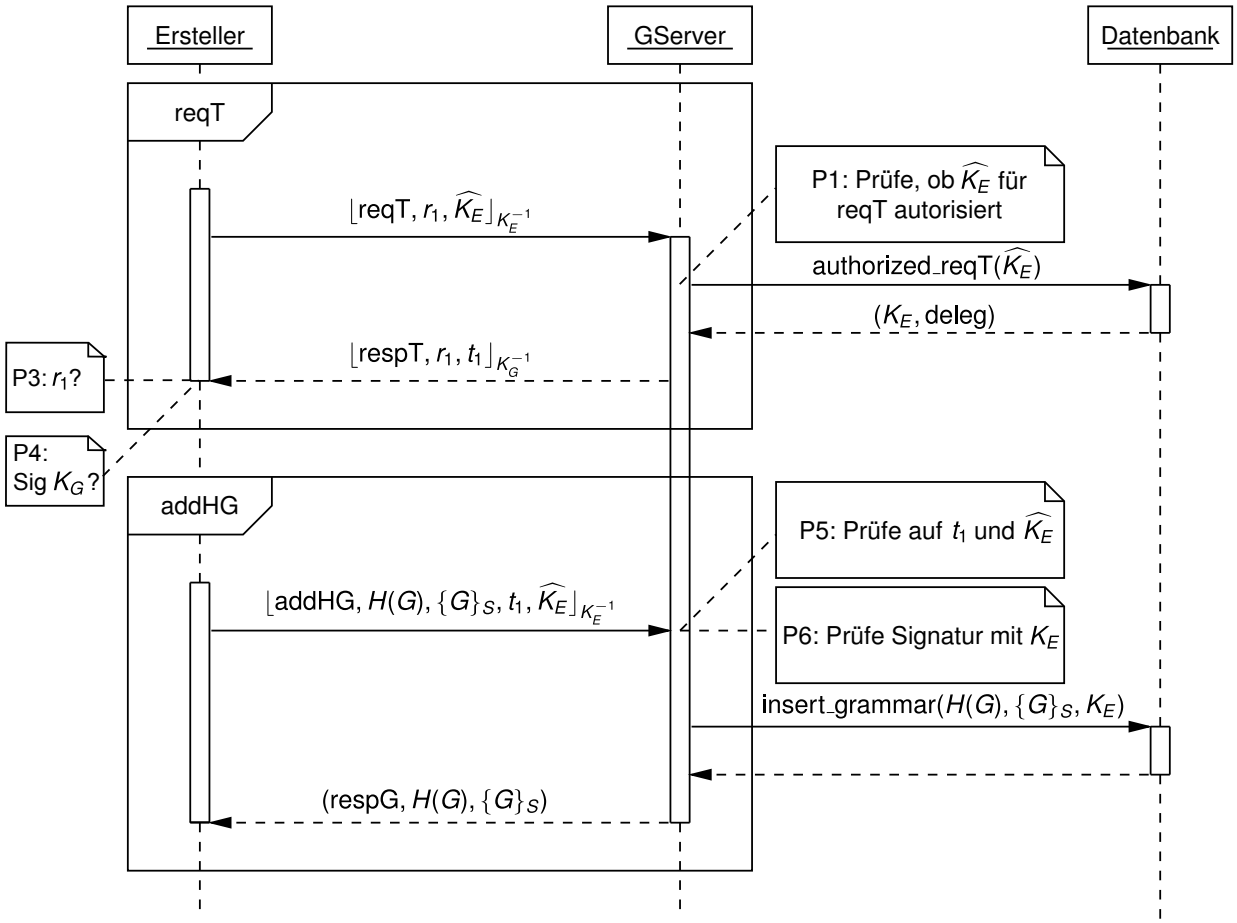


Abbildung 5.6: Token-Anforderung und Einstellen einer verschlüsselten Grammatik

Abbildung 5.6 zeigt die Beantragung eines Tokens durch E. Die durchgeführten Prüfschritte sind mit P*x* gekennzeichnet. In Prüfschritt P1 wird das Vorliegen der Autorisierung für $\widehat{K_E}$ geprüft. Ist $\widehat{K_E}$ berechtigt, so liegen im Server auch der öffentliche Schlüssel K_E und die Information vor, ob $\widehat{K_E}$ das Recht zur Delegation besitzt.

Der Server sendet die signierte Antwort *respT* zurück und speichert t_1. Der Client prüft in P3 die Gleichheit des zurückgesendeten Nonce-Wertes r_1 und in P4 die Signatur der Nachricht mit K_G.

Der Einsteller der Grammatik verwendet das Token t_1 für die signierte Nachricht *addHG* mit der unter S verschlüsselten Grammatik G. Auf Server-Seite wird in P5 geprüft, ob die korrekte Kombination aus Token t_1 und $\widehat{K_E}$ vorliegt. In P6 wird die Signatur des Einstellers mit K_E geprüft und im Erfolgsfall die verschlüsselte Grammatik und ihr Hashwert in die Datenbank mit Zuordnung zu $\widehat{K_E}$ geschrieben. Der Server schickt die definierte Antwort *respG* an den Einsteller.

In Ergänzung zu den in Abschnitt 4.8 definierten Anforderungen wurde zusätzlich eine Zählfunktion für ausgestellte Token integriert. Dieser Zähler kann beispielsweise zur Abrechnung

von erstellten Berechtigungen dienen oder die Einhaltung von vorgegebenen Kontingenten erzwingen.

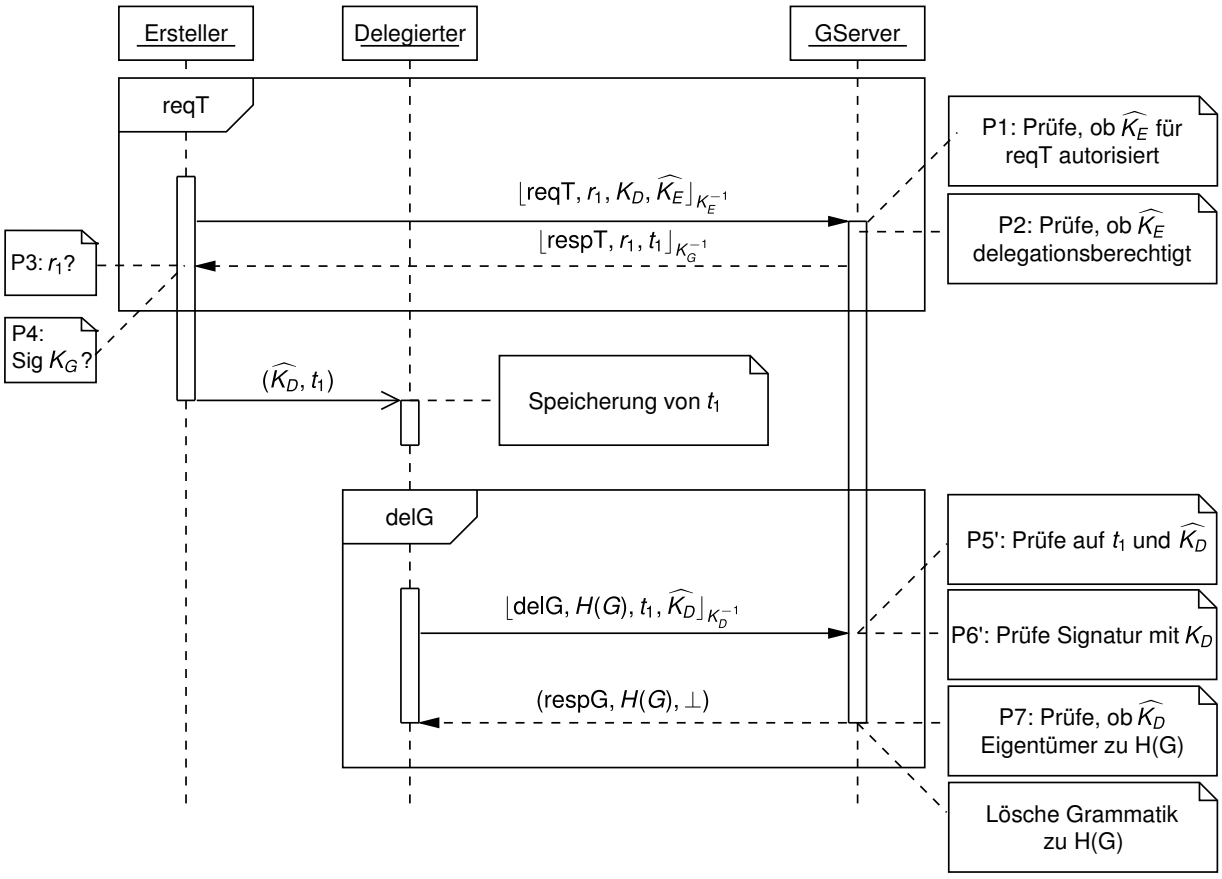


Abbildung 5.7: Löschen einer Grammatik mit Delegation

Abbildung 5.7 zeigt die Anforderung eines Token zur Weitergabe an einen Dritten, der bereits zuvor eine Grammatik eingestellt hat und diese löschen möchte.

Die *reqT*-Nachricht ist um den optionalen Schlüssel K_D des Dritten (z. B. kodiert durch ein Zertifikat) erweitert. Die Prüfung auf Server-Seite wird um P2 ergänzt (Delegationsberechtigung für $\widehat{K_E}$).

Nach Erhalt des Tokens wird dieser vom Ersteller an den Delegierten weitergegeben, den dieser bei sich zur späteren Verwendung speichert. Der Delegierte sendet die Löschnachricht *delG* mit dem Token t_1 und dem Hashwert der Grammatik an den Server. Dieser führt wie bei der Einstellung der Grammatiken die Prüfschritte P5 und P6 aus. Vor Ausführung der Löschung wird in P7 noch geprüft, ob die Zuordnung des Datenbankeintrags von $\mathrm{H}(G)$ zu $\widehat{K_D}$ vorliegt. Abschließend antwortet der Server definitionsgemäß mit einer *respG*-Nachricht auf die Löschanfrage.

5.4.3 Sicherheitseigenschaften der Grammatikverteilung

Die formale Prüfung, ob der Dienst die Eigenschaften erfüllt, wird mit Hilfe einer Proverif-Modellierung durchgeführt. Dabei wurden Erkenntnisse aus einer ersten Analyse durch eine Erweiterung des Protokolls in Kapitel 5.4.4 aufgegriffen und deren Wirksamkeit in einem zweiten Lauf bestätigt.

Das vollständige Proverif-Modell des Grammatik-Servers einschließlich der genannten Ergänzung ist in Anhang F enthalten.

5.4.3.1 Teilprozesse

Die Modellierung basiert dabei auf einer Reihe von Teilprozessen, die jeweils einen Teil der Funktionalität der Anwendungsfälle übernehmen. Diese Prozesse stellen den Grammatikserver und einen Client C sowie einen Delegationsberechtigten D dar. Dabei wurden bis auf den Abruf von Grammatiken lediglich die Funktionen Einstellung (*addHG*) und Löschung (*delG*) modelliert, da sie schützenswerte Ressourcen betreffen. Diese Teilprozesse sind:

processG Prozess des Grammatikservers (GServer) für Beantragung von Token (*reqT*), Einstellung (*addHG*) und Löschung (*delG*) von Grammatiken.

processGreqG Prozess des Grammatikservers, der auf Abruf einer Grammatik unter Angabe des Hashwertes antwortet

processCreqT Client, der für sich selbst Token beantragt

processDreq Delegationsberechtigter, der für Dritte Token beantragt

processCaddHG Einsteller-Prozess für verschlüsselte Grammatiken

processCreqG Client-seitiger Prozess zum Abruf einer Grammatik über Hashwert

processCdelG Client-seitiger Prozess zur Löschung von Grammatiken

Die Kommunikation zwischen C, D und GServer erfolgt dabei ausschließlich über den öffentlichen Kanal *net*, und kann somit vollständig vom Angreifer kontrolliert werden.

5.4.3.2 Ereignisse

Zur Überprüfung der Sicherheitseigenschaften ist es zunächst notwendig, Ereignisse im Ablauf der Protokollbearbeitung zu definieren, die im Rahmen von sog. queries in Beziehung gesetzt werden, um (ggf. injektive) kausale Zusammenhänge innerhalb des Ablaufs nachweisen zu können.

Vor Instantiierung der unter Replikation stehenden oben vorgestellten Teilprozesse werden zunächst Schlüsselpaare für alle Beteiligten einschließlich des Angreifers erzeugt.

Je nach untersuchter Variante werden die Schlüssel in die Autorisierungstabelle aufgenommen und mit den Rechten zur Token-Beantragung und/oder Delegation versehen. Für jede berechtigte Schlüsselreferenz wird ein Ereignis *authKeyid* mit der Referenz als Parameter erzeugt.

Bei der Token-Beantragung wird vor Aussendung des Request auf der Seite des Anfragenden ein Ereignis *reqToken* bzw. *reqDelegToken* erzeugt, das mit dem Zufallswert $r1$ und der Schlüsselreferenz des Berechtigten parametrisiert ist.

Sobald der Server ein Token ausgestellt hat, ist dies durch das Ereignis *genToken* mit den Parametern $r1$, Schlüsselreferenz des Berechtigten und Schlüsselreferenz des Anfragenden dokumentiert.

Die Anfragen zur Einstellung und Löschung von Grammatiken erzeugen vor Aussendung auf Client-Seite die Ereignisse *insertRequest* bzw. *deleteRequest*, denen auf Server-Seite die Ereignisse *grammarInsert* bzw. *grammarDelete* gegenüber stehen.

5.4.3.3 Eigenschaftsprüfungen

Die im vorausgehenden Abschnitt definierten Ereignisse können in Form von Abfragen zur Prüfung von Eigenschaften eingesetzt werden.

Zunächst wird für jedes Ereignis geprüft, ob es überhaupt im Rahmen des Protokolllaufs erreicht werden kann. Danach werden die folgenden Prüfungen durchgeführt.

Der Einstellung der (verschlüsselten) Grammatik G muss eine Autorisierung des Einstellers $\widehat{K_1}$ vorausgehen. Dies kann entweder direkt durch Aufnahme in die Autorisierungstabelle mit dem Recht zur Beantragung von Token geschehen oder durch die dedizierte Beantragung eines Token durch einen Delegationsberechtigten $\widehat{K_2}$ unter Angabe der Schlüsselreferenz $\widehat{K_1}$ des Einstellers als Berechtigten erfolgen (injektive Beziehung). Mit den eingeführten Ereignissen ausgedrückt, lautet dies:

$$[inj]\,\mathsf{grammarInsert}(G, \widehat{K_1}) \quad \rightsquigarrow \quad \mathsf{authKeyid}(\widehat{K_1}) \quad \vee \quad ([inj]\,\mathsf{genToken}(r, \widehat{K_1}, \widehat{K_2}) \ \wedge \ \mathsf{authKeyidDeleg}(\widehat{K_2})) \tag{GS-P1}$$

Eine Grammatik G soll nur dann durch einen Client $\widehat{K_1}$ gelöscht werden können, wenn dieser die Grammatik auch zuvor eingestellt hat. Dabei soll auf jeden Löschvorgang genau ein Einstellvorgang kommen (injektive Beziehung):

$$[inj]\,\mathsf{grammarDelete}(G, \widehat{K_1}) \quad \rightsquigarrow \quad [inj]\,\mathsf{grammarInsert}(G, \widehat{K_1}) \tag{GS-P2}$$

Ein angenommener Löschauftrag muss in Analogie zur Einstellung von einem vertrauenswürdigen, d. h. autorisierten Client kommen.

$$[inj]\,\mathsf{grammarDelete}(G, \widehat{K_1}) \quad \rightsquigarrow \quad \mathsf{authKeyidReq}(\widehat{K_1}) \;\vee\; ([inj]\,\mathsf{genToken}(r, \widehat{K_1}, \widehat{K_2}) \;\wedge\; \mathsf{authKeyidDeleg}(\widehat{K_2})) \tag{GS-P3}$$

Jedes Senden eines Einstell- oder Löschauftrages zu einer Grammatik mit Hashwert H darf nur genau einmal pro Anfrage des Clients $\widehat{K}$ durch den Server ausgeführt werden.

$$[inj]\,\mathsf{grammarInsert}(H, \widehat{K}) \quad \rightsquigarrow \quad [inj]\,\mathsf{insertRequest}(H, \widehat{K}) \tag{GS-P4}$$

$$[inj]\,\mathsf{grammarDelete}(H, \widehat{K}) \quad \rightsquigarrow \quad [inj]\,\mathsf{deleteRequest}(H, \widehat{K}) \tag{GS-P5}$$

Mit diesen Prüfungen sind die geforderten Eigenschaften aus Kapitel 4.8.3 abgedeckt.

Durch die Einführung eines Zählers in der Implementierung zur Umsetzung von Abrechnungsmechanismen kommt eine weitere Anforderung hinzu: Jede Anforderung eines Token mit Nonce-Wert r_1 durch einen autorisierten Client $\widehat{K_1}$ für eine zu berechtigenden Schlüsselreferenz $\widehat{K}$ darf nur zu genau einer Ausstellung eines Token führen, da andererseits ein Angreifer durch Wiedereinspielen von Requests dem Berechtigten Schaden zufügen kann. Diese Forderung wird formuliert als:

$$[inj]\,\mathsf{genToken}(r, \widehat{K}, \widehat{K_1}) \quad \rightsquigarrow \quad [inj]\,\mathsf{reqToken}(r, \widehat{K}) \;\vee\; [inj]\,\mathsf{reqDelegToken}(r, \widehat{K}) \tag{GS-P6}$$

5.4.4 Analyse und Erweiterung

Die im vorigen Abschnitt vorgestellten Prüfungen wurden durch entsprechende Ereignisse und Abfragen in der Proverif-Modellierung des Dienstes (vgl. Anhang F) berücksichtigt.

Die Analyse wurde in vier Szenarien S1 bis S4 durchgeführt. In der ersten Stufe S1 wurde nur der Client-Prozess berechtigt, den Dienst zu nutzen, jedoch kein Delegationsrecht vergeben. In der zweiten Stufe wurde dem Client das direkte Nutzungsrecht entzogen und dem Delegationsprozess die erweiterte Autorisierung mit Delegationsrecht gegeben.

In den Szenarien 3 und 4 erhält der Angreifer einen für die Dienstnutzung bzw. die Delegation autorisierten Schlüssel.

5.4.4.1 Erster Analyselauf

Im ersten Analyselauf zeigte sich ein Problem des Dienstentwurfs. Da bei der Definition der Eigenschaften in Kapitel 4.8.3 die durch die Implementierung hinzugekommene Forderung nach kontrollierter Ausstellung von Token (GS-P6) nicht berücksichtigt wurde, war das Prüfziel (GS-P6) nicht erfüllbar. Allerdings war es möglich, die Eigenschaft

$$\mathsf{genToken}(r, \widehat{K}, \widehat{K_1}) \quad \rightsquigarrow \quad \mathsf{reqToken}(r, \widehat{K}) \ \vee \ \mathsf{reqDelegToken}(r, \widehat{K}) \qquad \text{(GS-P6')}$$

nachzuweisen. Im Unterschied zu (GS-P6) ist die Korrespondenz nicht injektiv. Das bedeutet, dass zwar zur Auslösung der Token-Generierung die Anforderung durch einen Berechtigten erfolgt sein muss, jedoch kann es sich bei der empfangenen reqT-Nachricht um eine Wiedereinspielung handeln. Das Proverif-Werkzeug liefert einen entsprechenden Trace, der diese Vorgehensweise demonstriert.

5.4.4.2 Ergänzung des Protokolls

Zur Beseitigung der Schwachstelle wurde den bereits entworfenen Protokollschritten noch ein zusätzlicher Nachrichtenaustausch vorangestellt. Hierbei initialisiert der Client eine Token-Ausstellung mit einer initReqT-Nachricht, die der Server mit einer initRespT-Nachricht beantwortet. In dieser Antwort erhält der Client einen Zufallswert $r0$, den er in seine reqT-Anfrage aufnehmen muss.

Damit ergibt sich die angepasste Version des Protokolls zur Anforderung eines Tokens gemäß Abbildung 5.8. Durch die Einführung von r_0 sind alle durch K_E^{-1} signierten Anfragen individualisiert und damit für den Server unterscheidbar bzw. dem aktuellen Protokolllauf zuzuordnen.

