Anonymity Enhancing Protocol
AEP

Tóth Gergely

Betreuer:

Institut für Telematik
Prof. em. Dr. Dr. h.c. mult. Gerhard Krüger
Fakultät für Informatik, Universität Karlsruhe (TH)
Daniel Müller
Frank Pählke

Lehrstuhl für Messtechnik und Informationssysteme
Technische und Wirtschaftswissenschaftliche Universität Budapest
Hornák Zoltán

Karlsruhe, den 31. Mai 2002

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 31. Mai 2002

Tóth Gergely

Abstract

Anonymity Enhancing Protocol (AEP)

Mit der Verbreitung der Netzwerkkommunikation tauchte zuerst der Bedarf an Protokollen und Standards auf, die **sicherheitstechnische** Funktionalitäten wie Vertraulichkeit, Integritätssicherung oder Nichtabstreitbarkeit anbieten. Heutzutage stehen schon zahlreiche solche, gut untersuchte, weitverbreitete, zuverlässige und ohne tieferes technisches Wissen anwendbare Protokolle zur Verfügung. Mit der Verbreitung ihrer Anwendung tauchten nach der Behebung der wichtigsten Probleme (der einfachen, sicheren und authentischen Kommunikation) allerdings neue auf.

In jüngster Zeit gewinnt der **Schutz der Privatsphäre** (*privacy*) und damit verbunden das während der Kommunikation erreichbare höchstmögliche Niveau an Anonymität immer mehr an Bedeutung. Zur Befriedigung dieser neuen Anforderungen stehen nur Teillösungen zur Verfügung. Ziel von AEP ist die Spezifikation eines Protokolls, das die bewährten und bekannten sicherheitstechnischen Verfahren mit den bereits bekannten Anonymitätsverfahren kombiniert und eine allgemein benutzbare, selbständige Anonymitätsschicht definiert.

Es existieren schon zahlreiche Schichten in der Netzwerkkommunikation, die sicherheitstechnische Funktionalitäten anbieten: z. B. SSL, TLS oder SSH aus der Welt der Internet oder WTLS aus der WAP-Welt. Diese setzen eine Client-Server-Architektur voraus, unterstützen einen PKI-basierten Schlüsselaustausch und ermöglichen Vertraulichkeit, Integritätssicherung und Server- bzw. Clientauthentizität zu erreichen.

Bei der Planung von AEP wurden die Eigenschaften dieser Schichten berücksichtigt. Neben der einheitlichen und einfachen Benutzung wurde die Ersetzbarkeit der verschiedenen Anonymitätsverfahren innerhalb des Protokolls als Voraussetzung definiert.

Die zwei bekanntesten Methoden zur Bereitstellung von Anonymität sind die Verfahren der blinden Unterschrift und der Pseudoidentität. Das ursprünglich von David Chaum für anonyme Bezahlungen im Internet (*DigiCash*) entwickelte Protokoll der **blinden Unterschrift** ermöglicht nicht zurückverfolgbare Anonymität. Das andere Grundverfahren wird **Pseudoidentität** genannt. Dabei wird die Identität des Subjekts geschützt, sie kann jedoch im Falle eines Missbrauchs zurückverfolgt werden.

Bei der Spezifikation des Protokolls wurde zuerst die geeignete Präsentationssprache ausgewählt. Danach wurden mit ihrer Hilfe die in den verschiedenen Szenarien auftauchende abstrakten Teilnehmern und ihre Tätigkeiten beschrieben. Schließlich wurde die in den konkreten Fällen ausgetauschten Nachrichten spezifiziert, für die AEP als Kommunikationssprache XML benutzt.

Das so entstandene Protokoll stellt auch verschiedene Möglichkeiten zur Verteidigung gegen die, mit der Anonymität gleichzeitig auftretenden, Missbrauchsmöglichkeiten zur Verfügung. Neben der theoretischen Spezifikation wurde auch eine JAVA-Implementierung des Protokolls bereitgestellt, das die Anonymitätsfunktionalitäten über TCP/IP und SSH2 anbietet.

Das Protokoll stellt eine **einheitliche**, **einfach benutzbare Umgebung** für die Realisierung verschiedener Anonymitätsverfahren zur Verfügung. Mit seiner Hilfe können Anonymitäts-dienste **ohne tiefgehendes technisches Wissen** in Anspruch genommen werden. Ähnlich zu den verschiedenen Sicherheitsprotokollen befindet sich AEP auch in einer **eigenen Protokollschicht** und ermöglich die **Ersetzbarkeit und Erweiterbarkeit der verschiedenen Anonymitätsverfahren**.

Abstract

Anonymity Enhancing Protocol (AEP)

With the spreading of network communication first came the need for protocols and standards supplying **security-related** functionality that provide services such as confidentiality, integrity protection or non-repudiation. Today several such protocols exist that are widely known, reliable and usable without deep technical knowledge. After solving the most important problems, the broadening of their usage brought up new security-related needs besides the simple, confidential and authentic communication.

In latest time protecting personal data (**privacy**) and thus the highest possible achievable level of anonymity during network communication are becoming more and more emphasized. To meet these new needs only partial solutions exist. The aim of AEP is to specify a network protocol that combines the known successful security methods and basic solutions with the already existing anonymity providing technologies and to define a generally usable, independent anonymity network layer.

Several network communication layers exist that provide security-related functionality. Such is SSL, TLS or SSH from the world of the Internet, or WTLS known from the WAP world. These usually assume client-server communication, during which they provide PKI based key exchange, encryption, integrity-protection, server side and optionally client side authentication

During the planning of AEP the properties of these layers have been considered. Besides the standardized and easy usage the changeability of the different anonymity methods within AEP have been defined as requirements.

For providing anonymity the two most widely used methods are blind signatures and pseudo-identities. The three-sided (bank, customer, merchant) **blind-signature** protocol has originally been developed by David Chaum for anonymous payments through the Internet (*DigiCash*) and provides non-traceable anonymity. The other basic method is the **pseudo-identity**, where the real identity of the subject is protected, however in case of abuse the real identity can be traced back.

For the specification first a suitable presentation language has been chosen. In the next phase with the help of this presentation language the possible partners and their actions in the different situations have been described. Finally the network communication for the several possible use cases has been specified, for which AEP uses XML.

The resulting protocol provides protection measures against possible abuses and means to recover from an abuse. During the specification a JAVA implementation of the protocol has also been developed, AEP provides anonymity functionality over TCP/IP and SSH2.

AEP provides a **standardized**, **easy-to-use framework** for the realization of different anonymity methods. With its help anonymity functionality can be accessed **without deep technical knowledge**. Like security protocols, it resides in a **separate protocol layer** and provides possibilities for **changing and extending the anonymity methods**.

Összefoglaló

Anonymity Enhancing Protocol (AEP)
Anonimitási szolgáltatásokat nyújtó hálózati protokoll

A hálózati kommunikáció elterjedésével elsőként a **biztonságtechnikai** funkcionalitást nyújtó protokollok és szabványok iránt jelentkezett igény, melyek olyan szolgáltatásokat tudnak nyújtani, mint bizalmasság, integritás-védelem, vagy letagadhatatlanság. Napjainkban már számos ilyen ismert, elterjedt, megbízható és mély szakmai tudás nélkül is alkalmazható protokoll áll rendelkezésre. Ezek használatának elterjedésével a legégetőbb problémák megoldása után a kommunikációval szemben új biztonsági követelmények merültek fel az "egyszerű", titkos és hiteles kommunikáción túl.

Az utóbbi időben egyre nagyobb hangsúlyt kap a **személyi és személyes adatok védelme** (*privacy*) és ennek megfelelően a kommunikáció során elérhető minél nagyobb szintű anonimitás. Ennek az új igénynek a kielégítésére csak részmegoldások léteznek. AEP célja egy olyan hálózati protokoll specifikációja, amely ötvözi a már bevált és ismert biztonságtechnikai eljárásokat, alapmegoldásokat, valamint a már meglevő anonimitási módszereket és általánosan alkalmazható külön hálózati rétegben megvalósított anonimitási szolgáltatásokat nyújt.

Számos olyan hálózati kommunikációs réteg ismert, amely biztonságtechnikai funkcionalitást nyújt. Ilyen az Internet világából ismert SSL, vagy a TLS, valamint az SSH, de hasonló a WAP-ban használt WTLS is. Ezek általában kliens-szerver kommunikációt tételeznek fel, melyek során biztosítják a PKI alapú kulcscserét, titkosítást, integritás-védelmet valamint a szerver és esetleg kliens oldali azonosítást is.

AEP tervezésénél ezen rétegek tulajdonságait tartottuk szem előtt. Követelményként fogalmazódott meg az egységes és könnyű használat mellett, hogy a protokollnak biztosítania kell a különböző anonimitási technikák cserélhetőségét.

Anonimitás biztosítására a két legjobban elterjedt módszer a vak-aláírás és az álnév. Az eredetileg David Chaum által internetes anonim fizetésre kidolgozott háromszereplős (bank, ügyfél, szolgáltató) **vak-aláírás** protokoll (DigiCash) teljes, visszakövethetetlen anonimitást biztosít. A másik alapmódszer az **álnév** (*pszeudo-identitás*), amely esetében az alany identitása védelmet élvez, de visszaélés esetén az álnév használója visszakövethető.

A protokoll specifikálásához először egy megfelelő leíró nyelvet választottunk. Ezek után ennek a leíró nyelvnek a segítségével a következő fázisban a különböző esetekben megjelenő lehetséges absztrakt kommunikációs partnerek és azok lehetséges tevékenységeit írtuk le. Végül a konkrét esetekhez tartozó hálózati kommunikációt specifikáltuk, melyhez a protokoll az XML szabvány leíró nyelvet használja.

Az így elkészült protokoll megoldási módszereket kínál az anonimitással együtt felmerülő visszaélések elleni védekezésre és azok utólagos felderítésének támogatására is. A munka során elkészült a protokoll JAVA implementációja, melyben az anonimitási funkcionalitást a protokoll a TCP/IP-t felhasználó SSH2 biztonságtechnikai réteg felett bocsátja rendelkezésre.

A protokoll **egységes, könnyen használható felületet** biztosít a különböző anonimitási módszerek megvalósításához, segítségével anonimitási szolgáltatásokat **mély szakmai tudás nélkül** lehet igénybe venni. A biztonságtechnikai protokollok mintájára **külön rétegben** helyezkedik el, valamint biztosítja az **anonimitási módszerek cserélésének, kibővítésének lehetőségét**.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

Die rapide Entwicklung der letzten Jahrzehnten im Bereich Computertechnik, Hardware, Software und Kommunikation hat die weitgehende Elektronisierung und Integration der Datenverarbeitungssysteme ermöglicht. Diese Tendenz hat gezeigt, dass die Netzwerkkommunikation eine Schlüsselrolle in der Informatik unserer Zeit spielen wird.

Bei der Kommunikation ist der zuverlässige und immer schnellere **Datentransport** zwischen den Endpunkten ein Grundkriterium. Für dessen Realisation existieren schon seit längerem verschiedene standardisierte Lösungen. Dank der hierarchischen Komposition der Netzwerkprotokolle können oben liegende Schichten den Anwendern Dienste bereitstellen, die ohne tiefes, technisches Wissen effektiv benutzt werden können.

Ein weiteres Kriterium für die Kommunikation ist die **Sicherheit**. Dabei müssen Lösungen für Probleme gefunden werden wie entferntes Authentifizieren (ist wirklich derjenige am anderen Ende der Verbindung, für den er sich ausgibt), Vertraulichkeit (nur diejenigen sollen die Nachricht lesen können, für die sie bestimmt war), Integritätssicherung (genau das, was abgesandt wurde, soll auch ankommen) oder Nichtabstreitbarkeit (falls irgendetwas ankommt, soll es möglich sein zu beweisen, dass der Absender es geschickt hat).

Für diese Aufgaben existieren schon standardisierte, mathematisch fundierte Methoden, die sich auch in der Praxis bewährt haben. Obwohl das ISO-OSI Referenzmodell keine selbständige Sicherheitsschicht vorsieht (die obengenannten Aufgaben werden hauptsächlich für die Anwendungsschicht zugeordnet), hat es sich in der Praxis gezeigt, dass im Kommunikationsmodell eine Schicht benötigt wird, die sich dediziert nur mit solchen Sicherheitsaufgaben befasst. Es ist wichtig zu bemerken, dass solche Sicherheitsschichten schon existieren. Eine ihrer grundlegenden Eigenschaften ist, dass sie sich an keinen bestimmten (Verschlüsselungs-, Signier-, Hash, usw.) Algorithmus binden, sondern diese sind im Protokoll austauschbar, damit das Protokoll selbst langzeitig benutzbar bleibt.

Mit der zunehmenden Verbreitung der Sicherheitsschichten taucht in letzter Zeit immer öfter der Bedarf an **Anonymität** auf. Da während der Kommunikation der Datentransport sicher abläuft, der Datenstrom unabhörbar und unmodifizierbar ist, und beide Seiten in der Identität der Anderen sicher sein können, taucht der Wunsch auf, diese Identität jetzt zu verbergen. Da immer größere Datenmengen zum Beispiel durch das Internet fließen und immer größere Datenbanken mit Gewohnheiten und persönlichen Daten von Benutzern verwaltet werden, ist neben der Gesetzgebung (zum Beispiel Datenschutzgesetz) die Anonymität ein Schutz des Teilnehmers vor Überwachung. Damit zusammen kommt auch die Notwendigkeit, sich gegen Missbrauch zu schützen, die eng zusammen mit den Anonymitätsmethoden behandelt werden muss.

Zur Realisation der verschiedenen Niveaus von Anonymität existieren schon geeignete Mechanismen. Aber es fehlt eine Zusammenfassung, eine Vereinheitlichung, die in den Bereichen der Sicherheit schon geschehen ist. Damit zusammen ist es auch nötig, eine Protokollschicht – eine "Black Box" – zu haben, die nur die Anonymitätsfunktionalität erledigt, ohne dass der Anwender tiefgehende Kenntnisse über das Innenleben der Anonymitätsmechanismen aufweisen müsste.

Diese Arbeit hat die Zielsetzung, ein Netzwerkprotokoll (**AEP** – *Anonymity Enhancing Protocol*) zu spezifizieren, das die Funktionalität einer abstrakten Sicherheitsschicht benutzt und darauf basierend die Verwirklichung verschiedener Anonymitätsmethoden[1] in einer einheitlichen, einfach benutzbaren Weise ermöglicht.

1.1 Zielsetzung von AEP

Im ISO-OSI-Referenzmodell wurde keine Schicht für die Realisierung von Anonymität vorgesehen, solche Aufgaben müssen in der Anwendungsschicht erledigt werden. Aus Anwendersicht ist es aber nötig, eine selbständige Schicht – die **Anonymitätsschicht** – zur Behandlung von Problemen im Zusammenhang mit der Anonymität zu haben, die für verschiedene Aufgaben verschiedene Niveaus an Anonymität bereitstellen kann, die standardisierte, austauschbare und leicht konfigurierbare Methoden benutzt und zu deren Anwendung keine Fachkenntnisse auf dem Gebiet der Anonymität erforderlich sind. Zielsetzung von AEP ist, all das zu ermöglichen und diese Anonymitätsschicht zu verwirklichen.

Ein weiteres Ziel von AEP ist es, zu ermöglichen, dass ähnlich wie bei den verschiedenen Sicherheitsschichten, wo die kryptographischen Algorithmen austauschbar sind, die Anonymitätsverfahren innerhalb von AEP durcheinander **ersetzbar** sind. Es soll möglich sein, dass aus Anwendersicht nur das gewünschte Niveau an Anonymität bestimmt wird. Wie AEP das dann erreicht, wird innerhalb des Protokolls abgemacht. Dadurch wird die Benutzbarkeit und Robustheit von AEP drastisch erhöht.

AEP behandelt zur Zeit nur einen speziellen Bereich der Anonymität. Dieser ist durch eine Client-Server-Architektur gekennzeichnet. In einem solchen Fall ist der Client, oder in AEP das **Subjekt** derjenige, der das höchstmögliche Niveau an Anonymität während der Inanspruchnahme eines Dienstes erreichen will. Ihm steht der Server gegenüber, in AEP der **Dienstanbieter**, der solche anonym benutzbaren Dienste anbietet. Bei der Kommunikation ist es wichtig, dass nur das Subjekt anonym ist, der Dienstanbieter ist identifizierbar. Es muss auch berücksichtigt werden, dass das Subjekt und der Dienstanbieter entgegengesetzte Interessen haben: Das Subjekt möchte das höchstmögliche Niveau an Anonymität erreichen; der Dienstanbieter will dagegen das Subjekt – meistens aus geschäftlichen oder wirtschaftlichen Gründen, aber auch als Gegenmaßname zum potenziellen Missbrauch – völlig identifizieren. Um diesen Gegensatz aufzulösen, wird im AEP eine dritte Partei, die **Anonymitätsinstanz** eingeführt.

Im Weiteren wird AEP auch dadurch beschränkt, dass der Dienstanbieter nicht einen völlig öffentlichen Dienst betreibt[2], sondern die Dienstleistung nur für eine bestimmte Zielgruppe erbringt, das heißt, dass vor der eigentlichen Dienstleistung eine anonyme Autorisierung durchgeführt werden muss. Wird der Dienstanbieter durch eine Firma verkörpert, so ist diese Zielgruppe die Gruppe der Kunden, welche die Dienstleistung gegen Entgelt in Anspruch nehmen. Ist der Dienstanbieter eine staatliche oder öffentliche Einrichtung, so ist diese Gruppe zum Beispiel die Gruppe der Bürger des Landes.

---

[1] Obwohl das Endziel von AEP ist, dass eine beliebige Anonymitätsmethode in seinem Rahmen verwirklichbar sein soll, hat die in dieser Arbeit vorgestellte Spezifikation noch einige Beschränkungen, die später erläutert werden.

[2] Es gibt auch Dienstanbieter, die keine Beschränkung bezüglich der Dienstanwender haben. Dieser Fall wird jedoch in AEP nicht behandelt.

AEP wird dafür ausgelegt, für zwei Probleme eine Lösungsumgebung bereitzustellen. Das eine ist, wie das Subjekt Daten (falsche Identitäten, Erlaubnisse, usw.), die er später zur Autorisierung brauchen wird, während des **Initialisierungsprozesses** erhält. Das andere ist, wie diese beim **Autorisierungsprozess** verwendet werden können. Damit hat AEP das Ziel, ein anonymes Autorisierungssystem zu beschreiben. Wie nach der Autorisierung die eigentliche Dienstleistung erbracht wird, ist nicht Teil der AEP-Spezifikation.

1.2 Gliederung der Arbeit

Die Gliederung der Arbeit folgt dem, im Folgenden beschriebenem Schema.

Im Kapitel 2 werden die grundlegenden Techniken behandelt, die benutzt werden, um die Funktionalität von AEP zu erbringen. Diese können grob in zwei Kategorien eingeordnet werden: in Abschnitt 2.1 werden verschiedenen Sicherheitsmechanismen behandelt, und in Abschnitt 2.2 wird eine kurze Zusammenfassung über Anonymität gegeben.

Kapitel 3 fasst die AEP-Systemarchitektur kurz zusammen, stellt die, in Kapitel 4 ausführlich behandelten, Protokollschichten vor, beschreibt das Verhältnis dieser Schichten zu anderen Schichten der Netzwerkarchitektur und erklärt die Rollen der Teilnehmer, die AEP benutzen. Ziel des Kapitels ist, eine grobe Übersicht über AEP und über ein System, dass AEP benutzt, zu geben.

Kapitel 4 stellt die AEP-Spezifikation vor. Zuerst wird der AEP-Dienstzugangspunkt spezifiziert (Abschnitt 4.1), dann werden die beiden Sprachen zur Datenbeschreibung vorgestellt (Abschnitt 4.2), schließlich werden die Kommunikationsvorgänge während des AEP-Protokollablaufs ausführlich behandelt (Abschnitt 4.3).

Kapitel 5 beschreibt die Implementierung der Version 0.2 von AEP. Nach allgemeinen Bemerkungen (Abschnitt 5.1) werden Implementierungsdetails bezüglich der Sicherheitsschicht behandelt (Abschnitt 5.2), anschließend wird die Implementierung der Anonymitätsschicht ausführlich beschrieben (Abschnitt 5.3).

Kapitel 6 diskutiert zuerst verschiedene Ideen zur möglichen Weiterentwicklung der Spezifikation und der Implementierung des Protokolls bzw. des Testsystems (Abschnitt 6.1), und dann wird eine Zusammenfassung der bisherigen Arbeit gegeben (Abschnitt 6.2).

Anhang A listet die in der vorliegenden Arbeit benutzten Abkürzungen auf. Anhang B erklärt die Notationen, die zur Beschreibung eines Dienstzugangspunkts benutzt werden. In Anhang C werden die Präsentationssprache und die Kommunikationssprache spezifiziert. In Anhang D werden die verschiedenen Datentypen, die in AEP benutzt werden, in der Präsentationssprache definiert. Anhang E beinhaltet eine kurze Benutzeranleitung, anhand derer die verschiedenen implementierten Anwendungen installiert und konfiguriert werden können. Schließlich listet Anhang F die benutzen Referenzen auf.

2. Grundlagen

In diesem Kapitel werden die Grundlagen der Techniken behandelt, die benutzt werden, um die Funktionalität von AEP zu erbringen. Diese können grob in zwei Kategorien eingeordnet werden:

- Zuerst werden sicherheitsbezogene Probleme und Lösungsmethoden behandelt. In diesem Abschnitt werden zuerst die verschiedenen Probleme – im Bezug auf die Sicherheit der Netzwerkkommunikation – vorgestellt, und dann gibt eine kurze Zusammenfassung der verschiedenen Techniken und Methoden zu deren Lösung.
- Dann werden die Grundlagen der Anonymität erläutert. Nach der Einführung verschiedener Anonymitätsniveaus werden bekannte Anonymitätsmechanismen beschrieben. Eine kurze Zusammenfassung ethischer Probleme und gesetzlicher Aspekte wird auch gegeben.

Im Folgenden werden Methoden beschrieben, die als Protokollschichten implementiert wurden, oder implementiert werden könnten. Die meistverbreitete Schichtenarchitektur für die Netzwerkkommunikation ist das ISO-OSI-Referenzmodell[3].

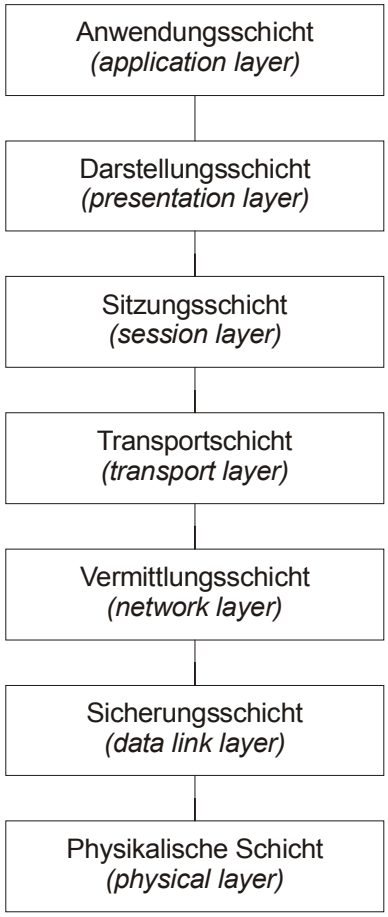


Abbildung 1 – **Das ISO-OSI Referenzmodell**

[3] *ISO – Open Systems Interconnection Reference Model* [ISO-1]

2.1 Sicherheit

Sicherheit ist ein Bedürfnis eines jeden. Ein Anbieter will, dass nur autorisierte Benutzer die angebotenen Dienste in Anspruch nehmen können. Sowohl Benutzer als auch Anbieter wollen, dass die Dienste problemlos erreichbar sind, dass die Kommunikation gesichert abläuft und dass keine unbefugten Dritten mitlauschen oder den Datenverkehr verändern.

Solche und ähnliche Probleme beschäftigen schon seit Jahrhunderten die Wissenschaftler[4]. Mit der Verbreitung der Netzwerkkommunikation und des e-Business wird dieser Themenbereich heutzutage intensiv behandelt.

Im Bezug auf Sicherheit ist zuerst der Begriff der **Schwachstelle** *(vulnerability)* wichtig. Eine Schwachstelle ist eine Eigenschaft des Systems, die gemäß der Ziele des Systems unerwünscht ist. Das Vorhandensein von Schwachstellen zieht die Entstehung von **Bedrohungen** *(threat)* mit sich. Durch die Elimination der Schwachstellen werden auch die Bedrohungen beseitigt. Wird eine Bedrohung realisiert, so spricht man von einem **Angriff** *(attack)*, der von einem **Angreifer** *(attacker)* ausgeführt wird. Durch den Angriff entsteht **Schaden** *(damage)*. Angriffe können in zwei Gruppen unterteilt werden: **passive Angriffe** brauchen nur lesbaren Zugriff auf die Daten, wogegen **aktive Angriffe** diese auch modifizieren und deswegen schreibbaren Zugriff benötigen.

In den Folgenden wird zuerst die Notwendigkeit der Sicherheit bewiesen, dann werden sowohl für Dienste als auch für die Kommunikation die sicherheitsbezogene Bedrohungen, die Anforderungen und schließlich die möglichen Maßnahmen behandelt.