5.4.4.3 Analyse der erweiterten Protokollversion

Die Ergebnisse der Proverif-Analyse sind in Tabelle 5.2 dargestellt.

Die Analyse zeigt, dass die Prüfungen GS-P1, GS-P2 und GS-P3 für alle Szenarien S1 bis S4 erfüllt sind. Damit sind alle Veränderungen in der Datenbank nur durch autorisierte Anwender möglich. Die Mandanten-Trennung (Löschen von Grammatiken nur durch den Einsteller) ist ebenfalls in allen Fällen wirksam.

Für die Fälle S1 und S2 konnte darüber hinaus gezeigt werden, dass, sofern kein Angreifer zur Dienstnutzung autorisiert ist, für die Aktionen „Ausstellung eines Tokens“ sowie „Einstellen und Löschen von Grammatiken“ jeweils eine entsprechende Request-Erstellung durch einen vertrauenswürdigen Beteiligten vorausgeht.

Für die Fälle S3 und S4 ist dieser Nachweis nicht möglich, da in diesem Fall der Angreifer die Requests erzeugt und somit die Ziele per se nicht erfüllt sein können. Die negativen Tests sind daher nicht als Einschränkung des Dienstes zu werten. Somit erfüllt der hier spezifizierte und modellierte Dienst alle an ihn gestellten Anforderungen.

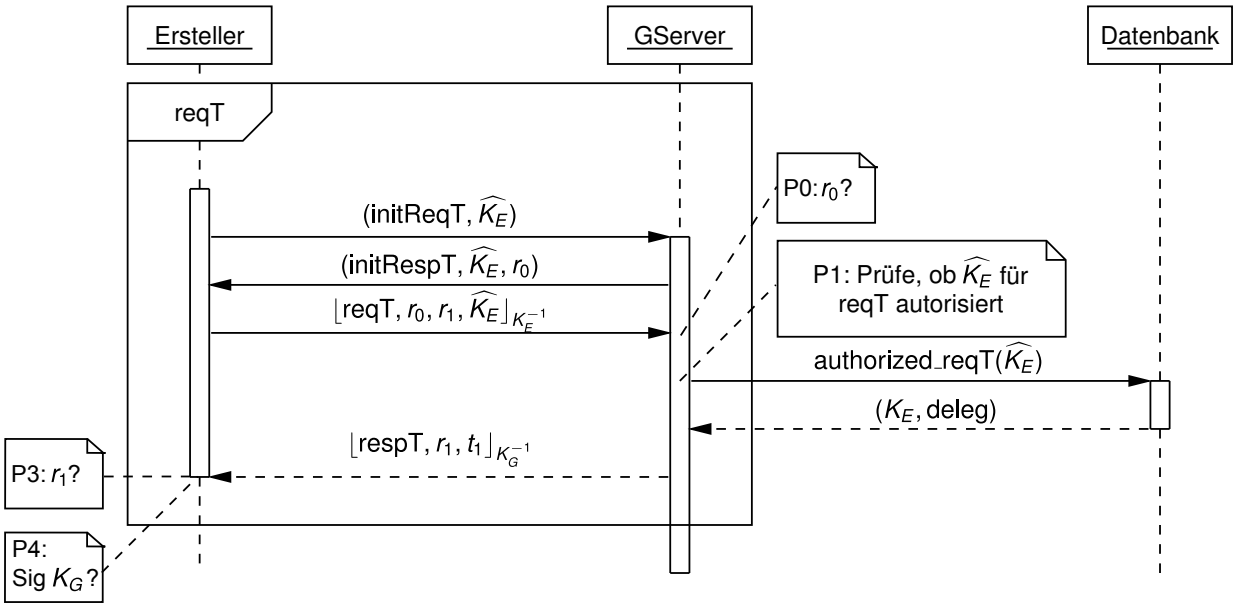


Abbildung 5.8: erweiterte Token-Anforderung

5.5 Beschränkung und Delegation

Nachdem die Realisierung der grammatikbasierten Zertifikate und des Grammatikverteildienstes vorgestellt und die Verfahren einer formalen Bewertung unterzogen wurden, sollen in diesem Abschnitt noch die Delegationsmechanismen sowie die Anwendung von Beschränkungen bei der Anforderung von Zusicherungen, der Berechtigung von Bestätigern und der Auswertung von grammatikbasierten Zertifikaten betrachtet werden.

Hierzu wird zunächst die Abbildung der verschiedenen Beschränkungsarten auf GC-Einschränkungen vorgestellt, wie sie für das Anwendungsfeld der Sicherheitsauditierungen vorgenommen werden kann. Im Anschluss wird ein Mechanismus zur Verwendung von Modulen in Grammatiken und seine Anwendung bei Speicherung und Auswertung von Beschränkungen diskutiert. Abschließend wird eine formale Bewertung hinsichtlich der Ausdrucksstärke und Anwendbarkeit der Delegation durch Definition einer Semantik auf Basis der SVO-Logik gegeben.

5.5.1 Delegationszertifikat

In Abschnitt 4.7.3 wurde vorgeschlagen, die Delegationszusicherung in Form eines GC auf Basis einer Grammatik G_D zu kodieren. Im Rahmen der Implementierung wurde eine solche Grammatik erstellt und in Anhang C.4 aufgenommen.

Anhand von Delegationszertifikaten auf Basis der Grammatik G_D lässt sich nochmals die Verbesserung der Verständlichkeit durch Rückgewinnung des Primärtextes belegen. Für eine Delegationsaussage zur BSI-Grundschutzmaßnahme 5.72 (vgl. Tabelle 4.1) für *Max Mustermann* mit

		S1	S2	S3	S4	Erläuterung
C	Auth	**T**	**F**	**F**	**F**	Autorisierung Anfragen C
	Deleg	**F**	**F**	**F**	**F**	Autorisierung Delegation C
D	Auth	**F**	**T**	**F**	**F**	Autorisierung Anfragen D
	Deleg	**F**	**T**	**F**	**F**	Autorisierung Delegation D
A	Auth	**F**	**F**	**T**	**T**	Autorisierung Anfragen Angreifer
	Deleg	**F**	**F**	**F**	**T**	Autorisierung Delegation Angreifer
GS-P1		**T**	**T**	**T**	**T**	Grammatikeinstellung nur nach Autorisierung
GS-P2		**T**	**T**	**T**	**T**	Löschung einer Grammatik nur durch Inhaber
GS-P3		**T**	**T**	**T**	**T**	Löschung nur nach Autorisierung
GS-P4		**T**	**T**	**F**	**F**	nur eine Einstellung pro Einstellanfrage
GS-P5		**T**	**T**	**F**	**F**	nur eine Löschung pro Löschanfrage
GS-P6		**T**	**T**	**F**	**F**	nur eine Token-Erstellung pro Anfrage

Tabelle 5.2: Ergebnisse der Proverif-Analyse des Grammatikverteildienstes

einer Beschränkung des zu prüfenden Systems auf *cntsrv1* und Festlegung der Prüfmethode auf *Portscan* ergibt sich folgender (in S-Expressions dargestellter) Syntaxbaum

```
(Satz (Berechtigter (CommonName "Max Mustermann")
(EmailAddress "max@example.com")) (GrammatikName "BSI-M.5-72")
(GrammatikReferenz (Algorithmus 'SHA256')
(Hashwert 0x05619a2d17bdf7be682269e3c09da133f258da9ba73e0970be14acf578cd0570)
(Uri "http://security.example.com/grammars/m572.gcg")) (Weitergabe 3)
(Beschraenkungen (Und (Pfad "//Methode/'Portscan'")
(Pfad "//Server/'cntsrv1'"))))
```

Diese kompakte Information kann durch Rückgewinnung mit G_D wie folgt dargestellt werden:

> Hiermit wird an "Max Mustermann" ("max@example.com") das Recht
> zur Erstellung von Aussagen gemäß der Grammatik "BSI-M.5-72" erteilt.
>
> Die Grammatik hat den SHA256-Hashwert
> 0x05619a2d17bdf7be682269e3c09da133f258da9ba73e0970be14acf578cd0570.
>
> Die Grammatik kann unter der Adresse
> "http://security.example.com/grammars/m572.gcg"
> abgerufen werden.

Diese Berechtigung kann über maximal 3 nachfolgende Delegationsvorgänge weitergegeben werden.

Die auf Basis dieser Delegation gemachten Aussagen müssen folgende Bedingungen erfüllen:

und(
"//Methode/'Portscan'"
;
"//Server/'cntsrv1'"
)

Bei der Festlegung dieses Formates wurden zur Angabe des Berechtigten lediglich der volle Name und die E-Mail-Adresse berücksichtigt. Diese Kombination reicht zur Identifikation für die meisten, insbesondere Firmen-internen, Anwendungsfälle aus. Es ist dann jedoch notwendig, dass auch auf der Ebene der *Issuer*-Angaben im verwendeten GC-Zertifikat diese Informationen enthalten sind. Ergänzend können hier für geschlossene Anwendergruppen ggf. auch zusätzlich Personalnummern o. ä. aufgenommen werden. Bei Anwendung in öffentlichen Bereichen ermöglicht eine Schlüsselkennung die eindeutige Identifizierung.

Die Länge der auf eine Delegation folgenden Zertifikatskette hängt vom gewählten Wert *Weitergabe* ab. Bei der Modellierung von Beispielen hat sich gezeigt, dass eine Maximallänge (ausgehend von der initialen Berechtigung am ZAS) von $w_{\max} = 2$ sinnvollerweise bei Sicherheitsauditierungen nicht überschritten werden sollte. In der Regel gibt es einen direkten Abstimmungsbedarf zwischen den Auditoren und den Zusichernden, so dass viele Zwischenstufen in der Delegation entweder die Kommunikation erschweren oder überhaupt nicht relevant sind. Ein typisches Szenario ist der Kontakt zwischen Auditor und Teamleiter, der die zuständigen Mitarbeiter benennt und damit berechtigt. Maximal sollte jedoch nur noch eine weitere Führungsebene einbezogen werden.

5.5.2 Beschränkungsarten

Das bei Sicherheitsauditierungen in der Praxis häufig angewendete mehrstufige Konzept von Vorgaben bzw. Anforderungen kann durch die Beschränkungsmechanismen (Abschnitt 4.6) und Delegationszertifikate abgebildet werden.

Es werden im Rahmen einer Auditierung die Ebenen von Festlegungen betrachtet: *zentrale Festlegungen*, *Festlegung der Kompetenz* und *systemspezifische Festlegungen*.

Die zentralen Festlegungen werden durch die Sicherheitsrichtlinie des Unternehmens (engl. security policy) und auf das ganze Unternehmen anzuwendende Vorschriften wie z. B. Datensicherheitsstandards der Herausgeber von Zahlungsverkehrskarten [PCI10] gebildet. Diese Richtlinien legen grundsätzliche Anforderungen an Verfahren und Betriebsmodi von Systemen fest. Beispiele hierzu sind vorgeschriebene Sicherheitsprüfungen für Personal, Passwort-Richtlinien (Länge,

Zeichenvorrat, Änderungsfrequenz), Härtungsvorgaben für Betriebssysteme oder Algorithmenvorgaben für Verschlüsselungsmechanismen. Die daraus resultierenden Beschränkungen werden sollen als *Policy-Beschränkung* bezeichnet werden.

Die Festlegung der Kompetenz für die Beauftragten setzt die Verantwortungs- und Fähigkeitsbereiche der Zusichernden in *Kompetenz-Beschränkungen* um.

Da eine allgemein formulierte Grammatik sowie ein Prüfplan eine Menge konkreter Systeme abdecken, müssen die durch den ZAS akzeptierten Zusicherungen auf die tatsächlich durch einen Mitarbeiter betreuten Systeme eingeschränkt werden können. Ebenfalls ist es an dieser Stelle sinnvoll, mögliche Aussageformen entsprechend der Fähigkeiten des Mitarbeiters einzuschränken. Sieht eine Zusicherungsklasse verschiedene Verfahren oder Werkzeuge zur Erhebung der angeforderten Informationen zu, so kann es z. B. sinnvoll sein, die Zusicherung auf die Verfahren einzugrenzen, für die der Mitarbeiter bereits geschult wurde.

Um eine Eingrenzung der Systeme vornehmen zu können, müssen Zusicherungen Identifikationsdaten für ein System enthalten. Beispiele hierfür sind der logische Systemname oder Seriennummern von Geräten. Werden Zusicherungen zu Personen wie z. B. Schulungsnachweise benötigt, kommen Kennungen wie bei den Delegationszertifikaten zum Einsatz.

Mit *systemspezifischen Beschränkungen* können zusätzliche Festlegungen für einzelne zu untersuchende Systeme getroffen werden. Diese kommen z. B. bei Systemen mit erhöhtem Schutzbedarf zum Einsatz, bei denen die Festlegungen der allgemein gültigen Sicherheitsrichtlinie nicht ausreichen.

5.5.3 Skalierung durch Module

Sollen übergreifende Sicherheitsrichtlinien auf unterschiedliche Untersuchungsfelder anwendbar sein, so sollte soweit möglich eine einheitliche Darstellung für die zu beschränkenden Teile einer Zusicherung eingesetzt werden.

Zu diesem Zweck wurde ein Mechanismus entworfen, der die Aufnahme vorgefertigter Bausteine in zu erstellende anwendungsspezifische Grammatiken ermöglicht, ohne die Eigenschaften des vorgestellten Ansatzes wesentlich zu verändern. Lediglich eine Einschränkung der Benennung von Nicht-Terminalen ist zu beachten.

Jeder Baustein hat dabei folgende Eigenschaften

- Jeder Baustein besteht aus Produktionsregeln, die – unter Hinzunahme der standardmäßig definierten Lexeme – eine abgeschlossene Sprache bilden, d. h. für jedes Nicht-Terminal auf der rechten Seite einer Regel existiert eine Produktionsregel.
- Alle Nicht-Terminale erhalten einen Baustein-spezifischen Prefix, der für alle frei erstellten Grammatiken als reserviert gilt.
- Das Startsymbol mit Baustein-spezifischen Prefix ist bekannt.

Ein Baustein ist somit eine eigene Grammatik $G_B =< \mathcal{N}_B, \mathcal{T}_B, \mathcal{P}_B, S_B >$. Die Aufnahme eines solchen Bausteins erfolgt durch einen Include-Befehl innerhalb der aufnehmenden Grammatik. Dabei wird die aufzunehmende Grammatik durch ihren Hashwert sowie eine Quellenangabe referenziert.

Wird G_B in eine Grammatik G' aufgenommen, so ergibt sich daraus eine resultierende Grammatik G durch Vereinigung der korrespondierenden Mengen.

$$\begin{aligned} \mathcal{P} &= \mathcal{P}' \cup \mathcal{P}_B \\ \mathcal{N} &= \mathcal{N}' \cup \mathcal{N}_B \quad \text{mit } \mathcal{N}' \cap \mathcal{N}_B = \emptyset \\ \mathcal{T} &= \mathcal{T}' \cup \mathcal{T}_B \\ S &= S' \end{aligned}$$

Die Produktionsregeln $\mathcal{P}$ müssen dann so umgeformt werden, dass mindestens eine Regel auf ihrer rechten Seite S_B enthält, damit die Einbindung der Grammatik G_B wirksam wird.

Ein Beispiel für einen solchen Baustein mit dem Prefix *Gm012* und dem Startsymbol *Gm012_-Server* ist eine Grammatik zur Benennung eines untersuchten Serversystems die u. a. den folgenden Text erfolgreich validiert:

```
Für den virtuellen Server "vlsrv01" im Hostsystem "vhost17"
mit Betriebssystem "RedHat", Version "5.9",
wird folgende Zusicherung gegeben:
```

Dieser Baustein kann beispielsweise in die Bestätigung der Maßnahme M 5.72 (vgl. Tabelle 4.1) in die erste Regel aufgenommen werden:

Mn5_72 → ***Gm012_Server*** 'Auf dem Server wurden alle nicht benötigten Dienste abgeschaltet.\n' *Pruefung*? 'Die aktiven Dienste sind:\n' *Dienste*?

Durch die Prefixnotation können nun einheitliche XPath-Ausdrücke für alle Zusicherungen verwendet werden, die diesen Baustein eingebunden haben. So erfüllt jede Zusicherung, die sich auf einen virtuellen Server bezieht, den Prüfausdruck

```
//Gm012_Typ/'virtuell'
```

Bei mehrfacher Einbindung des Bausteins kann es notwendig sein, durch eine Erweiterung des XPath-Ausdrucks die gewünschte Stelle der Verwendung zu präzisieren.

5.5.4 Speicherung und Prüfung von Beschränkungen

Grundlage für alle Auditierungsaktivitäten bildet das in Abschnitt 1.2.3 beschriebene allgemeine Prüfverfahren, bei dem nach einer Erhebung der Gegebenheiten der Infrastruktur ein Prüfplan erstellt wird. Der Prüfplan legt die zu untersuchenden Systeme und Prozesse sowie die dabei zu

betrachtenden Fragestellungen fest. Diese Anforderungen sowie die Policy- und systemspezifischen Beschränkungen müssen auf dem Server hinterlegt und bei Eingang von Zusicherungen im zentralen Auditierungsserver geprüft werden.

Hierzu wird eine Zuordnung aus Selektionskriterien und XPath-Prüfausdrücken verwendet. Als Selektionskriterium kann der Hashwert einer Grammatik, ein XPath-Ausdruck oder die Kombination aus beiden verwendet werden.

Wird nur ein Grammatik-Hashwert verwendet, so wird der Prüfausdruck auf jede Zusicherung unter dieser Grammatik angewendet. Dies wird bei Policy-Beschränkungen angewendet, die direkt einer Grammatik (z. B. Auditierung der Authentisierungsmodule von Servern) zugeordnet werden.

Die Verwendung eines XPath-Ausdrucks als Selektor kann bei Policy-Beschränkungen genutzt werden, die bezüglich eines Grammatikbausteins aufgestellt werden. Hierbei wird das Startsymbol des Bausteins als Selektionskriterium verwendet. Auf diese Weise wird automatisch jede Grammatik durch die Beschränkung abgedeckt, die den entsprechenden Baustein einbindet.

Die Verwendung der Kombination aus Grammatik-Hashwert und XPath-Ausdrucks kann für die Festlegung von Anforderungen und systemspezifischen Beschränkungen genutzt werden. Der XPath-Ausdruck bestimmt dabei ein konkretes System. Dabei wird eine Grammatik zur Identifikation von Systemen referenziert und durch den XPath-Ausdruck mit der Identität des gewünschten Systems parametrisiert.

Die Kompetenzen der Bestätiger werden nicht initial in der Datenbank des ZAS gespeichert. Sie werden durch die Delegationszertifikate kodiert, die im Verlauf des Ausrollens von Rechten an den ZAS übermittelt werden. Lediglich die Anfänge der Delegationsketten werden zentral hinterlegt, d. h. welche Grammatiken durch wen mit welchen Beschränkungen delegiert werden dürfen. Hierzu werden weitere Details im nächsten Kapitel betrachtet. Der ZAS nimmt auch u. U. notwendige Rückrufe einer Delegation vom jeweiligen Ersteller entgegen, so dass er die Funktion des Statusauskunftsdienstes implizit erfüllt.

Sind alle Anforderungen und Beschränkungen hinterlegt, so wird ein Auditauftrag erteilt, in dessen Rahmen die Mitarbeiter die Prüfpläne erhalten. In diesem Zusammenhang kann auch die Anfragenummer verteilt werden, die in die Zusicherung zur Etablierung der Frische aufgenommen wird.

Die Erhebung des Ist-Zustandes erfolgt durch die Bestätiger in der herkömmlichen Weise mit beliebigen manuellen oder automatisierten Verfahren. Die Bestätiger dokumentieren die Ergebnisse in Form von grammatikbasierten Zertifikaten, die an den ZAS gesendet werden.

Bei Erhalt einer Zusicherung prüft der ZAS zunächst (neben der eigentliche Zertifikatsprüfung), ob die Policy- und systemspezifischen Beschränkungen sowie die aus der Delegationskette vorliegenden Beschränkungen eingehalten werden. Ist dies der Fall, so wird ermittelt, auf welche Anforderungen die Zusicherung passt und diese kann dann als erfüllt markiert werden.

5.5.5 Formale Beschreibung der Delegation

Wie bereits die Zusicherungen für die Auditierungen soll auch für Delegationszertifikate eine formale Beschreibung zur Festlegung einer Semantik und zum Nachweis der Funktionsfähigkeit angegeben werden.