2.1.1 Notwendigkeit

In den Anfängen der Rechentechnik waren Computer ohne Netzanbindung zentral aufgestellt und nur wenige der heute aktuellen Sicherheitsaspekte waren von Bedeutung. Da die Rechner von außen nicht erreichbar waren, wurden hauptsächlich physikalische Zugriffskontrollsysteme angewandt, damit ein möglicher Angreifer die Rechner nicht erreichen konnte.

Mir der Verbreitung der Netzwerkkommunikation tauchten jedoch völlig neue Aspekte der Sicherheit auf. Rechner wurden in Netzwerke integriert, Zugriffe von außerhalb der Organisation sind auch erlaubt oder sogar erforderlich bzw. erwünscht. Heute, mit der Verbreitung des e-Business, sind die netzwerkbezogenen Aspekte der Sicherheit von äußerster Wichtigkeit.

2.1.2 Dienste

In der Netzwerkkommunikation von heute überwiegen diejenigen Systeme, die gemäß einer Client-Server-Architektur aufgebaut sind. In einem solchen Fall nimmt der Client Dienste, die

[4] Eines der bekanntesten Probleme ist das Problem der byzantinischen Generäle. Konstantinopel ist unter Belagerung durch Ottomane Bataillone. Obwohl die Bataillone stark sind, sie können einzeln die Stadt nicht einnehmen, sie müssen ihren Angriff koordinieren und zur selben Zeit starten. Die Generäle der Bataillone können miteinander mit Hilfe von Kurieren kommunizieren. Es gibt jedoch unter der Generäle auch einige Verräter, die aus verschiedensten politischen Gründen den Angriff sabotieren wollen. Verallgemeinert könnte man das Problem folgendermaßen formulieren: Können konkurrierende Prozesse sich koordinieren falls sich einige unter ihnen fehlerhaft verhalten.

von einem Server angeboten werden, in Anspruch. Der Client ist Benutzer des Servers eines Anbieters.

Das wichtigste Medium, auf dass sich auch diese Arbeit konzentriert, ist das Internet. Mit dessen Ausbreitung und der Intensivierung des Datenverkehrs darüber, gewinnen Dienste im Internet immer mehr an Gewicht in der Wirtschaft. Deswegen ist auch ihre Sicherheit von äußerster Wichtigkeit.

2.1.2.1 Bedrohungen

Die Bedrohungen bei Netzwerkdiensten können grob in zwei Gruppen unterteilt werden: entweder kann ein berechtigter Benutzer den Dienst nicht erreichen, oder ein nicht berechtigter Benutzer nimmt den Dienst in Anspruch.

2.1.2.1.1 Nichtverfügbarkeit *(denial of service)*

DoS- (*Denial of Service*) Attacken verursachen, dass der normale Benutzer eines Dienstes diesen nicht mehr in Anspruch nehmen kann, weil der Dienstanbieter nicht in der Lage ist, den Dienst anzubieten. Dies kann zum Beispiel dadurch verursacht werden, dass Angreifer die Rechner des Dienstanbieters mit zu vielen – möglicherweise sinnlosen – Abfragen überlasten. Ein Spezialfall von DoS ist **DDos**[5] (*Distributed DoS*), falls diese Abfragen von vielen verschiedenen Rechnern zum Dienstanbieter geschickt werden. DoS- und DDoS-Angriffe gehören in die Gruppe der aktiven Angriffe.

2.1.2.1.2 Unberechtigtes Benutzten *(illegitimate use)*

Im Falle eines unberechtigten Benutzens wird ein Dienst in Anspruch genommen, der nicht autorisiert wurde. Dies kann mehrere Ursachen haben. Entweder gelangt der Angreifer in Besitz von Informationen, mit deren Hilfe er sich für einen anderen ausgeben kann, oder der Angreifer kann den Autorisierungsprozess des Dienstanbieters so beeinflussen oder umgehen, dass er den unberechtigten Zugriff erlangt. Solche Angriffe werden in die Gruppe der aktiven Angriffe eingestuft.

2.1.2.2 Sicherheitsbezogene Anforderungen

Systeme, die Dienste anbieten, sollten mindestens den folgenden Anforderungen genügen. Diese sollen erreichen, dass einerseits berechtigte Benutzer geschützt werden, andererseits Unbefugte in erster Linie abgewehrt oder im Falle eines gelungenen Angriffs verfolgt werden können.

2.1.2.2.1 Verfügbarkeit *(availability)*

Berechtigte Benutzer sollten immer fähig sein, die angebotenen Dienste in Anspruch zu nehmen. Dienstanbieter sollten Maßnahmen einleiten, damit die Inanspruchnahme von Diensten durch berechtigte Benutzer vom restlichen Datenverkehr unterschieden werden kann und eine höhere Priorität bekommt.

---

[5] Anfang 2000 hatte ein solcher DDoS Angriff die bekannte Website Yahoo lahmgelegt [YAHOODDOS-1].

2.1.2.2.2 Berechtigtes Benutzen *(legitimate use)*

Dienste sollten nur berechtigten Benutzern angeboten werden. Dienstanbieter sollten Mechanismen benutzen, um die berechtigten Benutzer sicher zu identifizieren. Es sollten auch Schutzmaßnahmen eingeführt werden, damit unberechtigte Dritte zu Identifizierungsinformationen (*credentials*) keinen Zugriff bekommen.

2.1.2.2.3 Verfolgbarkeit *(accountability)*

Benutzer, die sicherheitskritische Dienste in Anspruch nehmen, sollen verantwortlich für ihre Aktivitäten sein. Die Inanspruchnahme und der Benutzer sollten, falls benötigt, miteinander verbindbar sein.

2.1.2.3 Sicherungsmechanismen

Im Folgendem werden Mechanismen vorgestellt, die es einerseits ermöglichen, den Server, der den Dienst anbietet, von Angriffen abzuschirmen, und andererseits Methoden bereitstellen, um befugte Benutzer von Unbefugten zu unterscheiden.

2.1.2.3.1 Firewalls

Firewalls werden benutzt, um den eigentlichen Server, der den Dienst anbietet und die Anfragen bedient, vom Netz abzutrennen und abzuschirmen. Ziel einer Firewall ist es, mögliche Angriffe abzuwehren und nicht bis zum Server durchzulassen, jedoch normalen Datenverkehr transparent durchfließen zu lassen.

2.1.2.3.2 Zugriffskontrollsysteme

Zugriffskontrollsysteme wurden entwickelt, um die Autorisierung der möglichen Benutzer durchzuführen. Sie entscheiden, ob jemand, der eine Anfrage an den Server schickt, diese auch bearbeitet zurückbekommt, oder ob die Anfrage abgelehnt wird.

2.1.3 Kommunikation

Das Problem, die Kommunikation zwischen zwei Endpunkten zu schützen, beschäftigte die Wissenschaftler schon seit der Antike. Zuerst wurden verschiedene Verschlüsselungsalgorithmen[6] erfunden, um die Vertraulichkeit zu ermöglichen. Damit zusammen kam dann auch der Wunsch nach Authentizität und Nichtabstreitbarkeit[7].

Die letzten zwei Jahrzehnte brachten dann einen Umbruch. Die verschiedenen Algorithmen wurden Standardisiert, ihre mathematischen Grundlagen wurden tiefgehend erforscht (und zum Teil auch bewiesen), und sie wurden dann in standardisierten **Kryptosystemen** zusammengefasst, deren Funktionalitäten durch standardisierte Schnittstellen benutzt werden können.

---

[6] Verschlüsselungsalgorithmen wurden schon seit langem benutzt. Die bekannteste unter ihnen könnte jedoch der – während des zweiten Weltkriegs benutzte – Enigma sein.

[7] Um Authentizität und Nichtabstreitbarkeit zu gewährleisten, wurden schon seit der Antike verschiedene Verfahren benutzt. Der Wachstempel auf einem Brief oder die sogar heutzutage benutzte Unterschrift sind nur einige der Beispiele.

Es existiert schon eine Reihe von solchen Kryptonsystemen, die sich in der Praxis bewährt haben. **SSH** (*Secure Shell*, [SSH-1]) wird angewendet, um einen gesicherten Zugriff auf entfernte Rechner zu ermöglichen. **SSL** (*Secure Sockets Layer*, [SSL-1] [SSL-2]) und sein Nachfolger, **TLS** (*Transport Layer Security*, [TLS-1]) werden hauptsächlich für sicheres Websurfen und für gesicherte Datenübertragung im Web benutzt, und **WTLS** (*Wireless Transport Layer Security*, [WTLS-1]) wurde für dieselben Ziele für WAP-fähige mobile Geräte konzipiert. Eine der grundlegenden Eigenschaften solcher Kryptosysteme ist, dass sie sich an keinen bestimmten (Verschlüsselungs-, Signier-, Hash, usw.) Algorithmus binden, sondern dass diese **austauschbar**[8] sind, damit das System selbst langzeitig benutzbar bleibt.

Es hat sich in der Praxis gezeigt, dass solche Kryptosysteme als selbständige Protokollschichten implementiert werden müssen. Gründe dafür sind grundsätzlich die folgenden:

- die Entwicklung eines solchen Kryptosystems benötigt von den Implementierern spezielles Wissen und andere Betrachtungsweisen als bei anderen Kommunikationsschichten;
- für die Benutzer ist es bequem und sicher, falls die erforderlichen Algorithmen in einer Schicht zusammengefasst und durch eine standardisierte Schnittstelle anwendbar sind.

2.1.3.1 Bedrohungen

Die folgenden Bedrohungen können im Allgemeinen in Bezug auf die Kommunikation klassifiziert werden. Ein Kommunikationskanal, der zwei Endteilnehmer verbindet, ist meistens diesen Bedrohungen ausgesetzt.

2.1.3.1.1 Abhören *(eavesdropping)*

Benutzen zwei Kommunikationspartner eine Datentransportmethode, die es erlaubt, dass auch Dritte auf sie Zugriff bekommen können, so besteht die Bedrohung eines Abhörens[9]. Wird der Datentransport nicht geschützt[10], so können Dritte die Daten lesen. Angriffe die das Abhören ermöglichen, werden in die Kategorie der passiven Angriffe eingestuft.

2.1.3.1.2 Verändern *(tampering)*

Es gibt auch Datentransportmethoden, die es ermöglichen, dass – unter bestimmten speziellen Bedingungen – Dritte die Daten mitten in der Datentransportroute verändern[11]. So können

---

[8] Ein gutes Beispiel dafür wäre die Benutzung des DES [DES-1] (ein symmetrisches Verschlüsselungsverfahren mit 56 Bit effektiver Schlüssellänge). Anfang der 90-er war DES ziemlich populär, aber mit der Entwicklung der Technologie wurde seine Anwendung immer unsicherer. Doch dank der Austauschbarkeit der Algorithmen in den verschiedenen Sicherheitsprotokollen, konnte man sich zum Beispiel auf 3DES (DES dreimal nacheinander, mit 112 oder 168 Bit effektiver Schlüssellänge) umstellen, ohne dass man das Protokoll selbst austauschen musste.

[9] In einem lokalen Netzwerk (LAN, *local area network*) des Typs Ethernet [ETHERNET-1] können z. B. alle, die auf das Netzwerk Zugriff haben, die Kommunikation der anderen mithören.

[10] In den 80er wurde das TELNET Protokoll [TELNET-1] weitgehend zur entfernten Administration von Rechnern benutzt, obwohl es keine Möglichkeit anbot, die transportierte Daten zu vor Unbefugten zu schützen.

[11] Die heutzutage so oft benutzten Protokolle IP [IP-1] und darüber TCP [TCP-1] haben eine Reihe von Schwachstellen, für denen es auch schon oftmals benutzte Angriffmöglichkeiten existieren. Um diese Schwachstellen zu beheben, wurde IPSEC [IPSEC-1] entwickelt.

Daten ankommen, die nicht abgeschickt wurden, oder es kann auch vorkommen, dass diejenigen, die versandt wurden, nicht den Empfänger erreichen. Solche Angriffe gehören in die Kategorie der aktiven Angriffe.

2.1.3.1.3 Wiedereinspielen *(replay)*

Eine weitere Bedrohung ist es, falls die Möglichkeit besteht, dass Nachrichten von einem Teilnehmer den anderen – zeitlich verschoben – mehrmals erreichen. Der Empfänger kann dies als zwei selbständige und verschiedene Nachrichten interpretieren. Diese Angriffe werden als aktive Angriffe eingestuft.

2.1.3.1.4 Maskerade *(impersonation)*

Falls ein Dritter sich für jemand anderes, als er selbst ist, ausgeben kann, so wird derjenige, mit dem er eine Kommunikation aufbaut, in der Identität des Kommunikationspartners belogen und so einem Angriff ausgesetzt. Solche Angriffe werden in die Gruppe der aktiven Angriffe eingeteilt.

2.1.3.2 Sicherheitsbezogene Anforderungen

Für die Kommunikation können in erster Linie die folgenden Anforderungen definiert werden. Diese sollen die oben genannten Bedrohungen eliminieren.

2.1.3.2.1 Vertraulichkeit *(confidentiality)*

Vertraulichkeit bedeutet, dass die Daten, die durch einen Kommunikationskanal fließen, nur von Berechtigten gelesen und richtig interpretiert werden können. Unbefugte dürfen die Daten nicht lesen können, auch nicht, wenn sie Zugriff auf den Kanal selbst erlangen. Diese Anforderungen können durch **Verschlüsselungsalgorithmen** erfüllt werden.

2.1.3.2.2 Integritätssicherung *(data integrity)*

Werden bestimmte Protokolle bzw. Netzwerke benutzt, so kann es nicht verhindert werden, dass Dritte die Daten verfälschen. Es ist aber zu gewährleisten, dass eine solche Veränderung der Daten zuverlässig vom Empfänger erkannt wird. Diese Funktionalität wird meistens mit Hilfe **kryptographischer Prüfsummen** erbracht.

2.1.3.2.3 Authentisierung *(authentication)*

Während des Aufbauens eines Kommunikationskanals müssen sich beide Parteien in der Identität des Anderen sicher sein. Werden sensitive Informationen übermittelt, so ist es äußerst wichtig, dass diese nicht in die Hände von Unbefugten gelangen. Die zwei meistverbreiteten Methoden für Authentisierung sind einerseits **Passworte**, andererseits digitale Signaturen während **Challenge-Response**-Verfahren.

2.1.3.2.4 Nichtabstreitbarkeit *(non-repudiation)*

Vor allem in geschäftlicher, insbesondere finanzieller Kommunikation wird die Nichtabstreitbarkeit der Nachrichten vorausgesetzt. Die bestgeeignete Methode für das Erreichen der Nichtabstreitbarkeit ist die **digitale Signatur**.

2.1.3.3 Methoden

Im Folgenden werden die grundlegendsten Methoden beschrieben, die zur Sicherung einer Kommunikation benutzt werden können. Heute angewandte Sicherheitsschichten bieten zumeist alle diese Methoden an. [HAC-1] und [SCHNEIER-1] behandeln diesen Bereich ausführlich.

2.1.3.3.1 Verschlüsselung *(ciphering)*

Verschlüsselungsverfahren werden benutzt, um die Vertraulichkeit der Nachrichten zu gewährleisten. Die sinntragende Nachricht wird vom Absender mit Hilfe eines Schlüssels verschlüsselt. Die so entstehende Bitfolge ist für einen Dritten uninterpretierbar und damit vor ihm sicher. Die verschlüsselte Nachricht erreicht den Empfänger, der sie dann mit einem – möglicherweise anderen – Schlüssel entschlüsselt und dann versteht.

2.1.3.3.1.1 Symmetrische Verschlüsselung

Um ein symmetrisches Verschlüsselungsverfahren zu benutzen, brauchen beide Seiten der Kommunikation denselben geheimen Schlüssel *(secret key)*. Der selbe geheime Schlüssel wird dann sowohl bei der Verschlüsselung als auch bei der Entschlüsselung benutzt. Deswegen ist der Schlüsselaustausch ein zentrales Thema bei solchen Verschlüsselungsverfahren.

Symmetrische Verschlüsselungsalgorithmen können in zwei Untergruppen eingeteilt werden:

- **Blockchiffren** bearbeiten einen größeren Datenblock (z.B. 64 Bits) auf einmal. Die bekanntesten Vertreter sind **DES** [DES-1], **3DES** [DES-1] und **AES** [AES-1].
- **Stromchiffren** arbeiten auf dem Niveau der Grundeinheiten des Datentransports (z.B. auf einem Bit oder einem Byte). Die bekanntesten Vertreter sind **A5** (für die Sicherung der GSM Kommunikation) und **RC4**/**Arcfour** [RSA-1] [ARCFOUR-1] (für die Sicherung der Kommunikation im Internet).

2.1.3.3.1.2 Asymmetrische Verschlüsselung

Bei asymmetrischer Verschlüsselung hat jede Partei zwei Schlüssel, einen geheimen *(private key)*, der vor allen anderen Personen geheim gehalten werden muss, und einen öffentlichen *(public key)*, der den Kommunikationspartnern bekannt sein muss. Daten, die mit einem öffentlichen Schlüssel verschlüsselt sind, können nur mit dem dazugehörenden geheimen Schlüssel entschlüsselt werden, und dies gilt auch umgekehrt. Falls man also jemandem eine verschlüsselte Nachricht schicken will, so muss die ursprüngliche Nachricht nur mit dem öffentlichen Schlüssel des Empfängers verschlüsselt werden und so ist es schon gewährleistet, dass niemand außer dem Empfänger die verschlüsselte Nachricht lesen kann.

Ein wesentlicher Vorteil der asymmetrischen Verfahren ist, dass geheime Schlüssel nicht ausgetauscht werden müssen, jeder muss seinen geheimen Schlüssel verbergen. Für die Kommunikation muss nur die authentische Übertragung des öffentlichen Schlüssels erreicht werden. Andererseits haben solche Verfahren den Nachteil, dass sie gegenüber symmetrischen Verschlüsselungsverfahren wesentlich langsamer sind und die Schlüssellänge – bei vergleichbarer Sicherheit – auch wesentlich größer ist.

Die zwei bekanntesten Vertreter sind der **RSA**-Algorithmus [RSA-2] und der **EC**-Algorithmus *(Elliptic Curve)* [EC-1].

2.1.3.3.2 Zertifikate *(certificates)*

Zertifikate werden benutzt, um die Identität eines Teilnehmers der Kommunikation zu beweisen. Sie enthalten sowohl den Namen des Subjekts als auch seinen öffentlichen Schlüssel. Zertifikate werden von **Zertifizierungsstellen** (*CA, Certification Authority*) ausgestellt, und um ihre Authentizität beweisen zu können, enthalten sie eine digitale Signatur der CA über die vorher genannten Daten. Das meistverbreitete Format eines Zertifikats wird durch den **X.509** Standard [X509-1] der ITU beschrieben.

2.1.3.3.3 Hashen *(hashing)*

Hashfunktionen sind Einwegfunktionen, die im Allgemeinen selten selbständig benutzt werden, aber ihre Ausgabedaten werden als Eingabe anderer kryptographischer Verfahren (z.B. für die Berechnung kryptographischer Prüfsummen und digitaler Signaturen) verwendet. Das Definitionsbereich einer Hashfunktion ist die Menge aller möglichen Bitfolgen (Texte), und der Wertebereich ist die Menge aller möglichen Bitfolgen einer bestimmten, für die Hashfunktion charakteristischen, Länge. Die folgenden Eigenschaften sind für eine Hashfunktion wichtig:

- Der Hashwert eines bestimmten Textes ist aus dem Text vergleichsweise einfach und schnell berechenbar.
- Aus dem Hashwert ist der ursprüngliche Text nicht berechenbar.
- Einen Text zu finden, dessen Hashwert eine gegebene Bitfolge ist, ist schwierig und es existieren keine effektiven – bekannten – Algorithmen dafür.

Die zwei bekanntesten Hashfunktionen sind **SHA-1** mit einem Hashwert der Länge 160 Bits [SHS-1] und **MD5** mit einem Hashwert der Länge 128 Bits [MD5-1].

2.1.3.3.4 Kryptographische Prüfsumme *(MAC, Message Authentication Code)*

Kryptographische Prüfsummen werden benutzt, um die Integrität einer Nachricht zu sichern. Mit Hilfe einer Hashfunktion und eines geheimen Schlüssels wird für die zu sendende Nachricht eine Bitfolge, die kryptographische Prüfsumme errechnet. Diese wird ans Ende der Nachricht angehängt. Der Empfänger berechnet mit Hilfe derselben Hashfunktion und desselben geheimen Schlüssels die MAC der empfangenen Nachricht. Falls diese mit der empfangenen MAC übereinstimmt, dann kann der Empfänger in der Integrität der Nachricht sicher sein, denn ein möglicher Angreifer, der die Nachricht verändern kann, kann die MAC ohne dem geheimen Schlüssel nicht berechnen. Der meistverbreitete Vertreter kryptographischer Prüfsummen ist **HMAC** [HMAC-1], wobei je nach benutzten Hashfunktion zwischen **HMAC-HD5**, **HMAC-SHA1** usw. unterschieden wird.

2.1.3.3.5 Zeitstempel *(timestamp)*

Zeitstempel werden benutzt um das Wiedereinspielen der gesendeten Nachrichten zu verhindern. Der Absender hängt ans Ende der Nachricht – jedoch nicht vor der kryptographischen Prüfsumme – den Zeitpunkt, den sogenannten **Zeitstempel**, des Absendens an, und schickt die so entstandene Nachricht ab. Der Empfänger vergleicht den Zeitstempel mit seiner eigenen Uhr, und falls die Differenz unter einem bestimmten Grenzwert liegt, so wird die Nachricht akzeptiert, andernfalls als ein Duplikat verworfen.

2.1.3.3.6 Digitale Signatur *(digital signature)*

Digitale Signaturen werden benutzt, um sicherzustellen, dass der Absender einer Nachricht die Nachricht nicht abstreiten kann. Für die Erstellung digitaler Signaturen wird auch ein geheimer Schlüssel *(private key)* und zu deren Verifikation der öffentliche Schlüssel *(public key)* benutzt. Für diese gelten dieselben Bemerkungen wie für die Schlüssel bei einem asymmetrischen Verschlüsselungsverfahren.

Um die digitale Signatur zu erstellen, wird zuerst die Nachricht mit einer Hashfunktion in einen Hashwert umgewandelt. Dann wird der Hashwert je nach Signaturtyp entweder mit einem asymmetrischen Verschlüsselungsverfahren verschlüsselt oder mit einem Signaturalgorithmus kodiert. Die bekanntesten Vertreter der ersten Gruppe sind **MD5withRSA** (Hashwert des MD5 mit RSA verschlüsseln) und **SHA1withRSA** (Hashwert des SHA1 mit RSA verschlüsseln), der bekannteste Vertreter der zweiten Gruppe ist **DSA** [DSA-1].

Da zur Erstellung der Signatur der geheime Schlüssel des Absenders erforderlich ist und da nur der Absender im Besitz des geheimen Schlüssels ist, kann aus dem Vorhandensein einer digitalen Signatur am Ende einer Nachricht geschlossen werden, dass der Absender die Nachricht erstellt hat.

2.1.3.4 SSH2

Um die oben genannten Methoden und Techniken an einem konkreten Beispiel zu veranschaulichen, wird in diesem Abschnitt die Funktionsweise von SSH2 (*Secure Shell*) [SSH-1] näher erläutert.

SSH2 ist ein Client-Server- Sicherheitsprotokoll, bestehend aus drei Unterprotokollen (*Transport Layer Protocol*, *Authentication Protocol* und *Connection Protocol*), die die meisten der im vorigen Abschnitt vorgestellten Techniken verwenden.

Während des Aufbaus einer SSH2-Verbindung wird zuerst das Transport Layer Protocol gestartet, nach dessen Fertigwerden das Authentication Protocol, und als letztes das Connection Protocol. Dieses ermöglicht den gewünschten geschützten und transparenten Datentransfer für darüberliegende Schichten.

2.1.3.4.1 SSH2 Transport Layer Protocol

Das Transport Layer Protocol bietet die folgenden Dienste für die anderen Unterprotokolle an:

- Sicherung der Kommunikation durch symmetrische Verschlüsselung (gegen Abhören), kryptographische Prüfsummen (gegen Veränderung) und Sequenznummern (gegen Wiedereinspielen)
- Authentisierung des Servers mittels digitaler Signaturen (gegen Maskerade).

Um diese Dienste anbieten zu können, werden Nachrichten zwischen den beiden Seiten ausgetauscht (die benötigten Nachrichtenpakete werden gemäß der später erläuterten Paketbildungsmethode erstellt, es werden aber keine Komprimierung, Verschlüsselung und kryptographische Prüfsummen benutzt).

Diese Nachrichten werden benötigt, um das Folgende abzuwickeln:

- Client und Server vereinbaren die Schlüsselaustausch-, Komprimier-, MAC- und Verschlüsselungs-Algorithmen;
- Client und Server tauschen mittels des vereinbarten **Schlüsselaustauschalgorithmus** (*key exchange algorithm*) – der meistens auf asymmetrischen Verschlüsselungsverfahren und digitalen Signaturen basiert – kryptographische Daten untereinander aus, aus denen dann jede Seite für sich selbst die verschiedenen geheimen Schlüssel (für MAC und Verschlüsselung) ableitet.

Das SSH2 Transport Layer Protocol schickt Nachrichten nur in Form von Paketen, die gemäß der folgenden Methode erstellt werden, ab.