Analog zu den grammatikbasierten Zertifikaten wird für die Delegationszertifikate D die Schreibweise

$$D = \lfloor \mathrm{GDC}(B, \mathrm{H}(G), R, w, \mathcal{M}) \rfloor_{K_A^{-1}}$$

eingeführt. Hierbei steht B für den zu Berechtigenden, $\mathrm{H}(G)$ ist der Hashwert der Grammatik für die Basisberechtigung, R ist die anzuwendende Beschränkung, w der Indikator für das Recht zur Weiterdelegation über eine bestimmte Zahl von Stufen (Ganzzahl, $w \geq 0$) und $\mathcal{M}$ enthält die weiteren Meta-Daten, die bei Bedarf herausgezogen werden können. Für R kann $\perp$ geschrieben werden, wenn keine Einschränkungen vorliegen.

Die Basisberechtigung ist nach Abschnitt 4.7.2 durch $\mathscr{L}(G)$ gegeben. Beschränkungen R_ν werden aus der Menge $\mathcal{X}$ der zulässigen XPath-Ausdrücke und den Operatoren $\mathrm{and}(.,.)$, $\mathrm{or}(.,.)$ und $\mathrm{not}(.)$ gebildet.

Da Delegationszertifikate oftmals keinen Nonce-Wert enthalten, da sie i. d. R. lange Gültigkeitsdauern haben und nicht explizit angefragt werden, wird für Delegationszertifikate angenommen, dass für sie die Zusicherung der Aktualität durch einen Statusauskunftsdienst übernommen wird und diese Auskunft durch das *current*-Prädikat ausgedrückt.

Um die Berechtigung zur Delegation der Kompetenz zur Erstellung von Zusicherungen formal zu fassen, wird die *provides*-Beziehung zwischen einem Prinzipal und einer Menge von Aussagen sowie der Anzahl n erlaubter weiterer Delegationsschritte eingeführt.

$$Q \text{ provides } (\mathscr{L}(G, r_0), n) \qquad \text{mit } n \geq 0 \tag{5.7}$$

Wie zuvor kann $r_0 = \perp$ sein, so dass $\mathscr{L}(G, r_0) = \mathscr{L}(G)$ ist.

Die Wahrheitsbedingungen lauten damit mit den Ganzzahlen $n > 0$ und $w \geq 0$

$$(r, t) \vDash Q \text{ provides } (\mathscr{L}(G, r_0), n)$$

g.d.w. die Aussage

$$(r, t) \vDash Q \text{ says } \mathrm{GDC}(P, \mathrm{H}(G), r_1, w, \mathcal{M})$$

für alle $0 \leq w < n$ die Aussagen

$$(r, t') \vDash P \text{ controls } \mathscr{L}(G, \mathrm{and}(r_0, r_1)) \quad \text{für alle } t' \geq 0$$

und

$$(r, t') \vDash P \text{ provides } (\mathscr{L}(G, \mathrm{and}(r_0, r_1)), w) \quad \text{für alle } t' \geq 0$$

impliziert.

Zur vollständigen Abdeckung des Wertebereichs wird außerdem festgelegt, dass für $n = 0$ gilt:

$$\models P \text{ provides } (\mathscr{L}(G, r_0), 0)$$

$\mathscr{L}(G, r_0)$ bildet die maximale Aussagenmenge, die Q delegieren kann. Bei der Delegation kann eine zusätzliche Einschränkung r_1 angegeben werden, die jedoch auch auf $r_1 = \bot$ gesetzt werden kann, so dass die volle Aussagenmenge weiter delegiert wird.

Da die SVO-Logik keine Vergleichsoperatoren für Ganzzahlen enthält, wird hierzu ein neues Prädikat GT (für *greater than*) bereit gestellt:

$$\text{GT}(n, m) \Leftrightarrow n > m$$

Mit diesen Mitteln kann die Interpretation für Delegationszertifikate durch eine SVO-Regel definiert werden.

$$(Q \text{ says } \text{GDC}(P, \text{H}(G), r_1, w, \mathcal{M}) \ \wedge \ Q \text{ provides } (\mathscr{L}(G, r_0), n) \ \wedge \ \text{GT}(n, w)) \rightarrow$$
$$(P \text{ controls } \mathscr{L}(G, \text{and}(r_0, r_1)) \ \wedge \ P \text{ provides } (\mathscr{L}(G, \text{and}(r_0, r_1)), w)) \quad \text{(GC3)}$$

Die Korrektheit der Implikation ist für $n > 0$ wieder unmittelbar aus den Wahrheitsbedingungen ersichtlich. Für $n = 0$ ist $\text{GT}(n, w)$ für alle w falsch und damit die Implikation ebenfalls wahr und damit korrekt.

5.5.6 Prüfablauf

Mit den Ableitungsschritten des vorangegangenen Abschnitts kann die Auswertung einer Delegationskette durchgeführt werden. Als Beispiel wird eine Kette vom Sicherheitsmanagement S, über einen Abteilungsleiter A, einen Teamleiter T bis zu einem Mitarbeiter M gebildet. Ausgehend von S wird A mit dem *provides*-Recht für zwei Stufen ausgestattet. Dies wird in der lokalen Datenbank abgelegt und daher als Prämisse modelliert.

Die weiteren Bestätigungen durch A und T werden in Form eines GDC an S übermittelt, da S als Initiator der Delegationskette am Ende die Zusicherungen von M erhält. Für die GDC und eine Zusicherung Z von M werden dabei folgende Kurzschreibweisen definiert:

$$\text{GDC}_T = \text{GDC}(T, \text{H}(G), r_1, 1, \mathcal{M})$$
$$\text{GDC}_M = \text{GDC}(M, \text{H}(G), r_2, 0, \mathcal{M})$$
$$Z = \text{GC}(X, \text{H}(G), N, \mathcal{M})$$

Die Prämissen sind

Kenntnis der öffentlichen Schlüssel

S believes $\mathrm{PK}_\sigma(A, K_A)$ (P1)

S believes $\mathrm{PK}_\sigma(T, K_T)$ (P2)

S believes $\mathrm{PK}_\sigma(M, K_M)$ (P3)

Zweistufiges Delegationsrecht für A

S believes A provides $(\mathscr{L}(G, r_0), 2)$ (P4)

Delegationsnachweise

S believes S received $\lfloor \mathrm{GDC}_T \rfloor_{K_A^{-1}}$ (P5)

S believes S received $\lfloor \mathrm{GDC}_M \rfloor_{K_T^{-1}}$ (P6)

Gültige Signaturen der GDC

S believes $\mathrm{SV}(\lfloor \mathrm{GDC}_T \rfloor_{K_A^{-1}}, K_A, \mathrm{GDC}_T)$ (P7)

S believes $\mathrm{SV}(\lfloor \mathrm{GDC}_M \rfloor_{K_T^{-1}}, K_T, \mathrm{GDC}_M)$ (P8)

GDC sind aktuell

S believes current(A, GDC_T) (P9)

S believes current(T, GDC_M) (P10)

Gültig signierte Zusicherung von M

S believes S received $\lfloor Z \rfloor_{K_M^{-1}}$ (P11)

S believes $\mathrm{SV}(\lfloor Z \rfloor_{K_M^{-1}}, K_M, Z)$ (P12)

Korrekte Sprach- und Berechtigungsprüfung

S believes $\mathrm{GCV}(X, G, p)$ (P13)

S believes $\mathrm{GCR}(X, \mathrm{and}(\mathrm{and}(r_0, r_1), r_2))$ (P14)

S believes fresh(N) (P15)

Prüfung der Signaturen und Zuordnung der Nachrichten zu den Sendern

S believes A said GDC_T (C1)

durch Ax6, Nec, P1, P5, P7, Ax1, MP

S believes T said GDC_M (C2)

durch Ax6, Nec, P2, P6, P8, Ax1, MP

S believes M said Z (C3)

durch Ax6, Nec, P3, P11, P12, Ax1, MP

Durch die Nachweise der Aktualität von einem Statusauskunftsdienst können die Delegations-

nachweise als aktuelle Äußerungen betrachtet werden:

$$S \text{ believes } A \text{ says } \mathrm{GDC}_T \tag{C4}$$

durch Ax23, Nec, C1, P9, Ax1, MP

$$S \text{ believes } T \text{ says } \mathrm{GDC}_M \tag{C5}$$

durch Ax23, Nec, C2, P10, Ax1, MP

Aus der Delegationsaussage GDC_T kann die Berechtigung von T und daraus die Berechtigung von M abgeleitet werden:

$$S \text{ believes } T \text{ provides } (\mathscr{L}(G, \mathrm{and}(r_0, r_1)), 1) \tag{C6}$$

durch GC3, Nec, C4, P4, GT$(2, 1)$, Ax1, MP

$$S \text{ believes } M \text{ controls } \mathscr{L}(G, \mathrm{and}(\mathrm{and}(r_0, r_1), r_2)) \tag{C7}$$

durch GC3, Nec, C5, C6, GT$(1, 0)$, Ax1, MP

Durch Nachweis der Frische der Zusicherung gilt diese ebenfalls als aktuell geäußert:

$$S \text{ believes } \mathrm{fresh}(Z) \tag{C9}$$

durch Ax19, Nec, P15, Ax1, MP

$$S \text{ believes } M \text{ says } Z \tag{C10}$$

durch Ax21, Nec, C9, C3, Ax1, MP

Die Kompetenz für p folgt aus der Prüfung der Grammatik und der Restriktionen.

$$S \text{ believes } M \text{ controls } p \tag{C11}$$

durch GC2, Nec, C7, P13, P14, Ax1, MP

Die zurückgewonnene Aussage wird M zugeordnet.

$$S \text{ believes } M \text{ says } p \tag{C12}$$

durch GC1, Nec, C10, P13, Ax1, MP

Mittels der M zugewiesenen Kompetenz kann die Aussage übernommen werden:

$$S \text{ believes } p \quad \square$$

durch Ax18, Nec, C11, C12, Ax1, MP

Damit wurde gezeigt, dass die Übernahme der notwendigen Überzeugungen auf Basis der aufgestellten Prämissen möglich ist. Aus den Schritten dieses Nachweises können direkt die Funktionsbausteine und deren Anwendungsreihenfolge für eine Implementierung des Delegationsmechanismus abgelesen werden.

5.5.7 Eigenschaften der Delegation

Die Implementierung und formale Analyse haben die Eigenschaften der spezifizierten und realisierten Delegationsmechanismen verdeutlicht.

Es ist mit dem gewählten Ansatz sowohl möglich, die Rechte zur Abgabe einer Aussage sehr feingranular einzuschränken als auch dieses Recht an weitere Beteiligte weiterzugeben.

Besonderes Augenmerk verdient dabei die Eigenschaft, dass die Weitergabe von Aussagerechten in ihrer Wirksamkeit vom Kenntnisstand des Akzeptors abhängt, der die Auswertung der Delegationszertifikate durchführt. Liegen dem Akzeptor umfangreiche Aussagerechte für den Aussteller der Delegation vor, so kann auch der Berechtigte entsprechend umfangreiche Rechte erhalten und umgekehrt. Je nach vorliegenden Berechtigungsprämissen oder Delegationszertifikaten kann also die Sicht auf die Rechte der Beteiligten unterschiedlich sein. Die Angabe eines absoluten Rechteumfanges ist in diesem Modell nicht möglich, da jeweils nur die subjektive Sicht des Akzeptors betrachtet wird. Dieses Modell trägt damit dem Umstand Rechnung, dass auch bei der menschlichen Delegation jeder einzelne über die jeweiligen Zuständigkeiten eines Delegierenden informiert werden muss.

Das Verfahren koppelt das Recht zur Erstellung von Zusicherungen und das Recht zur Weiterdelegation der Delegation durch (GC3). Diese Zusammenlegung wurde zur Vereinfachung der Inhalte der Delegationszertifikate vorgenommen. Alternativ könnten statt des einen Indikators w auch zwei Rechte unterschieden werden. In diesem Fall müsste sich ein Delegationsberechtigter u. U. selbst das Zusicherungsrecht erteilen.

Die Anzahl der Stufen für eine Delegation ist durch den Initiator frei wählbar. Für die Implementierung wurde hier für den Anwendungsfall jedoch die Einschränkung von $w_{\max} = 2$ eingeführt, da bei Auditierungen längere Delegationsketten nicht sinnvoll sind.

Der Grundmechanismus, einen Sprachumfang auf verschiedenen Ebenen immer weiter einzuschränken, ermöglicht keine Ausnahmen. Soll z. B. eine unternehmensweite Sicherheitsrichtlinie aus technischen Gründen für ein System teilweise außer Kraft gesetzt werden, so darf die Richtlinie nicht auf das System angewendet werden und es müssen alle Beschränkungen, die dennoch gelten sollen, gesondert modelliert werden. Dies bedeutet für solche Fälle zusätzlichen Aufwand.

Bei Einsatz von Delegationszertifikaten ist es zwingend notwendig, die Aktualität der Zertifikate nachzuweisen. Hierdurch wird in jedem Fall ein Mechanismus für einen Rückruf und die zentrale Verteilung von Zustandsinformation benötigt. Wird zur Erhebung und Auswertung der Zusicherungen ein zentraler Auditierungsserver eingesetzt, so fallen diese Funktionen zusammen und der Aufwand ist vergleichsweise gering.

6 Zusammenfassung

6.1 Erreichte Ziele

Im ersten Kapitel wurden Anforderungen an einen neuen Ansatz für signierte Zusicherungen bei Auditierungen gestellt. Mit den neu entworfenen grammatikbasierten Zertifikaten und den zugehörigen Mechanismen können diese Anforderungen erfüllt werden.

Durch die Kodierung von Zusicherungen durch abstrakte Syntaxbäume auf Basis von kontextfreien Grammatiken wurde eine eineindeutige Beziehung zwischen den natürlichsprachlichen Zusicherungen und den im Zertifikat enthaltenen strukturierten Daten geschaffen.

Damit ist der hohe Gestaltungsspielraum der Grammatiken nutzbar, um *flexibel* unterschiedlichste Zusicherungsarten zu unterstützen, sofern die Zusicherungen auf beliebige aber feste Textbausteine eingeschränkt werden können. Hierbei können auch umfangreiche Textpassagen verwendet werden, da durch die zusätzliche Bereitstellung der Grammatiken beim Akzeptor der Zertifikate das entwickelte Optimierungsverfahren zur Streichung redundanter Anteile aus dem Syntaxbaum eingesetzt werden kann.

Durch den Einsatz von Signaturen in Kombination mit Rückrufmechanismen, die an den X.509-Standard angelehnt wurden, sind die Zusicherungen authentisierbar. Durch die beweisbare Zuordnung zu einem Ersteller können die Zusicherungen im Rahmen entsprechender organisatorischer Festlegungen eine hohe *Verbindlichkeit* ermöglichen. Durch die technische Möglichkeit in den GC auch Signaturen des Primärtextes zu übertragen, lassen sich auch qualifizierte Signaturen mit hoher Rechtsverbindlichkeit in den Ansatz integrieren.

Da der Primärtext sowohl bei der Erstellung als auch bei der Auswertung der GC angezeigt werden kann, lässt sich eine sehr hohe *Verständlichkeit* der Zusicherungen erreichen, da diese umgangssprachlich oder in der Fachsprache des jeweiligen Anwendungsfalles erstellt werden können. Bei der Parametrisierung von Anforderungen und Einschränkungen können semantisch sinnvoll belegte Bezeichner verwendet werden, so dass die XML-Daten der Zusicherungen und XPath-Ausdrücke der Beschränkungen ebenfalls mit Termen der anwendungsspezifischen Sprache gestaltet werden können.

Mit diesen Kodierungsformen ist es möglich, Anforderungen von Zusicherungen, Beschränkungen des Sprachumfangs von Zusicherungen, Compliance-Prüfungen und die Zusicherungen

selbst so bereit zu stellen, dass für diese eine *vollautomatische Auswertung* realisiert werden kann.

Für die Berechtigung zur Erstellung von Zusicherungen wurde ein skalierbarer, mehrstufiger Delegationsmechanismus bereit gestellt, über den z. B. ausgehend vom Sicherheitsmanagement über z. B. zwei Führungsebenen Mitarbeitern die notwendige Kompetenz zugesprochen werden kann. Dabei ist es möglich, den Kompetenzumfang in jeder Delegationsstufe weiter einzuschränken.

Für alle Mechanismen wurden Datenformate spezifiziert und eine prototypische Implementierung erstellt, die ihre Funktionsfähigkeit belegt.

Alle zur Anwendung der grammatikbasierten Zertifikate notwendigen Interpretationsvereinbarungen wurden formal spezifiziert und einer Analyse auf Basis der SVO-Logik unterzogen. Auf diese Weise konnte gezeigt werden, welche Überzeugungen aus den grammatikbasierten Zertifikaten abgeleitet werden können und welche Prämissen dazu notwendig sind.

Die wichtigste Interpretationsvereinbarung ist (GC1). Anwender müssen beim Verfahren der signierten abstrakten Syntaxbäume (s-s) die Gleichsetzung der Äußerung von XML-kodierten Daten mit dem zurückgewonnenen Primärtext akzeptieren. Dies ist nicht notwendig, wenn mit dem p-s-Verfahren gearbeitet wird (Primärsignatur und Übertragung der strukturierten Daten). In allen Fällen kann die Interpretationsvereinbarung (GC2) zur Berechtigungsprüfung im Hinblick auf einen für die Zusicherung vorgegebenen Sprachumfang eingesetzt werden. Die dritte Vereinbarung (GC3) legt die Interpretation der Delegationszertifikate fest und ermöglicht die Nutzung von Delegationsketten mit beschränkbarer Länge.

Die formalen Analysen bilden klare Vorgaben für Implementierungen, da sie alle notwendigen Prüfschritte exakt dokumentieren, so dass sich für Realisierungen ergibt, welche Daten vorgehalten und welche algorithmischen Verfahrensschritte ausgeführt werden müssen. Dadurch ist eine sichere Realisierung der Anwendung grammatikbasierter Zertifikate deutlich vereinfacht.

6.2 Grenzen des Verfahrens

Das vorgestellte Verfahren weist durch einige seiner Eigenschaften und durch die zur Etablierung notwendigen Aufwände hinsichtlich der Anwendbarkeit Grenzen auf.

Es wurde vorausgesetzt, dass die GC ein bestehendes Auditierungssystem unterstützen, d. h. dass grundsätzlich die angeforderten Zusicherungen z. B. durch schriftliche Prüfpläne und Sicherheitsrichtlinien bekannt sind. Die Auswertung der XPath-Ausdrücke ist trotz der Verwendung von Begriffen des Anwendungsbereichs für den Endanwender nicht ohne entsprechende Unterrichtung möglich. Somit ist es also nicht sinnvoll, lediglich im Rahmen der elektronisch kodierten Festlegungen bei Autorisierungen, Prüfausdrücken für Policy-Vorgaben oder Delegationen neue Beschränkungen aufzunehmen; eine begleitende Dokumentation von Anforderungen und Beschränkungen in Textform bleibt weiterhin notwendig.

Desweiteren werden Einschränkungen, die im Rahmen der genannten Prüfausdrücke aufgenommen werden, nicht in den automatisch generierten Eingabewerkzeugen berücksichtigt. Diese prüfen zwar die Korrektheit in Bezug auf die Basisberechtigung, also $\mathscr{L}(G)$, nicht jedoch alle im Nachgang geforderten oder ausgeschlossenen Wertebereiche. Die Überschreitung des Kompetenzbereiches durch einen Ersteller einer Zusicherung kann daher u. U. nicht durch das Eingabewerkzeug erkannt werden, sondern führt erst bei der Auswertung der Zusicherung zur korrekten Ablehnung.

Teilweise erzwingen Formulierungen bestimmte Konstrukte in der Grammatik und damit in den strukturierten Daten, die beim direkten Entwurf eines Austausch-Formates vermieden werden könnten. Dies wird durch das Prinzip verursacht, dass es keine leeren Knoten (d. h. Blätter, die ein Nicht-Terminal repräsentieren) im AST gibt und somit jede Information durch ein Terminal kodiert werden muss. Dadurch werden teilweise einfache binäre Entscheidungen durch einen Knoten mit einem Wort aus dem jeweiligen Zusicherungstext dargestellt.