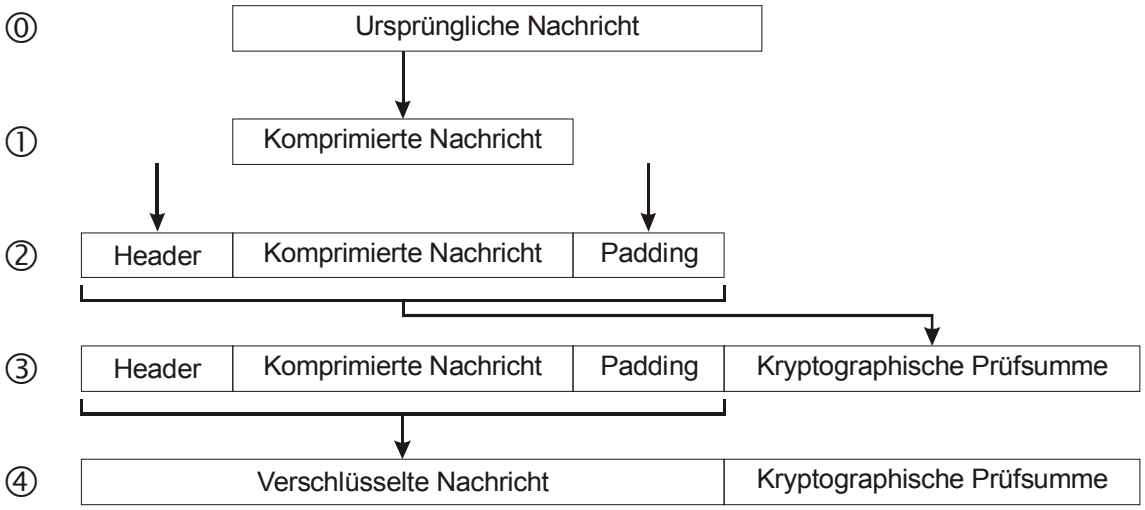


Abbildung 2 – **Paketbildung des SSH2 Transport Layer Protocol**

Die einzelnen Phasen beinhalten die folgende Transformationen:

⓪ Die ursprüngliche Nachricht wird vom Anwender des SSH2 Transport Layer Protocol zusammengestellt.

① Falls Komprimierung (z. B. ZLIB [ZLIB-1] [ZLIB-2]) vereinbart wurde, wird die ursprüngliche Nachricht komprimiert.

② Header-Daten (Paketlänge, Paddinglänge) werden am Anfang und Padding (Auffüllung mit zufälligen Bytes, damit die Länge des zu verschlüsselnden Blocks eine Mehrfache der Blockgröße des Verschlüsselungsalgorithmus wird) am Ende der komprimierten Nachricht angehängt.

③ Aus den Header-Daten, der komprimierten Nachricht und dem Padding wird mit Hilfe der für jedes Paket erhöhten Sequenznummer die kryptographische Prüfsumme (z. B. HMAC-SHA1) ermittelt und nach dem Padding angehängt. Die Sequenznummer wird nicht durch den Kanal transportiert, sie wird nur bei der Berechnung der kryptographischen Prüfsumme benutzt.

④ Schließlich werden die Header-Daten, die komprimierte Nachricht und der Padding mit dem Verschlüsselungsalgorithmus (z. B. 3DES) verschlüsselt.

2.1.3.4.2 SSH2 Authentication Protocol

Nachdem das Transport Layer Protocol erfolgreich initialisiert wurde, kann das Authentication Protocol gestartet werden. Dieses dient dazu, den Client in der Kommunikation zu authentisieren. Dieses Unterprotokoll benutzt die Dienste des Transport Layer Protocol, das heißt, dass die benötigten Nachrichten bevor sie abgeschickt werden, durch das Transport Layer Protocol noch transformiert werden.

Welche Möglichkeiten zur Authentisierung des Clients benutzt werden können, wird vom Server entschieden. Die drei meistbenutzten Methoden sind die Folgenden:

- "`none`": Der Client wird nicht authentisiert. Der Server erlaubt jedem, das Connection Protocol zu starten.
- "`password`": Der Client wird mittels eines **Benutzernamens** (*username*) und eines dazugehörenden **Passworts** (*password*) authentisiert.
- "`publickey`": Der Client wird mittels eines Benutzernamens und durch einer digitalen Signatur authentisiert. Diese wird mit dem zum Benutzername gehörenden geheimen Schlüssel erstellt. Dieser geheimer Schlüssel muss zum dem Server bekannten öffentlichen Schlüssel passen.

2.1.3.4.3 SSH2 Connection Protocol

Nachdem das Authentication Protocol beendet wurde, kann das Connection Protocol gestartet werden. Dieses Unterprotokoll bietet im wesentlichen zwei Dienste an:

- Die Möglichkeit mehrere gesicherte und transparente Datenkanäle über die angebotene Schnittstelle zu benutzen.
- Flusskontrolle der Datenströme.

Das Connection Protocol ermöglicht es, **Kanäle** (*channels*) in der SSH2-Kommunikation zu öffnen. Jeder Kanal ist in sich selbständig, und benutzt das Transport Layer Protocol, um die durchfließenden Daten zu transportieren. Oberhalb von SSH2 liegende Schichten können diese Kanäle zum Datentransfer benutzen.

Kanäle haben die folgenden Eigenschaften:

- Kanäle können von beiden Seiten geöffnet werden (falls die andere Seite dem Kanalöffnungsantrag zustimmt) und ermöglichen bidirektionalen Datentransfer.
- Der Datenstrom, der durch einen Kanal fließt, wird von der Absenderkanalinstanz in Pakete aufgeteilt und von der Empfängerkanalinstanz wieder in einen Datenstrom zusammengestellt.
- Um **Flusskontrolle** zu ermöglichen, kann für jeden Kanal jede Seite für die von der anderen Seite gesendeten Daten eine maximale Paketlänge und eine maximale Sendefenstergröße bestimmen.
  - Pakete, die die Kanalinstanz abschickt, können nicht größer sein als die von der anderen Seite vorgegebene maximale Paketlänge.
  - Mit jedem Kanal wird auf beiden Seiten ein aktuelles Sendefenster assoziiert. Dieses Sendefenster wird während der Initialisierung des Kanals auf die maximale Sendefenstergröße eingestellt. Bei jedem Abschicken eines Pakets wird das aktuelle Sendefenster um die Länge des abgeschickten Pakets verkleinert. Falls das

Sendefenster aufgebraucht ist, kann kein Datenpaket mehr abgeschickt werden. Jede Seite kann jedoch der anderen Seite signalisieren, dass das Sendefenster um einen bestimmten Betrag erweitert werden kann.

2.2 Anonymität

Unter **Anonymität** versteht man das Verbergen der Identität während eines Kommunikationsvorgangs, an dem mehrere Parteien teilnehmen. Unter **Privacy**[12] versteht man das Verbergen persönlicher Daten während der Kommunikation. es hat sich gezeigt, dass der Wunsch nach dem höchstmöglichen Niveau an Anonymität und Privacy mehr an Bedeutung gewinnt. Genau deswegen wird auch der Schutz der persönlichen Daten immer wichtiger; es gibt auch gesetzliche Bestimmungen dafür [BDSG-1].

Anonymität kann als ein Unterbereich von Privacy betrachtet werden. AEP beschränkt sich überwiegend auf den Problemkreis Anonymität und behandelt nur Methoden, deren primäre Aufgabe im Verbergen der Identität besteht (obwohl sie möglicherweise auch zur Lösung von Problemen im Bereich Privacy benutzt werden könnten).

2.2.1 Notwendigkeit von Anonymität – Pro und Kontra

Es gibt keine einheitliche Beurteilung, ob Anonymität etwas Gutes oder etwas Böses ist. Fokussiert man auf die Vorteile, die durch Anonymität entstehen, wie die Beiträge zu den verschiedenen persönlichen Freiheiten, könnte man denken, dass diese die Nachteile überwiegen. Betrachtet man Straftaten, wo Verbrecher wegen ihrer Anonymität nicht zur Rechenschaft gezogen werden können, könnte man argumentieren, dass Anonymität deswegen verboten werden müsste.

Im Folgenden werden die Vor- und Nachteile ausführlicher diskutiert und schließlich die Position von AEP vorgestellt.

2.2.1.1 Pro

Mit den immer größeren Datenbanken, die staatlichen Einrichtungen und multinationalen Konzernen zur Verfügung stehen, und den immer besseren Suchmaschinen, die das Internet ohne Unterbrechung durchsuchen, werden immer mehr – persönliche – Daten über einen jeden zusammengesammelt und mit modernen Techniken verwaltet. Nachrichten, die man in eine *newsgroup* abgeschickt hat, Daten, die ein Familienmitglied über seine Verwandten im WWW veröffentlicht hat, Besuchergewohnheiten, die die verschiedenen Portale speichern – all dies kann dann zu einem **Profil**[13] zusammengestellt werden, ohne dass man davon etwas weiß.

Eine der Möglichkeiten, sich gegen das Erstellen eines Profils zu wehren, ist die Anonymität. Falls man die Identität des Subjekts, das beobachtet wird, nicht kennt, dann ist die zusammengesammelte Information auch nicht "gegen" es verwendbar. So wird Anonymität zu einer der Schlüsseltechniken beim Schutz der Privatsphäre im Internet.

---

[12] Entspricht etwa "**Schutz der Privatsphäre**".

[13] [FROOMKIN-2] behandelt ausführlich die Erstellung von Profils (*profilen*) und auch mögliche Gegenmaßnahmen.

Ein anderer Vorteil ist, dass man anonym viel leichter über Themen spricht, die sonst peinlich wären, wie z. B. über die politische Überzeugung, über Geschlechtskrankheiten oder über Alkoholismus. Anonymität kann somit zur allgemeinen Aufklärung beitragen und helfen, gesellschaftliche Ziele zu erreichen.

2.2.1.2 Kontra

Mit dem Wissen, dass man anonym ist, und deswegen die begangenen Straftaten nicht bestraft werden können, ist viel einfacher diese zu begehen, die Hemmschwelle zum Begehen von Straftaten wird viel geringer.

Urheberrechtlich geschützte Programme, Filme, Musik oder Bücher werden auf so genannten *warez-sites* ohne Genehmigung der Besitzer veröffentlicht und dann auch von dort heruntergeladen.

So genannte *spam*-E-Mails, die den Empfänger auf Dienste aufmerksam machen sollen, werden jedem Besitzer einer E-Mail-Adresse anonym einem regelmäßig zugeschickt. Gibt man die eigene E-Mail-Adresse unvorsichtig an der falschen Stelle an, so wird man von solchen belästigenden Nachrichten überflutet.

Angriffe auf Dienstanbieter, die im vorigen Kapitel schon erläutert wurden, werden in der Regel anonym verübt. Angreifer wenden die neuesten Techniken an, um ihre Identität zu verbergen oder sich als jemand anderes auszugeben.

2.2.1.3 Position von AEP

Die Meinung des Autors ist, dass Anonymität in sich selbst weder etwas Gutes, noch etwas Böses ist. Die Beurteilung hängt lediglich von der Art der Benutzung ab.

Um zu verhindern, dass Anonymität allein wegen ihrer Nachteile nicht ermöglicht wird, müssen die Möglichkeiten entwickelt, erforscht, standardisiert und gegebenenfalls auch gesetzlich reguliert werden. Es ist äußerst wichtig, dass Missbrauch entweder nicht ermöglicht wird oder mindestens die Möglichkeit besteht, ihn zu verfolgen. Deswegen ist die Zielsetzung von AEP auch, eine **anonymen Autorisierungsprozess** zu beschreiben, wobei jedes der beiden Wörter eine wichtige Rolle spielt: Einerseits wird Anonymität ermöglicht, andererseits wird aber auch eine Autorisierung durchgeführt.

Eine maßgebende Organisation im Themenbereich Anonymität ist die Gruppe der **Cypherpunks**. Ihr Kredo [CYPHERP-1] könnte man so zusammenfassen: Anonymität durch Technologie, nicht durch Gesetzgebung. AEP schließt jedoch die Gesetzgebung nicht aus, behandelt selbst allerdings nur die technologischen Aspekte.

2.2.2 Anonymitätsniveaus

In diesem Kapitel werden fünf Niveaus der Anonymität – beginnend mit dem schwächsten – beschrieben.

Für AEP sind nur die Niveaus 3 und 4 von Interesse. Diese erfüllen nämlich die Kriterien, die vorher schon beschrieben wurden. Für diese beiden Niveaus kann durch AEP ein einheitlicher anonymer Autorisierungsprozess beschrieben werden.

Das Niveau 1 bietet keine Anonymität, und ist deshalb uninteressant im Bezug auf AEP. Obwohl Niveau 2 schon einige Aspekte der Anonymität mit sich bringt, so ist die effektive Anonymität ziemlich gering, und immer noch uninteressant für AEP. Niveau 5 erfüllt die

Voraussetzung nicht, dass eine anonyme Autorisierung durchgeführt werden muss, und wird deswegen in AEP nicht behandelt.

2.2.2.1 Keine Anonymität [Niveau 1]

Das schwächste Niveau an Anonymität ist ihr **vollständiges Fehlen**. Während der Kommunikation ist die Identität des Subjekts bekannt. In diesem Fall entstehen bei der Verfolgung eines Missbrauchs keine Probleme durch Anonymität.

2.2.2.2 Zurückverfolgbares Subjekt [Niveau 2]

Das Subjekt besitzt bereits eine zweite Identität, mit der es normalerweise während der Kommunikation identifiziert werden kann. Somit bleibt seine echte Identität im Allgemeinen verborgen. Diese zweite Identität kann einerseits ein **Deckname**[14] oder andererseits eine **Pseudoidentität**[15] sein.

Damit wird der Kommunikationsvorgang schon in zwei Phasen unterteilt. Zuerst muss das Subjekt eine zweite Identität bekommen. Dies geschieht während des **Initialisierungsprozesses.** Erst dann kann das Subjekt die Dienstleistung in Anspruch nehmen, wobei es **anonym autorisiert** werden muss.

Jedoch kann der Dienstanbieter die echte Identität des Subjekts zu jeder Zeit und uneingeschränkt erfahren. Die Zuordnung zwischen der echten und der zweite Identität und die Dienstleistung werden von derselben Partei ausgeführt, somit kann der Dienstanbieter im Falle eines Missbrauchs oder nur aus puren wirtschaftlichen Interesse die echte Identität des Subjekts erfahren.

Ein typisches Beispiel wäre der Benutzername im Rechenzentrum der Universität. Der Benutzername "*um37*" ist in sich anonym, jedoch kann der Dienstanbieter – und in diesem Fall andere Benutzer auch – die Identität zum Beispiel mit dem UNIX-Kommando "`finger um37`" schnell erfahren.

2.2.2.3 Begrenzt zurückverfolgbares Subjekt [Niveau 3]

Der Unterschied zum vorherigen Niveau ist, dass die Dienstleistung und die Zuordnung zwischen der zweite Identität und der echten Identität nicht mehr von derselben Partei durchgeführt werden. Das Eine wird noch immer vom Dienstanbieter erbracht, für das Andere ist jedoch die hier erstmals eingeführte **Anonymitätsinstanz**[16] zuständig.

Es ist wichtig zu bemerken, dass unter normalen Umständen der Dienstanbieter nur die zweite Identität des Subjekts kennt. Im Falle eines Missbrauchs kann jedoch der Dienstanbieter mit Hilfe der Anonymitätsinstanz die echte Identität erfahren.

---

[14] Der Deckname ist ein anderer Name als der echte, jedoch von dem Subjekt selbst ausgewählt, und deswegen besitzt er meistens eine persönliche Bedeutung für seinen Besitzer.

[15] Die Pseudoidentität ist ein meistens keine Bedeutung tragendes Zufallswort (wie zum Beispiel "*um37*"), das dem Subjekt vom Dienstanbieter oder von der Anonymitätsinstanz zugewiesen wird.

[16] Die Anonymitätsinstanz ist eine Art vertrauenswürdige dritte Partei (*trusted third party*), die die echten und zweiten Identitäten in einer Datenbank verwaltet.

2.2.2.4 Nicht zurückverfolgbares Subjekt [Niveau 4]

Wird dieses Niveau an Anonymität in der Kommunikation erreicht, so hat weder der Dienstanbieter noch die Anonymitätsinstanz eine Möglichkeit, die Identität des Subjekts zu erfahren, während dieses eine Dienstleistung in Anspruch nimmt.

Damit jedoch die Autorisation des Subjekts geprüft werden kann – natürlich ohne seine echte Identität zu wissen –, muss das Subjekt im Besitz einer speziellen **Erlaubnis** sein. Solche Erlaubnisse werden von der Anonymitätsinstanz[17] ausgestellt.

Eine wichtige Anforderung an die Erlaubnis ist, dass keine Möglichkeit bestehen darf, sie der echten Identität des Subjekts zuzuordnen. Die Tatsache, dass das Subjekt eine Erlaubnis bekommen hat, eine bestimmte Art von Dienstleistungen in Anspruch zu nehmen, muss aber gespeichert werden.

2.2.2.5 Volle Anonymität [Niveau 5]

Bei voller Anonymität ist die Identität des Subjekts während der Inanspruchnahme der Dienstleistung nicht bekannt. Diese wird jedoch auch nicht benötigt, denn der Dienstanbieter betreibt einen völlig öffentlichen Dienst, und braucht keine Autorisation.

Zwei Unterarten können unterschieden werden:

- [Niveau 5a]: Der Dienstanbieter speichert einige Daten (Beschreibung des Dienstes, Zeit der Inanspruchnahme, Daten über den Weg zum Subjekt). Typisches Beispiel wäre das Surfen im Web, wobei die meisten Dienstanbieter den Names der besuchten Seiten, den Zeitpunkt und die IP-Adresse des Subjekts speichern.
- [Niveau 5b]: Der Dienstanbieter speichert keine Daten. Beispiel: das starten eines Programms unter Windows 9x.

2.2.3 Anonymitätsverfahren

Die Anonymitätsmethoden können im Allgemeinen in die Kategorie der **PET**[18] (*Privacy Enhancing Technology*) eingestuft werden. In diesem Abschnitt werden für jede Anonymitätsmethode die in den verschiedenen Phasen der Kommunikation teilnehmenden Parteien vorgestellt, die Anforderungen beschrieben, und schließlich werden die Etappen selbst erläutert.

Die hier vorgestellten Anonymitätsmethoden brauchen Dienste einer Sicherheitsschicht um die gewünschte Funktionalität zu erbringen.

2.2.3.1 Anonymitätsniveau des begrenzt zurückverfolgbaren Subjekts

2.2.3.1.1 Methode der Pseudoidentität

Diese Methode wird generell in Zugriffskontrollsystemen benutzt.

---

[17] In diesem Fall wird die Rolle der Anonymitätsinstanz ein bisschen erweitert, sie arbeitet jetzt als eine Art Erlaubniserteilungsinstanz (*permission authority*).

[18] Verschiedene PETs und ihre Anwendungsmöglichkeiten werden in [PET-1] [PET-2] [SZEKELY-1] weitgehend behandelt.

2.2.3.1.1.1 Teilnehmer

Drei verschiedene Rollen können unterschieden werden:

- **Subjekt**: nimmt Dienstleistungen anonym, mit Hilfe seiner falschen Identität, die er von der Anonymitätsinstanz bekommen hat, in Anspruch;
- **Dienstanbieter**: bietet die Dienstleistungen den verschiedenen Subjekten an, obwohl er normalerweise nur ihre falsche Identität kennt;
- **Anonymitätsinstanz**: verwaltet die Zuordnung zwischen falschen und echten Identitäten, und führt die anonyme Autorisierung des Subjekts für den Dienstanbieter durch.

2.2.3.1.1.2 Anforderungen

Die folgenden Anforderungen sichern dem Subjekt das Erreichen des gewollten Anonymitätsniveaus:

- nur die Anonymitätsinstanz kann die echte zu der falschen Identität zuordnen;
- der Dienstanbieter kennt nur die falsche Identität des Subjekts;
- die Anonymitätsinstanz autorisiert das Subjekt für den Dienstanbieter;
- im Falle eines Missbrauchs kann der Dienstanbieter die echte Identität des Subjekts mit Hilfe der Anonymitätsinstanz erfahren.

2.2.3.1.1.3 Beschreibung der Methode

Die folgenden Phasen können unterschieden werden:

- Während des **Initialisierungsprozesses** meldet sich das Subjekt bei der Anonymitätsinstanz an. Während dessen wird das Subjekt identifiziert, bekommt die falsche Identität, und die Anonymitätsinstanz speichert die Zuordnung zwischen der echten und falschen Identität bzw. Daten zum anonymen Autorisierungsprozess. Während dieser Phase wird auch bestimmt, welche Berechtigungen das Subjekt bei den verschiedenen Dienstanbietern hat. Der Initialisierungsprozess wird bei jedem Subjekt nur einmal durchgeführt.

  Der Initialisierungsprozess benötigt von der Sicherheitsschicht die Authentisierung beider Seiten, Integritätssicherung und die Vertraulichkeit der Kommunikation.

- Die nächste Phase ist die **anonyme Autorisierung** des Subjekts für die Inanspruchnahme der Dienstleistung beim Dienstanbieter. Das Subjekt teilt dem Dienstanbieter seine falsche Identität, welche Dienste es in Anspruch nehmen will, und Daten für die anonyme Authentisierung mit. Der Dienstanbieter leitet diese weiter zu der Anonymitätsinstanz, die einerseits die Authentisierung durchführt, und andererseits darauf basierend die Autorisation ermittelt. Falls zum Schluss die Anonymitätsinstanz die Inanspruchnahme des Dienstes bewilligt, so kann die Dienstleistung erfolgen.

  Dieser Prozess benötigt von der Sicherheitsschicht zwischen Subjekt und Dienstanbieter die Authentisierung des Dienstanbieters, Integritätssicherung und Vertraulichkeit der Kommunikation; zwischen Anonymitätsinstanz und Dienstanbieter beidseitige Authentisierung, Integritätssicherung und Vertraulichkeit der Kommunikation.

Im Falle eines Missbrauchs kann der Dienstanbieter zum Beispiel durch ein gerichtliches Verfahren mit Hilfe der Anonymitätsinstanz die echte Identität des Subjekts erfahren und rechtliche Schritte einleiten.

2.2.3.2 Anonymitätsniveau des nicht zurückverfolgbaren Subjekts

2.2.3.2.1 Methode der blinden Unterschrift

Die Methode wurde ursprünglich von **David Chaum** beschrieben [CHAUM-1] [CHAUM-2] [CHAUM-3] und bildet die Grundlage der anonymen Bezahlungsmöglichkeit **DigiCash** [DIGICASH-1] im Internet. Bei DigiCash werden die Rollen so konkretisiert, dass das Subjekt ein anonymer Einkäufer ist, der Dienstanbieter bleibt der selbe oder wird zu einem Geschäft, und die Rolle der Anonymitätsinstanz wird von einem Bank übernommen. Doch die ursprüngliche Idee kann noch um vieles mehr erweitert werden. Zum Beispiel kann diese Methode zu anonymen elektronischen Abstimmungen benutzt werden (das Subjekt ist der Wähler, der Dienstanbieter ist dann die "Urne", und die Anonymitätsinstanz ist derjenige, der die Stimmzettel ausgibt).

2.2.3.2.1.1 Plastische Veranschaulichung

Dieser Abschnitt versucht, die Methode der blinden Unterschrift plastisch zu veranschaulichen. Dazu wird eine Analogie in der normalen Welt beschrieben. Die eigentliche, digitale und kryptographisch basierte Methode wird ausführlich im Abschnitt 4.3.3.3 behandelt.

Für die Veranschaulichung wird das Beispiel eines Kaufvorgangs benutzt. Die Teilnehmer sind der Einkäufer (das Subjekt), die Bank (die Anonymitätsinstanz) und der Verkäufer (der Dienstanbieter). In diesem Szenario erwirbt der Einkäufer zuerst eine Banknote (siehe Abbildung 3) und benutzt diese dann zum anonymen Einkauf.

① Der Einkäufer nimmt eine Blankobanknote, ein Kohlepapier und einen Umschlag.

② Der Einkäufer legt das Indigopapier auf die Banknote und steckt beide in den Umschlag.

③ Der Einkäufer schließt den Umschlag und schickt diesen der Bank, mit der Bitte, den Umschlag mit einem bestimmten Betrag (z. B. EUR 1) abzustempeln.

④ Die Bank bekommt einen geschlossenen Umschlag und die Information über den gewünschten Betrag und über den Einkäufer.

⑤ Die Bank sucht den zu dem gewünschten Betrag passenden Stempel aus.

⑥ Die Bank stempelt den Umschlag ab (wegen des Kohlepapiers wird dadurch auch die Blankobanknote abgestempelt), zieht den gewünschten Betrag vom Konto des zukünftigen Einkäufers ab und schickt den abgestempelten Umschlag zurück.

⑦ Der zukünftige Einkäufer erhält den abgestempelten Umschlag.

⑧ Der zukünftige Einkäufer überprüft den Umschlag (ob er nicht geöffnet wurde) und öffnet ihn. Die Bank darf den Umschlag nicht öffnen, denn dann könnte sie die Banknote so markieren (eventuell so, dass der Einkäufer es nicht bemerkt), dass später zu der Banknote die Identität des Einkäufers zugeordnet werden könnte.

⑨ Der zukünftige Einkäufer entnimmt aus dem Umschlag die von der Bank abgestempelte Banknote.