Da die Eingabe von Zusicherungen im Primärtextformat im Vergleich zur formularbasierten Eingabemethode vergleichsweise aufwändig ist, ist davon auszugehen, dass das letztgenannte Eingabeverfahren dominieren wird. Die Primärtextdarstellung wird in solchen Anwendungsfällen hauptsächlich während einer Einführungsphase relevant sein, in der Anwender die Bedeutung der von ihnen abgegebenen Zusicherungen erlernen. Später wird sie im täglichen Betrieb eine untergeordnete Rolle spielen und lediglich zu Nachweisen z. B. im Fall externer Audits oder bei juristischen Fragen herangezogen werden.

Da die Erstellung von Grammatiken und das Einrichten von Berechtigungen trotz der unterstützenden Werkzeuge einen unvermeidbaren Aufwand zur Bereitstellung erzeugt, ist davon auszugehen, dass das Verfahren nur in Fällen zum Einsatz kommt, die durch eine standardisierte Grundlage (z. B. IT-Grundschutz) eine breite Anwendergruppe ansprechen oder die durch periodische Ausführung zu einer großen Zahl von Nutzungen führen. Für Fälle von akut auftretendem Prüfbedarf wird das Verfahren eine untergeordnete Bedeutung haben.

6.3 Ausblick

Mit dieser Arbeit wurde das Grundprinzip der grammatikbasierten Zertifikate eingeführt, bewertet und erprobt. Die dabei erkannten Grenzen stellen einen Themenbereich dar, in dem weitere Untersuchungen sinnvoll sind.

Eine wichtige Aufgabenstellung ist die Endanwender-taugliche Darstellung der durch XPath-Ausdrücke spezifizierten Anforderungen und Einschränkungen. Hierbei können vor allem die automatische Erstellung von Beauftragungstexten zur Erstellung von Auditierungen sowie die Bereitstellung angepasster Eingabewerkzeuge, die die jeweils gültigen Einschränkungen berücksichtigen, den Nutzwert erhöhen.

Eine weitere Fragestellung ist die Prüfung der Anwendbarkeit in anderen Anwendungsfeldern als der Auditierung von IT-Systemen. Hierbei ist insbesondere der Bereich online abgeschlossener Rechtsgeschäfte, die Kauf- und Dienstleistungsverträge umfassen, ein viel versprechender

Bereich. Bei solchen Geschäften ist häufig die Akzeptanz umfassender Bedingungen notwendig, gleichzeitig muss jedoch aber auch der jeweils gewählte Dienstleistungsumfang, ggf. mit einer Reihe von Optionen, beauftragt oder eine verbindliche Bestellung diverser Waren bestätigt werden.

Gerade beim Einsatz im kommerziellen Bereich, jedoch auch bei stark verteilten Auditierungsszenarien z. B. in Großunternehmen, stellen sich darüber hinaus architekturelle Fragen nach der sinnvollen Platzierung von Funktionen wie Grammatikserver und Statusauskunftsdiensten. Die daraus resultierenden Vertrauensbeziehungen bedingen ggf. weitere Berechtigungsmechanismen oder Kompetenzannahmen, deren Modellierung über den hier vorgestellten Ansatz hinausgeht.

Die bisher nur prototypische Implementierung sollte in produktiv nutzbare Werkzeuge überführt werden. Hierbei ist insbesondere die Gestaltung der Komponente für den Ersteller von Zusicherungen interessant, da für alle Bereiche, in denen diese Rolle ad-hoc durch Anwender übernommen wird, keine aufwändige Programmverteilung stattfinden kann. Ein möglicher Ansatzpunkt ist die Kombination aus Web-basierten Eingabeformularen zur Erstellung der strukturierten Daten bei gleichzeitiger Integration einer Rückgewinnungs- und Anzeigekomponente im Web-Browser, die für den Anwender die Funktionen Anzeige und Signatur der zu bestätigenden Daten in einem von ihm kontrollierten Bereich vertrauenswürdig bereit stellen.

Um eine solche allgemein verwendbare Infrastruktur realisieren zu können, ist es zudem unverzichtbar, durch öffentlich zugängliche Standardisierung der grammatikbasierten Zertifikate eine Basis für den interoperablen Einsatz zu schaffen. Diese Maßnahme ist für unternehmensinterne Lösungen nicht zwingend notwendig, jedoch für die Nutzung außerhalb der geschlossenen Anwendergruppen kann sie, ggf. verbunden mit der Bereitstellung entsprechender Funktionsbibliotheken, grammatikbasierte Zertifikate als einen Baustein zur Realisierung verbindlicher und für den Anwender nachvollziehbarer elektronischer Kommunikations- und Geschäftsprozesse über das Gebiet der Auditierung von IT-Systemen hinaus etablieren.

A Übersicht über SVO-Logik

A.1 Sprachelemente

Formel[1]	Semantik
$\lfloor X \rfloor_{k^{-1}}$	mit dem privaten Schlüssel k^{-1} signiertes Datum X
$\{X\}_k$	X verschlüsselt unter k
$P \overset{k}{\longleftrightarrow} Q$	k ist ein zwischen P und Q geteiltes Geheimnis
$\mathrm{PK}_\sigma(P,k)$	k ist ein öffentlicher Signaturschlüssel von P
$\mathrm{PK}_\varphi(P,k)$	k ist ein öffentlicher Verschlüsselungsschlüssel von P
$\mathrm{PK}_\delta(P,k)$	k ist ein öffentlicher Schlüssel von P zur Schlüsselaushandlung
$\mathrm{SV}(X,k,Y)$	Die Formel ist wahr, wenn mit k geprüft werden kann, dass X aus Y durch Verwendung des zu k gehörenden Signaturschlüssel hervorgeht.
P received X	P hat die Nachricht X empfangen.
P has X	P ist in Besitz der Nachricht X.
P said X	P hat (irgendwann) in der Vergangenheit die (Teil-)Nachricht X gesendet.
P says X	P hat im aktuellen Protokolllauf die (Teil-)Nachricht X gesendet.
fresh(X)	X war in keiner Nachricht enthalten, die vor dem aktuellen Protokolllauf gesendet wurde.
P believes φ	P kann sich so verhalten, als ob φ wahr ist.
P controls φ	P hat die Kompetenz (Jurisdiktion) über φ. Mit anderen Worten: wenn P φ sagt, so gilt φ als wahr.
X from Q	Nachricht X, die von Q stammt
$\tilde{K}$	komplementärer Schlüssel zu K ($\tilde{K} = K$ im symmetrischen Fall, $\tilde{K} = K^{-1}$ im asymmetrischen Fall)
	Ergänzungen durch die vorliegende Arbeit
current(P,X)	Wenn P X gesagt hat, so sagt P X auch im aktuellen Protokolllauf.
P provides (L,n)	P kann das Recht zur Erstellung von Zusicherungen aus der Sprache L vergeben. n weitere Delegationsstufen sind möglich.
GT(n,m)	Ganzzahl n ist größer als Ganzzahl m

[1] mit P, Q: Prinzipale, X, Y: Nachrichten, k: Schlüssel und φ: Formel, n,m: Ganzzahlen

A.2 Axiome

A.2.1 Überzeugung

$$P \text{ believes } \varphi \wedge P \text{ believes } (\varphi \rightarrow \psi) \rightarrow P \text{ believes } \psi \tag{Ax1}$$

$$P \text{ believes } \varphi \rightarrow \varphi \tag{Ax2}$$

$$P \text{ believes } \varphi \rightarrow P \text{ believes } (P \text{ believes } \varphi) \tag{Ax3}$$

$$\neg(P \text{ believes } \varphi) \rightarrow P \text{ believes } (\neg P \text{ believes } \varphi) \tag{Ax4}$$

A.2.2 Quellenzuordnung

$$(P \overset{K}{\longleftrightarrow} Q \wedge R \text{ received } \{X \text{ from } Q\}_K) \rightarrow (Q \text{ said } X \wedge Q \text{ has } K) \tag{Ax5}$$

$$(\mathrm{PK}_\sigma(Q, K) \wedge R \text{ received } X \wedge \mathrm{SV}(X, K, Y)) \rightarrow Q \text{ said } Y \tag{Ax6}$$

A.2.3 Schlüsselaushandlung

$$((\mathrm{PK}_\delta(P, K_p)) \wedge (\mathrm{PK}_\delta(Q, K_q))) \rightarrow P \overset{F_0(K_p,K_q)}{\longleftrightarrow} Q \tag{Ax7}$$

$$\varphi \equiv \varphi[F_0(K, K')/F_0(K', K)] \tag{Ax8}$$

A.2.4 Empfang von Nachrichten

$$P \text{ received } (X_1, \ldots, X_n) \rightarrow P \text{ received } X_i \tag{Ax9}$$

$$(P \text{ received } \{X\}_K \wedge P \text{ has } \tilde{K}) \rightarrow P \text{ received } X \tag{Ax10}$$

$$P \text{ received } \lfloor X \rfloor_K \rightarrow P \text{ received } X \tag{Ax11}$$

A.2.5 Besitz von Information

$$P \text{ received } X \rightarrow P \text{ has } X \tag{Ax12}$$

$$P \text{ has } (X_1, \ldots, X_n) \rightarrow P \text{ has } X_i \tag{Ax13}$$

$$(P \text{ has } X_1 \wedge \ldots \wedge P \text{ has } X_n) \rightarrow (P \text{ has } F(X_1, \ldots, X_n)) \tag{Ax14}$$

A.2.6 Verständnis von Informationen

$$P \text{ believes } (P \text{ has } F(X)) \rightarrow P \text{ believes } (P \text{ has } X) \tag{Ax15}$$

A.2.7 Aussagen

$$P \text{ said } (X_1, \ldots, X_n) \rightarrow (P \text{ said } X_i \wedge P \text{ has } X_i) \tag{Ax16}$$

$$P \text{ says } (X_1, \ldots, X_n) \rightarrow (P \text{ said } (X_1, \ldots . X_n) \wedge P \text{ says } X_i) \tag{Ax17}$$

A.2.8 Jurisdiktion

$$(P \text{ controls } \varphi \wedge P \text{ says } \varphi) \rightarrow \varphi \tag{Ax18}$$

A.2.9 Frische

$$\text{fresh}(X_i) \rightarrow \text{fresh}(X_1, \ldots, X_n) \tag{Ax19}$$

$$\text{fresh}(X_1, \ldots, X_n) \rightarrow \text{fresh}(F(X_1, \ldots, X_n)) \tag{Ax20}$$

A.2.10 Prüfung von Frische

$$(\text{fresh}(X) \wedge P \text{ said } X) \rightarrow P \text{ says } X \tag{Ax21}$$

A.2.11 Symmetrische Eignung von geteilten Schlüsseln

$$P \overset{K}{\longleftrightarrow} Q \equiv Q \overset{K}{\longleftrightarrow} P \tag{Ax22}$$

A.2.12 Ergänzung: Aktualität

$$Q \text{ said } X \wedge \text{current}(Q, X) \rightarrow Q \text{ says } X \tag{Ax23}$$

A.3 Ergänzungen für grammatikbasierte Zertifikate

A.3.1 Interpretation für grammatikbasierte Zertifikate

$$Q \text{ says } \mathrm{GC}(X, \mathrm{H}(G), \mathcal{M}) \wedge \mathrm{GCV}(X, G, p) \rightarrow Q \text{ says } p \qquad \text{(GC1)}$$

A.3.2 Kompetenz für Sprachelemente

$$P \text{ controls } \mathscr{L}(G, r) \wedge \mathrm{GCV}(X, G, p) \wedge \mathrm{GCR}(X, r) \rightarrow P \text{ controls } p \qquad \text{(GC2)}$$

A.3.3 Delegation

$$\begin{aligned}(Q \text{ says } \mathrm{GDC}(P, \mathrm{H}(G), r_1, w, \mathcal{M}) \ \wedge \ Q \text{ provides } (\mathscr{L}(G, r_0), n) \ \wedge \ \mathbf{GT}(n, w)) \rightarrow \\ (P \text{ controls } \mathscr{L}(G, \mathrm{and}(r_0, r_1)) \ \wedge \ P \text{ provides } (\mathscr{L}(G, \mathrm{and}(r_0, r_1)), w))\end{aligned} \qquad \text{(GC3)}$$

B Anwendungsbeispiele für formale Analysemethoden

Dieser Anhang enthält ausführliche Beispiele zur Demonstration der Anwendung der formalen Analysemethoden SVO und Proverif aus Kapitel 3.

B.1 SVO: Prüfung signierter Daten mit PKI-Diensten

Als Beispiel für die Anwendung der SVO-Logik soll der Empfang eines signierten Datums und die zugehörige Auswertung von Zertifikaten betrachtet werden. Alice (A) erhält von Bob (B) ein von ihm signiertes Datum D sowie das zur Prüfung notwendige Zertifikat. Alice verfügt lediglich über den Schlüssel K_T des ausstellenden Trust Centers T sowie die Zertifikate seiner OCSP-Dienste und prüft die Gültigkeit des Zertifikats von Bob gegen das Ausstellerzertifikat und den OCSP-Responder O des Trust Centers.

Das Zertifikat, das B vorlegt, enthält neben Schlüssel K_B und Identität B noch eine Seriennummer S, die Ausstellerangabe T und einen Verweis auf den zuständigen OCSP-Dienst O:

$$\lfloor S, T, B, K_B, O \rfloor_{K_T^{-1}} \tag{B.1}$$

Der OCSP-Responder verwendet das Datum g zur Kennzeichnung des Status *good* (alternativ sind r: revoked, u: unknown) gemäß RFC 2560 [MAM+99].

Für die Modellierung werden keine Zeitangaben in den Zertifikaten betrachtet, sondern es wird davon ausgegangen, dass die Kommunikation innerhalb des Gültigkeitszeitraums aller Zertifikate stattfindet, da Zeitangaben in den Zertifikaten ohne Nutzung von Abfrageprotokollen auf der Anwenderseite geprüft werden können und somit schon vor Anwendung des untersuchten Protokollablaufs ihre Eignung sichergestellt werden kann.

B.1.1 Zielfestlegung

Es soll gezeigt werden, dass A zu der Überzeugung kommen kann, dass das Datum D von B gesendet wurde.

$$A \text{ believes } B \text{ said } D$$

B.1.2 Protokollbeschreibung

Am Kommunikationsablauf aus drei Nachrichten sind A, B und der OCSP-Dienst (O) beteiligt.

$$
\begin{aligned}
&(1) \quad B \mapsto A : (\lfloor D \rfloor_{K_B^{-1}}, \lfloor S, T, B, K_B, O \rfloor_{K_T^{-1}}) \\
&(2) \quad A \mapsto O : (H(T), S, N) \\
&(3) \quad O \mapsto A : \lfloor H(T), S, g, N \rfloor_{K_O^{-1}}
\end{aligned}
$$

Mit der ersten Nachricht (1) erhält A das signierte Datum und das Zertifikat von B. Mit Nachricht (2) fragt A bei O nach einer Statusauskunft bezüglich des Zertifikats mit dem Hashwert über die Ausstellerkennung $H(T)$, der Seriennummer S und fügt einen Zufallswert N (Nonce) hinzu. Mit Nachricht (3) antwortet O mit der Statusauskunft.

Bei der Analyse von Authentisierungsprotokollen kann die Zeit in Epochen eingeteilt werden, die durch den Beginn und das Ende eines Protokolllaufs definiert werden. Die aktuelle Epoche ist somit die Zeit nach der ersten Nachricht des zu untersuchenden Laufes eines Authentisierungsprotokolls, in der eine Nachricht frisch sein kann (vgl. Tabelle 3.1).

B.1.3 Prämissen aus initialem Wissen

Im ersten Modellierungsschritt werden die Prämissen notiert, die das initiale Wissen und die getroffenen Annahmen durch A beschreiben.

$$
\begin{aligned}
&A \text{ believes } \mathrm{PK}_\sigma(T, K_T) && \text{(P1)} \\
&A \text{ believes } \mathrm{PK}_\sigma(O, K_O) && \text{(P2)} \\
&A \text{ believes } T \text{ controls } \mathrm{PK}_\sigma(B, K_B) && \text{(P3)} \\
&A \text{ believes } T \text{ controls } (O \text{ controls current}(T, Z)) && \text{(P4)} \\
&A \text{ believes current}(T, O \text{ controls current}(T, Z)) && \text{(P5)} \\
&A \text{ believes fresh}(N) && \text{(P5)}
\end{aligned}
$$

A kennt die öffentlichen Schlüssel von T und jedem infrage kommenden O und kann diese korrekt zuordnen (P1, P2). Bei der Modellierung wird nur ein beliebiger aber fester OCSP-Responder berücksichtigt.

(P3) besagt, dass A dem Trust Center die Kompetenz einräumt, Zertifizierungsaussagen zu machen. Daneben kann das Trust Center Aussagen zur Aktualität der Zertifikate an O delegieren (P4). Diese Delegation im Rahmen der Ausstellung kann nicht widerrufen werden und ist daher zu allen Zeiten aktuell (P5). Z steht dabei für ein beliebiges Zertifikat.

A darf die Frische des von ihr erzeugten Zufallswertes N voraussetzen (P5).

B.1.4 Prämissen aus empfangenen Nachrichten

Bei der Modellierung der Nachrichten genügt es, die von A empfangenen Nachrichten zu betrachten. Die von A gesendete Nachricht wird bereits durch (P5) abgedeckt, da die Kenntnis der Frische der einzige relevante Aspekt ist, der zum Wissen von A beiträgt. Die Nachrichten sind somit

$$A \text{ received } (\lfloor D \rfloor_{K_B^{-1}}, \lfloor S, T, B, K_B, O \rfloor_{K_T^{-1}})$$
$$A \text{ received } \lfloor H(T), S, g, N \rfloor_{K_O^{-1}}$$

Die Wiedererkennung bzw. das Verständnis der Elemente in den empfangenen Nachrichten wird durch die folgenden Prämissen wiedergegeben und damit die subjektive Sichtweise berücksichtigt. Dabei wurde die erste Nachricht bereits in die Bestandteile zerlegt und die jeweiligen Interpretationen der Felder durch SVO-Terme gefasst.

$$A \text{ believes } A \text{ received } \lfloor \langle D \rangle_{*A} \rfloor_{K_B^{-1}} \tag{P6}$$
$$A \text{ believes } A \text{ received } \lfloor \langle S \rangle_{*A}, T, \mathrm{PK}_\sigma(B, \langle K_B \rangle_{*A}), O \text{ controls current}(T, Z) \rfloor_{K_T^{-1}} \tag{P7}$$
$$A \text{ believes } A \text{ received } \lfloor \text{current}(T, \langle S \rangle_{*A}, T, \mathrm{PK}_\sigma(B, \langle K_B \rangle_{*A})), N \rfloor_{K_O^{-1}} \tag{P8}$$

In (P6) und (P7) sind für A das signierte Datum, die Seriennummer des Zertifikats sowie der Prüfschlüssel des Signierers neue Daten und werden daher in der $\langle X \rangle_{*A}$-Schreibweise notiert. Bei der Interpretation der Nachrichten wird die Nennung von B und K_B definitionsgemäß als Subjekt-Schlüssel-Beziehung interpretiert. Die Nennung von O wird als Kompetenzaussage gewertet, dass O die Aktualität von Zertifikaten zusichern kann, die T erstellt hat. Aus der Antwort des OCSP-Responders in (P8) wird der Status g als Aktualitätszusicherung für das Zertifikat mit der Seriennummer S von Aussteller T und damit für das vorliegende Zertifikat interpretiert.

Zur Erhöhung der Lesbarkeit wird für die weiteren Umformungen bei den Größen K_B, S und D auf die Notation $\langle X \rangle_{*A}$ verzichtet, da keine der Größen im Weiteren als Funktionsergebnis interpretiert wird.

Für die Zertifikatsaussage wird die Abkürzung Z_B eingeführt:

$$Z_B =_{\text{def}} (S, T, \mathrm{PK}_\sigma(B, K_B), O \text{ controls current}(T, Z)) \tag{B.2}$$

A kann die Signatur des Zertifikats erfolgreich mit dem Schlüssel von T prüfen.

$$A \text{ believes } \mathrm{SV}(\lfloor Z_B \rfloor_{K_T^{-1}}, K_T, Z_B) \tag{P9}$$

Ebenso seien die Prüfungen für die anderen Signaturen erfolgreich.

$$I =_{\text{def}} \text{current}(T, (S, T, \mathrm{PK}_\sigma(B, K_B))), N \tag{B.3}$$
$$A \text{ believes } \mathrm{SV}(\lfloor I \rfloor_{K_O^{-1}}, K_O, I) \tag{P10}$$
$$A \text{ believes } \mathrm{SV}(\lfloor D \rfloor_{K_B^{-1}}, K_B, D) \tag{P11}$$

Mit den aufgestellten Prämissen und den Axiomen der SVO-Logik kann nun der Nachweis geführt werden, dass A von der Authentizität der signierten Nachricht überzeugt werden kann.