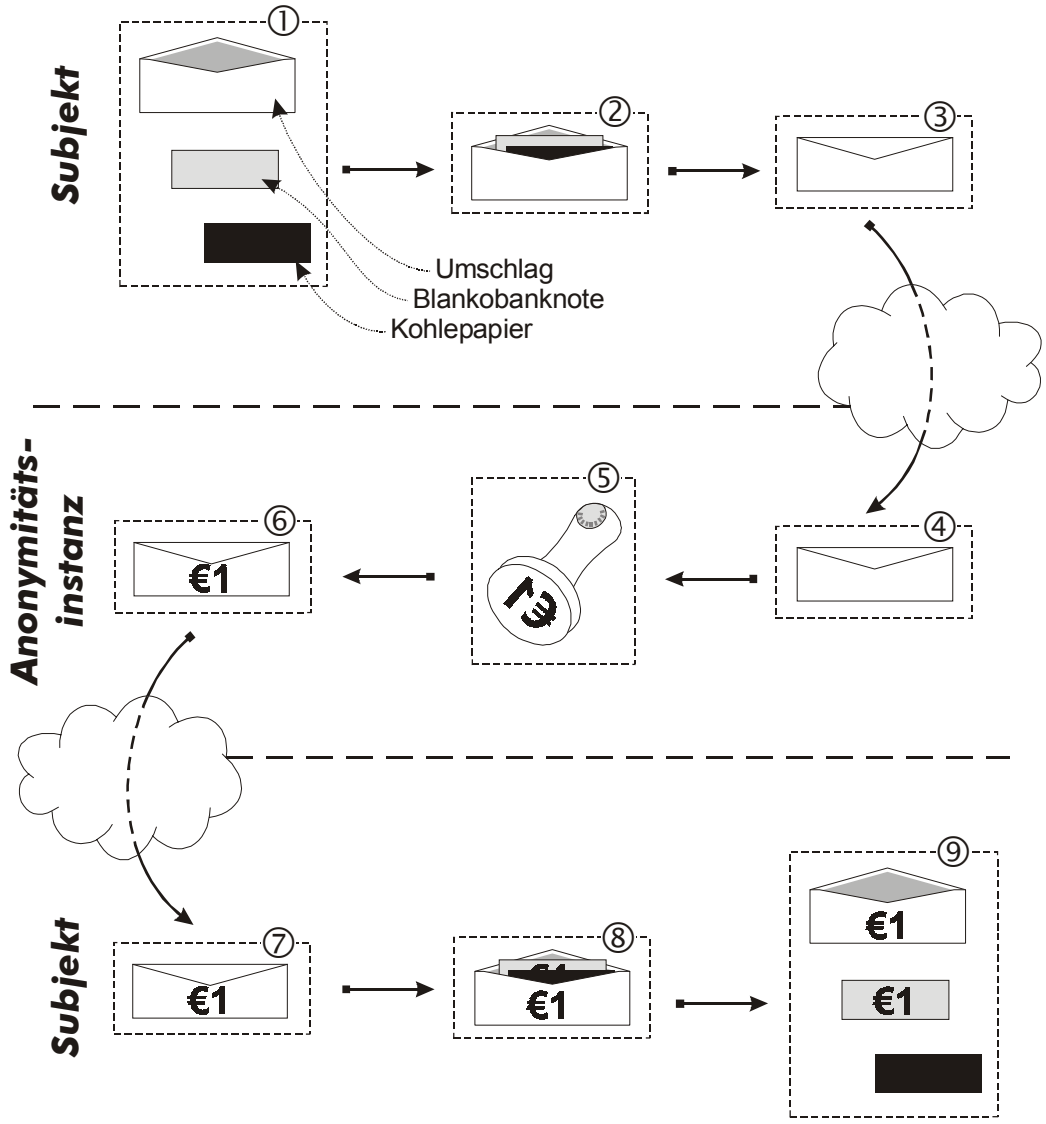


Abbildung 3 – **Veranschaulichung der Technik der blinden Unterschrift**

Die folgenden Bedingungen müssen eingehalten werden:

- Der Umschlag darf nur vom zukünftigen Einkäufer geöffnet werden, damit die Bank keine speziellen Bemerkungen (aus denen man später auf die Identität des Besitzers der Banknote schließen könnte) auf die Banknote schreiben kann.
- Die blanko Banknote darf keine Informationen über den zukünftigen Einkäufer tragen.
- Der Stempel muss für den gleichen Betrag immer der gleiche sein, für verschiedene Beträge müssen jedoch verschiedene Stempel benutzt werden.
- Es müssen standardisierte Beträge benutzt werden, damit man aus dem Betrag des Einkaufs oder der Banknote nicht auf die Identität des Einkäufers schließen kann.

Später, während des eigentlichen Einkaufs, benutzt der Einkäufer die so erhaltenen Banknoten beim Verkäufer. Falls der Einkäufer selbst seine Identität nicht preisgibt, so kann weder der Verkäufer noch die Bank aus den Banknoten auf die Identität des Einkäufers schließen. Die Authentizität der Banknoten wird mit dem vorhandenen Stempel bewiesen.

2.2.3.2.1.2 Teilnehmer

Während der Kommunikation können drei verschiedene Rollen unterschieden werden:

- **Subjekt**: mit Hilfe der Erlaubnis, welche die Anonymitätsinstanz ausgestellt hat, nimmt es einen Dienst anonym in Anspruch;
- **Dienstanbieter**: bietet Subjekten Dienste an, hat aber keine Möglichkeit, die Identität der Subjekte zu erfahren, Subjekte können nur mit gültigen Erlaubnissen Dienste in Anspruch nehmen;
- **Anonymitätsinstanz**: stellt für die Subjekte spezielle Erlaubnisse aus, mit der sie die Dienste der Dienstanbieter in Anspruch nehmen können, bzw. überprüft die Gültigkeit solcher Erlaubnisse während des Autorisierungsprozesses.

2.2.3.2.1.3 Anforderungen

Die folgenden Anforderungen sichern dem Subjekt das Erreichen des gewollten Anonymitätsniveaus:

- die einmalige Inanspruchnahme der Dienstleistung muss durch eine Erlaubnis von der Anonymitätsinstanz bewilligt werden, während dieser Bewilligung ist die echte Identität des Subjekts bekannt;
- mit der Erlaubnis kann die Dienstleistung eventuell bei verschiedenen Dienstanbietern in Anspruch genommen werden;
- aus der Erlaubnis darf die Identität des Subjekts nicht ermittelbar sein;
- Erlaubnisse können nur von der Anonymitätsinstanz ausgestellt werden, ihre Authentizität kann von allen Teilnehmern überprüft werden, die Erlaubnis spezifiziert den Art der Dienstleistung;
- während der Inanspruchnahme der Dienstleistung ist die Identität des Subjekts nicht bekannt;
- um zu verhindern, dass eine Erlaubnis mehrmals benutzt wird, wird dieser unmittelbar vor der Dienstleistung von der Anonymitätsinstanz überprüft;
- die Anonymitätsinstanz notiert das Ereignis, dass eine konkrete Erlaubnis bei einem konkreten Dienstanbieter benutzt wurde.

2.2.3.2.1.4 Beschreibung der Methode

Die folgenden Phasen können unterschieden werden:

- Während des **Initialisierungsprozesses** wird eine Erlaubnis ausgestellt. Dieser Prozess besteht aus zwei Unteretappen:
- Das Subjekt teilt der Anonymitätsinstanz die gewünschte Art der Dienstleistung, die es in Anspruch nehmen will, mit. Falls das Subjekt die nötigen Berechtigungen[19] hat, wird dann eine Vorerlaubnis (*pre-permission*) ausgestellt und Daten über die Erstellung bei der Anonymitätsinstanz gespeichert.

[19] Im Falle vom elektronischem Geld, bedeuten diese Berechtigungen im Allgemeinen, dass das Subjekt genügend echtes Geld auf seinem Konto hat.

- Danach führt das Subjekt allein noch bestimmte Veränderungen an dieser Vorerlaubnis durch, um die endgültige Erlaubnis zu bekommen, damit die Zuordnung zwischen dem Subjekt und der Erlaubnis nicht mehr hergestellt werden kann. Dabei müssen die folgenden drei Anforderungen erfüllt werden:
  - ⇨ Das Subjekt darf nicht fähig sein, allein eine Vorerlaubnis zu generieren.
  - ⇨ Nur aus Vorerlaubnissen sollen Erlaubnisse generierbar sein.
  - ⇨ Alle Teilnehmer sollen leicht überprüfen können, dass die Erlaubnis aus einer Vorerlaubnis generiert wurde.

  Dieses Prozess benötigt von der Sicherheitsschicht die Authentisierung beider Seiten, Integritätssicherung und Vertraulichkeit der Kommunikation.
- Während der **anonymen Autorisierung** benutzt das Subjekt das (die) von ihm allein bekannte(n) Erlaubnis(se)[20], um bei dem Dienstanbieter eine Dienstleistung in Anspruch zu nehmen.

  Zuerst teilt das Subjekt dem Dienstanbieter mit, welche Dienstleistung es in Anspruch nehmen will, und übermittelt auch die benötigten Erlaubnisse. Der Dienstanbieter überprüft die Erlaubnisse selbst, um falsche Erlaubnisse auszufiltern. Danach übermittelt der Dienstanbieter diese Erlaubnisse an die Anonymitätsinstanz, damit diese mögliche Zweitbenutzungen überprüfen kann. Werden die Erlaubnisse von der Anonymitätsinstanz akzeptiert, so kann die Dienstleistung erfolgen. Falls die Erlaubnisse von der Anonymitätsinstanz akzeptiert worden sind, so verlieren sie ihre Gültigkeit und die Anonymitätsinstanz speichert relevante Daten über die Benutzung der konkreten Erlaubnisse bei dem konkreten Dienstanbieter.

  Dieser Prozess benötigt von der Sicherheitsschicht zwischen Subjekt und Dienstanbieter die Authentisierung des Dienstanbieters, Integritätssicherung und Vertraulichkeit der Kommunikation; zwischen Anonymitätsinstanz und Dienstanbieter beidseitige Authentisierung, Integritätssicherung und Vertraulichkeit der Kommunikation.

2.2.3.3 Volle Anonymität

Im Folgenden werden verschiedene Mechanismen vorgestellt, die dem Subjekt volle Anonymität garantieren. Diese Mechanismen wurden für die Anonymisierung verschiedener Internet-Dienste konzipiert.

2.2.3.3.1 Anonyme E-Mail

Die in diesem Abschnitt vorgestellten Mechanismen dienen dazu, um E-Mail-Kommunikation zu anonymisieren.

In diesem Kontext sind zwei Begriffe wichtig zu nennen. Unter **Absenderanonymität** (*sender anonymity*) versteht man, dass der Empfänger die Identität des Absenders nicht erfahren kann. **Empfängeranonymität** (*receiver anonymity*) bedeutet, dass man anonym E-Mails empfangen kann, dass der Absender die Identität des Empfängers nicht ermitteln kann, der Absender kennt nur die Pseudoidentität des Empfängers.

[20] Wie der Dienstanbieter und das Subjekt es abmachen, für welche Dienstleistungen welche Art von Erlaubnissen benötigt werden, ist irrelevant für die Anonymität selbst und wird auch in AEP nicht behandelt.

Für die Einstufung wurde die Systematik aus [PET-2] übernommen.

2.2.3.3.1.1 Anonymer Remailer des Typs 0

Gemäß der schon vorgestellten Klassifizierung der Anonymitätsmechanismen könnte ein anonymer Remailer des Typs 0 ins – für AEP uninteressante – Anonymitätsniveau des zurückverfolgbaren Subjekts eingestuft werden, er wird jedoch, um die historische Entwicklung der Remailer zu zeigen, auch in diesem Abschnitt vorgestellt.

Ein anonymer Remailer des Typs 0 bietet sowohl Absender- als auch Empfängeranonymität an. Der bekannteste Vertreter dieser Gruppe war `anon.penet.fi`, der aber wegen gesetzlichen Problemen geschlossen wurde.

Absenderanonymität wird dadurch erreicht, dass man E-Mail über einen solchen Remailer schickt. Der Remailer entfernt alle Daten, aus denen man auf die Identität des Absenders schließen könnte, und schickt die Nachricht zum Empfänger.

Empfängeranonymität wird erreicht, da Subjekte eine anonyme E-Mail-Adresse beim Remailer bekommen können. E-Mails an diese Adresse werden an das Subjekt weitergeleitet, der Absender hat jedoch keine Möglichkeit, die Identität des Empfängers zu ermitteln.

Ein anonymer Remailer des Typs 0 hat jedoch mehrere Nachteile:

- Bezüglich der Absenderanonymität müssen Subjekte dem Betreiber des Remailers vertrauen, dass die Nachrichten anonym weitergeleitet werden.
- Bezüglich Empfängeranonymität müssen Subjekte dem Betreiber des Remailers vertrauen, dass die Datenbank, die die Zuordnung zwischen echter und anonymer E-Mail-Adresse enthält, nicht veröffentlicht wird. Subjekte müssen außerdem der Widerstandsfähigkeit des Remailers gegen Angriffen vertrauen.
- Angreifer mit entsprechenden technischen Möglichkeiten können durch das Abhören der eingehenden und ausgehenden Daten des Remailers beide Arten der Anonymität kompromittieren.

2.2.3.3.1.2 Anonymer Remailer des Typs 1

Um die obengenannten Schwachstellen des Typs 0 zu eliminieren, wurden die Remailer des Typs 1 eingeführt. Diese bieten nur Empfängeranonymität an, damit keine Datenbank geschützt werden muss. Des weiteren wurden die folgenden Mechanismen eingeführt:

- Ein Remailer des Typs 1 kann verschlüsselte Nachrichten empfangen. Falls man die ursprüngliche Nachricht mit dem öffentlichen Schlüssel des Remailers verschlüsselt, so kann nur der Remailer die ursprüngliche Nachricht wiederherstellen und ans Ziel weiterleiten.
- Remailers des Typs 1 unterstützen Verkettung. Dies bedeutet, dass sich in der Weiterleitungskette der E-Mail nicht nur einer, sondern mehrere Remailer befinden.

  Verknüpft man Verschlüsselung und Verkettung, so entsteht eine beachtliche Verbesserung der Anonymität. Der Absender erstellt eine verschachtelte Nachricht (siehe Abbildung 4). Diese wird dem ersten Remailer zugeschickt. Dieser entschlüsselt die Nachricht mit seinem geheimen Schlüssel und bekommt eine Zieladresse und wieder eine verschlüsselte Nachricht. Diese wird dann an die Zieladresse weitergeleitet. Befinden sich dort ein Remailer des Typs 1, so tut er das selbe. Zum

Schluss erhält dann der Empfänger die Nachricht von dem Remailer, der am Ende der Kette war.

Der Vorteil dieser Verschachtelung ist, dass keiner der Remailer den Absender zum Empfänger zuordnen kann. Ein Remailer kennt entweder den Absender oder den Empfänger oder keines von beiden. Um den Absender und Empfänger der E-Mail einander zuordnen zu können, bräuchte der Angreifer Zugriff auf alle Remailer in der Kette, was bei einer großen Anzahl von verwendeten Remailer schwierig zu bewältigen ist.

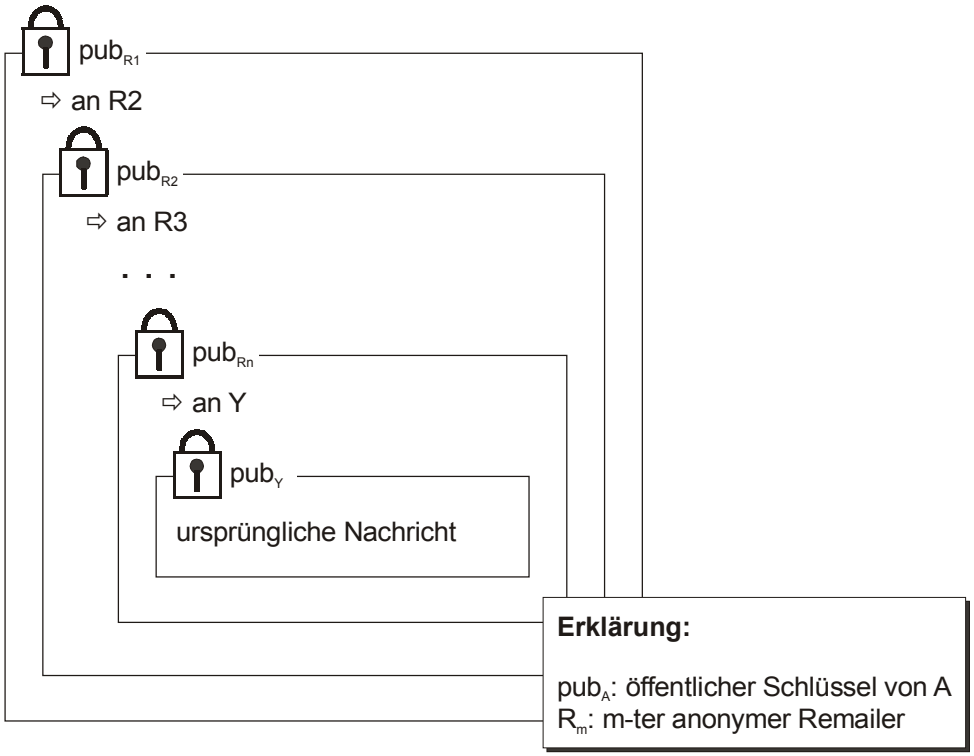


Abbildung 4 – **Verschachtelte Nachricht bei der Kombination von Verschlüsselung und Verkettung mit anonymen Remailer des Typs 1**

- Schließlich vertauschen Remailer des Typs 1 zufallsbestimmt die Reihenfolge der ausgehenden Nachrichten.

2.2.3.3.1.3 Anonymer Remailer des Typs 2

Anonyme Remailer des Typs 2 – die heute aufzufindenden Remailer – benutzen noch die folgenden Mechanismen:

- Nachrichten zwischen den Remailer haben immer dieselbe Länge. Falls eine ursprüngliche Nachricht länger ist, als diese Konstante, dann wird die Nachricht vom ersten Remailer aufgeteilt und vom letzten wieder zusammengestellt; falls die Nachricht kürzer ist als diese Konstante, dann wird sie mit Leerzeichen aufgefüllt.
- Da die Anonymität immer eine "Sicherheit in der Menge" voraussetzt, senden Remailer des Typs 2 einander falsche, keine echte Nachricht tragende Zufallsnachrichten.

2.2.3.3.1.4 Typ "newnym"

Systeme des Typs "newnym" wurden konzipiert, um Empfängeranonymität auf einem höheren Niveau der Sicherheit bereitzustellen. Anstatt wie bei einem anonymen Remailer des Typs 0 eine Datenbank mit anonymen und echten E-Mail-Adressen zu verwalten, speichern Remailer des Typs "newnym" nur eine verschachtelte Nachricht (die die Methoden der Verschlüsselung und Verkettung verwendet) und die Adresse eines anonymen Remailers des Typs 1 zusammen mit der anonymen E-Mail-Adresse.

Empfängt ein solcher Remailer eine Nachricht, die an eine von ihm verwaltete anonyme E-Mail-Adresse geschickt wurde, so wird diese Nachricht zusammen mit der gespeicherten verschachtelten Nachricht dem anonymen Remailer zugeschickt. Der entschlüsselt die erste Schicht der verschachtelten Nachricht und bekommt die Adresse eines zweiten Remailers, an den die ursprüngliche und die verschachtelte (schon einmal entschlüsselte) Nachricht gesendet werden. Der letzte Remailer bekommt die Nachricht zusammen mit der verschlüsselten echten E-Mail-Adresse des Empfängers.

2.2.3.3.1.5 Newsgroups

Eine ziemlich einfache Technik ist es, anonyme Nachrichtengruppen (*newsgroups*), wie z.B. `alt.anonymous.messages`, zu benutzen. Der Absender verschlüsselt die ursprüngliche Nachricht mit dem öffentlichen Schlüssel des Empfängers und schickt die Nachricht der Nachrichtengruppe. Die einzige über den Empfänger ableitbare Information ist, dass er die Nachrichtengruppe liest.

Diese Technik ist ähnlich der altbewährten Methode, die verschlüsselte Nachricht als Anzeige in einem Tagesblatt aufzugeben. Sie hat jedoch den Nachteil, dass sie mit den Ressourcen überhaupt nicht sparsam umgeht.

2.2.3.3.2 Anonymes Websurfen

Ähnlich zu der Funktionsweise der anonymen Remailer wurden Mechanismen fürs anonyme Websurfen entwickelt.

2.2.3.3.2.1 Anonymer Proxy – Typ 0

Ein **anonymer Proxy** führt außer dem transparenten Datentransfer lediglich eine Operation aus: er löscht alle relevanten Daten aus der Kommunikation, aus denen man auf die Identität oder Adresse des Subjekts schließen könnte. Im Allgemeinen bedeutet dies, dass der anonyme Proxy die IP-Adresse des Subjekts durch seine eigene IP-Adresse ersetzt.

Die Benutzung eines solchen anonymen Proxy ist einfach und braucht außer dem ursprünglichen Webbrowser keine zusätzlichen Programme. Ihre Benutzung ist einfach: entweder stellt man im Browser den gewünschten anonymen Proxy als Proxy ein oder man surft zu der Seite der Proxy und gibt dann dort die anonym aufzusuchende Adresse ein.

Der bekannteste Vertreter dieser Gruppe ist **www.anonymizer.com** [ANON-1].

2.2.3.3.2.2 MIX – Typ 1

Die Entwicklung von Typ 0 zum Typ 1 ist im Wesentlichen ähnlich wie beim anonymen Remailer. Das Subjekt muss für das Erreichen einer höheren Stufe der Sicherheit seiner

Anonymität zusätzliche Softwarekomponenten auf seinem Rechner installieren. Die folgenden Techniken werden in dieser Gruppe eingeführt:

- Die Kommunikation zwischen Subjekt und den Anonymitätsservern (von ihrem Erfinder **David Chaum** als **MIXe** bezeichnet) wird verschlüsselt.
- Mehrere solche MIXe werden in einem Kommunikationsvorgang benutzt.
- Nachrichtenpakete von mehreren Benutzer werden von den MIXen umkodiert und umsortiert.

Der bekannteste Vertreter solcher MIXe ist **JAP** [JAP-1].

3. Systemarchitektur

Dieses Kapitel fasst das durch AEP spezifizierte System kurz zusammen, stellt die im folgenden Kapitel ausführlich behandelten Protokollschichten vor, beschreibt das Verhältnis dieser Schichten zu anderen Schichten der Netzwerkarchitektur und erklärt die Rollen der Teilnehmer, die AEP benutzen.

Ziel dieses Kapitels ist, eine grobe Übersicht über AEP und seine Umgebung zu geben, die tiefgehende Spezifikation erfolgt dann im nächsten Kapitel.

3.1 Protokollaufbau

Sowohl theoretische Untersuchungen als auch praktische Erfahrungen haben die Existenz einer selbständigen Sicherheitsschicht gerechtfertigt. Aus ähnlichen Gründen wird im durch AEP spezifizierten System die Funktionalität der Anonymität von einer selbständigen Anonymitätsschicht erbracht. Diese benutzt die Dienste der Sicherheitsschicht, um die eigenen Dienste anbieten zu können.

In der konkreten Implementierung, die später behandelt wird, wurde SSH2 als Sicherheitsschicht (über TCP/IP) ausgewählt und sowohl diese als auch die Anonymitätsschicht wurden in JAVA realisiert.

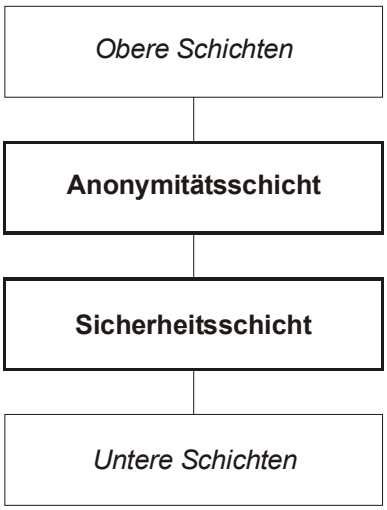


Abbildung 5 – **Allgemeine Schichtenarchitektur**

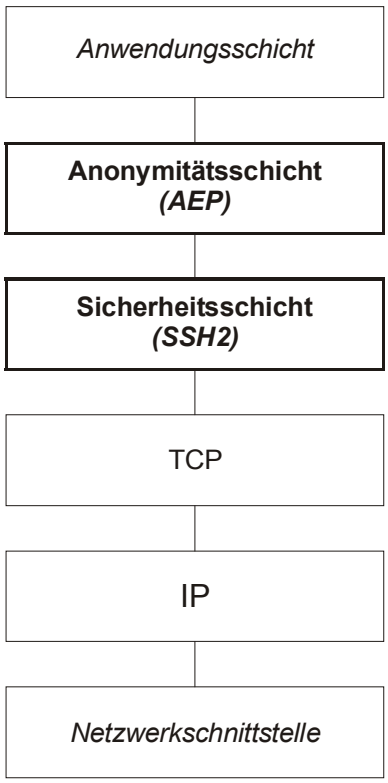


Abbildung 6 – **Die implementierte Schichtenarchitektur von AEP**

3.2 Protokollablauf

In diesem Abschnitt wird zuerst der Ablauf eines anonymen Autorisierungsprozesses im Allgemeinen beschrieben. Dann wird die durch AEP spezifizierte Architektur und der dort realisierte Ablauf vorgestellt.