B.1.5 Durchführung von Ableitungsschritten

Mit den gegebenen Prämissen und Axiomen wird im Rahmen einer Ableitung versucht, das definierte Ziel herzuleiten. Die bei den Ableitungsschritten verwendeten Schreibweisen werden im Folgenden dargestellt. Für die Anwendung des i-ten Axioms (Ax*i*) der SVO-Logik auf den konkreten Anwendungsfall werden dessen allgemeinen Variablen durch konkrete Bezeichner aus dem untersuchten Anwendungsfall ersetzt, dargestellt durch eine Substitutionsfunktion $\sigma(\tau)$, die alle x_i durch y_i ersetzt (Schreibweise $\{y_i/x_i\}$). Durch die Nec-Regel wird diese dann den Überzeugungen von A hinzugefügt.

$$\frac{\vdash \sigma(\text{Ax}i)}{A \text{ believes } \sigma(\text{Ax}i)}\ \text{Nec} \qquad \text{mit } \sigma(\tau) = \tau\{y_1/x_1\}\ldots\{y_n/x_n\}$$

Für weitere Schlussfolgerungen kann die Kombination aus Axiom (Ax1) und der Modus-Ponens-Regel angewendet werden.

$$\frac{A \text{ believes } \varphi \wedge A \text{ believes } (\varphi \rightarrow \psi),\quad (A \text{ believes } \varphi \wedge A \text{ believes } (\varphi \rightarrow \psi)) \rightarrow A \text{ believes } \psi}{A \text{ believes } \psi}\ \text{MP}$$

Dieser Ausdruck kann auch kürzer geschrieben werden:

$$\frac{A \text{ believes } \varphi \wedge A \text{ believes } (\varphi \rightarrow \psi)}{A \text{ believes } \psi}\ \text{Ax1, MP}$$

B.1.6 Ableitung

Im ersten Schritt stellt A die Authentizität des Zertifikates fest. Das zugehörige Axiom ist (Ax6). Dieses wird durch entsprechende Ersetzung und Übernahme in die Überzeugungen von A anwendbar gemacht.

$$\frac{\vdash \sigma((\text{PK}_\sigma(Q,K) \wedge R \text{ received } X \wedge \text{SV}(X,K,Y)) \rightarrow Q \text{ said } Y)}{\vdash A \text{ believes } \Big(\text{PK}_\sigma(T,K_T) \wedge A \text{ received } \lfloor Z_B \rfloor_{K_T^{-1}} \wedge \text{SV}(\lfloor Z_B \rfloor_{K_T^{-1}}, K_T, Z_B) \rightarrow T \text{ said } Z_B\Big)}\ \text{Nec} \quad \text{(C1)}$$

$$\text{mit } \sigma(\tau) = \tau\{T/Q\}\{K_T/K\}\{A/R\}\{\lfloor Z_B \rfloor_{K_T^{-1}}/X\}\{Z_B/Y\}$$

Das der Formelzählung vorangestellte „C“ steht für Schlussfolgerung (Conclusio). Damit wird jeweils das Ergebnis des Ableitungsschrittes referenziert.

Mittels der übernommenen Formel (C1) und den Prämissen (P1), (P7) und (P9) sowie der Schreibweise aus (B.2) kann die Nachricht Z_B dem Absender T über die *said*-Relation zugeordnet werden.

$$\varphi = \overbrace{\text{PK}_\sigma(T,K_T)}^{\text{aus P1}} \wedge \overbrace{A \text{ received } \lfloor Z_B \rfloor_{K_T^{-1}}}^{\text{aus P7}} \wedge \overbrace{\text{SV}(\lfloor Z_B \rfloor_{K_T^{-1}}, K_T, Z_B)}^{\text{aus P9}}$$

$$\psi = T \text{ said } Z_B$$

$$\frac{\overbrace{A \text{ believes } \varphi}^{P1,P7,P9} \wedge \overbrace{A \text{ believes } (\varphi \rightarrow \psi)}^{C1}}{A \text{ believes } (T \text{ said } Z_B)} \text{ Ax1, MP} \tag{C2}$$

Da jeder weitere Ableitungsschritt genau diesem Schema folgt, wird im Weiteren nur jeweils das abgeleitete Ergebnis und die dazu verwendeten Prämissen, Zwischenergebnisse und Ableitungsregeln angegeben.

Nach (C2) ist T said $(S, T, \mathrm{PK}_\sigma(B, K_B), O$ controls current$(T, Z))$ gegeben. Mit (Ax16) können Teilaussagen betrachtet werden:

$$A \text{ believes } T \text{ said } O \text{ controls current}(T, Z) \tag{C3}$$
durch Ax16, Nec, C2, Ax1, MP

Da diese Aussage per Festlegung in (P5) als aktuell gilt, ist auch die Äußerung aktuell.

$$A \text{ believes } T \text{ says } O \text{ controls current}(T, Z) \tag{C4}$$
durch Ax23, Nec, P5, C3, Ax1, MP

A akzeptiert die Aussage von T über O, da mit (P4) die notwendige Kompetenz vorliegt.

$$A \text{ believes } O \text{ controls current}(T, Z) \tag{C5}$$
durch Ax18, Nec, C4, P4, Ax1, MP

Die Auskunft des OCSP ist authentisch.

$$A \text{ believes } O \text{ said } (\text{current}(T, (S, T, \mathrm{PK}_\sigma(B, K_B))), N) \tag{C6}$$
durch Ax6, Nec, P2, P8P10, Ax1, MP

Da N frisch ist, ist auch die Auskunft des OCSP frisch.

$$A \text{ believes fresh}(\text{current}(T, (S, T, \mathrm{PK}_\sigma(B, K_B))), N) \tag{C7}$$
durch Ax19, Nec, P5, Ax1, MP

Damit ist die Äußerung der Auskunft aktuell.

$$A \text{ believes } O \text{ says } (\text{current}(T, (S, T, \mathrm{PK}_\sigma(B, K_B))), N) \tag{C8}$$
durch Ax21, Nec, C7, C6, Ax1, MP

Es wird nur ein Teil der Aussage benötigt.

$$A \text{ believes } O \text{ says current}(T, (S, T, \mathrm{PK}_\sigma(B, K_B))) \tag{C9}$$
durch Ax17, Nec, C8, Ax1, MP

Da A durch die Aussage von T von der Kompetenz von O überzeugt ist, wird die Auskunft als Fakt übernommen.

$$A \text{ believes current}(T, (S, T, \mathrm{PK}_\sigma(B, K_B))) \tag{C10}$$
durch Ax18, Nec, C5, C9, Ax1, MP

Es wird der Zertifizierungsteil der Aussage von T betrachtet.

$$A \text{ believes } T \text{ said } (S, T, \mathrm{PK}_\sigma(B, K_B)) \tag{C11}$$
durch Ax16, Nec, C2, Ax1, MP

Da die Aktualitätsaussage des OCSP vorliegt, ist die Äußerung aktuell.

$$A \text{ believes } T \text{ says } (S, T, \mathrm{PK}_\sigma(B, K_B)) \tag{C12}$$
durch Ax23, Nec, C11, C10, Ax1, MP

Damit wird auch die Teilaussage der Schlüsselzuordnung aktuell von T geäußert.

$$A \text{ believes } T \text{ says } \mathrm{PK}_\sigma(B, K_B) \tag{C13}$$
durch Ax17, Nec, C12, Ax1, MP

T hat als Trust-Center die Kompetenz, Aussagen über Schlüsselzuordnungen zu machen (P3). Damit übernimmt A die Schlüsselzuordnung.

$$A \text{ believes } \mathrm{PK}_\sigma(B, K_B) \tag{C14}$$
durch Ax18, Nec, P3, C13, Ax1, MP

Die Signaturprüfung belegt die Authentizität der Daten D.

$$A \text{ believes } B \text{ said } D \qquad \square \tag{C15}$$
durch Ax6, Nec, C14, P6, P11, Ax1, MP

B.2 Proverif: Anwendungsbeispiel PKI-Dienste

Zur Demonstration der Anwendung von Proverif wird das gleiche Beispiel wie im Fall der SVO-Logik verwendet (vgl. Kapitel B.1).

Während es bei der Betrachtung auf Basis der SVO-Logik ausreicht, die Beobachtungen durch A zu modellieren, werden in Proverif auch B und der OCSP-Dienst modelliert.

Das Verhalten der Beteiligten wird entsprechend der vorgestellten Modellierungsmöglichkeiten hinsichtlich der Generierung von Daten, Sende- und Empfangsvorgänge und Signatur- und Prüfoperationen dargestellt.

```
1  (* Beispiel Signierte Daten und OCSP-Abfrage *)
2  set traceDisplay = long.
3  set verboseRules = true.
4
5  (* Verschlüsselung mit symmetrischen Schlüsseln *)
6  type key.
7  fun senc(bitstring, key): bitstring.
8  reduc forall m: bitstring, k: key; sdec(senc(m, k), k) = m.
9
10 (* Asymmetrische Verschlüsselung *)
11
12 type skey.
13 type pkey.
14
15 fun pk(skey): pkey.
16 fun aenc(bitstring, pkey): bitstring.
17
18 reduc forall m: bitstring, sk: skey; adec(aenc(m, pk(sk)), sk) = m.
19
20 (* Hash-Funktion *)
21 fun H(bitstring):bitstring.
22
23 (* Asymmetrische Schlüssel und Funktionen für Signaturen *)
24
25 type sskey.
26 type spkey.
27
28 fun spk(sskey): spkey.
29 fun sign(bitstring, sskey): bitstring.
30
31 reduc forall m: bitstring, ssk: sskey; getmess(sign(m, ssk)) = m.
32 reduc forall m: bitstring, ssk: sskey; checksign(sign(m, ssk), spk(ssk)) = m.
33
34 (* Distinguished Name *)
35 type distname.
36
37 (* Beteiligte *)
38 const T:distname. (* Trust Center *)
39 const O:distname. (* OCSP Service *)
40 const A:distname. (* Alice *)
41 const B:distname. (* Bob *)
42
43 (* Konstanten für OCSP-Auskünfte *)
44 type ocspstatus.
45 const   good:ocspstatus.
46 const revoked:ocspstatus.
47 const unknown:ocspstatus.
48
49 (* Konvertierungsfunktion *)
50 fun makeissuerref(distname):bitstring [data].
51
52 (* Allgemeines Kommunikationsmedium *)
```

```
53  free c:channel.
54
55  (* Tabelle mit vertrauenswürdigen Zertifikaten
56   vgl. CA-Listen in Browsern *)
57  table trustedKeys(distname, spkey).
58
59  (* Datenbank des OCSP *)
60  table ocspDB(bitstring, bitstring).
61
62  (* Ereignisse *)
63  event gotDatafrom(bitstring, distname).
64  event newdata(bitstring, distname).
65
66  (* Prüfungen *)
67  (* Erreichbarkeit des finalen Ereignisses *)
68  query daten:bitstring, sender:distname;
69        event(gotDatafrom(daten, sender)).
70
71  (* Daten werden nur als authentisch gesehen, wenn sie zuvor
72   vom Sender geschickt wurden *)
73  query daten:bitstring, sender:distname;
74        event(gotDatafrom(daten, sender)) ==> event(newdata(daten, sender)).
75
76  let userA =
77     (* empfange signiertes Datum und Zertifikat *)
78     in(c, (signedData:bitstring, cert:bitstring));
79     (* Zerlege Zertifikat *)
80     let (serial:bitstring,
81        issuer:distname,
82        subject:distname,
83        subjectkey:spkey,
84        ocspref:distname) = getmess(cert) in
85     (* hole Schlüssel aus Vertrauensliste *)
86     get trustedKeys(= issuer, pkT) in
87     (* und prüfe die Signatur des Zertifikats *)
88     let (= serial, = issuer, = subject, = subjectkey, = ocspref)
89     = checksign(cert, pkT) in
90        (* Abfrage am OCSP mit nonce N *)
91        new N:bitstring;
92        out(c, (H(makeissuerref(issuer)), serial, N));
93        (* warte auf Antwort vom OCSP-Dienst *)
94        in(c, ocspresp:bitstring);
95        (* Hole Prüfschlüssel entsprechend OCSPref aus Zertifikat *)
96        get trustedKeys(= ocspref, pkO) in
97           (* Signaturprüfung mit gleichzeitiger Prüfung auf 'good' *)
98           let (= H(makeissuerref(issuer)), = serial, = N, = good)
99           = checksign(ocspresp, pkO) in
100          (* prüfe Signatur der Daten *)
101          let data:bitstring = checksign(signedData, subjectkey) in
102          (* Daten werden als authentisch von subject akzeptiert *)
103          event gotDatafrom(data, subject)
```

```
104  .
105
106  let userB(skB:sskey, BobsZert:bitstring) =
107     (* generiere neues Datum *)
108     new d:bitstring;
109     (* Ereignis zur Verwendung in Prüfung *)
110     event newdata(d, B);
111     (* Übertragung von signierten Daten und Zertifikat *)
112     out(c, (sign(d, skB), BobsZert))
113  .
114
115  let OCSP(skO:sskey) =
116     (* Warte auf Anfrage *)
117     in(c, (issuerhash:bitstring, serial:bitstring, nonce:bitstring));
118     get ocspDB(= issuerhash, = serial) in
119        (* Zertifikat vorhanden *)
120        out(c, sign((issuerhash, serial, nonce, good), skO))
121  .
122
123  process
124     (* Schlüssel für Trust Center und OCSP erstellen ... *)
125     new skT:sskey;
126     new skO:sskey;
127     (* ... und in die Liste der vertrauenswürdigen Schlüssel eintragen *)
128     let pkT = spk(skT) in insert trustedKeys(T, pkT);
129     let pkO = spk(skO) in insert trustedKeys(O, pkO);
130     (* Schlüssel für Bob *)
131     new skB:sskey;
132     let pkB = spk(skB) in
133        (* Ausstellung des Zertifikats *)
134        new s:bitstring;
135        (* Zertifikat auch im OCSP hinterlegen *)
136        insert ocspDB(H(makeissuerref(T)), s);
137        let Bcer = sign((s, T, B, pkB, O), skT) in
138     (* Prozesse der (nach der Ausstellung) Beteiligten instantiieren *)
139     ((!userA) | (!userB(skB, Bcer)) | (!OCSP(skO)))
140
```

Die Erreichbarkeit des Authentisierungsschrittes ist gegeben.

```
The event gotDatafrom(d_1550,B) is executed.
A trace has been found.
RESULT not event(gotDatafrom(daten_824,sender_825)) is false.
```

Die Korrespondenz zur Sicherstellung der Authentizität der empfangenen Daten ist ebenfalls nachweisbar.

```
Starting query event(gotDatafrom(daten,sender)) ==>
event(newdata(daten,sender))
```

```
goal reachable: begin(newdata(d[!1 = @sid_821],B)) ->
end(gotDatafrom(d[!1 = @sid_821],B))
RESULT event(gotDatafrom(daten,sender)) ==>
event(newdata(daten,sender)) is true.
```

Durch Einführung eines Phasenmodells können für das Modell auch die Sperrung des Zertifikats und die Notwendigkeit der Prüfung des Nonce-Wertes nachgewiesen werden.

In die Prozesse wird eine Teilung in die Phasen 0 und 1 integriert. In Phase 1 veröffentlicht B seinen privaten Schlüssel (Schlüsselkompromittierung). Der OCSP-Dienst antwortet mit dem Status „revoked“. A empfängt und prüft Nachrichten auf die gleiche Weise wie in der ersten Phase.

Durch die negative Auskunft des OCSP-Dienstes ist das Ereignis gotDatafrom’ nicht erreichbar.

Wird die Prüfung des Nonce-Wertes unterbunden (Austausch des Terms =N’ durch N”:bitstring), so gelingt es dem Angreifer durch Kenntnis des privaten Schlüssels von B und Wiedereinspielen einer in Phase 0 ausgestellten OCSP-Auskunft den Prüfpunkt gotDatafrom’ zu erreichen. Die Korrespondenz des Prüfpunktes mit dem Kontrollereignis beim Ausstellen der Nachricht schlägt fehl.

C Verwendete Grammatiken

Dieser Anhang enthält einige Grammatiken, die im Rahmen der Mechanismen und Beispiele verwendet wurden.

C.1 Meta-Grammatik

Die nachstehenden Meta-Grammatik, die in ANTLRv3-Syntax notiert ist, definiert die Form der Grammatikdateien für grammatikbasierte Zertifikate. Mit Parsern auf Basis dieser Meta-Grammatik kann somit die Wohlgeformtheit einer GC-Grammatik überprüft werden.

```
/*
ANTLRv3-Grammatik zur Prüfung von Grammatiken für grammatikbasierte Zertifikate
(C) 2012 Matthias Kabatnik
*/

grammar GCG;

options {
   output=AST;
   ASTLabelType=CommonTree;
}

gcgGrammatik
   :      (knotenRegel | nonKnotenRegel)+ EOF
   ;

knotenRegel
   :   TOKIDENTIFIER  ':'  rhs ';'
   ;

nonKnotenRegel
   :   (TOKIDENTIFIER|NTOKIDENTIFIER) INV ':'  rhs ';'
   ;


rhs   :      (terminaleVariante|nonTerminaleVariante)
                                          ( '|' (terminaleVariante|nonTerminaleVariante) )*
     ;

terminaleVariante
   :   (terminal INV quantor?)*
      terminalerBaustein quantor?
      (terminal INV quantor?)*
   ;

nonTerminaleVariante
```

```
   :   ((terminal VIS?| LEXEM) quantor?)*
      nonTerminalerBaustein quantor?
      ((((terminal VIS?)|LEXEM)|nonTerminalerBaustein) quantor?)*
   ;

terminalerBaustein
   :   terminal | LEXEM
   ;

nonTerminalerBaustein
   :   NTOKIDENTIFIER | TOKIDENTIFIER
   ;

terminal   :      CHAR_LITERAL | STRING_LITERAL | WSMAKRO
   ;

quantor   :   '?' | '*' | '+'
   ;

// Lexikalische Regeln

INV:   '!'   // markiert Terminale, Lexeme und terminale Knoten als
   ;         // unsichtber (INVisble)

VIS:   '^'   // erzwingt die Sichtbarkeit eines Terminals in einem
   ;         // inneren Knoteni (VISible)

WSMAKRO   :   'WS'   // Lexem als Platzhalter für einzelne Leerzeichen
   ;

/* Die folgenden Lexer-Regeln wurden teilweise aus der ANTLRv3-Grammatik
  (Copyright (c) 2005-2007 Terence Parr  All rights reserved) übernommen
*/

SL_COMMENT
    :   '//'
        ~('\r'|'\n')*
      '\r'? '\n'
      {$channel=HIDDEN;}
   ;

ML_COMMENT
   :   '/*'  .* '*/' {$channel=HIDDEN;}
   ;

CHAR_LITERAL
   :   '\'' LITERAL_CHAR '\''
   ;

STRING_LITERAL
   :   '\'' LITERAL_CHAR LITERAL_CHAR* '\''
   ;

fragment
LITERAL_CHAR
   :   ESC
   |   ~('\''|'\\')
   ;

fragment
ESC   :   '\\' ( 'n' | 'r' |'t'
      |   'b'
      |   'f'
      |   '"'
      |   '\''
```

```
      |   '\\'
      |   '>'
      |   'u' XDIGIT XDIGIT XDIGIT XDIGIT
      |   . // unknown, leave as it is
      )
   ;

fragment
XDIGIT   :   '0' .. '9' | 'a' .. 'f' | 'A' .. 'F'
   ;

INT   :   '0'..'9'+
   ;

NTOKIDENTIFIER   : 'a'..'z' ID
   ;

TOKIDENTIFIER   // ID, die auch als Token (Knotenname) des AST zulässig ist
   :   'A'..'Z'('A'..'Z'|'_'|'0'..'9')* 'a'..'z' ID
   ;

fragment
ID   :   ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*
   ;

LEXEM   :   ('A'..'Z'|'_'|'0'..'9')+
   ;

WS   :   ( ' ' | '\t' | '\r'? '\n')+ {$channel=HIDDEN;}
   ;
```