3.2.1 Allgemeiner Ablauf eines anonymen Autorisierungsprozesses

Bei der Analyse der Anonymitätsverfahren, die in AEP verwirklicht werden sollen (Verfahren des Niveaus 3 und 4), wurden die folgenden Ähnlichkeiten unter ihnen festgestellt:

- Teilnehmer können drei verschieden Rollen annehmen: das **Subjekt** will das höchstmögliche Niveau an Anonymität bei einer Dienstinanspruchnahme erreichen; der **Dienstanbieter** bietet diesen Dienst an, will jedoch das Subjekt aus verschiedenen Gründen identifizieren; die **Anonymitätsinstanz** balanciert diesen Gegensatz aus.
- Das Schema der Kommunikation ist bei den verschiedenen Verfahren ähnlich:

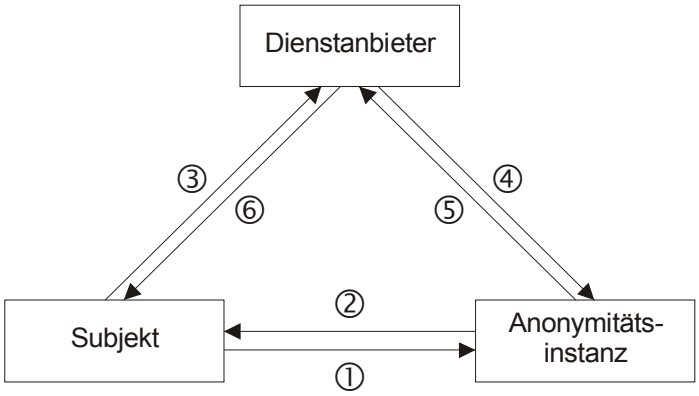


Abbildung 7 – **Allgemeiner Ablauf eines anonymen Autorisierungsprozesses**

- **Initialisierungsprozess**

  Die Kommunikation läuft zwischen Subjekt und Anonymitätsinstanz ab, während beide Teilnehmer identifizierbar sind.

  ① Das Subjekt vereinbart mit der Anonymitätsinstanz das Niveau der späteren Anonymität und den Typ der Dienstleistung.

  ② Die Anonymitätsinstanz stellt die für die spätere Dienstleistung benötigten **Anonymitätsdokumente** aus und speichert relevante Daten über diesen Vorgang.

- **Anonyme Autorisierung**

  Während der Kommunikation erreicht das Subjekt das gewünschte Niveau an Anonymität, alle anderen Teilnehmer sind identifizierbar.

  ③ Das Subjekt signalisiert anonym dem Dienstanbieter den Wunsch, einen konkreten Dienst in Anspruch zu nehmen, und übergibt die benötigten Anonymitätsdokumente.

④ Der Dienstanbieter veranlasst mit diesen Anonymitätsdokumenten bei der Anonymitätsinstanz die anonyme Autorisierung des Subjekts.

⑤ Die Anonymitätsinstanz überprüft die Authentizität der Anonymitätsdokumente und die Berechtigung des Subjekts. Falls das Subjekt berechtigt ist, den Dienst in Anspruch zu nehmen, wird eine Bewilligung an den Dienstanbieter geschickt und die Benutzung der Anonymitätsdokumente gespeichert.

⑥ Nach dem Empfang der Bewilligung führt der Dienstanbieter die Dienstleistung aus.

3.2.1.1 Bemerkungen

- Der Unterschied zwischen einem konkreten Dienst und dem Typ des Dienstes ist wie folgt:
  - Konkrete **Dienste** (*service*) könnten beispielsweise das – gebührenpflichtige – Abrufen aktueller Börseninformationen über das Internet oder das nur Studenten der Universität erlaubte Ausleihen eines Buches aus der Universitätsbibliothek sein.
  - Der **Typ des Dienstes** (*serviceType*) ist die Bezeichnung der Berechtigung, die zum Dienst benötigt wird. Für die obigen Beispiele wäre dann der Typ des Dienstes die Summe, die das Subjekt bezahlen muss, z. B. "5 €", oder die Bezeichnung der Gruppe, die berechtigt ist die Bibliothek zu benutzen, z. B. "UNI-KA Studenten".
- Die Anonymitätsinstanz erfährt nur den Typ des Dienstes und nicht die Bezeichnung des konkreten Dienstes. Dadurch hat sie keine Informationen über die Dienste, die das Subjekt in Anspruch genommen hat, falls sie überhaupt während der anonymen Autorisierung die Identität des Subjekts kennt.

3.2.2 Kommunikationsvorgänge im Zusammenhang mit AEP

Die Kommunikationsvorgänge im Zusammenhang mit AEP können in drei Phasen eingeteilt werden:

- Kommunikation zur Administration der beiden Serverinstanzen (Anonymitätsinstanz und Dienstanbieter),
- Kommunikation während des Initialisierungsprozesses,
- Kommunikation bei der anonymen Inanspruchnahme des Dienstes und der anonymen Autorisierung.

Abbildung 8 veranschaulicht all dieser Kommunikationsvorgänge.

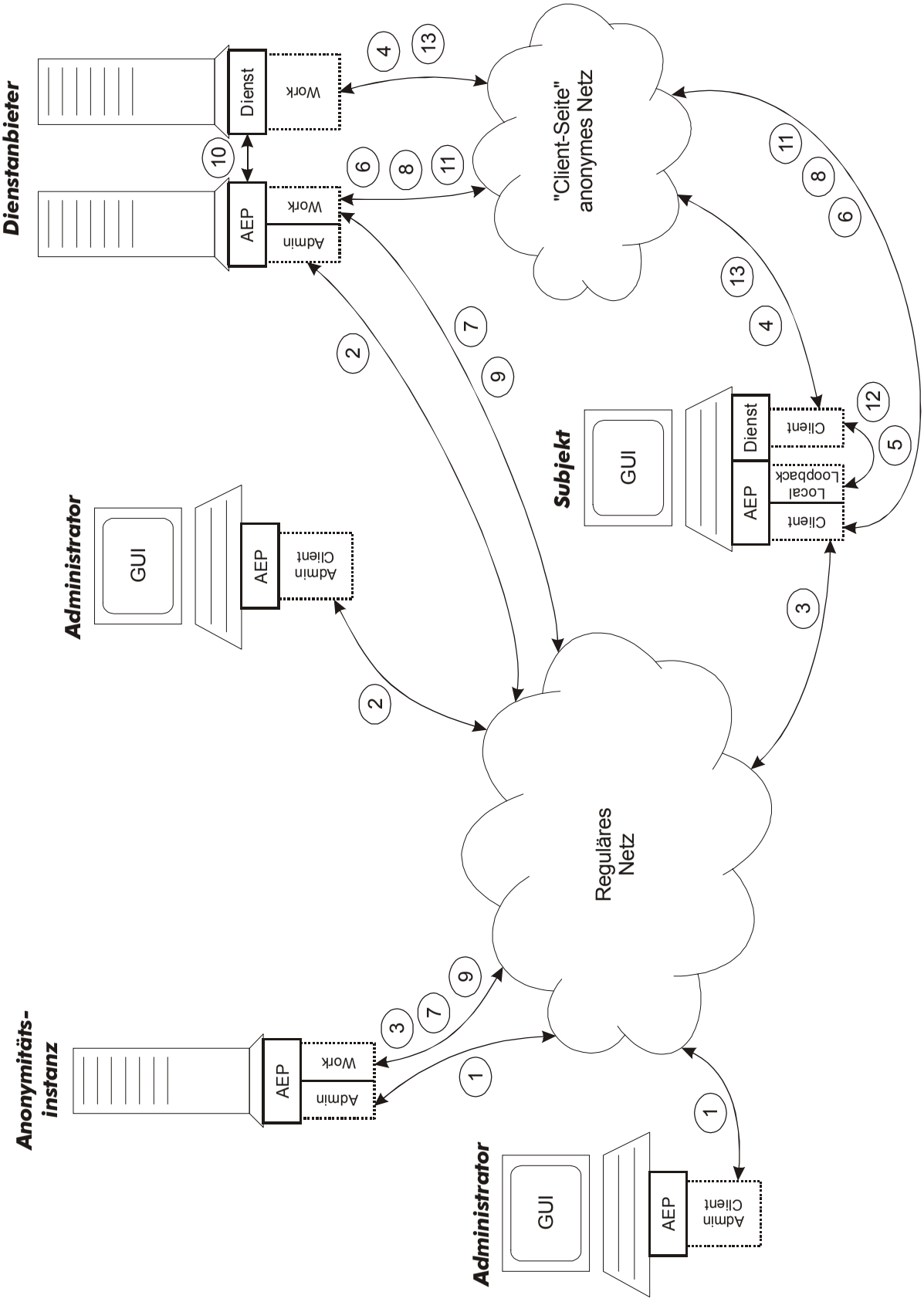


Abbildung 8 – **Kommunikationsvorgänge im Zusammenhang mit AEP**

3.2.2.1 Administration

AEP ermöglicht es, sowohl die Anonymitätsinstanz als auch den AEP-Server des Dienstanbieters über das Netz zu verwalten. Die Phasen (1) und (2) deuten an, dass diese ferne Administration über ein reguläres Netzwerk ablaufen kann.

> In der konkreten Implementierung wird diese ferne Administration über SSH2 Kanäle ermöglicht, wobei beide Seiten der Kommunikation authentisiert werden.

3.2.2.2 Initialisierungsprozess – Erwerb der Anonymitätsdokumente

In dieser Phase läuft die Kommunikation zwischen dem Subjekt und der Anonymitätsinstanz ab (3). Die Kommunikation kann über das reguläre Netz abgewickelt werden, der Kommunikationskanal wird von der Sicherheitsschicht geschützt.

Während dieser Phase vereinbaren die zwei Teilnehmer das spätere Niveau an Anonymität, das Subjekt teilt der Anonymitätsinstanz den Typ des Dienstes mit, und beide tauschen eventuell kryptographische Daten aus. Schließlich bekommt das Subjekt die Anonymitätsdokumente (*tokens*).

> In der konkreten Implementierung wird die Kommunikation über SSH2 abgewickelt, wobei beide Seiten authentisiert werden.

3.2.2.3 Anonyme Inanspruchnahme des Dienstes

In dieser Phase wird der Datenaustausch auf zwei Teilstrecken unterteilt. Einerseits tauschen Subjekt und Dienstanbieter Daten aus, andererseits Dienstanbieter und Anonymitätsinstanz. Während dieses Vorgangs ist das Subjekt gegenüber dem Dienstanbieter – und eventuell auch gegenüber der Anonymitätsinstanz, abhängig vom Niveau der Anonymität – anonym.

> Im Folgenden wird das Beispiel einer anonymen Abfrage einer HTML Seite über das Internet mittels eines herkömmlichen Browser behandelt.

Der ganze Vorgang wird von einer *"Dienst-Client"*-Applikation des Subjekts gestartet (4). Diese nimmt Kontakt mit derjenigen Instanz des Dienstanbieters auf, die den eigentlichen Dienst leistet. Bei diesem Vorgang ist das Subjekt anonym.

> Dies bedeutet, dass das Subjekt eine HTML Seite vom Webserver des Dienstanbieters mittels HTTPS abfragt, welcher dann z. B. mit einem `meta-refresh`-Operation die benötigten Daten weiterschickt. Um die Anonymität sicherzustellen, muss der Datenverkehr über ein anonymisierender Netzwerk gehen.
>
> - Abfrage durch HTTPS an den Webserver (in der konkreten Implementierung ein J2EE Server [J2EE-1]):
>   `GET request_service.jsp HTTP/1.1`
>
> - Antwort in Form einer HTML Seite über HTTPS:
>   ```
>   <meta http-equiv="refresh"
>   content="0;URL=http://localhost:2226/?serviceID=abcd&
>   serviceType=1EUR">
>   ```

Die *"Dienst-Client"*-Applikation muss dann der AEP-Instanz des Subjekts die relevanten Daten über den Dienst und den dazu benötigten anonymen Autorisierungsprozess mitteilen (5). Dazu betreibt die AEP-Instanz einen lokalen *"loopback"*-Server, der diese relevanten Daten empfängt.

Der lokale *"loopback"*-Server empfängt die HTTP-GET-Nachricht des Webbrowsers und bearbeitet sie.

- Abfrage über HTTP:
  `GET /?serviceID=abcd&serviceType=1EUR HTTP/1.1`

Die AEP-Instanz des Subjekts nimmt anonym mit der AEP-Instanz des Dienstanbieters Kontakt auf, und übermittelt relevante Daten über den gestarteten Dienst und auch die Anonymitätsdokumente, die für die Autorisierung benötigt werden (6).

Dies wird über SSH2 abgewickelt, wobei aber nur der Dienstanbieter authentisiert wird. Um die Anonymität sicherzustellen, muss der Datenverkehr über ein anonymisierendes Netzwerk gehen.

Die AEP-Instanz übermittelt die Anonymitätsdokumente und den Typ des beanspruchten Dienstes an die Anonymitätsinstanz (7). Die Anonymitätsinstanz hat jetzt zwei Möglichkeiten:

- Entweder enthalten die Anonymitätsdokumente alle benötigten Daten, die zur Autorisierung des Subjekts benötigt werden (Anonymitätsniveau des nicht zurückverfolgbaren Subjekts). In diesem Fall werden die Dokumente überprüft, und falls sie gültig sind, wird eine Bewilligung an die AEP-Instanz des Dienstanbieters geschickt. [Es geht mit (10) weiter]
- Oder die Anonymitätsinstanz braucht die Authentisierung des Subjekts (Anonymitäts-niveau des begrenzt zurückverfolgbaren Subjekts). In diesem Fall schickt die Anonymitätsinstanz dem Subjekt durch die AEP-Instanz des Dienstanbieters einen *Challenge*, einen zufälligen Text, den das Subjekt digital signieren muss, womit dann seine Identität bestätigt und es authentisiert wird. [Es geht mit (8) weiter]

  Dies wird über SSH2 abgewickelt, wobei beide Seiten authentisiert werden.

Die AEP-Instanz des Dienstanbieters schickt den *Challenge* unverändert an die AEP-Instanz des Subjekts weiter (8). Diese signiert ihn und schickt das Ergebnis als *Response* zurück

Dies wird über SSH2 abgewickelt, wobei aber nur der Dienstanbieter authentisiert wird. Um die Anonymität sicherzustellen, muss der Datenverkehr über ein anonymisierendes Netzwerk gehen.

Die AEP-Instanz des Dienstanbieters schickt die *Response* an die Anonymitätsinstanz (9). Jetzt muss diese alle benötigten Daten haben, um die Autorisation des Subjekts zu entscheiden. Falls die Dienstleistung erfolgen kann, wird eine Bewilligung an die AEP-Instanz des Dienstanbieters zurückgeschickt.

Dies wird über SSH2 abgewickelt, wobei beide Seiten authentisiert werden.

Falls die AEP-Instanz des Dienstanbieters die Bewilligung bekommen hat, wird diejenige Instanz, die die tatsächliche Dienstleistung erbringt, darüber benachrichtigt, dass die Dienstleistung autorisiert wurde (10).

In der konkreten Implementierung wird die ID der Dienstabfrage freigeschaltet, indem der J2EE-Webserver und die AEP-Instanz des Dienstanbieters mittels CORBA-IIOP kommunizieren.

Die AEP-Instanz des Dienstanbieters benachrichtigt die AEP-Instanz des Subjekts darüber, dass die Dienstleistung erfolgen kann (11).

Dies wird über SSH2 abgewickelt, wobei aber nur der Dienstanbieter authentisiert wird. Um die Anonymität sicherzustellen, muss der Datenverkehr über ein anonymisierendes Netzwerk gehen.

Der lokale *"loopback"*-Server des Subjekts schickt eine Antwort (12) auf die Anfrage (5) der *"Dienst-Client"*-Applikation, damit diese dann die Inanspruchnahme des Dienstes beginnt.

- Antwort in Form einer HTML-Seite über HTTP:
```
<meta http-equiv="refresh"
content="0;URL=https://serviceprovider/get_service.jsp?
serviceID=abcd">
```

Die *"Dienst-Client"*-Applikation des Subjekts nimmt Kontakt mit der diensterbringenden Instanz des Dienstanbieters auf (13). Diese erbringt den Dienst.

- Abfrage über HTTPS an den Webserver:
```
GET get_service.jsp?serviceID=abcd HTTP/1.1
```
- Dienst als Antwort in Form einer HTML-Seite über HTTPS

3.2.2.4 Bemerkungen

3.2.2.4.1 Kommunikationskanäle

In der Implementierung können mehrere Operationen über denselben Kommunikationskanal ablaufen. So werden die folgenden Kanäle benutzt (Nummerierung gemäß Abbildung 8): (1), (2), (3), (4), (5)+(12), (6)+(8)+(11), (7)+(9), (10), (13).

3.2.2.4.2 Skalierbarkeit

Falls der Dienstanbieter strikt vor jeder Inanspruchnahme von Diensten eine anonyme Autorisierung durchführen lässt, kann dies zur Überlastung der Anonymitätsinstanz führen. Dieses Problem kann z.B. durch die Einführung von **Sitzungen** (*sessions*) in der Kommunikation zwischen Subjekt und Dienstanbieter behoben werden. Für eine Gruppe von Diensten wird dann die Autorisierung desselben Subjekts nur einmal durchgeführt. Dies kann z. B. dadurch erreicht werden, dass das Subjekt einen zufälligen Sitzungsschlüssel am Ende einer gelungenen anonymen Autorisierung erwirbt und dann damit auch weitere Dienste in Anspruch nehmen kann. Unter Umständen kann dieser Sitzungsschlüssel mit der *serviceID* übereinstimmen.

3.2.2.4.3 *"serviceID"*

Die **serviceID** wird benutzt, um den konkreten Dienst, den das Subjekt in Anspruch nehmen will, zu identifizieren. Dadurch kann der Zusammenhang zwischen dem anonymen Autorisierungsprozess mittels AEP und der eigentlichen Dienstnutzung bzw. Dienstleistung erstellt werden. Die einfachste und vielleicht auch effizienteste Realisierung der *serviceID* sind lange zufällige Zahlen. Jeder Dienstnutzung, für die der Dienstanbieter eine anonyme Autorisierung des Subjekts mittels AEP durchführen will, wird eine solche lange zufällige Zahl zugeordnet, die nur der Dienstanbieter und das Subjekt kennen.

Beim Start der Dienstnutzung erhält das Subjekt die zufällige *serviceID*. Danach startet es die anonyme Autorisierung mit dieser *serviceID*. Falls der Dienstanbieter während dieses Kommunikationsvorgangs die Bewilligung von der Anonymitätsinstanz bekommt, so wird der Instanz des Dienstanbieters, die die Dienstleistung erbringen soll, signalisiert, dass diese *serviceID* autorisiert wurde. Falls dann das Subjekt den Dienst mittels der *serviceID* in Anspruch nehmen will, so wird das auch gelingen.

3.2.2.4.4 "Client-Seite" anonymes Netzwerk

Um die Anonymität nicht zu verletzten, muss gewährleistet werden, dass der Dienstanbieter durch AEP oder durch die Dienstinanspruchnahme keine Informationen über den Standort des Subjekts erhält.

AEP wurde in sich so konzipiert, dass nach dem Ablauf der Kommunikation sicher ist, dass der Dienstanbieter keine Daten über das Subjekt erfahren konnte.

Es muss jedoch berücksichtigt werden, dass auch unterliegende Schichten der Netzwerkarchitektur (sowohl in der AEP-Kommunikation als auch während der Dienstinanspruchnahme) Daten über die Kommunikationspartner preisgeben können. So kann man in einer TCP/IP-Kommunikation die IP-Adresse des Kommunikationspartners leicht herausfinden, ausgehend von dieser dann möglicherweise auch den geographischen Standort, wenn nicht sogar gleich die Identität.

Deshalb müssen Techniken benutzt werden, um diese Schichten zu anonymisieren. Für diesen Zweck kann das sogenannte **"Client-Side" anonymes Netzwerk** mit der dazugehörenden Netzwerkarchitektur konzipiert werden.

Die Spezifikation eines solchen Netzes ist nicht Ziel dieser Arbeit. Es existieren schon implementierte Techniken, die für bestimmte Protokolle der hier genannten Anforderungen genügen (siehe z. B. Abschnitt 2.2.3.3).

> Für die Anonymisierung eines webbasierten Dienstes kann z. B. JAP [JAP-1] benutzt werden, für die Anonymisierung der SSH2 Kommunikation könnte theoretisch ein anonymer SSH2-Proxy benutzt werden, der die schon vorgestellten Techniken der Verkettung, Verschlüsselung und Umordnung benutzt. NAT (*Network Address Translation*) [KRÜGER-1] ist möglicherweise die einfachste Möglichkeit, die "Client-Seite" anonymes Netz zu realisieren.

4. AEP-Spezifikation

In diesem Kapitel wird die AEP-Spezifikation vorgestellt. Zuerst wird der AEP-SAP spezifiziert, dann werden die beiden Sprachen zur Datenbeschreibung beschrieben, schließlich wird die Kommunikation im Zusammenhang mit AEP ausführlich behandelt.

4.1 Spezifikation des AEP-SAP

In diesem Abschnitt wird der Dienstzugangspunkt (*SAP*, *service access point*) der Anonymitätsschicht (AEP) definiert. Dabei werden die herkömmlichen Notationen (die im Anhang B ausführlich beschrieben werden) benutzt.

4.1.1 Initialisierungsprozess

Während des Initialisierungsprozesses läuft die Kommunikation zwischen Subjekt und Anonymitätsinstanz ab, wobei das Subjekt Anonymitätsdokumente erwirbt. Der Ablauf des Initialisierungsprozesses wird in der folgenden Abbildung veranschaulicht.

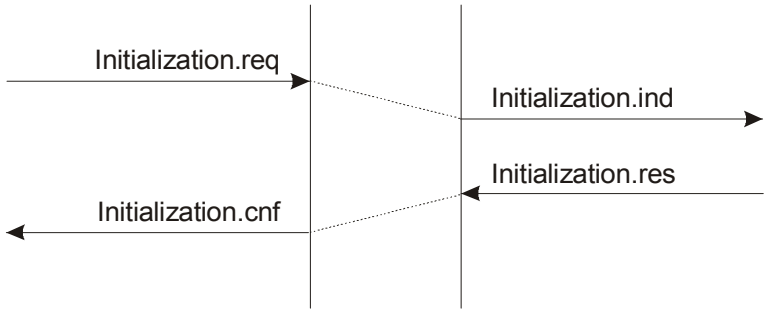


Abbildung 9 – **Ablauf des Initialisierungsprozesses**

4.1.1.1 Dienstelement AEP-Initialization

Der Initialisierungsprozess wird allein durch dieses Dienstelement abgewickelt.

	Dienstelement AEP-Initialization			
Parameter	req	ind	res	cnf
Adresse der Anonymitätsinstanz	M	M(=)		
Beanspruchtes Anonymitätsniveau	M	M(=)		
Typ des Dienstes	M	M(=)		
Anonymitätsdokumente			M	M(=)

4.1.2 Anonyme Autorisierung

Die anonyme Autorisierung beinhaltet zwei Kommunikationsvorgänge. Einerseits kommunizieren Subjekt und Dienstanbieter, andererseits Dienstanbieter und Anonymitätsinstanz.

Beide Kommunikationsvorgänge laufen über dieselben Dienstelemente ab, sie werden vom Dienstanbieter verknüpft. Der Ablauf der anonymen Autorisierung wird in der folgenden Abbildung veranschaulicht.

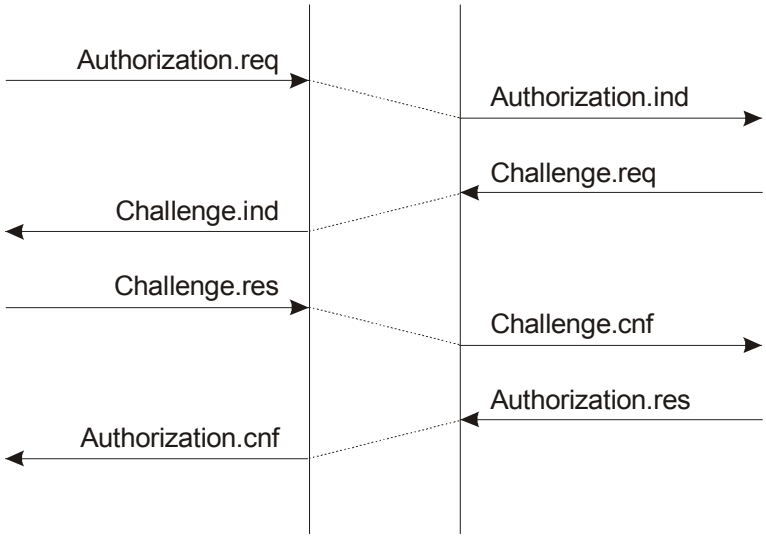


Abbildung 10 – **Ablauf der anonymen Autorisierung**

4.1.2.1 Dienstelement AEP-Authorization

Dieses Dienstelement bildet den Rahmen der anonymen Autorisierung und wird bei jeder anonymen Autorisation benutzt.

Zwischen Subjekt und Dienstanbieter ist die Benutzung des Feldes "*Beschreibung des Dienstes*" erforderlich, damit die anonyme Autorisation mit Hilfe von AEP und die eigentliche Dienstinanspruchnahme verbunden werden können. Dieses Feld darf jedoch während der Kommunikation zwischen Dienstanbieter und Anonymitätsinstanz nicht benutzt werden.

Das Feld "*Daten über die Dienstinanspruchnahme*" wird auch nur bei der Kommunikation zwischen Subjekt und Dienstanbieter benutzt.

Parameter	**Dientstelement AEP-Authorization**			
	req	ind	res	cnf
Adresse der Partner-AEP-Instanz	M	M(=)		
Beschreibung des Dienstes	O	C(=)		
Anonymitätsdokumente	M	M(=)		
Daten über die Dienstinanspruchnahme			O	C(=)

4.1.2.2 Dienstelement AEP-Challenge

Dieses Dienstelement wird nur im Falle des Anonymitätsniveaus des begrenzt verfolgbaren Subjekts benutzt. Mit seiner Hilfe wird ein Text (die **Challenge**) von der Anonymitätsinstanz über den Dienstanbieter zum Subjekt geschickt. Das Subjekt signiert diese Challenge und schickt dann die resultierende **Response** über den Dienstanbieter zu der Anonymitätsinstanz zurück.

Parameter	**Dientstelement AEP-Challenge**			
	req	ind	res	cnf
Challenge	M	M(=)		
Response			M	M(=)

4.1.3 Verbindungsabbau

Im Normalfall wird die Kommunikation zwischen zwei AEP-Instanzen zum passenden Zeitpunkt automatisch beendet.

- Während des Initialisierungsprozesses veranlasst die Anonymitätsinstanz die Beendigung der Kommunikation nach dem Absenden der `AEP-Initialization.res`.
- Während der anonymen Autorisierung:
  - Zwischen Dienstanbieter und Anonymitätsinstanz: Die Anonymitätsinstanz veranlasst die Beendung der Kommunikation nach dem Absenden der `AEP-Authorization.res`.
  - Zwischen Subjekt und Dienstanbieter: Der Dienstanbieter veranlasst die Beendigung der Kommunikation nach dem Absenden der `AEP-Authorization.res`.

Unter bestimmten Umständen kann es jedoch vorkommen, dass der Kommunikationsvorgang nicht mehr fortgesetzt werden kann. In einem solchen Fall kann das Dienstelement `AEP-Alert` benutzt werden, um der anderen Seite zu signalisieren, dass die Kommunikation beendet wird.

Parameter	**Dientstelement AEP-Alert**	
	req	ind
Begründung	M	M(=)

4.2 Sprachen zur Datenbeschreibung

In diesem Abschnitt werden die zwei Sprachen zur Datenbeschreibung, die von AEP benutzt werden, beschrieben.

Gemäß der ISO-OSI (Schicht 6 und 7) kann die Präsentationssprache mit der abstrakten Syntax und die Kommunikationssprache mit der Transfersyntax verglichen werden.

4.2.1 Präsentationssprache

Die **Präsentationssprache** (detaillierte Spezifikation siehe Abschnitt C.1 im Anhang) ist die Sprache, die die von der Anonymitätsschicht benutzten Datenstrukturen beschreibt. Die Definitionen aller möglichen Datentypen, die von AEP benutzt werden können, werden in der Präsentationssprache beschrieben.

Die Syntax der Präsentationssprache ist ähnlich zur Syntax der Präsentationssprache von TLS [TLS-1], WTLS [WTLS-1] und der Programmiersprache C [C-1]. Sie wurde so spezifiziert, dass sie leicht zu benutzen und zu verstehen ist.

Beispiel:

```
basic RString;

enum {NONE, TRACEABLE, RESTRICTED_TRACEABLE, NON_TRACEABLE,
FULL} AnonymityLevel;

struct {
   AnonymityLevel anonymityLevel;
   RString name;
   Extension parameters;
   Extension ext;
} AnonymitySuite;
```

4.2.2 Kommunikationssprache

Die Anonymitätsschicht tauscht die Daten mit ihrer Umgebung gemäß der **Kommunikationssprache** aus (detaillierte Spezifikation siehe Abschnitt C.2 im Anhang).

Ziel bei der Spezifikation der Architektur von AEP war, dass die Präsentationssprache auch in zukünftigen Versionen unverändert bleibt, aber die Kommunikationssprache bei Bedarf verändert oder ersetzt werden kann. Demnach wird die Kommunikationssprache nicht durch eine selbständige Definition angegeben, sondern durch einen Übersetzer, die Datenstrukturen der Präsentationssprache in Elemente der Kommunikationssprache umwandelt (und auch umgekehrt).

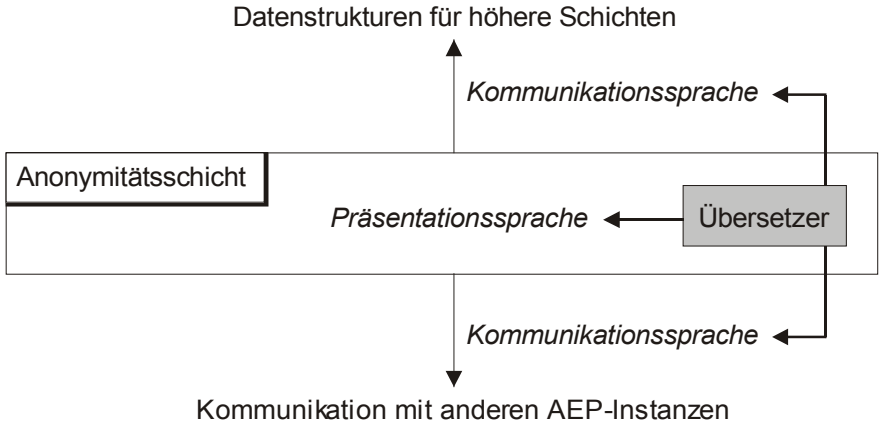


Abbildung 11 – **Zusammenhang zwischen Kommunikationssprache und Präsentationssprache**

Bei der Bestimmung der Kommunikationssprache wurden die folgenden Kriterien berücksichtigt:

- Es sollen Bibliotheken in mehreren Programmiersprachen für die Bearbeitung existieren,
- Sie soll schnell interpretierbar sein,
- Menschen sollen sie leicht lesen können.

Während der konkreten Implementierung wurde XML [XML-1] für die Kommunikationssprache ausgewählt.

Beispiel:

```
<anonymitySuite type="AnonymitySuite">
  <anonymityLevel value="RESTRICTED_TRACEABLE" />
  <name value="PseudoIdentity" />
</anonymitySuite>
```

In zukünftigen Implementierungen kann jedoch berücksichtigt werden, dass, falls die menschliche Lesbarkeit vernachlässigt wird, viel kompaktere Sprachen für die Kommunikationssprache in Betracht gezogen werden können. In Kommunikationsvorgängen, bei denen die Bandbreite kritisch ist, kann die relative große Länge von XML Dokumenten ein Nachteil sein. Falls jedoch z. B. die Komprimierung, die SSH2 anbietet, benutzt wird, so kann dieses Problem möglicherweise entschärft werden.

4.3 Die AEP-Kommunikation

Die Anonymitätsschicht erbringt die gewünschte Funktionalität durch Nachrichten formuliert in der Kommunikationssprache. Damit die verschiedenen Anonymitätstechniken einheitlich behandelt werden können, wurden diese Nachrichten so spezifiziert, dass sie so allgemein wie nur möglich gestaltet sind bzw. die Realisierung aller erwähnten konkreten Anonymitätstechniken ermöglichen. Dieser Abschnitt stellt diese Nachrichten und ihre Reihenfolge vor.

Im Folgenden wird zuerst die allgemeine, abstrakte Kommunikation beschrieben, dann werden die Details der konkreten Anonymitätstechniken behandelt.

4.3.1 Voraussetzungen

Damit die Sicherheitsschicht, die von der Anonymitätsschicht benötigt wird, die gewünschte Funktionalität erbringen kann, ist es nötig, dass alle Teilnehmer der Kommunikations-vorgänge

- in Besitz digitaler Ausweise sind, mit deren Hilfe ihnen ein eindeutiger Name zugeordnet werden kann (*identification*), und
- fähig sind, ihre Identität mit Hilfe kryptographischer Methoden zu beweisen (*authentication*).

Für die Realisierung dieser Anforderungen wird in der Praxis ein neuer Typ von Teilnehmer in den Kommunikationsvorgängen benötigt. Dieser ist die so genannte **Zertifizierungsstelle** (*certification authority*), deren Aufgabe es ist, als Einrichtung der **PKI** [PKI-1] [FROOMKIN-1] (*public key infrastructure*) fälschungssichere Ausweise auszustellen. Es ist nötig, dass jeder Eigentümer eines solchen Ausweises in Besitz eines geheimen und des dazugehörenden öffentlichen Schlüssels ist.

Der Ausweis wird von der CA ausgestellt und von ihr auch digital signiert. Sie beinhaltet neben der Bezeichnung der CA (*issuer*) den öffentlichen Schlüssel und die Bezeichnung (*DN, distinguished name*) des Besitzers.

4.3.2 Allgemeine Kommunikation

Gemäß der Analyse der verschiedenen Anonymitätstechniken werden die folgenden drei relevanten Typen von Teilnehmern der AEP-Kommunikation unterschieden: **Subjekt**, **Dienstanbieter** und **Anonymitätsinstanz**.

Die anonyme Dienstinanspruchnahme wird in zwei Phasen erreicht:

- Während des **Initialisierungsprozesses** erwirbt das Subjekt die Anonymitäts-dokumente, die in der nächsten Phase benötigt werden. Während dessen kommuniziert es mit der Anonymitätsinstanz, wobei seine Identität bekannt ist.
- Während der **anonymen Autorisierung** nimmt das Subjekt anonym Kontakt mit dem Dienstanbieter auf. Die Autorisierung der Dienstinanspruchnahme wird von der Anonymitätsinstanz durchgeführt.

Dieser Abschnitt stellt alle Nachrichtentypen von AEP vor. In einem konkreten Kommunikationsvorgang kann es jedoch vorkommen, dass nicht alle Typen benötigt werden. In den folgenden Kommunikationsdiagrammen werden die optionalen Nachrichten mit einem Stern (*) gekennzeichnet. Alle anderen Nachrichten kommen in jedem Kommunikations-vorgang vor. Die Reihenfolge der Nachrichten kann in AEP nicht verändert werden.

Während der Kommunikation wird die Schichtenarchitektur von unten aufgebaut. Da die Anonymitätsschicht sich oberhalb der Sicherheitsschicht befindet, wird zuerst die sichere Verbindung zwischen den beiden Seiten aufgebaut. Erst danach startet die Anonymitätsschicht ihren Nachrichtenaustausch mittels des gesicherten Datentransports.

4.3.2.1 Initialisierungsprozess

Die Aufgabe des Initialisierungsprozesses ist es, das später während der anonymen Autorisierung benutzte Anonymitätsverfahren zu vereinbaren und die dazu benötigte Anonymitätsdokumente dem Subjekt zu übermitteln. Das Kommunikationsdiagramm des Initialisierungsprozesses sieht folgendermaßen aus:

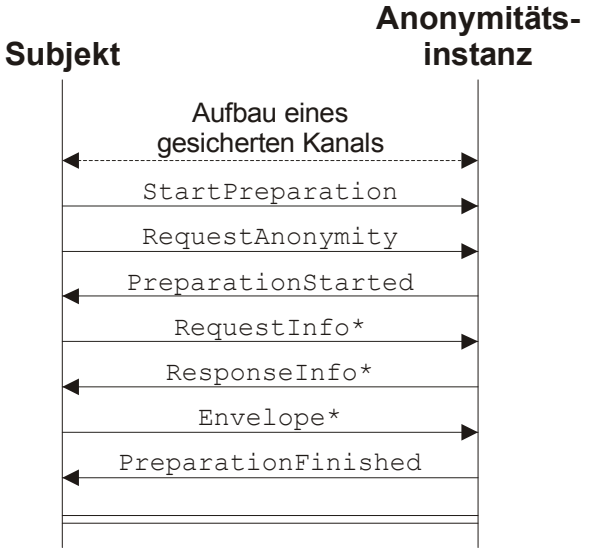


Abbildung 12 – **Kommunikationsdiagramm des allgemeinen Initialisierungsprozesses**

Der Initialisierungsprozess beschränkt sich auf das Subjekt und die Anonymitätsinstanz. Dabei sind beide Seiten identifizierbar. Während des Kommunikationsvorgangs wird zuerst die Verbindung durch die Sicherheitsschicht aufgebaut. Diese Phase benötigt die Authentisierung beider Seiten, Vertraulichkeit und Integritätssicherung der Kommunikation.

Die Bedeutung der Nachrichten ist die Folgende (Bemerkung: AS ist die Bezeichnung des Subjekts bzw. AA der Anonymitätsinstanz):

- `StartPreparation` (AS⇨AA): Neben der Versionsnummer der benutzten AEP-Spezifikation[21] übermittelt das Subjekt eine Liste von Bezeichnungen der von ihm unterstützten Anonymitätsverfahren.
- `RequestAnonymity` (AS⇨AA): In dieser Nachricht übermittelt das Subjekt den Typ des späteren Dienstes und das gewünschte Anonymitätsniveau.
- `PreparationStarted` (AA⇨AS): Die Anonymitätsinstanz teilt dem Subjekt mit, welches Anonymitätsverfahren aus der in der `StartPreparation`-Nachricht übermittelten Liste für die Anonymitätsdokumente und damit auch für die spätere Dienstinanspruchnahme ausgewählt wurde.
- `RequestInfo*` (AS⇨AA): Falls das Subjekt bestimmte Daten von der Anonymitätsinstanz für spätere Nachrichten oder für die Durchführung bestimmter kryptographischer Verfahren braucht, so kann es diese mit Hilfe dieser optionalen Nachricht von der Anonymitätsinstanz abfragen.
- `ResponseInfo*` (AA⇨AS): Als Antwort auf die `RequestInfo`-Nachricht übermittelt die Anonymitätsinstanz dem Subjekt die gewünschten Daten mit Hilfe dieser optionalen Nachricht.
- `Envelope*` (AS⇨AA): Falls das Subjekt der Anonymitätsinstanz kryptographische Daten übermitteln muss, so kann diese optionale Nachricht für diesen Zweck benutzt werden.
- `PreparationFinished` (AA⇨AS): Als letzte Nachricht übermittelt die Anonymitätsinstanz dem Subjekt die ausgestellten Anonymitätsdokumente. Bei der Anonymitätsinstanz können eventuell Daten über diese Ausstellung gespeichert werden.

4.3.2.1.1 Bemerkungen

- Der Initialisierungsprozess kann auch in zwei Phasen unterteilt werden.
  - Mit den `StartPreparation`, `RequestAnonymity` und `Preparation-Started` Nachrichten werden zuerst das später benutzte Anonymitätsverfahren vereinbart. Diese Nachrichten sind allgemein und werden unabhängig vom vereinbarten Anonymitätsverfahren immer in der gleichen Weise benutzt.
  - Die `RequestInfo`, `ResponseInfo`, `Envelope` und `Preparation-Finished` Nachrichten werden schon vom vereinbarten Anonymitätsverfahren beeinflusst.

[21] Die vorliegende Spezifikation hat die Versionsnummer **0.2**.

- Damit die Flexibilität des Protokolls nicht verloren geht, kann jede Nachricht um zusätzliche Daten erweitert werden.
- Ein Kommunikationskanal wird nur für die einmalige Durchführung des Initialisierungsprozesses benutzt. Nach der `PreparationFinished`-Nachricht wird der Kanal von der Anonymitätsinstanz automatisch geschlossen.
- Es kann vorkommen, dass das Subjekt die erworbenen Anonymitätsdokumente noch verändern muss. Z. B. im Falle des Anonymitätsverfahrens blinder Unterschrift (siehe 2.2.3.2.1) führt das Subjekt noch bestimme kryptographische Transformationen an den erworbenen Anonymitätsdokumenten durch.
- Anonymitätsdokumente in der AEP-Kommunikation werden als so genannte *Tokens* realisiert. Ein solches Token ist das Ergebnis eines Initialisierungsprozesses. Es ist als eine komplexe Datenstruktur vorzustellen und beinhaltet die folgenden Daten:
  - Verbindliche Felder:
    - ⇨ Bezeichnung des Anonymitätsniveaus, das mit der Hilfe des Tokens während der Dienstinanspruchnahme erreicht wird
    - ⇨ Bezeichnung des Anonymitätsverfahrens, das während der anonymen Autorisierung benutzt wird
    - ⇨ Typ des Dienstes, für den das Token ausgestellt wurde
    - ⇨ Information über eventuelle Beschränkungen der Benutzung des Tokens (z. B. "kann nur einmal benutzt werden", "kann mehrfach benutzt werden" usw.)
    - ⇨ Ausstellungsdatum
  - Optionale Felder:
    - ⇨ Eindeutiges Kennzeichen (*UID*, *unique identifier*) des Tokens
    - ⇨ Intervall der Gültigkeit
    - ⇨ Digitale Signatur der Anonymitätsinstanz

4.3.2.2 Anonyme Autorisierung

Die anonyme Autorisierung mittels AEP wird nach einer Dienstleistungsanforderung vom Subjekt gestartet. In dieser Phase nehmen alle drei Seiten teil. Es werden zwei Kommunikationskanäle aufgebaut: zwischen Subjekt und Dienstanbieter bzw. zwischen Dienstanbieter und Anonymitätsinstanz. Ziel der anonymen Autorisierung ist festzustellen, ob das Subjekt den angeforderten Dienst in Anspruch nehmen darf. Dies wird von der Anonymitätsinstanz gemäß der übermittelten Anonymitätsdokumente entschieden.

Während dieser Kommunikationsvorgänge ist das Subjekt für den Dienstanbieter (und eventuell auch für die Anonymitätsinstanz) anonym und erreicht das Anonymitätsniveau, das in den Anonymitätsdokumenten, die es während des Initialisierungsprozesses erworben hat, spezifiziert ist.

Während der Kommunikationsvorgänge werden zuerst die Verbindungen durch die Sicherheitsschichten aufgebaut. Zwischen Subjekt und Dienstanbieter wird die Authentisierung des Dienstanbieters, Vertraulichkeit und Integritätssicherung der Kommunikation benötigt; zwischen Dienstanbieter und Anonymitätsinstanz die Authentisierung beider Seiten, Vertraulichkeit und Integritätssicherung der Kommunikation.

Das Kommunikationsdiagramm der anonymen Autorisierung sieht folgendermaßen aus:

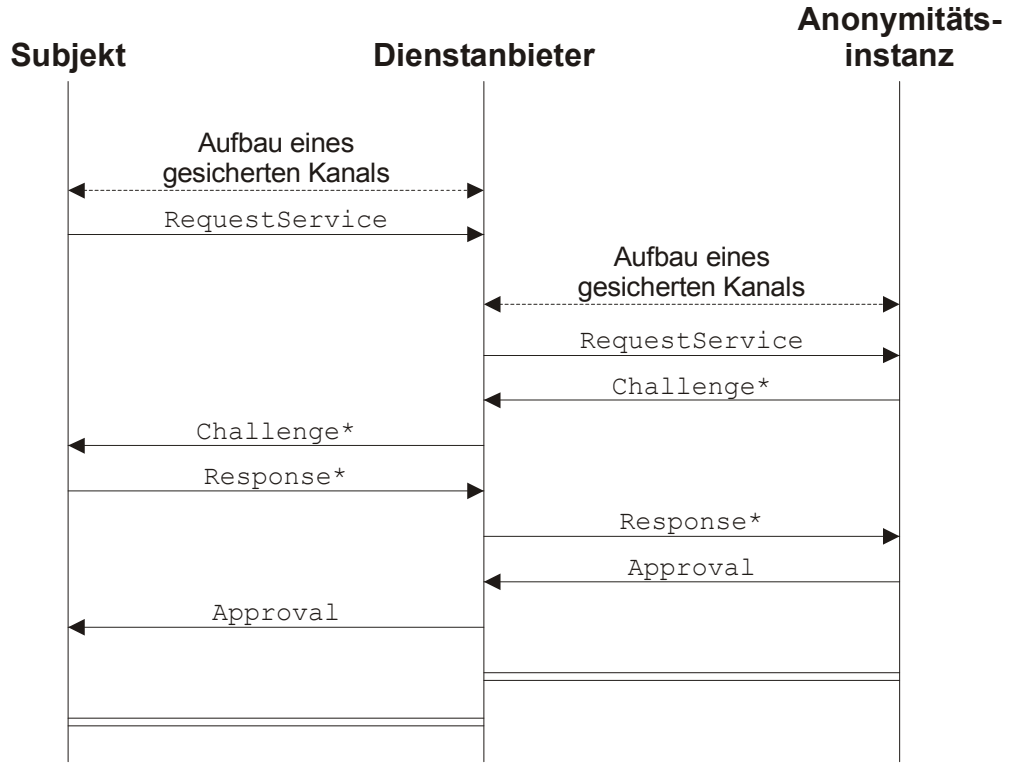


Abbildung 13 – **Kommunikationsdiagramm der allgemeinen anonymen Autorisierung**

Die Bedeutung der Nachrichten ist wie folgt (Bemerkung: AS ist die Bezeichnung des Subjekts, SP des Dienstanbieters und AA der Anonymitätsinstanz):

- `RequestService` (AS⇨SP): In dieser Nachricht übermittelt das Subjekt Daten über den angeforderten Dienst (z. B. *serviceID*) und die dazu benötigten Anonymitäts-dokumente.
- `RequestService` (SP⇨AA): Der Dienstanbieter leitet die Nachricht des Subjekts weiter zu der Anonymitätsinstanz, löscht daraus jedoch alle relevanten Daten im Bezug auf den eigentlichen Dienst.
- `Challenge*` (AA⇨SP): Falls die Anonymitätsdokumente allein die Autorisierung des Subjekts nicht ermöglichen und das Subjekt für die Anonymitätsinstanz authentisiert werden muss, so schickt diese dem Subjekt über den Dienstanbieter einen Text, den das Subjekt digital signieren muss.
- `Challenge*` (SP⇨AS): Der Dienstanbieter leitet die von der Anonymitätsinstanz erhaltene optionale `Challenge`-Nachricht unverändert ans Subjekt weiter.
- `Response*` (AS⇨SP): Als Antwort auf die `Challenge`-Nachricht schickt das Subjekt durch den Dienstanbieter die digitale Signatur des zufälligen Textes zur Anonymitätsinstanz.
- `Response*` (SP⇨AA): Der Dienstanbieter leitet die vom Subjekt erhaltene optionale `Response`-Nachricht unverändert an die Anonymitätsinstanz weiter.

- `Approval` (AA⇨SP): Falls die Anonymitätsinstanz die Berechtigung des Subjekts ermitteln konnte, so wird dem Dienstanbieter diese Nachricht als Bewilligung zugeschickt.
- `Approval` (SP⇨AS): Nachdem der Dienstanbieter die Bewilligung von der Anonymitätsinstanz erhalten hat, wird dieselbe Nachricht auch ans Subjekt geschickt, damit dieses die Inanspruchnahme des angeforderten Dienstes beginnen kann.

4.3.2.2.1 Bemerkungen

- AEP ist zur Zeit eigentlich auf die anonyme Autorisierung eines Subjekts ausgelegt. Das bedeutet, dass der Ergebnis von AEP die Antwort auf eine Ja oder Nein Frage ist, nämlich ob das Subjekt einen Diensttyp in Anspruch nehmen darf (oder auch nicht). Wie der Dienstanbieter diesen Dienst in der Wirklichkeit danach leistet, ist zur Zeit nicht Teil der AEP Spezifikation. (Damit das Protokoll jedoch untersucht werden kann, wurde in der Testimplementierung auch dieser Themenbereich behandelt. Siehe die Bemerkungen 3.2.2.4.2 und 3.2.2.4.3.)
- Damit die Flexibilität des Protokolls nicht verloren geht, kann jede Nachricht um zusätzliche Daten erweitert werden.
- Ein Kommunikationskanal wird nur für die einmalige Durchführung einer anonymen Autorisierung benutzt. Zwischen Anonymitätsinstanz und Dienstanbieter wird der Kanal von der Anonymitätsinstanz nach dem Absenden der `Approval`-Nachricht geschlossen. Zwischen Dienstanbieter und Subjekt wird der Kanal vom Dienstanbieter nach dem Absenden der `Approval` Nachricht geschlossen.

4.3.2.3 Spezialfälle

Bislang wurde nur das beschrieben, was während des normalen Ablaufs der AEP-Kommunikation vorkommt. Ein Protokoll muss jedoch auch auf Spezialfälle vorbereitet sein.

4.3.2.3.1 Fehler

Falls ein Teilnehmer der AEP-Kommunikation einen Zustand erreicht, der nicht der AEP-Spezifikation entspricht, so kann er den Kommunikationsvorgang abbrechen. Um dem Partner signalisieren zu können, dass die Kommunikation absichtlich beendet wurde, wurde für diesen Zweck in der AEP-Spezifikaion eine spezielle Nachricht (`Alert`) eingeführt.

Nach dem Absenden einer `Alert` Nachricht wird der Kommunikationskanal vom Absender geschlossen. Da die Nachricht auch eine Begründung der Unterbrechung der Kommunikation enthält, kann der Empfänger aus der Nachricht auf die Ursache des Fehlers schließen.