C.2 XML-Schema für grammatikbasierte Zertifikate

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:gc="http:///GCertSchema/
    GrammarbasedCertificate/GrammarbasedCertificate.xsd" xmlns:xsd="http://www.w3.org/2001/
    XMLSchema" targetNamespace="http:///GCertSchema/GrammarbasedCertificate/
    GrammarbasedCertificate.xsd">
   <xsd:element name="GrammarbasedCertificate">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Version" type="string"></xsd:element>
            <xsd:element name="SerialNumber" type="xsd:hexBinary">
            </xsd:element>
            <xsd:element name="Issuer" type="gc:entityType"></xsd:element>
            <xsd:element name="Validity" type="gc:validityType"></xsd:element>
            <xsd:element name="GrammarReference" type="gc:grammarReferenceType"></xsd:element>
            <xsd:element name="Assertion" type="gc:assertionType" maxOccurs="unbounded"></xsd:
                element>
            <xsd:element name="Nonce" type="xsd:base64Binary" minOccurs="0"></xsd:element>
            <xsd:element name="Extensions" type="gc:extensionsType" minOccurs="0"></xsd:element
                >
         </xsd:sequence>
          <attribute name="ID" type="string" fixed="tbsGCertificate"></attribute>
      </xsd:complexType>
   </xsd:element>

   <xsd:complexType name="validityType">
      <xsd:sequence>
         <xsd:element name="NotBefore" type="xsd:timeInstant"></xsd:element>
         <xsd:element name="NotAfter" type="xsd:timeInstant" minOccurs="0"></xsd:element>
```

```
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="grammarReferenceType">
   <xsd:sequence>
      <xsd:element name="DigestMethod" type="gc:digestMethodType"></xsd:element>
      <xsd:element name="DigestValue" type="xsd:base64Binary"></xsd:element>
      <xsd:element name="URI" type="xsd:anyURI" minOccurs="0"></xsd:element>
      <xsd:element name="GCGrammarServer" type="xsd:anyURI" minOccurs="0"></xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="digestMethodType">
   <xsd:restriction base="string">
      <xsd:enumeration value="sha256"></xsd:enumeration>
      <xsd:enumeration value="sha384"></xsd:enumeration>
      <xsd:enumeration value="sha512"></xsd:enumeration>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="assertionType">
   <xsd:sequence>
      <xsd:element name="ParseTree" type="gc:wildcardType"></xsd:element>
      <xsd:element name="Validity" type="gc:validityType" minOccurs="0"></xsd:element>
      <xsd:element name="GrammarReference" type="gc:grammarReferenceType" minOccurs="0"></
          xsd:element>
      <xsd:element name="PrimarySignature" type="gc:primarySignatureType" minOccurs="0"></
          xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="wildcardType">
    <xsd:sequence maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
     </xsd:sequence>
 </xsd:complexType>

<xsd:complexType name="extensionsType">
   <xsd:sequence>
      <xsd:element name="AuthorityKeyIdentifier" type="xsd:base64Binary" minOccurs="0"></xsd
          :element>
      <xsd:element name="AuthorityInformationAccess" type="gc:authorityInformationAccessType
          " minOccurs="0" maxOccurs="unbounded"></xsd:element>
      <xsd:element name="CRLDistributionPoint" type="gc:cRLDistributionPointType" minOccurs
          ="0"></xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="authorityInformationAccessType">
   <xsd:sequence>
      <xsd:element name="AccessMethod" type="gc:accessMethodType"></xsd:element>
      <xsd:element name="AccessLocation" type="xsd:anyURI"></xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="accessMethodType">
   <xsd:restriction base="string">
      <xsd:enumeration value="ocsp"></xsd:enumeration>
      <xsd:enumeration value="caIssuers"></xsd:enumeration>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="cRLDistributionPointType">
   <xsd:sequence>
```

```
         <xsd:element name="CRLDistributionPointName" type="xsd:anyURI" minOccurs="0" maxOccurs
             ="unbounded"></xsd:element>
         <xsd:element name="CRLIssuer" type="gc:entityType" minOccurs="0"></xsd:element>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="primarySignatureType">
      <xsd:sequence>
         <xsd:element name="SignatureAlgorithm" type="gc:signatureAlgorithmType"></xsd:element>
         <xsd:element name="SignatureValue" tpye="xsd:base64Binary"></xsd:element>
         <xsd:element name="X509Certificate" tpye="xsd:base64Binary" minOccurs="0"></xsd:
             element>
         <xsd:element name="PrimarySigner" tpye="gc:entityType" minOccurs="0"></xsd:element>
         <xsd:element name="PrimaryAuthorityKeyIdentifier" tpye="xsd:base64Binary" minOccurs
             ="0"></xsd:element>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:simpleType name="signatureAlgorithmType">
      <xsd:restriction base="string">
         <xsd:enumeration value="sha256WithRSAEncryption"></xsd:enumeration>
         <xsd:enumeration value="sha384WithRSAEncryption"></xsd:enumeration>
         <xsd:enumeration value="sha512WithRSAEncryption"></xsd:enumeration>
         <xsd:enumeration value="ecdsa-with-SHA256"></xsd:enumeration>
         <xsd:enumeration value="ecdsa-with-SHA384"></xsd:enumeration>
         <xsd:enumeration value="ecdsa-with-SHA512"></xsd:enumeration>
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="entityType">
      <xsd:sequence>
         <xsd:choice maxOccurs="unbounded">
            <xsd:element name="C" type="string"></xsd:element>
            <xsd:element name="countryName" type="string"></xsd:element>
            <xsd:element name="CN" type="string"></xsd:element>
            <xsd:element name="commonName" type="string"></xsd:element>
            <xsd:element name="L" type="string"></xsd:element>
            <xsd:element name="localityName" type="string"></xsd:element>
            <xsd:element name="ST" type="string"></xsd:element>
            <xsd:element name="stateOrProvinceName" type="string"></xsd:element>
            <xsd:element name="title" type="string"></xsd:element>
            <xsd:element name="street" type="string"></xsd:element>
            <xsd:element name="streetAddress" type="string"></xsd:element>
            <xsd:element name="SN" type="string"></xsd:element>
            <xsd:element name="surname" type="string"></xsd:element>
            <xsd:element name="GN" type="string"></xsd:element>
            <xsd:element name="givenName" type="string"></xsd:element>
            <xsd:element name="name" type="string"></xsd:element>
            <xsd:element name="O" type="string"></xsd:element>
            <xsd:element name="organizationName" type="string"></xsd:element>
            <xsd:element name="OU" type="string"></xsd:element>
            <xsd:element name="organizationalUnitName" type="string"></xsd:element>
            <xsd:element name="initials" type="string"></xsd:element>
            <xsd:element name="emailAddress" type="string"></xsd:element>
            <xsd:element name="serialNumber" type="string"></xsd:element>
         </xsd:choice>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

C.3 Grammatik für Anforderungen

```
Satz: 'Die benötigten Zusicherungen müssen folgende Eigenschaften erfüllen:'
```

```
      Beschraenkungen;

Beschraenkungen
      :  Regel '\n'| NoRestrictions;

NoRestrictions
      :  '\n'! 'Keine' ' Beschraenkungen\n'!;

Regel!: Pfad
      | Oder
      | Und
      | Inversion;

Pfad : '\n'! STRING;

Inversion : '\nnicht(' Regel '\n)';

Oder: '\noder(' Regel '\n;'  Regel '\n)';

Und: '\nund(' Regel '\n;' Regel '\n)';
```

C.4 Grammatik für Delegationszusicherungen

```
Satz: 'Hiermit wird an ' Berechtigter
      'das Recht\nzur Erstellung von Aussagen gemäß der Grammatik '
       GrammatikName ' erteilt.\n\n'
       GrammatikReferenz Weitergabe Beschraenkungsblock ;

Berechtigter:   CommonName ' (' EmailAddress ') ' ;

CommonName:   STRING ;

EmailAddress: STRING ;

GrammatikReferenz: 'Die Grammatik hat den ' Algorithmus '-Hashwert\n   '
       Hashwert '.\n\n' Uri?  GrammatikServer?;

Algorithmus: 'SHA256' |   'SHA384' |'SHA512' ;

Weitergabe:   'Diese Berechtigung kann über maximal '! INT
         ' nachfolgende Delegationsvorgänge\nweitergegeben werden.\n\n'!;

Beschraenkungsblock!:
      'Die auf Basis dieser Delegation gemachten Aussagen müssen folgende\n'
      'Bedingungen erfüllen:\n' Beschraenkungen ;

Beschraenkungen:  Regel '\n'| NoRestrictions;

NoRestrictions :  'Keine' ' Beschraenkungen\n'!;

Regel!: Pfad | Oder | Und | Inversion;
```

```
Pfad : '\n'! STRING;

Inversion : '\nnicht(' Regel '\n)';

Oder : '\noder(' Regel '\n;'  Regel '\n)';

Und : '\nund(' Regel '\n;' Regel '\n)';

Hashwert :   HEX;

Uri :   'Die Grammatik kann unter der Adresse\n   '! STRING
      '\nabgerufen werden.\n\n'!;

GrammatikServer:
         'Die Grammatik kann vom Grammatikserver unter der Adresse\n   '!
         STRING '\n abgerufen werden\n\n'! ;

GrammatikName :   STRING ;
```

D Erstellung von Grammatiken

D.1 Beispiel für Top-Down-Verfahren

Dieser Abschnitt beschreibt ausführlich die Ableitung einer Grammatik aus einer konkreten Zusicherung durch Anwendung der in Kapitel 5.1.4 beschriebenen Vorgehensweise. Dabei werden die in Anhang D.2 definierten Editor-Makros eingesetzt.

Der Ausgangstext ist eine simple Zusicherung, die in Kapitel 4.3 bereits als Beispiel verwendet wurde. Bei der Darstellung der Umformungen werden nur die jeweils relevanten Regeln dargestellt. Die Zeilenumbrüche und Einrückungen wurden zur kompakteren Darstellung gegenüber der Originaldatei verändert.

Zunächst wird das Beispiel im Editor so erfasst.

```
Auf dem Server cntsrv1 wurden alle nicht benötigten Dienste abgeschaltet.
Die Korrektheit der Dienstkonfiguration wurde mittels nmap-Befehl geprüft.
Die aktiven Dienste sind:
ssh auf Port 22
http auf Port 80
https auf Port 443
```

Durch Anwendung des **gca**-Makros wird aus diesem Text die initiale Grammatik erstellt:

```
Satz:
  'Auf dem Server cntsrv1 wurden alle nicht benötigten Dienste abgeschaltet.\n'
  'Die Korrektheit der Dienstkonfiguration wurde mittels nmap-Befehl geprüft.\n'
  'Die aktiven Dienste sind:\n'
  'ssh auf Port 22\n'
  'http auf Port 80\n';
  'https auf Port 443\n';
```

Durch Markieren von cntsrv1, Makro **gcsr** für Nicht-Terminal *Server* und mehrfache Anwendung von **gcra** in der *Server*-Regel wird eine Auswahl von Terminalen für die Nennung des untersuchten Servers erstellt:

```
Satz:
  'Auf dem Server ' Server
  ' wurden alle nicht benötigten Dienste abgeschaltet.\n'
  'Die Korrektheit der Dienstkonfiguration wurde mittels nmap-Befehl geprüft.\n'
  'Die aktiven Dienste sind:\n'
  'ssh auf Port 22\n'
  'http auf Port 80\n';
  'https auf Port 443\n';

Server: 'cntsrv1' | 'cntsrv2' | 'cntsrv3' | 'mailsrv';
```

Markieren des Terminals 'Die Korrektheit ... geprüft\n' und Anwendung von **gctr** mit dem optionalen Nicht-Terminal *Pruefung* formt die *Satz*-Regel um:

```
Satz:
  'Auf dem Server ' Server
  ' wurden alle nicht benötigten Dienste abgeschaltet.\n'
  Pruefung?
  'Die aktiven Dienste sind:\n'
  'ssh auf Port 22\n'
  'http auf Port 80\n';
  'https auf Port 443\n';

Pruefung:
  'Die Korrektheit der Dienstkonfiguration wurde mittels nmap-Befehl geprüft.\n'
;
```

Die Teilzeichenkette 'nmap-Befehl' wird mit **gcsr** in eine *Methode*-Regel ausgelagert und durch **gcra** um weitere Alternativen ergänzt:

```
Pruefung:
  'Die Korrektheit der Dienstkonfiguration wurde mittels ' Methode
  ' geprüft.\n';

Methode:   'nmap-Befehl' | 'Portscan' | 'netstat-Befehl';
```

Die Aufzählung der Dienste wird durch eine Markierung von 'ssh auf Port ...' bis '... Port 443\n' mit dem Makro **gctr** in eine *Dienste*-Regel ausgelagert. Das in der Regel *Satz* aufgenommene Nicht-Terminal *Dienste* wird als optional markiert:

```
Satz:
  'Auf dem Server ' Server
  ' wurden alle nicht benötigten Dienste abgeschaltet.\n'
  Pruefung?
  'Die aktiven Dienste sind:\n'
  Dienste?

Dienste:   'ssh auf Port 22\n' 'http auf Port 80\n' 'https auf Port 443\n';
```

Aus der Regel *Dienste* wird die Terminale 'http auf Port 80\n' und 'https auf Port 443\n'gelöscht und das verbleibende Terminal mit **gcst** in eine *Dienst*-Regel gewandelt. Das Nicht-Terminal *Dienst* wird +-quantisiert.

```
Dienste:   Dienst+ ;

Dienst:   'ssh auf Port 22\n';
```

Die Teilzeichenketten 'ssh' und '22' werden mit **gcsr** in eine *Dienstname*- und eine *Port*-Regel ausgelagert. Bei Dienstname werden durch **gcra** weitere Namensalternativen aufgenommen.

```
Dienst:    Dienstname ' auf Port ' Portnummmer '\n';

Dienstname:   'ssh' | 'dns' | 'smtp' | 'http' | 'https' | 'ntp';

Portnummer:   '22';
```

Durch Markieren des Terminals '22' und Anwendung von **gcex** mit dem Lexem *INT* wird die *Portnummer*-Regel zu:

```
Portnummer:   INT;
```

Abschließend kann noch eine zusätzliche Knotenebene mit dem Kürzel der Grundschutzmaßnahme durch Markieren aller Elemente der *Satz*-Regel und Makro **gctr** mit Nicht-Terminal *Mn5_72* erreicht werden.

```
Satz: Mn5_72;

Mn5_72:  'Auf dem Server ' Server
         ' wurden alle nicht benötigten Dienste abgeschaltet.\n'
         Pruefung
         'Die aktiven Dienste sind:\n'
         Dienste?;
```

D.2 Definitionen der vim-Makros für Umformungen

```
" Makros zur Unterstützung der Erstellung von gc-Grammatiken
" (c) M. Kabatnik, 2012
"
" gca - gc Anfang
" Alle Zeilen in Hochkomma mit explizitem \n und Grundstruktur
nmap gca :se tw=0<CR>:%s/^/<Tab><Tab><Tab><Tab>'/<CR>:%s/$/\\n' /<CR>1GOSatz
   <CR><Tab><Tab><Tab>:<Esc>$Go<Tab><Tab><Tab>;<Esc>

" gcmt - gc markiere Terminal
" dabei Cursor innerhalb eines mit Hochkomma definierten Terminal-Strings
```

```
nmap gcmt ?'<CR>vf'

" gctr - gc Terminal zu Regel umwandeln
" Markiertes Terminal oder Elementfolge in Regel verschieben
vmap gctr di<gcName><gcQuant><Esc>$Go<CR><gcName><gcInv><CR><Tab><Tab><Tab
   >:<Tab><Esc>p$Go<Tab><Tab><Tab>;<Esc>:%s/<gcName>/gcxID/g<CR>:%s/<gcQuant
   >/gcxQuant/g<CR>:%s/<gcInv>/gcxInv/g<CR>

" Zusatzfunktionen für gctr und gcsr
cmap gcxID <C-R>=inputdialog("Bezeichner")<CR>
cmap gcxQuant <C-R>=inputdialog("Quantifizierer (leer, ?, +, *)")<CR>
cmap gcxInv <C-R>=inputdialog("Knotenunterdrückung (leer, !)")<CR>

" gcsr - gc Substring aus Terminal zu Regel umwandeln
" Markierten Terminalteil in Regel verschieben
vmap gcsr mndi' <gcName><gcQuant> '<Esc>$Go<CR><gcName><gcInv><CR><Tab><Tab
   ><Tab>:<Tab>'<Esc>pa'<Esc>$Go<Tab><Tab><Tab>;<Esc>:%s/<gcName>/gcxID/g<CR
   >:%s/<gcQuant>/gcxQuant/g<CR>:%s/<gcInv>/gcxInv/g<CR>'n:%s/''//g<CR>

" gcra - gc Regel Alternative
" Cursor innerhalb Regel (nicht auf Semikolon), Alternative hinzufügen
nmap gcra /;<CR>i<Bar><Tab>' '<CR><Tab><Tab><Tab><Up><End><Left><BS>

" gcsa - Substring als AST-relevant markieren
" Terminalteil für AST auswählen (durch Markierung)
vmap gcsa di'! '<Esc>pa' '<Esc>/'<CR>a!<ESC>:s/ \?''! \?//g<CR>

" gcex - Ersetzen einer Elementfolge durch Nicht-Terminal oder Lexem
" Die markierten Elemente (Terminale, Nicht-Terminale und Lexeme) werden
   ersetzt
vmap gcex c<gcNonTermOrLexem><Esc>:s/<gcNonTermOrLexem>/gcxNTOL/g<CR>

" Zusatzfunktion für gcex
cmap gcxNTOL <C-R>=inputdialog("Nicht-Terminal oder Lexem")<CR>

" gcda - gc delete alternative
" Löschen der Regelalternative unter dem Cursor
nmap gcda ?[:<Bar>]<CR><Right>v/[<Bar>;]<CR><Left>d:s/\(:\)\?<Bar>\(<Bar>\<
   Bar>;\)\?/\1\2/g<CR>

" gczf - Einfalten der Regel unter dem Cursor
" Es wird ein eingefalteter vim-Bereich (zf) angelegt
nmap gczf /;<CR>?:<CR>?\a<CR>bv/;<CR>zf<Down>/\a<CR>^
```