4.3.2.3.2 Missbrauch

Die Teilnehmer können die folgenden Maßnahmen im Falle eines eventuellen Missbrauchs ergreifen:

- Anonymitätsinstanz
  - Falls der Missbrauch während des Initialisierungsprozesses entdeckt wird, so wird die Anonymitätsinstanz die Anonymitätsdokumente nicht ausstellen.

  - Falls der Missbrauch während der anonymen Autorisierung entdeckt wird, so wird die Bewilligung dem Dienstanbieter nicht erteilt.
  - Falls es festgestellt wird, dass Anonymitätsdokumente Unbefugten ausgestellt wurden, so können diese abhängig vom benutzten Anonymitätsverfahren widerrufen werden. Z. B. können Pseudoidentitäten widerrufen werden, Anonymitätsdokumente des Verfahrens blinder Unterschrift aber nicht.

- Dienstanbieter
  - Falls der Dienstanbieter feststellt, dass ein anonymes Subjekt nach der erfolgreichen anonymen Autorisierung einen Missbrauch begangen hat, so kann er die Freigabe der echten Identität des Subjekts bei der Anonymitätsinstanz veranlassen. Das ist jedoch nicht bei allen Anonymitätsverfahren möglich (nur bei denjenigen, die das Anonymitätsniveau des beschränkt zurückverfolgbaren Subjekts realisieren).
- Subjekt
  - Falls das Subjekt während des Initialisierungsprozesses die gewünschten Anonymitätsdokumente nicht erlangt, während der anonymen Autorisierung keine Bewilligung erhält, oder trotz Bewilligung den Dienst nicht in Anspruch nehmen kann, so kann es dies nur durch Methoden, die nicht Teil der AEP-Spezifikation sind, korrigieren.

4.3.3 Bemerkungen zu konkreten Anonymitätsverfahren

In diesem Abschnitt wird die Realisierung verschiedener Anonymitätsverfahren mit Hilfe von AEP behandelt.

4.3.3.1 Pseudoidentität (*pseudo-identity*)

Diese Methode wird im Abschnitt 2.2.3.1.1 eingestuft und behandelt. Mit ihrer Hilfe kann das Niveau des begrenzt zurückverfolgbaren Subjekts erreicht werden. In diesem Abschnitt wird die Realisierung dieser Methode innerhalb von AEP beschrieben.

4.3.3.1.1 Initialisierungsprozess

- Nach der erfolgreichen Vereinbarung dieser Methode schickt die Anonymitätsinstanz gleich als letzte Nachricht (`PreparationFinished`) die Anonymitätsdokumente. Diese enthalten die Pseudoidentität des Subjekts für den gewünschten Diensttyp. Diese Pseudoidentität ist eine große zufällige Zahl, die das Subjekt für die Anonymitätsinstanz eindeutig identifiziert, sie kann jedoch von anderen (z. B. vom Dienstanbieter) nicht der echten Identität des Subjekts zugeordnet werden.

4.3.3.1.2 Anonyme Autorisierung

- Während der anonymen Autorisierung schickt das Subjekt zuerst die Anonymitätsdokumente dem Dienstanbieter (`RequestService`), dieser leitet dann die Nachricht weiter zur Anonymitätsinstanz.

- Die Anonymitätsinstanz schickt die `Challenge`-Nachricht zurück zum Dienstanbieter, welcher dann diese zum Subjekt weiterleitet.
- Der in der `Challenge`-Nachricht enthaltene zufällige Text muss dann vom Subjekt digital signiert werden, damit seine Identität bestätigt werden kann. Diese digitale Signatur muss mit demjenigen geheimen Schlüssel erstellt werden, welchen das Subjekt beim Verbindungsaufbau der Sicherheitsschicht während des Initialisierungsprozesses benutzt hat. Die so erstellte digitale Signatur wird dann in der `Response`-Nachricht dem Dienstanbieter zurückgeschickt und zur Anonymitäts-instanz weitergeleitet.
- Falls die Identität damit bewiesen wurde, so erteilt die Anonymitätsinstanz die Bewilligung, und schickt dem Dienstanbieter die `Approval`-Nachricht, welche schließlich auch das Subjekt erreicht.

4.3.3.2 Einmalige Pseudoidentität (*one-time pseudo-identity*)

Diese Methode ist eine Erweiterung der im Abschnitt 2.2.3.1.1 eingestuften und behandelten Methode der Pseudoidentität. Mit ihrer Hilfe kann das Niveau des begrenzt zurückverfolgbaren Subjekts erreicht werden.

Bei der Verwendung der Methode der Pseudoidentität wird das Subjekt während der anonymen Autorisierung immer mit derselben Pseudoidentität identifiziert. Erhält der Dienstanbieter Informationen, durch die er die Pseudoidentität mit der echten Identität des Subjekts verknüpfen kann, so kann die Anonymität aller vorherigen anonymen Aktivitäten des Subjekts aufgehoben werden.

Die hier vorgestellte Methode behebt diese Schwäche. Die Grundlage dafür ist, dass gemäß diesem Verfahren das Subjekt beliebig viele Pseudoidentitäten erstellen kann. Die Anonymitätsinstanz kann die Autorisierung trotzdem durchführen, und der Dienstanbieter sieht bei jeder Autorisierung eine andere Pseudoidentität.

In diesem Abschnitt werden die kryptographischen Grundlagen behandelt und die Realisierung innerhalb AEP beschrieben.

4.3.3.2.1 Kryptographische Grundlagen

In diesem Abschnitt werden die kryptographischen Grundlagen der Methode der einmaligen Pseudoidentität näher betrachtet. Die während des Initialisierungsprozesses und der anonymen Autorisierung durchzuführenden Transformationen werden beschrieben und an einem konkreten Beispiel auch vorgeführt.

4.3.3.2.1.1 Initialisierungsprozess

Während des Initialisierungsprozesses werden eigentlich keine kryptographischen Operationen in AEP durchgeführt, das einzig wichtige, was abgewickelt wird, ist das Folgende:

1. Das Subjekt bekommt von der Anonymitätsinstanz den öffentlichen RSA-Schlüssel dieses Verfahrens. Dieser besteht aus einem Exponenten (`OTPIPubExp`) und einem Modulus (`OTPIMod`).

   Z. B. sei `OTPIPubExp` = 10001_{16} und `OTPIMod` = $513AFF69_{16}$.

4.3.3.2.1.2 Anonyme Autorisierung

Die Anonymitätsdokumente, die das Subjekt während des Initialisierungsprozesses erhalten hat, enthalten ein eindeutiges Kennzeichen (`pseudoIdentity`), mit dem diese identifiziert werden. Dieses Kennzeichen muss der Anonymitätsinstanz durch den Dienstanbieter übermittelt werden, so dass der Dienstanbieter das ursprüngliche Kennzeichen nicht erfährt, die Anonymitätsinstanz die anonyme Autorisierung aber durchführen kann.

1. Das Subjekt generiert eine zufällige Zahl (`cipherKey`), die gleich lang ist wie das Kennzeichen der Anonymitätsdokumente (`pseudoIdentity`). Dann rechnet das Subjekt die folgenden zwei Zahlen aus.

   (1)

   $\texttt{OTpseudoIdentity} = \texttt{pseudoIdentity} \oplus \texttt{cipherKey}$

   Bitweise Exklusiv-Oder (XOR) Operation ($\oplus$) von `pseudoIdentity` mit `cipherKey`:

   $0 \oplus 0 = 0;\ 0 \oplus 1 = 1;\ 1 \oplus 0 = 1;\ 1 \oplus 1 = 0$

   (2)

   $\texttt{encryptedCipherKey} = \texttt{cipherKey}^{\texttt{OTPIPubExp}} \bmod \texttt{OTPPIMod}$

   RSA-Operation von `cipherKey` mit dem vorher erhaltenen öffentlichen RSA-Schlüssel.

   > Sei $\texttt{pseudoIdentity} = 24A64772_{16}$ und $\texttt{cipherKey} = 3A2ECD01_{16}$,
   > dann wird $\texttt{OTpseudoIdentity} = 1E888A73_{16}$ und
   > $\texttt{encryptedCipherKey} = 509DA2A6_{16}$.

2. Die Anonymitätsinstanz entschlüsselt mit Hilfe des Exponenten des passenden geheimen Schlüssels (`OTPIPrivExp`) die verschlüsselten Daten:

   (3)

   $\texttt{cipherKey} = \texttt{encryptedCipherKey}^{\texttt{OTPIPrivExp}} \bmod \texttt{OTPIMod}$

   RSA-Operation von `encryptedCipherKey` mit dem geheimen RSA-Schlüssel.

   (4)

   $\texttt{pseudoIdentity} = \texttt{OTpseudoIdentity} \oplus \texttt{cipherKey}$

   XOR-Operation von `OTpseudoIdentity` mit `cipherKey`.

   > Entsprechend sei $\texttt{OTPIPrivExp} = 4EAAFE01_{16}$, damit ergibt sich
   > $\texttt{cipherKey} = 3A2ECD01_{16}$ und $\texttt{pseudoIdentity} = 24A64772_{16}$ wie gehabt.

4.3.3.2.2 Realisierung in AEP

Dieser Abschnitt beschreibt, wie die im vorigem Abschnitt behandelten kryptographischen Transformationen innerhalb AEP realisiert werden, und in welchen Nachrichten die verschiedenen Zahlen übermittelt werden.

4.3.3.2.2.1 Initialisierungsprozess

Nachdem die Methode der einmaligen Pseudoidentität vereinbart wurde, werden die Nachrichten folgendermaßen gestaltet.

- Das Subjekt fordert den öffentlichen RSA-Schlüssel der Methode von der Anonymitätsinstanz an (`RequestInfo`-Nachricht).
- Die Anonymitätsinstanz antwortet auf die Anfrage mit dem Schlüssel (`ResponseInfo`-Nachricht). Somit erhält das Subjekt `OTPIMod` und `OTPIPubExp`.
- Die Anonymitätsinstanz schickt als letzte Nachricht (`PreparationFinished`) die Anonymitätsdokumente. Diese enthalten die Pseudoidentität des Subjekts (`pseudoIdentity`) für den gewünschten Diensttyp. Die so erhaltenen Anonymitätsdokumente unterscheiden sich im Wesentlichen nicht von denjenigen, die das Subjekt mit Hilfe des Verfahrens der Pseudoidentität erwirbt.

4.3.3.2.2.2 Anonyme Autorisierung

- Das Subjekt transformiert die Anonymitätsdokumente – gemäß (1) – und schickt das Ergebnis dann dem Dienstanbieter (`RequestService`), welcher die Nachricht zur Anonymitätsinstanz weiterleitet.
- Die Anonymitätsinstanz schickt die `Challenge`-Nachricht zurück zum Dienstanbieter, welcher sie dann zum Subjekt weiterleitet.
- Der in der `Challenge`-Nachricht enthaltene zufällige Text muss vom Subjekt digital signiert werden, damit seine Identität bestätigt werden kann. Diese digitale Signatur muss mit demjenigen geheimen Schlüssel erstellt werden, welchen das Subjekt beim Verbindungsaufbau der Sicherheitsschicht während des Initialisierungsprozesses benutzt hat.

  Das Subjekt erstellt die `Response`-Nachricht, welche neben der berechneten digitalen Signatur die gemäß (2) erstellte `encryptedCipherKey`-Zahl als Ergänzung (*extension*) enthält, und schickt die Nachricht dem Dienstanbieter, welcher sie zur Anonymitätsinstanz weiterleitet.
- Falls die Identität des Subjekts so bewiesen werden konnte, erteilt die Anonymitätsinstanz die Bewilligung, und schickt dem Dienstanbieter die `Approval` Nachricht, welche schließlich auch das Subjekt erreicht.

4.3.3.2.2.3 Bemerkungen

- Da das Subjekt während jeder anonymen Autorisierung einen neuen `cipherKey` generiert, wird die einmalige Pseudoidentität (`OTpseudoIdentity`), die der Dienstanbieter sieht, jedes mal eine andere.
- Der Dienstanbieter ist nicht in der Lage, die einmalige Pseudoidentität zu entschlüsseln und die eigentliche Pseudoidentität (`pseudoIdentity`) zu erfahren, da er dafür den geheimen Schlüssel des Verfahrens bräuchte (um zuerst `encryptedCipherKey` zu entschlüsseln und dann `OTpseudoIdentity`), den aber nur die Anonymitätsinstanz

kennt. Ohne den geheimen Schlüssel ist die Entschlüsselung des RSA-verschlüsselten `encryptedCipherKey` praktisch unmöglich (bei genügend langem Schlüssel).

4.3.3.3 Blinde Unterschrift (*blind signature*)

Diese Methode wurde von **David Chaum** beschrieben [CHAUM-1] [CHAUM-2] [CHAUM-3] und wird im Abschnitt 2.2.3.2.1 eingestuft und behandelt. Mit ihrer Hilfe kann das Niveau des nicht zurückverfolgbaren Subjekts erreicht werden. In diesem Abschnitt werden die kryptographischen Grundlagen behandelt und die Realisierung innerhalb von AEP beschrieben.

4.3.3.3.1 Kryptographische Grundlagen

In diesem Abschnitt werden die kryptographischen Grundlagen der Methode der blinden Unterschrift näher betrachtet. Die während des Initialisierungsprozesses und der anonymen Autorisierung durchzuführenden Transformationen werden beschrieben und an einem konkreten Beispiel auch vorgeführt.

4.3.3.3.1.1 Initialisierungsprozess

Während des Initialisierungsprozesses werden die folgenden kryptographischen Operationen durchgeführt. Als Ergebnis erlangt das Subjekt die Erlaubnis, dass in diesem Fall den wichtigsten Teil der Anonymitätsdokumente repräsentiert.

1. Das Subjekt bekommt von der Anonymitätsinstanz den öffentlichen RSA-Schlüssel, der zum Typ des Dienstes passt. Dieser besteht aus einem Exponenten (`STPubExp`) und einem Modulus (`STMod`).

   Z. B. sei `STPubExp = 65537` und `STMod = 1973776333`.

2. Das Subjekt generiert zwei zufällige Zahlen, das Kennzeichen (`randomNumber`) und den Umschlag (`randomEnvelope`). Das Subjekt bestimmt die folgende spezielle Zahl, wobei `randomNumber2` durch Verdoppelung der Ziffernfolge von `randomNumber` entsteht.

   Am Ende des Initialisierungsprozesses entsteht das Erlaubnis in Form einer RSA-Signatur von `randomNumber2` mit dem geheimen Schlüssel des Diensttyps. Um die RSA-Signatur zu überprüfen, muss daran zuerst eine RSA-Operation mit dem öffentlichen Schlüssel durchgeführt werden. Damit die Authentizität der so entstandenen Zahl überprüft werden kann, muss die ursprüngliche `randomNumber2` spezielle Eigenschaften haben. Die einfachste Möglichkeit, dass zu erreichen, ist die Verdopplung der Ziffernfolge.

   Da praktisch keiner außer der Anonymitätsinstanz eine Zahl generieren kann, die nach der erwähnten RSA-Operation in einer verdoppelten Ziffernfolge resultiert, wird somit die Authentizität erreicht.

   (1)

   $$\texttt{specNumberInEnvelope} = (\texttt{randomNumber2} \cdot \texttt{randomEnvelope}^{\texttt{STPubExp}}) \bmod \texttt{STMod}$$

   Falls `randomNumber = 6535`, `randomNumber2 = 65356535` und `randomEnvelope = 7823`, dann ist `specNumberInEnvelope = 389358663`.

3. Die Anonymitätsinstanz transformiert diese spezielle Zahl in eine andere. Diese Transformation ist eigentlich eine RSA-Operation. Dabei ist `STPrivExp` der Exponent des geheimen Schlüssels, dessen öffentlicher Schlüssel vorher schon benutzt wurde.

   (2)

   $$signedSpecNumberInEnvelope = specNumberInEnvelope^{STPrivExp} \bmod STMod$$

   Entsprechend sei `STPrivExp = 1209262233` und damit ist `signedSpecNumberInEnvelope = 1176235684`.

4. Das Subjekt transformiert diese spezielle Zahl in die endgültige Form wobei $randomEnvelope^{-1}$ die multiplikative Inverse von `randomEnvelope` modulo `STMod` ist.

   (3)

   $$randomEnvelope \cdot randomEnvelope^{-1} = 1 \bmod STMod$$

   (4)

   $$signedSpecNumber = signedSpecNumberInEnvelope \cdot randomEnvelope^{-1} \bmod STMod$$

   Im Beispiel ist $randomEnvelope^{-1} = 1323840524$ und `signedSpecNumber = 1613383882`.

Die durch diese Transformationen entstandene `signedSpecNumber` genügt auch der folgenden Gleichung und kann somit als die digitale RSA Signatur von `randomNumber2` betrachtet werden:

(5)

$$signedSpecNumber = randomNumber2^{STPrivExp} \bmod STMod$$

4.3.3.3.1.2 Anonyme Autorisierung

Während der anonymen Autorisierung kann die folgende kryptographische Transformation durchgeführt werden, um die Authentizität einer potenziellen Erlaubnis (`potentialPermission`) zu überprüfen:

(6)

$$potentialRandomNumber2 = potentialPermission^{STPubExp} \bmod STMod$$

Falls die Zahl `potentialRandomNumber2` eine verdoppelte Ziffernfolge ist, dann ist die Authentizität der Erlaubnis bewiesen.

> Falls `potentialPermission = 1613383882`, dann wird gemäß der Gleichung `potentialRandomNumber2 = 65356535` und dadurch wird die Authentizität der Erlaubnis bewiesen.
>
> Versucht man jedoch eine ungültige Erlaubnis, so wird das gleich erkannt. Sei z. B. `potentialPermission = 8765`, dann wird `potentialRandomNumber2 = 1886527860` und damit wird die Ungültigkeit der Erlaubnis bewiesen.

Die Anonymitätsinstanz muss jedoch auch sicherstellen, dass eine Erlaubnis nicht mehrmals benutzt wird.

4.3.3.3.1.3 Bemerkungen

- Am Ende des Initialisierungsprozesses erlangt das Subjekt in der Form der Zahl `signedSpecNumber` die Erlaubnis.
- Bis zum Beginn der anonymen Autorisierung kennt nur das Subjekt die endgültige Erlaubnis.
- Die Authentizität der Erlaubnis kann von jedem gemäß (6) überprüft werden, man benötigt dafür nur den öffentlichen RSA-Schlüssel, der zum Typ des Dienstes passt.
- Das Subjekt allein ist nicht in der Lage, eine gültige Erlaubnis zu erstellen. Es ist praktisch unmöglich, ohne den geheimen Schlüssel eine Zahl zu generieren[22], die dann als authentische Erlaubnis anerkannt wird. Deswegen muss eine authentische und noch nicht benutze Erlaubnis ohne weiteres Überprüfen akzeptiert werden.

4.3.3.3.2 Realisierung in AEP

Dieser Abschnitt beschreibt, wie die im vorigen Abschnitt behandelten kryptographischen Transformationen innerhalb von AEP realisiert werden, also in welchen Nachrichten die verschiedenen Zahlen übermittelt werden.

4.3.3.3.2.1 Initialisierungsprozess

Nachdem die Methode der blinden Unterschrift vereinbart wurde, werden die Nachrichten folgendermaßen gestaltet:

- Das Subjekt kann den zum Typ des Dienstes gehörenden RSA-Schlüssel von der Anonymitätsinstanz abfragen (`RequestInfo`-Nachricht).
- Die Anonymitätsinstanz antwortet auf die Abfrage mit dem Schlüssel (`ResponseInfo`-Nachricht). Somit erhält das Subjekt `STMod` und `STPubExp`.
- Das Subjekt erstellt die Zahl `specNumberInEnvelope` – gemäß (1) – und schickt diese der Anonymitätsinstanz (`Envelope`-Nachricht).
- Die Anonymitätsinstanz generiert die Zahl `signedSpecNumberInEnvelope` – gemäß (2) –, speichert relevante Daten über diesen Vorgang und schickt diese Zahl zurück zum Subjekt (`PreparationFinished` Nachricht).
- Das Subjekt transformiert die erhaltene Vorerlaubnis gemäß (4) in die endgültige Erlaubnis und speichert das Resultat.

4.3.3.3.2.2 Anonyme Autorisierung

Während der anonymen Autorisierung werden die folgenden Nachrichten in der AEP-Kommunikation benutzt:

- `RequestService` (AS ⇨ SP): Das Subjekt schickt die Anonymitätsdokumente (die Erlaubnis mit der `signedSpecNumber`-Zahl) dem Dienstanbieter.

---

[22] Falls die Voraussetzung besteht, dass der Schlüssel (`STMod` und `STPrivExp`) bzw. `randomNumber` lang genug (z. Z. 1024 Bits, siehe [SCHIPSEY-1]) sein müssen.

- `RequestService` (SP ⇨ AA): Der Dienstanbieter leitet diese Nachricht (ohne die Beschreibung des Dienstes) an die Anonymitätsinstanz weiter.
- `Approval` (SP ⇨ AA): Falls die Erlaubnis gemäß (6) überprüft wurde und noch unbekannt ist, so kann die Bewilligung dem Dienstanbieter erteilt werden.
- `Approval` (SP ⇨ AS): Die erhaltene Bewilligung wird ans Subjekt weitergeleitet.

4.3.3.3.2.3 Bemerkungen

- Die Anonymitätsinstanz stellt zu jedem Diensttyp einen anderen RSA Schlüssel bereit.
- Während der Erstellung der Vorerlaubnis (2) ist die Identität des Subjekts bekannt. Damit können Informationen über die Ausstellung einer Vorerlaubnis eines bestimmten Typs gespeichert werden.

  Im Falle der Ausstellung einer digitalen Banknote kann dabei der Betrag vom Konto des Subjekts abgezogen werden.

- Während der anonymen Autorisierung können bei der Anonymitätsinstanz Informationen über die Benutzung einer bestimmten Erlaubnis bei einem bestimmten Dienstanbieter gespeichert werden.

  Im Falle einer Bezahlung mit digitalen Banknoten kann der Betrag dem Dienstanbieter gutgeschrieben werden.

5. Das implementierte System

Während der Spezifikation des Protokolls wurde auch für wichtig erachtet, dass neben der theoretischen Spezifikation eine benutzbare Implementierung erstellt wird, auf die weitere Anwendungen aufbauen können.

Dieses Kapitel beschreibt die Implementierung der Version 0.2 von AEP. Zuerst werden Implementierungsdetails bezüglich der Sicherheitsschicht behandelt, dann die Implementierung der Anonymitätsschicht ausführlich beschrieben und schließlich wird vorgestellt, wie die Interaktion zwischen Benutzer und dem Beispielsystem abläuft.

5.1 Allgemeine Bemerkungen

5.1.1 Implementierungsumgebung

Die Implementierungsumgebung, welche während der ganzen Periode der Entwicklung von AEP und der Beispielsanwendungen benutzt wurde, wird in diesem Abschnitt vorgestellt.

5.1.1.1 Benutzte Programmiersprache

Für die Implementierung wurde die Programmiersprache JAVA [JAVA-1] ausgewählt. Es stehen mehrere Gründe dafür:

- JAVA ist eine weitverbreitete objektorientierte Programmiersprache.
- Implementierungen in JAVA sind unabhängig vom Betriebssystem, da die JAVA-Programme auf einer virtuellen Maschine laufen.
- JAVA bietet genügend allgemeine und leicht anwendbare Möglichkeiten für die Behandlung der XML-Kommunikationssprache (*JAXP*) [XML-2].
- Es stand eine frei verfügbare Implementierung des für die Sicherheitsschicht ausgewählten SSH2-Protokolls bereit. (Team MONICA JavaCom SSH2 v0.31, [MONICA-1]).

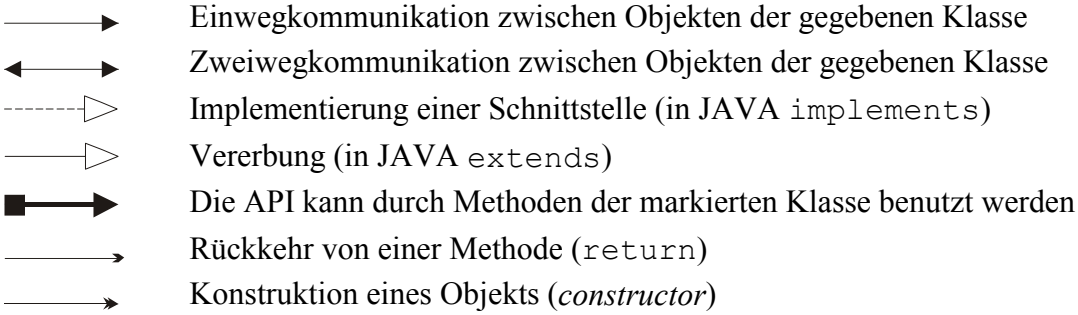
5.1.1.2 Benutzte Software

Für die Entwicklung wurde die folgende Software benutzt:

- Als Betriebsystem: SuSE Linux 7.3 [SUSE-1]
- Als graphische Oberfläche: KDE 2.2.1 [KDE-1]
- Als Entwicklungsumgebung: NetBeans IDE 3.3.1 [NETBEANS-1]
- Als Programmiersprache: J2SE 1.4 [J2SE-1]
- Als Webserver: J2EE 1.3.1 [J2EE-1]
- Als SQL-Server: MySQL 3.23.44 [MYSQL-1]

5.1.2 Benutze Notationen

Im Folgenden werden in Abbildungen die folgenden Notationen benutzt:



5.1.3 Auffinden der Implementierung

Anhang E beschreibt, wie die auf der beigefügten CD befindliche Implementierung installiert und ausprobiert werden kann.