E Beispiel eines XML-kodierten grammatikbasierten Zertifikats

```
<?xml version="1.0" encoding="UTF-8"?><gc:GrammarbasedCertificate xmlns:gc="http:///GCertSchema
    /GrammarbasedCertificate/GrammarbasedCertificate.xsd" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" ID="tbsGCertificate" xsi:schemaLocation="http:///GCertSchema/
    GrammarbasedCertificate/GrammarbasedCertificate.xsd GrammarbasedCertificate.xsd ">
        <Version>1.0</Version>
        <SerialNumber>05913426df2835c34919ff003666a121</SerialNumber>
        <Issuer>
                <emailAddress>matthias@kabatnik.net</emailAddress>
                <CN>Matthias Kabatnik</CN>
                <L>Leinfelden</L>
                <C>DE</C>
        </Issuer>
        <Validity>
                <NotBefore>2013-04-24T23:35:27Z</NotBefore>
                <NotAfter>2013-07-24T23:35:27Z</NotAfter>
        </Validity>
        <GrammarReference>
                <DigestMethod>sha256</DigestMethod>
                <DigestValue>c84on+SIZOWu39JR4jzSxi6/TdwPqTEwtDuWllsrdOk=</DigestValue>
                <URI>http://security.example.com/grammars/m173.gcg</URI>
                <GCGrammarServer>http://gcgsrv.example.com/grammarserver</GCGrammarServer>
        </GrammarReference>
        <Assertion>
                <ParseTree>
<Satz>
   <Ort>
      <GCELEMENT type="STRING">Haus 2, Raum RZ-01</GCELEMENT>
   </Ort>
   <Autorisierung>
      <AutKriterium>
         <GCELEMENT type="TOKEN">Besitz</GCELEMENT>
      </AutKriterium>
      <AutKriterium>
         <GCELEMENT type="TOKEN">Wissen</GCELEMENT>
      </AutKriterium>
   </Autorisierung>
   <Besucher>
      <Zuordnung>
         <GCELEMENT type="TOKEN">zugeordnet</GCELEMENT>
      </Zuordnung>
      <Aufsicht>
         <GCELEMENT type="TOKEN">sichergestellt</GCELEMENT>
      </Aufsicht>
   </Besucher>
   <Protokollierung>
      <Zutritt>
         <PZ_Berechtigt>
```

```
            <GCELEMENT type="TOKEN">elektronisch</GCELEMENT>
         </PZ_Berechtigt>
         <PZ_Besucher>
            <GCELEMENT type="TOKEN">schriftlich</GCELEMENT>
            <Abzeichnung>
               <GCELEMENT type="TOKEN">abgezeichnet</GCELEMENT>
            </Abzeichnung>
         </PZ_Besucher>
      </Zutritt>
      <Austritt>
         <PA_Berechtigt>
            <GCELEMENT type="TOKEN">elektronisch</GCELEMENT>
         </PA_Berechtigt>
         <PA_Besucher>
            <GCELEMENT type="TOKEN">schriftlich</GCELEMENT>
         </PA_Besucher>
      </Austritt>
   </Protokollierung>
   <Vereinzelung>
      <GCELEMENT type="TOKEN">eine</GCELEMENT>
   </Vereinzelung>
   <Passback>
      <GCELEMENT type="TOKEN">eine</GCELEMENT>
   </Passback>
</Satz>
                </ParseTree>
        </Assertion>
        <Nonce>AoqSG5dNfIcBZ3GVx+P90Q==</Nonce>
        <Extensions>
                <AuthorityKeyIdentifier>9CqNv5x5JBir+JvRmVTWSf9GlrE=</AuthorityKeyIdentifier>
        </Extensions>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#tbsGCertificate">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>M1qjouVeZYmx42mu/deY67ppNTw=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
Ar1JczWkjY+NEqltO5J9ZfI39xXFFsLk5PjWRXld0d05ooZ5zFFtejEfs3B5YR26EnBBtoWnyCVu
bd4umig5OvstVoQtIuXisLsmuyXZBHoJEVIlgREK0PG9gAldTOalPFFYVZjn+dSck5ByGgjM8bOM
kOOx9FfPp/+eseYG1epgSO1p0BH+Rv6rGYAQqX6Mi87zVHFDWdgjiG7t/Bco9egO6x0abM456f+4
hBtKAFh2X+p0/AQJkAUKUHMXdOJaBJaMLVQQTrcV0iZm1geCxnnuex02/HDp2g/gUj0rHIX2FnYP
wMdcb6ydetzFr9BWoffjdeyQ+uqb62whtgqdDg==
</ds:SignatureValue>
</ds:Signature></gc:GrammarbasedCertificate>
```

F Modellierung in Proverif

Die folgende Modellierung in Proverif stellt die Funktionalität des Grammatikverteildienstes mit Möglichkeit zur Delegation dar.

```
1  (* Analyse des Grammatikservers *)
2
3  (* Angaben zur Steuerung der Ausgabe der Analyseergebnisse *)
4  set traceDisplay = long.
5  set verboseRules = true.
6
7  (* Datentyp für Nachrichtentypen des Grammatikservers *)
8  type gsMessage.
9
10 const addG: gsMessage. const addHG: gsMessage. const respG: gsMessage.
11 const reqG: gsMessage. const delG: gsMessage. const reqT: gsMessage.
12 const respT:gsMessage. const initReqT:gsMessage. const initRespT:gsMessage.
13
14 (* Hashwert-Funktion ohne Reduktion, da nicht umkehrbar *)
15 fun hash(bitstring): bitstring.
16
17 (************************************************
18  * Definitionen für asymmetrische Kryptographie *
19  ************************************************)
20
21 type pkey.  (* public encryption key *)
22 type skey.  (* secret encryption key *)
23 type pskey. (* public signing key *)
24 type sskey. (* secret signing key *)
25
26 (* Einweg-Hashfunktion für KeyID-Bestimmung *)
27 fun kidhash(pskey):bitstring.
28
29 (* Public encryption key wird aus privatem encryption key abgeleitet *)
30 fun pk(skey):pkey.
31 (* Public signing key wird aus privatem signing key abgeleitet *)
32 fun psk(sskey):pskey.
33 (* Signatur mit Wert und Signaturschlüssel *)
34 fun sign(bitstring, sskey):bitstring.
35 (* Prüfe Signatur und gib signierte Daten zurück *)
36 reduc forall m: bitstring, k: sskey; checksign(sign(m, k), psk(k)) = m.
37 (* Prüfe signierten Hashwert gegen die Vergleichsdaten und gib referenzierte Daten zurück *)
```

```
38 reduc forall m: bitstring, k: sskey; checksignedhash(m, sign(hash(m), k), psk(k)) = m.
39
40 (* Zugriff auf signierte Inhalte ohne Prüfung der Signatur *)
41 reduc forall m: bitstring, k: sskey;  getsigneddata(sign(m, k)) = m.
42
43 (*********************************************
44  * Definitionen für symmetrische Kryptographie *
45  *********************************************)
46 type symkey. (* Symmetrischer Schlüssel *)
47
48 fun senc(bitstring, symkey):bitstring.
49 reduc forall x:bitstring, y:symkey; sdec(senc(x, y), y) = x.
50
51 (* einziger öffentlicher Kommunikationskanal ist das Netz *)
52 free net : channel.
53
54 (* verifizierte Annahmen zur Vertraulichkeit *)
55 not attacker (new sskC). (* privater Signaturschlüssel von C bleibt vertraulich *)
56 not attacker (new sskG). (* privater Signaturschlüssel von G bleibt vertraulich *)
57 not attacker (new G).   (* Grammatik bleibt vertraulich *)
58
59 (* Definition von Ereignissen *)
60 event authKeyidDeleg(bitstring).
61 event authKeyidReq(bitstring).
62 event reqToken(bitstring, bitstring).
63 event reqDelegToken(bitstring, bitstring).
64 event grammarInsert(bitstring, bitstring).
65 event grammarDelete(bitstring, bitstring).
66 event gotGrammar(bitstring).
67 event authKeyid(bitstring).
68 event genToken(bitstring, bitstring, bitstring).
69 event genTokenDeleg(bitstring, bitstring).
70 event deleteRequest(bitstring, bitstring).
71 event insertRequest(bitstring, bitstring).
72
73 (* Prüfung der Erreichbarkeit einzelner Ereignisse, erwartetes Ergebnis
74  ist: not event(xyz) .... is false. *)
75 query G:bitstring, K:bitstring; event(grammarInsert(G, K)).
76 query G:bitstring; event(gotGrammar(G)).
77 query K:bitstring; event(authKeyid(K)).
78 query R:bitstring, K:bitstring; event(reqToken(R, K)).
79 query R:bitstring, K:bitstring; event(reqDelegToken(R, K)).
80 query H:bitstring, K:bitstring; event(grammarDelete(H, K)).
81 query R:bitstring, K:bitstring, T:bitstring; event(genToken(R, K, T)).
82 query H:bitstring, K:bitstring; event(deleteRequest(H, K)).
83 query H:bitstring, K:bitstring; event(insertRequest(H, K)).
84
85 (* Prüfung der Protokollzusammenhänge*)
86 (* Neueinstellung nur durch legitimierte Anwender GS-P1 *)
87 query G:bitstring, K1:bitstring, K2:bitstring, R:bitstring;
88       inj-event(grammarInsert(G, K1)) ==> (event(authKeyid(K1)) ||
```

```
89      (inj-event(genToken(R, K1, K2)) && event(authKeyidDeleg(K2)) ).
90
91  (* Löschung nur, wenn vorher durch gleichen Akteur eingestellt GS-P2 *)
92  query G:bitstring, K:bitstring;
93      event(grammarDelete(G, K)) ==> event(grammarInsert(G, K)).
94
95  (* Löschung nur, wenn direkte Autorisierung des Antragstellers vorlag
96   oder für den Antragsteller einer Delegationstoken beantragt wurde GS-P3 *)
97  query G:bitstring, K1:bitstring, K2:bitstring, R:bitstring;
98      inj-event(grammarDelete(G, K1)) ==> (event(authKeyid(K1)) ||
99      (inj-event(genToken(R, K1, K2)) && event(authKeyidDeleg(K2)) )).
100
101 (* Neueinstellung nur nach individueller Beantragung GS-P4 *)
102 query H:bitstring, K:bitstring; inj-event(grammarInsert(H, K)) ==>
103     inj-event(insertRequest(H, K)).
104
105 (* Löschen nur nach individueller Beantragung GS-P5 *)
106 query H:bitstring, K:bitstring; inj-event(grammarDelete(H, K)) ==>
107     inj-event(deleteRequest(H, K)).
108
109 (* für jede Token-Erstellung liegt genau eine Anfrage (entweder
110  direkt oder mit Delegation vor GS-P6 *)
111 query R:bitstring, K1:bitstring, K2:bitstring;
112     inj-event(genToken(R, K1, K2)) ==> inj-event(reqToken(R, K1)) ||
113                                        inj-event(reqDelegToken(R, K1)).
114
115 (* Tabellen zur vertrauenswürdigen Speicherung von Daten *)
116 (* 'authorized' enthält autorisierte Identitäten und deren zusätzlichen Rechte *)
117         (* keyid, pubkey, tokreq, delegate *)
118 table authorized(bitstring, pskey, bool, bool).
119
120 (* Tabelle zur Speicherung von Tokens *)
121       (* keyid, token *)
122 table tokens(bitstring, bitstring).
123
124 (* Grammatikdatenbank *)
125         (* hash, Grammatik, Eigentümer *)
126 table grammars(bitstring, bitstring, bitstring).
127
128 (* Grammatikserver zur Verarbeitung geschützter Grammatiken *)
129 let processG(sskG: sskey) =
130   (* empfange Startnachricht *)
131   in(net, (= initReqT, keyid:bitstring));
132   (* schicke Anfangswert *)
133   new r0:bitstring;
134   out(net, (initRespT, keyid, r0));
135   (* empfange eine Nachricht *)
136   in(net, msg:bitstring);
137   (* P0: zerlege Nachricht und prüfe auf r0 und keyid *)
138   let (= reqT, = r0, r1:bitstring, delegKey:pskey, = keyid) = getsigneddata(msg) in
139   (* P1: prüfe die Autorisierung und hole zugehörigen öffentlichen Schlüssel *)
140   get authorized(= keyid, pubkey:pskey, = true, delegRight:bool) in
```

```
141     (* P2: prüfe Signatur der Daten *)
142     let (= reqT, = r0, = r1, = delegKey, = keyid) = checksign(msg, pubkey) in
143     (* erzeuge neuen Token *)
144     new token:bitstring;
145     if (kidhash(delegKey) = keyid) || (delegRight = true) then (* Delegationsrecht *)
146     (
147       event genToken(r1, kidhash(delegKey), keyid);
148       (* sende Token zurück *)
149       out(net, sign((respT, r1, token), sskG));
150
151       (* empfange eine Nachricht *)
152       in(net, msg2: bitstring);
153       (* P5: zerlege Nachricht und prüfe auf token und keyid *)
154       let (= addHG, grammarHash:bitstring,
155               G:bitstring,
156          = token,
157                  = kidhash(delegKey)) = getsigneddata(msg2) in
158       (
159         (* P6: prüfe Signatur *)
160         let (= addHG, = grammarHash, = G, = token, = kidhash(delegKey)) =
161           checksign(msg2, delegKey) in
162         (* stelle Grammatik ein *)
163         event grammarInsert(grammarHash, kidhash(delegKey));
164         insert grammars(grammarHash, G, kidhash(delegKey))
165           (* P5': zerlege Nachricht und prüfe auf token und keyid *)
166       ) else let (= delG, grammarHash:bitstring,
167         = token,
168                       = kidhash(delegKey)) = getsigneddata(msg2) in
169       (
170           (* P6': prüfe Signatur *)
171           let (= delG, = grammarHash, = token, = kidhash(delegKey)) =
172             checksign(msg2, delegKey) in
173           (* P8: prüfe Besitzer der Grammatik *)
174           get grammars(= grammarHash, G:bitstring, = kidhash(delegKey)) in
175           event grammarDelete(grammarHash, kidhash(delegKey))
176         )
177     )
178 .
179
180 let processGreqG =
181     (* empfange Request für Grammatik *)
182     in(net, (= reqG, grammarHash:bitstring));
183     (* hole Grammatik aus Datenbank *)
184     get grammars(= grammarHash, G:bitstring, owner:bitstring) in
185     (* generiere Antwort *)
186     out(net, (respG, G))
187 .
188
189 let processCreqT(sskC: sskey, pskG: pskey) =
190     (* beginne Token-Anforderung *)
191     out(net, (initReqT, kidhash(psk(sskC))));
```

```
192     (* empfange Init-Antwort des Servers *)
193     in(net, (= initRespT, = kidhash(psk(sskC)), r0:bitstring));
194     (* neuer Zufallswert für Request *)
195     new r1:bitstring;
196     event reqToken(r1, kidhash(psk(sskC)));
197     (* sende signierten Request an Server *)
198     out(net, sign((reqT, r0, r1, psk(sskC), kidhash(psk(sskC))), sskC));
199     (* empfange Antwortnachricht *)
200     in(net, msg:bitstring);
201     (* P3: prüfe auf r1 *)
202     (* P4: prüfe Signatur gegen Schlüssel von G *)
203     let(= respT, = r1, token:bitstring) = checksign(msg, pskG) in
204     (* C sendet Token an sich selbst *)
205     out(net, (kidhash(psk(sskC)), token))
206 .
207
208 let processDreqT(sskD: sskey, pskG: pskey, delegKey:pskey) =
209     (* Angreifer kann den Delegierten bestimmen *)
210 (* in(net,delegKey:pskey); *)
211     (* beginne Token-Anforderung *)
212     out(net, (initReqT, kidhash(psk(sskD))));
213     (* empfange Init-Antwort des Servers *)
214     in(net, (= initRespT, = kidhash(psk(sskD)), r0:bitstring));
215     (* neuer Zufallswert für Request *)
216     new r1:bitstring;
217     event reqDelegToken(r1, kidhash(delegKey));
218     (* sende signierten Request an Server *)
219     out(net, sign((reqT, r0, r1, delegKey, kidhash(psk(sskD))), sskD));
220     (* empfange Antwortnachricht *)
221     in(net, msg:bitstring);
222     (* P3: prüfe auf r1 *)
223     (* P4: prüfe Signatur gegen Schlüssel von G *)
224     let(= respT, = r1, token:bitstring) = checksign(msg, pskG) in
225     (* Sende Token an Delegierten (ggf. auch an C selbst) *)
226     out(net, (kidhash(delegKey), token))
227 .
228
229
230 let processCaddHG(sskC:sskey, pskG:pskey, G:bitstring, GK:symkey) =
231     (* Lese Token für C-Prozess ein *)
232     in(net, (= kidhash(psk(sskC)), token:bitstring));
233     event insertRequest(hash(G), kidhash(psk(sskC)));
234     (* schicke signierte Nachricht mit verschlüsselter Grammatik *)
235     out(net, sign((addHG, hash(G), senc(G, GK), token, kidhash(psk(sskC))), sskC))
236 .
237
238 let processCreqG(grammarHash:bitstring, GK:symkey) =
239     (* Sende Grammatik-Anfrage *)
240     out(net, (reqG, grammarHash));
241     (* empfange Antwort mit verschlüsselter Grammatik *)
242     in(net, (= respG, = grammarHash, cipher:bitstring));
```

```
243     (* entschlüssele Grammatik *)
244     let G:bitstring = sdec(cipher, GK) in
245     event gotGrammar(G)
246   .
247
248   let processCdelG(sskC:sskey, grammarHash:bitstring) =
249     (* Lese Token für C-Prozess ein *)
250     in(net, (= kidhash(psk(sskC)), token:bitstring));
251     event deleteRequest(grammarHash, kidhash(psk(sskC)));
252     (* schicke signierten Löschauftrag *)
253     out(net, sign((delG, grammarHash, token, kidhash(psk(sskC))), sskC))
254   .
255
256   (* Definition des Gesamtsystems *)
257   process
258     (* erzeuge Signaturschlüssel für Client-Prozess C *)
259     new sskC:sskey;
260     let pskC:pskey = psk(sskC) in
261     out(net, pskC);
262
263     (* erzeuge Signaturschlüssel für Client-Prozess D *)
264     new sskD:sskey;
265     let pskD:pskey = psk(sskD) in
266     out(net, pskD);
267
268     (* Schlüssel für den Grammatikdienst *)
269     new sskG:sskey;
270     let pskG:pskey = psk(sskG) in
271     out(net, pskG);
272
273     (* erzeuge symmetrischen Schlüssel für Grammatikverschlüsselung *)
274     new GK:symkey;
275      (* neue Grammatik *)
276      new G:bitstring;
277
278     (* erzeuge Schlüsselpaar für den Angreifer und stelle es zur
279    Verfügung *)
280     new sskA:sskey;
281     let pskA:pskey = psk(sskA) in
282     out(net, sskA);
283     out(net, pskA);
284
285      (* Autorisierungen für C, D und Angreifer *)
286      insert authorized(kidhash(psk(sskC)), psk(sskC), true, false);
287      event authKeyid(kidhash(psk(sskC)));
288      (* event authKeyidDeleg(kidhash(psk(sskC))); *)
289      (* Variante: Instanz D erhält Delegationsrecht ->
290    Parameter auf true setzen , events einkommentieren*)
291      insert authorized(kidhash(psk(sskD)), psk(sskD), false, false);
292      (* event authKeyid(kidhash(psk(sskD))); *)
293      (* event authKeyidDeleg(kidhash(psk(sskD))); *)
294     (* Variante: Angreifer wird ebenfalls mit Rechten ausgestattet ->
```

```
295  Parameter auf true setzen, events einkommentieren *)
296    insert authorized(kidhash(psk(sskA)), psk(sskA), false, false);
297    (* event authKeyid(kidhash(psk(sskA))); *)
298    (* event authKeyidDeleg(kidhash(psk(sskA))); *)
299
300    (* Alle Teilprozesse unter Replikation instantiieren *)
301    !processG(sskG) |
302    !processCreqT(sskC, pskG) |!processDreqT(sskD, pskG, pskC) |
303    !processCaddHG(sskC, pskG, G, GK) |
304    !processCreqG(hash(G), GK) |!processGreqG |!processCdelG(sskC, hash(G))
305
```

Literaturverzeichnis

[AB05] ABADI, MARTÍN and BRUNO BLANCHET: *Analyzing security protocols with secrecy types and logic programs*. Journal of the ACM, 52:102–146, January 2005.

[AF01] ABADI, MARTÍN and CÉDRIC FOURNET: *Mobile values, new names, and secure communication*. Proceedings of the 28th Annual ACM Symposium on Principles of Programming Languages (POPL01), pages 104–115, 2001.

[AG99] ABADI, MARTÍN and ANDREW D. GORDON: *A calculus for cryptographic protocols: The spi calculus*. Information and Computation, 148:36–47, 1999.

[AT91] ABADI, MARTÍN and MARK R. TUTTLE: *A semantics for a logic of authentication*. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, August 1991.

[BAF08] BLANCHET, BRUNO, MARTÍN ABADI, and CÉDRIC FOURNET: *Automated verification of selected equivalences for security protocols*. Journal of Logic and Algebraic Programming, 75(1):3–51, Feb–Mar 2008.

[Bal93] BALENSON, D.: *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers*. RFC 1423 (Historic), February 1993.

[BAN89] BURROWS, MICHAEL, MARTÍN ABADI, and ROGER NEEDHAM: *A Logic of Authentication*. Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences, 426(1871):233–271, December 1989.

[BBC+07] BERGLUND, ANDERS, SCOTT BOAG, DON CHAMBERLIN, MARY F. FERNÁNDEZ, MICHAEL KAY, JONATHAN ROBIE, and JÉRÔME SIMÉON: *W3c recommendation: Xml path language (xpath) 2.0*. http://www.w3.org/TR/2007/REC-xpath20-20070123/, January 2007.

[BBF+08] BARTEL, MARK, JOHN BOYER, BARB FOX, BRIAN LAMACCHIA, and ED SIMON: *XML-Signature Syntax and Processing*. World Wide Web Consortium (W3C), http://www.w3.org/TR/xmldsig-core/, 10. June 2008.

[BDPA09] BERTONI, GUIDO, JOAN DAEMEN, MICHAËL PEETERS, and GILLES VAN ASSCHE: *The road from panama to keccak via radiogatún*. In HANDSCHUH, HELENA, STEFAN LUCKS, BART PRENEEL, and PHILLIP ROGAWAY (editors): *Symmetric Cryptography*, number 09031 in *Dagstuhl Seminar Proceedings*, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[BFG07] BECKER, MORITZ Y., CEDRIC FOURNET, and ANDREW D. GORDON: *Design and semantics of a decentralized authorization language*. In *20th IEEE Computer Security Foundations Symposium (CSF)*, pages 3–15, 2007.

[BFG10] BECKER, MORITZ Y., CÉDRIC FOURNET, and ANDREW D. GORDON: *Secpal: Design and semantics of a decentralized authorization language*. Journal of Computer Security, 18(4):619–665, 2010.

[BFIK99] BLAZE, M., J. FEIGENBAUM, J. IOANNIDIS, and A. KEROMYTIS: *The KeyNote Trust-Management System Version 2*. RFC 2704 (Informational), September 1999.

[BFL96] BLAZE, MATT, JOAN FEIGENBAUM, and JACK LACY: *Decentralized Trust Management*. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, May 1996. IEEE.

[Bla01] BLANCHET, BRUNO: *An efficient cryptographic protocol verifier based on prolog rules*. In *Proceedings of the 14th IEEE workshop on Computer Security Foundations*, CSFW '01, pages 82–96, Washington, DC, USA, 2001. IEEE Computer Society.

[Bla09] BLANCHET, BRUNO: *Automatic verification of correspondences for security protocols*. Journal of Computer Security, 17:363–434, December 2009.

[Bla11] BLANCHET, BRUNO: *Proverif: Cryptographic protocol verifier in the formal model*. http://www.proverif.ens.fr/, 2011.

[Bun01a] BUNDESGESETZ: *Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften*. Bundesgesetzblatt Jahrgang 2001 Teil I Nr. 22, 21. Mai 2001.

[Bun01b] BUNDESVERORDNUNG: *Verordnung zur elektronischen Signatur*. Bundesgesetzblatt I S. 3074, 16. November 2001.

[Bun02] BUNDESGESETZ: *Bürgerliches Gesetzbuch in der Fassung der Bekanntmachung vom 2. Januar 2002 (BGBl. I S. 42, 2909; 2003 I S. 738)*. Bundesgesetzblatt, 2002.

[Bun08a] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *BSI Grundschutz-Tool*. https://www.bsi.bund.de/DE/Themen/weitereThemen/GSTOOL/gstool_node.html, 02 2008.

[Bun08b] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *BSI-Standard 100-2: IT-Grundschutz-Vorgehensweise, Version 2.0*. Bundesamt für Sicherheit in der Informationstechnik, Bonn, 2008.

[Bun08c] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Kryptographische Verfahren: Empfehlungen und Schlüssellängen (BSI-TR-02102)*. http://www.bsi.bund.de/literat/tr/tr02102/BSI-TR-02102.pdf, Juni 2008. Version 1.0.

[Bun11] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *IT-Grundschutzkataloge: 12. Ergänzungslieferung, Stand 2011*. https://www.bsi.bund.de, September 2011.

[CDF+07] CALLAS, J., L. DONNERHACKE, H. FINNEY, D. SHAW und R. THAYER: *OpenPGP Message Format*. RFC 4880 (Proposed Standard), November 2007. Updated by RFC 5581.

[CDH+05] COOPER, M., Y. DZAMBASOW, P. HESSE, S. JOSEPH und R. NICHOLAS: *Internet X.509 Public Key Infrastructure: Certification Path Building*. RFC 4158 (Informational), September 2005.

[CFS+03] CHOKHANI, S., W. FORD, R. SABETT, C. MERRILL und S. WU: *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*. RFC 3647 (Informational), November 2003.

[Cho59] CHOMSKY, NOAM: *iOn Certain Formal Properties of Grammars*. Information and Control, 2(2):137–167, 1959.

[CKPE05] CANTOR, SCOTT, JOHN KEMP, ROB PHILPOTT, and EVE MALER (EDS.): *Assertions and protocols for the oasis security assertion markup language (saml) v2.0*. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, 15. March 2005.

[CKPR03] CRUELLAS, JUAN CARLOS, GREGOR KARLINGER, DENIS PINKAS, and JOHN ROSS: *XML Advanced Electronic Signatures (XAdES)*. World Wide Web Consortium (W3C), http://www.w3.org/TR/XAdES/, 20. February 2003.

[CO04] CHADWICK, DAVID W. and ALEXANDER OTENKO: *Implementing Role Based Access Controls using X.509 Privilege Management - the PERMIS Authorisation Infrastructure*. In JERMAN-BLAZIC, BORKA, WOLFGANG SCHNEIDER, and TOMAZ KLOBUCAR (editors): *Security and Privacy in Advanced Networking Technologies*, NATO Science Series, pages 26–39. IOS Press, unknown 2004. Proceedings of the NATO Advanced Networking Workshop on Advanced Security Technologies in Networking, Bled, Slovenia, 15-18 September 2003.

[COB03] CHADWICK, DAVID W., ALEXANDER OTENKO, and EDWARD BALL: *Role-based access control with x.509 attribute certificates*. IEEE Internet Computing, 07(2):62–69, March/April 2003.

[Com12] COMMON CRITERIA RECOGNITION ARRANGEMENT: *Common Criteria for Information Technology Security Evaluation – Version 3.1 Revision 4*. http://www.commoncriteriaportal.org/cc/, September 2012.

[CSF+08] COOPER, D., S. SANTESSON, S. FARRELL, S. BOEYEN, R. HOUSLEY, and W. POLK: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280 (Proposed Standard), May 2008.

[CZO+08] CHADWICK, DAVID W, GANSEN ZHAO, SASSA OTENKO, ROMAIN LABORDE, LINYING SU, and TUAN ANH NGUYEN: *PERMIS: a modular authorization infrastructure*. Concurrency and Computation: Practice and Experience, 20(11):1341–1357, August 2008. Online ISSN: 1532-0634.

[DAdB+11] DERAISON, RENAUD, MICHAEL ARBOI, ALEXIS DE BERNIS, AXEL NENNKER, BEIRNE KORNARKSI, and BENOIT BRODARD: *Open Vulnerability Assessment System (OpenVAS)*. http://www.openvas.org/, 2011.

[Day73] DAY, J.D.: *Memo to FTP group: Proposal for File Access Protocol*. RFC 520, June 1973.

[DDLS00] DAMIANOU, NICODEMOS, NARANKER DULAY, EMIL LUPU, and MORRIS SLOMAN: *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems*. Imperial College Research Report DoC 2000/1, October 2000. http://www-dse.doc.ic.ac.uk/Research/policies/ponder/PonderSpec.pdf.

[DHR+98] DUSSE, S., P. HOFFMAN, B. RAMSDELL, L. LUNDBLADE, and L. REPKA: *S/MIME Version 2 Message Specification*. RFC 2311 (Historic), March 1998.

[DHRW98] DUSSE, S., P. HOFFMAN, B. RAMSDELL, and J. WEINSTEIN: *S/MIME Version 2 Certificate Handling*. RFC 2312 (Historic), March 1998.

[Dow03] DOWNNARD, IAN: *Public-key cryptography extensions into kerberos*. IEEE Potentials, 21(5):30–34, Januar 2003.

[DY83] DOLEV, D. and A. C. YAO: *On the security of public key protocols*. IEEE Transactions on Information Theory, IT29(12):198–208, March 1983.

[Ecl] ECLIPSE FOUNDATION: *Xtext project*. http://www.eclipse.org/Xtext/.

[EFL+99] ELLISON, C., B. FRANTZ, B. LAMPSON, R. RIVEST, B. THOMAS, and T. YLONEN: *SPKI Certificate Theory*. RFC 2693 (Experimental), September 1999.

[ets12] *Electronic Signatures and Infrastructures (ESI) – CMS Advanced Electronic Signatures (CAdES)*. Technical Report ETSI TS 101 733 V2.1.1, European Telecommunications Standards Institute, March 2012. 2012.

[EU-00] EU-DIREKTIVE: *Directive 1999/93/ec of the european parliament and of the council of 13 december 1999 on a community framework for electronic signatures*. Official Journal of the European Communities L13/12, 19. January 2000.

[Eur07] EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE (ETSI): *Electronic Signatures and Infrastructures (ESI) – Policy requirements for certification authorities issuing qualified certificates*. ETSI TS 101 456 V1.4.3 (2007-05), Mai 2007.

[Eur09] EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE: *Electronic Signatures and Infrastructures (ESI) – PDF Advanced Electronic Signature Profiles*. www.etsi.org, Juli 2009.

[FHT10] FARRELL, S., R. HOUSLEY, and S. TURNER: *An Internet Attribute Certificate Profile for Authorization*. RFC 5755 (Proposed Standard), January 2010.

[FK92] FERRAIOLO, D. and R. KUHN: *Role-based access controls*. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, pages 554–563, October 1992.

[Fri06] FRIEDL, JEFFREY: *Mastering Regular Expressions*. O'Reilly Media, Inc., 2006.

[Gam85] GAMAL, TAHER EL: *A public key cryptosystem and a signature scheme based on discrete logarithms*. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[GJ90] GRUNE, D. and C.J.H. JACOBS: *Parsing techniques: a practical guide*. Ellis Horwood series in computers and their applications. Ellis Horwood, 1990.

[GNY90] GONG, LI, ROGER NEEDHAM, and RAPHAEL YAHALOM: *Reasoning about Belief in Cryptographic Protocols*. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 234–248, Oakland, California, May 1990. IEEE.

[Gro03] GROSS, THOMAS: *Security Analysis of the SAML Single Sign-on Browser/Artifact Profile*. In *Proceedings of the 19th Annual Computer Security Applications Conference*, Las Vegas, Nevada, December 8-12 2003. Applied Computer Security Associates.

[HK00] HOWELL, JON and DAVID KOTZ: *A Formal Semantics for SPKI*. Technical Report TR2000-363, Dartmouth College, Computer Science, Hanover, NH, March 2000.

[Hoa85] HOARE, CHARLES ANTONY RICHARD: *Communicating Sequential Processes*. Prentice Hall, 1985.

[Hof99] HOFFMAN, P.: *Enhanced Security Services for S/MIME*. RFC 2634 (Proposed Standard), June 1999. Updated by RFC 5035.

[Hou07] HOUSLEY, R.: *Cryptographic Message Syntax (CMS) Multiple Signer Clarification*. RFC 4853 (Proposed Standard), April 2007.

[Hou09] HOUSLEY, R.: *Cryptographic Message Syntax (CMS)*. RFC 5652 (Standard), September 2009.

[Int08] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO): *ISO 32000-1:2008 – Document management – Portable document format – Part 1: PDF 1.7*. ISO Standard, 2008.

[ISO89] ISO 8807: *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*. International Standard Organization, Geneva, 1989.

[ISO04] ISO/IEC: *ISO/IEC 10118-3:2004 – Hash-functions – Part 3: Dedicated hash-functions*, 2004.

[IT05] ITU-T: *Recommendation X.509 / ISO/IEC 9594-8: Information Technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*, 8 2005.

[jCPB+12] CHANG, SHU JEN, RAY PERLNER, WILLIAM E. BURR, MELTEM SÃ¶NMEZ TURAN, JOHN M. KELSEY, SOURADYUTI PAUL, and LAWRENCE E. BASSHAM: *Third-round report of the sha-3 cryptographic hash algorithm competition*. Technical Report NISTIR 7896, National Institute of Standards and Technology (NIST), November 2012. http://dx.doi.org/10.6028/NIST.IR.7896.

[JMV01] JOHNSON, DON, ALFRED MENEZES, and SCOTT VANSTONE: *The Elliptic Curve Digital Signature Algorithm (ECDSA)*. International Journal of Information Security, 1(1):36–63, July 001. DOI: 10.1007/s102070100002.

[Kal93] KALISKI, B.: *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services*. RFC 1424 (Historic), February 1993.

[Kal98] KALISKI, B.: *PKCS #7: Cryptographic Message Syntax Version 1.5*. RFC 2315 (Informational), March 1998.

[Kec00] KECK, DIRK OLIVER: *Erkennung von Wechselwirkungen zwischen Mehrwertdiensten durch Analyse ihrer Konfigurationen und Protokollabläufe – 79. Bericht über verkehrstheoretische Arbeiten*. Dissertation, Universität Stuttgart, 2000.

[Kem89] KEMMERER, RICHARD A.: *Analyzing Encryption Protocols Using Formal Verification Techniques*. IEEE Journal on Selected Areas in Coimmunications, 7(4):448–457, May 1989.

[Ken93] KENT, S.: *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*. RFC 1422 (Historic), February 1993.

[KMV00] KOBLITZ, NEAL, ALFRED MENEZES, and SCOTT VANSTONE: *The state of elliptic curve cryptography*. Des. Codes Cryptography, 19:173–193, March 2000.

[KN93] KOHL, J. and C. NEUMAN: *The Kerberos Network Authentication Service (V5)*. RFC 1510 (Proposed Standard), September 1993. Obsoleted by RFC 4120.

[Lin93] LINN, J.: *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*. RFC 1421 (Historic), February 1993.

[Low96] LOWE, GAVIN: *Breaking and fixing the Needham-Schroeder public-key protocol using FDR*. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes on Computer Science*, pages 147–166. Springer, 1996.

[LPS05] LINSENBARDT, D., S. PONTIUS, and A. STURGEON: *Internet X.509 Public Key Infrastructure Warranty Certificate Extension*. RFC 4059 (Informational), May 2005.

[MAM+99] MYERS, M., R. ANKNEY, A. MALPANI, S. GALPERIN, and C. ADAMS: *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 2560 (Proposed Standard), June 1999. Updated by RFC 6277.

[Mil80] MILNER, ROBIN: *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[Min08] MINVIELLE, ROBERT: *Linux security auditing tool (lsat)*. http://usat.sourceforge.net/, April 2008.

[Mos05] MOSES, TIM: *eXtensible Access control Markup Language (XACML) Version 2.0*. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, February 2005.

[MPW92] MILNER, ROBIN, JOACHIM PARROW, and DAVID WALKER: *A calculus of mobile processes, parts i and ii*. Inf. Comput., 100(1):1–77, September 1992.

[MvOV96] MENEZES, A., P. VAN OORSCHOT, and S. VANSTONE: *Handbook of Applied Cryptography*. CRC Press, 1996.

[Mye98] MYERS, MICHAEL: *Revocation: Options and challenges*. In HIRSCHFELD, R. (editor): *Proceedings of Financial Cryptography: Second International Conference, FC'98, Anguilla, British West Indies*, volume 1465 of *Lecture Notes in Computer Science*, page 165ff, Heidelberg, February 1998. Springer-Verlag.

[Nat94] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST): *Digital Signature Standard (DSS)*. Number 186 in *Federal Information Processing Standards Publications (FIPS PUBS)*. National Institute of Standards and Technology (NIST), Geitherburg, MD, USA, May 1994.

[Nat02] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST): *Secure Hash Standard*. Number 180-2 in *Federal Information Processing Standards Publications (FIPS PUBS)*. National Institute of Standards and Technology (NIST), Geitherburg, MD, USA, August 2002.

[Nat04] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST): *Secure Hash Standard, Change Notice 1*. Number 180-2 in *Federal Information Processing Standards Publications (FIPS PUBS)*. National Institute of Standards and Technology (NIST), Geitherburg, MD, USA, Februar 2004.

[NE11] NAGIOS ENTERPRISES, LLC. http://www.nagios.com/, 2011.

[NT94] NEUMAN, B. CLIFFORD and THEODORE TS'O: *Kerberos: An Authentication Service for Computer Networks*. IEEE Communications Magazine, 32(9):33–38, September 1994.

[NYHR05] NEUMAN, C., T. YU, S. HARTMAN, and K. RAEBURN: *The Kerberos Network Authentication Service (V5)*. RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113.

[Oxf10] OXFORD UNIVERSITY COMPUTING LABORATORY: *Failures-Divergence Refinement - FDR2 User Manual*, October 2010. http://web.comlab.ox.ac.uk/projects/concurrency-tools/.

[Par07] PARR, TERENCE: *The Definitive ANTLR Reference: Building Domain-Specific Languages*. Pragmatic Programmers. Pragmatic Bookshelf, first edition, May 2007.

[PCI10] PCI SECURITY STANDARDS COUNCIL: *Payment Card Industry (PCI) – Datensicherheitsstandard – Anforderungen und Sicherheitsbeurteilungsverfahren*. https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf, October 2010.

[PL10] PARDUCCI, BILL and HAL LOCKHART: *eXtensible Access control Markup Language (XACML) Version 2.0*. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf, August 2010.

[PMH96] PETERSEN, HOLGER, MARKUS MICHELS und PATRICK HORSTER: *Taxonomie digitaler Signaturkonzepte*. In: *Proceedings Digitale Signaturen*, Darmstadt, 1996. Vieweg Verlag.

[RSA78] RIVEST, R. L., A. SHAMIR und L. ADLEMAN: *A method for obtaining digital signatures and public-key cryptosystems*. Commun. ACM, 21(2):120–126, 1978.

[RSA93] RSA SECURITY INC.: *PKCS #7 v1,5: RSA Cryptographic Message Syntax Standard*. ftp://ftp.rsasecurity.com/pub/pkcs/ps/pkcs-7.ps, November 1993. RSA Security Inc. Public-Key Cryptography Standards (PKCS).

[RSA12] RSA LABORATORIES, EMC CORPORATION: *PKCS #1 v2.2: RSA Cryptography Standard*. http://www.rsa.com/rsalabs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf, October 2012. EMC Corporation Public-Key Cryptography Standards (PKCS).

[SC01] SYVERSON, PAUL and ILIANO CERVESATO: *The Logic of Authentication Protocols*. In FOCARDI, R. and R. GORRIERI (editors): *Foundations of Security Analysis and Design — FOSAD'00*, number 2171 in *LNCS*, pages 63–136. Springer-Verlag, 2001.

[Sch01] SCHRÖER, FRIEDRICH WILHELM: *AMBER, An Ambiguity Checker for Context-free Grammars*. Technical report, Fraunhofer Institute for Computer Architecture and Software Technology, 2001. http://accent.compilertools.net/Amber.html.

[Sch07] SCHAAD, J.: *Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility*. RFC 5035 (Proposed Standard), August 2007.

[SNP04] SANTESSON, S., M. NYSTROM, and T. POLK: *Internet X.509 Public Key Infrastructure: Qualified Certificates Profile*. RFC 3739 (Proposed Standard), March 2004.

[SO96] SYVERSON, PAUL F. and PAUL C. VAN OORSCHOT: *A Unified Cryptographic Protocol Logic*. Technical report, NRL Publication 5540-227, Naval Research Lab, 1996.

[Ten11] TENABLE NETWORK SECURITY: *Nessus vulnerability scanner*. http://www.tenable.com/products/nessus, 2011.

[TT09] T7 E.V. and TELETRUST E.V.: *Common PKI Specifications for Interoperable Applications*. http://www.t7ev.org/uploads/media/Common-PKI_v2.0.pdf, 20. January 2009.

[Tur00] TURNBULL, JIM: *Cross-Certification and PKI Policy Networking*. http://www.entrust.com/resources/pdf/cross_certification.pdf, August 2000.

[vH11] HAUGWITZ, HANNES VON: *Advanced Intrusion Detection Environment*. http://aide.sourceforge.net/, 2011.

[Vin02] VINCK, BART: *USECA: UMTS Security Architecture*. Report AC336/ATEA/WP23/DS/P/08/1, Advanced Communications Technologies & Services Programme (ACTS), March 2002.

[vO93] OORSCHOT, PAUL C. VAN: *Extending cryptographic logics of belief to key agreement protocols*. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 233–243. ACM Press, November 1993.

[WKH97] WAHL, M., S. KILLE, and T. HOWES: *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. RFC 2253 (Proposed Standard), December 1997. Obsoleted by RFCs 4510, 4514, updated by RFC 3377.

[WVH05] WANG, J., D. D. VECCHIO, and M. HUMPHREY: *Extending the Security Assertion Marup Language to Support Delegation for Web Services and Grid Services*. In *Proceedings of the International Conference on Web Services*, Orlando, Florida, USA, 1005.

[ZT06] ZHU, L. and B. TUNG: *Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)*. RFC 4556 (Proposed Standard), June 2006. Updated by RFC 6112.