Die Implementierung wird auch von der Webseite des *Team MONICA* [MONICA-1] erreichbar sein.

5.2 Implementierung der Sicherheitsschicht – SSH2

Für die Sicherheitsschicht, die AEP benutzt, wurde SSH2 ausgewählt. Während der Implementierung von AEP wurde eine frei verfügbare JAVA-Implementierung des SSH2-Protokolls [MONICA-1] benutzt. Für SSH2 sprechen die folgenden Vorteile:

- SSH2 ist ein vom IETF [IETF-1] standardisiertes Protokoll (siehe [SSH-1]).
- SSH2 garantiert die Vertraulichkeit der Kommunikation (durch die Verwendung von Verschlüsselungsalgorithmen, wie z. B. 3DES [DES-1] oder AES [AES-1]).
- SSH2 bietet die Integritätssicherung der Kommunikation an (durch die Verwendung von kryptographischen Prüfsummen, wie z. B. HMAC-SHA1, HMAC-MD5 [HMAC-1] [SHS-1] [MD5-1]).
- SSH2 ermöglicht die Authentisierung sowohl des Servers als auch des Clients (mit Hilfe von digitalen Signaturen, wie z. B. DSS [DSS-1] oder RSA [RSA-2]).
- SSH2 benutzt die Client-Server-Architektur.
- Mit SSH2 ist gesicherte und transparente Übertragung von Daten möglich (in SSH2-Kanälen).

Die Architektur der Implementierung der SSH2-Sicherheitsschicht ist in Abbildung 14 dargestellt.

Im Allgemeinen kann ein Anwender mit Hilfe der `SSH2Facade` einen SSH2-Client oder SSH2-Server starten. Es gibt jedoch zwei Hilfsklassen, die die Startprozedur vereinfachen (`SSH2ClientImpl` und `SSH2ServerImpl`).

Die SSH2-Funktionalität kann dann durch Objekte der Klassen `SSH2ClientSide` und `SSH2ServerSide` (und durch die für jeden Client konstruierten `SSH2Server`-Objekte) erreicht werden.

Die effektive Kommunikation über TCP/IP (Paketbildung und Dekodierung, Verschlüsselung, Entschlüsselung, Bildung und Überprüfen kryptographischer Prüfsummen) wird durch einen `SSH2Tokenizer` erledigt.

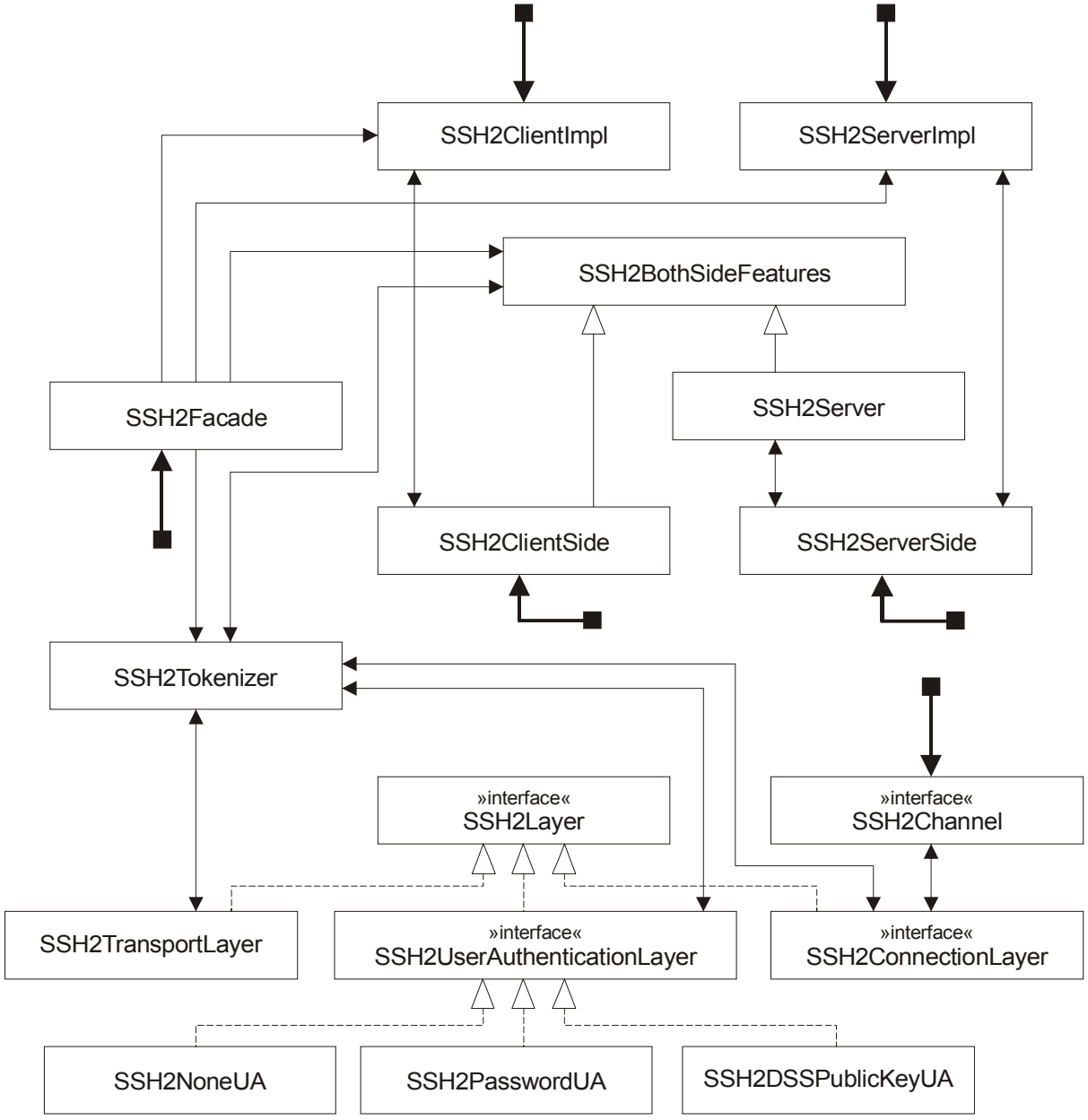


Abbildung 14 – **Architektur der SSH2 Sicherheitsschicht**

Der Schlüsselaustausch und die Authentisierung des Servers sind die Aufgaben des `SSH2TransportLayer` (*SSH2 transport layer protocol*, siehe 2.1.3.4.1).

Die Authentisierung eines Clients wird durch die `SSH2UserAuthenticationLayer` (*SSH2 authentication protocol*, siehe 2.1.3.4.2l) erledigt. Die derzeitige Implementierung unterstützt drei verschiedene Methoden für die Authentisierung des Clients: keine Authentisierung (`SSH2NoneUA` – *none*-Methode), Authentisierung mit Hilfe eines Passworts (`SSH2PasswordUA` – *password*-Methode) und Authentisierung mit einer digitalen Signatur nach dem DSS-Verfahren (`SSH2DSSPublicKeyUA` – *publickey*-Methode).

SSH2 transportiert Daten über transparente Kanäle (`SSH2Channel`). In einer SSH2-Verbindung über einen TCP/IP-Socket können mehrere solche Kanäle von beiden Seiten geöffnet werden. Diese Kanäle werden von der `SSH2ConnectionLayer` (*SSH2 connection protocol*, siehe 2.1.3.4.3) konstruiert. Darüberliegende Schichten können durch solche Kanäle gesichert miteinander kommunizieren.

5.3 Implementierung der Anonymitätsschicht – AEP

Während der Implementierung des Protokolls wurden zuerst die APIs (*application programming interfaces*) für die Behandlung der Präsentations- und der Kommunikationssprache erstellt. Danach wurde die SSH2-Implementierung ins Protokoll integriert. Damit wurde ermöglicht, Datenstrukturen der Präsentationssprache gemäß der Kommunikationssprache gesichert zu übertragen. In der nächsten Phase wurde der Kern des Protokolls, das Nachrichtenmanagement erstellt. Anschließend wurden die schon vorgestellten Anonymitätsverfahren in AEP implementiert. Parallel dazu wurden verschiedene Benutzeroberflächen und Beispielanwendungen bereitgestellt.

5.3.1 Behandlung der Präsentations- und Kommunikationssprache

Die erste Phase der Implementierung des Protokolls war die Erstellung der APIs für die Behandlung der Präsentations- (siehe Abschnitt C.1 im Anhang) und Kommunikations-sprachen (siehe Abschnitt C.2 im Anhang). Das Ergebnis sind die folgenden drei Teile der Implementierung:

- **CoreDefinition**: Die `CoreDefinition` transformiert in der Präsentationssprache (*langspec.in*, siehe Anhang D) formulierte Datenstrukturdefinitionen in eine kompaktere Form (*langspec.out*), die die weitere Benutzung durch AEP vereinfacht.
- **PLang-API**: die *PLang-API* ermöglicht, solche Variablen in JAVA zu benutzen, für deren Definition ein in der Präsentationssprache beschriebener Datentyp benutzt wurde. Die *PLang-API* wird benötigt, weil JAVA für bestimmte Datentypen der Präsentationssprache (*variant struct*, *enum*) keine nativen Datentypen besitzt.

  Zuerst wurde die **AEPType-API** erstellt (die ganze Klassenhierarchie der *AEPType-API* wird in der Abbildung 15 dargestellt). Diese enthält eine Klassenhierarchie, die ausgehend von einem allgemeinen und abstrakten AEP-Datentyp (`AEPType`) durch Vererbung ermöglicht, alle grundlegenden Datentypen der Präsentationssprache (`AEPStruct`, `AEPList`, `AEPBasic`, `AEPEnum`, usw.) zu benutzen. Gemäß dieser API, werden den Variablen, die einen *struct*, *list* oder *enum* repräsentieren, jeweils `AEPStruct`-, `AEPList`- oder `AEPEnum`-JAVA-Objekte zugeordnet. Dies bedeutet, dass z. B. zu einer `Alert`-Variablen genauso ein `AEPStruct`-Objekt zugeordnet wird, wie zu einer `DateType`-Variablen (sie werden jedoch verschieden initialisiert).

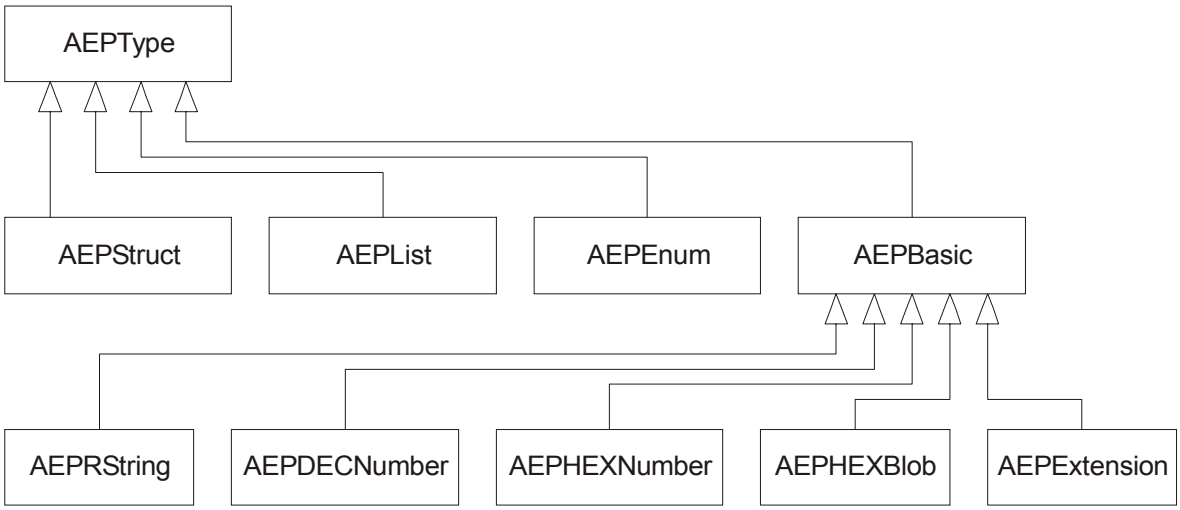


Abbildung 15 – **Klassenhierarchie der AEPType API**

Da die Benutzung der *AEPType-API* sich als ziemlich umständlich erwies, wurde für dieselbe Aufgabe als Ergänzung die **CG-API** entworfen. Die *CG-API* benutzt die *AEPType-API*, bietet jedoch für jeden Datentyp eine *wrapper*-Klasse an (z.B. `Alert` wird die Klasse `CG_Alert` zugeordnet, `DateType` die Klasse `CG_DateType`). Dadurch wird das Programmieren vereinfacht und viele Programmierfehler, die mit der *AEPType-API* nur während Laufzeit entdeckt werden konnten, können bereits zur Übersetzungszeit aufgeklärt werden.

Objekte der *AEPType-API* können mit der `CGManager` Klasse in Objekte der *CG-API* transformiert werden. Die Klassen der *CG-API* werden mit einem Werkzeug (`ClassGenerator`) automatisch generiert. Diese Klassen bieten eine bequeme und sichere Möglichkeit an und können überall, wo Variablen der *AEPType-API* benötigt werden, benutzt werden (z. B. die Klasse `CG_Alert` realisiert die Klasse `AEPStruct`, gemäß der Definition: `CG_Alert extends AEPStruct`).

Im Folgenden wird der Unterschied zwischen der *AEPType-API* und der *CG-API* demonstriert. Sei `token` eine `AEPStruct`-Variable, die die Anonymitätsdokumente (*token*) für einen bestimmten Diensttyp enthält. Beide Programmcodeteile erledigen dieselbe Aufgabe: Das eindeutige Kennzeichen (*tokenID*) der Anonymitätsdokumente (falls dieses existiert) soll ermittelt werden.

Das Kennzeichen kann gemäß der *AEPType-API* folgendermaßen ermittelt werden:

```
AEPType tokenSecurity = token.getField("security");
AEPType tokenID =
   ((AEPStruct)tokenSecurity).getField("tokenID");
String tokenIDString = ((AEPRString)tokenID)._toSring();
```

Gemäß der *CG-API* sieht das gleiche folgendermaßen aus (die Cast-Operationen wurden eliminiert, und anstatt Felder mit Strings zu identifizieren, sind sie durch Methoden erreichbar):

```
CG_Token tokenCG = (CG_Token)(CGManager.convertToCG(token));
String tokenIDString = tokenCG.CG_get_tokenSecurity().
   CG_get_tokenID()._toString();
```

- **CLang-API**: die *CLang-API* ermöglicht es, Variablen der Präsentationssprache (gemäß der *AEPType-API*) in ihre Repräsentation in der Kommunikationssprache zu transformieren und Variablen der Präsentationssprache mit gemäß der Kommunikationssprache erhaltenen Daten zu initialisieren.

  Dies bedeutet konkret, dass Variablen der *AEPType-API* mit Hilfe der *CLang-API* in Text – interpretierbar gemäß der Kommunikationssprache – umgewandelt werden können bzw. dass mit Hilfe der *CLang-API* Variablen der *AEPType-API* mit Text – interpretierbar gemäß der Kommunikationssprache – initialisiert werden können.

  Während der Implementierung wurde die Klasse `XMLCLang` erstellt, die die XML-Kommunikationssprache realisiert.

Abbildung 16 stellt die Relation der verschiedenen APIs zur Behandlung der Präsentations- und der Kommunikationssprache dar.

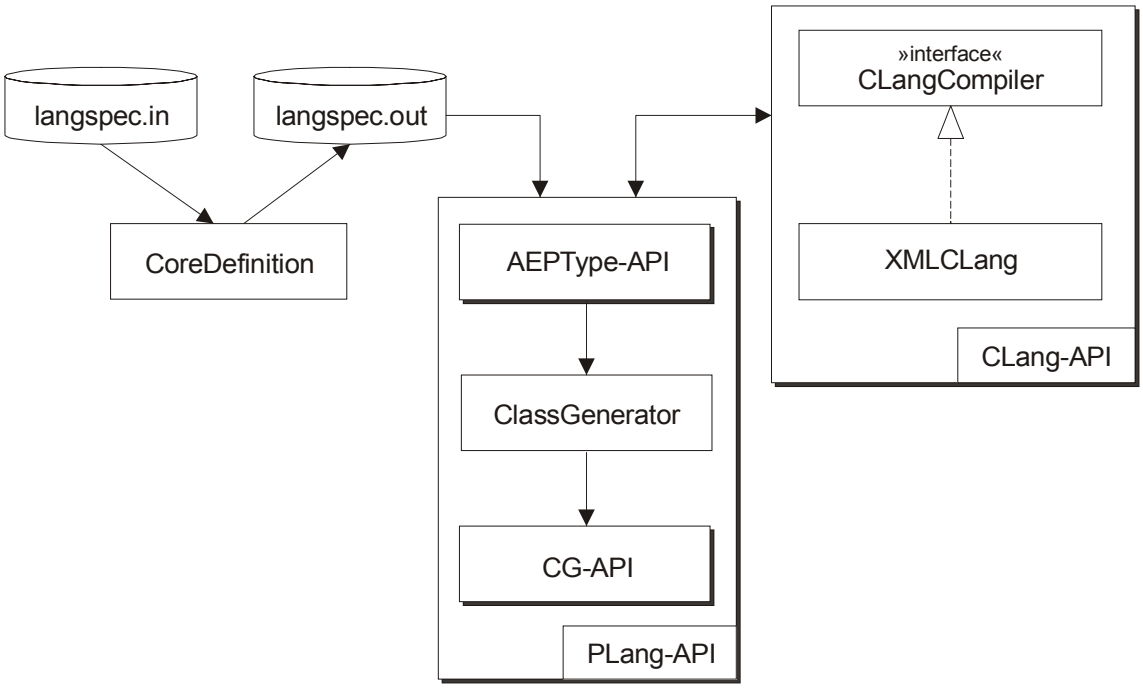


Abbildung 16 – **Relation der verschiedenen APIs zur Behandlung der Präsentations- und der Kommunikationssprache**

5.3.2 Verbindung zwischen AEP und der Sicherheitsschicht

Die Hauptklasse der AEP-Implementierung, die das Protokoll initialisiert, trägt den Namen `AEPMain`. Gemäß der Spezifikation können fünf verschiedene AEP-Instanzen unterschieden werden (bei der Anonymitätsinstanz, beim Dienstanbieter, beim Subjekt, beim Administrator der Anonymitätsinstanz und schließlich beim Administrator des Dienstanbieters) und alle diese werden von `AEPMain` gestartet (gemäß der gegebenen Konfigurationsdateien). Bei den zwei Serverinstanzen werden die SSH2-Server auch von `AEPMain` gestartet, die SSH2-Clienten werden jedoch von den Objekten der jeweiligen graphischen Oberfläche (siehe 5.3.5.1 und 5.3.5.2) gestartet.

Abbildung 17 stellt die Verknüpfungen zwischen Objekten der SSH2-Sicherheitsschicht und Objekten der AEP-Anonymitätsschicht dar.

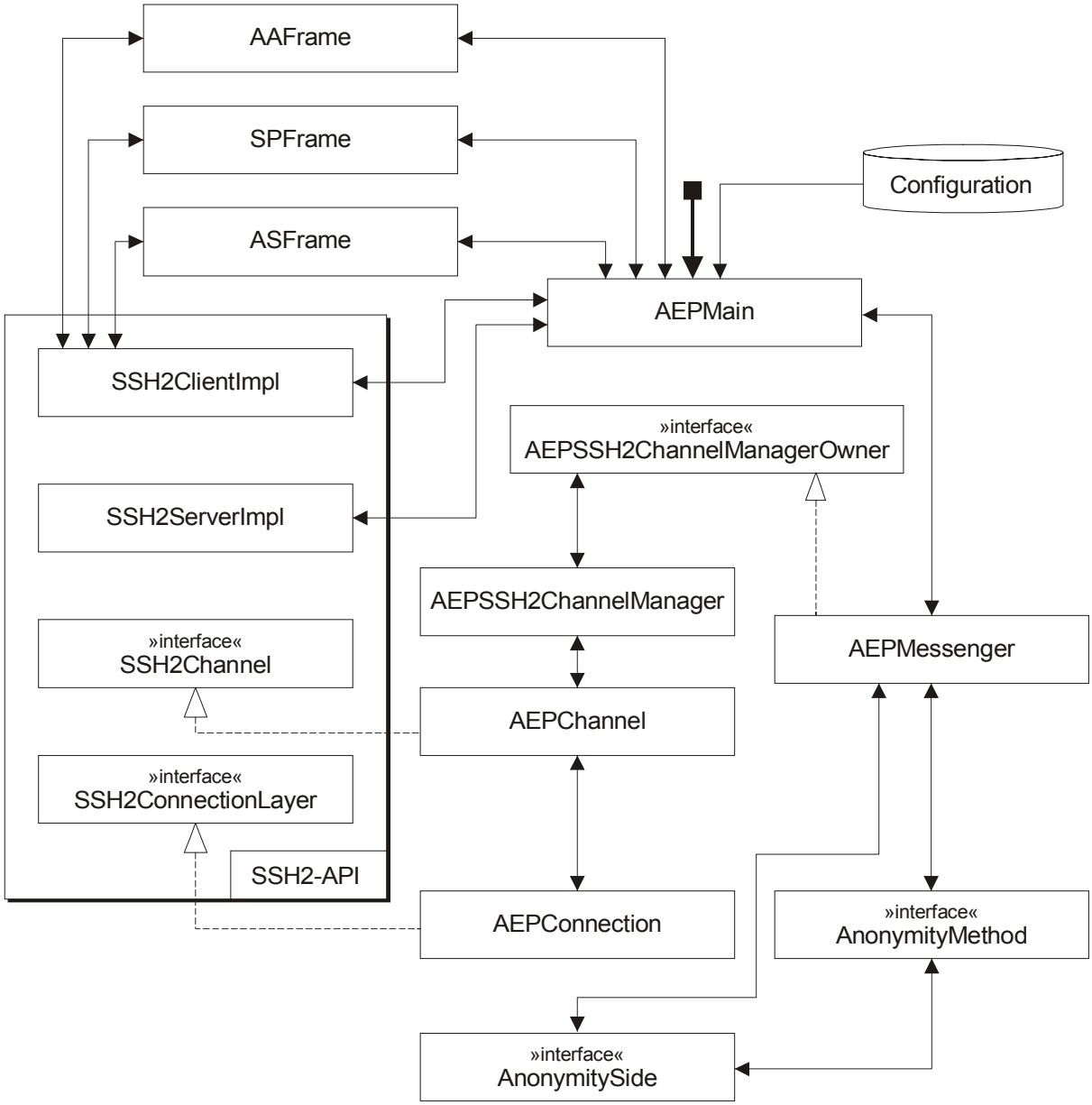


Abbildung 17 – **Verbindung von AEP und der SSH2-API**

Wurde ein SSH2-Kanal (in Form eines `AEPChannel` Objekts) erfolgreich aufgebaut, so wird ihm ein `AEPSSH2ChannelManager`-Objekt zugewiesen. Die Aufgabe solcher Objekte ist es, mittels der *CLang-API* die aus der SSH2-Verbindung gelesenen Daten in Nachrichten der Präsentationssprache zu transformieren und Nachrichten gemäß der Kommunikationssprache durch den zugewiesenen Kanal zu schicken.

Bei AEP-Kommunikationsvorgängen, die den Initialisierungsprozess oder die anonyme Autorisierung abwickeln, wird zu jedem `AEPSSH2ChannelManager`-Objekt ein `AEPMessenger`-Objekt konstruiert. Die Aufgabe solcher Objekte ist es, die für AEP

benötigten Nachrichten zu erstellen und der Partnerseite zu übermitteln oder die erhaltenen Nachrichten zu interpretieren. Konkret bedeutet dies, dass der Initialisierungsprozess bzw. die anonyme Autorisierung ursprünglich immer von einem `AEPMessenger`-Objekt gestartet wird (mit einer `StartPreparation`- bzw. einer `RequestService`-Nachricht).

Wurde während des AEP-Kommunikationsvorgangs schon ermittelt, welches Anonymitätsverfahren benutzt wird, so werden dementsprechend jeweils ein Objekt für das Verfahren und eines für die gegebene Rolle in AEP konstruiert. Diese Objekte implementieren einerseits das `AnonymityMethod`-Interface (z. B. die `PseudoIdentity`-Klasse für die Pseudoidentität) und andererseits das `AnonymitySide`-Interface (dementsprechend dann entweder `AAPseudoIdentity` bei der Anonymitätsinstanz, `SPPseudoIdentity` beim Dienstanbieter oder `ASPseudoIdentity` beim Subjekt). Diese `AnonymitySide`-Objekte werden für die Erstellung und Behandlung der für das Anonymitätsverfahren spezifischen Nachrichten benutzt.

5.3.3 Funktionsweise der Protokollimplementierung

In diesem Abschnitt wird das Innenleben der Protokollimplementierung ein bisschen näher betrachtet.

In Abbildung 18 werden die vier charakteristischen Sequenzen von Methodenaufrufen dargestellt. Obwohl diese vier illustrierten Phasen allgemein in der abgebildeten Reihenfolge ablaufen, kann es sehr wohl vorkommen, dass sich bestimmte Phasen mehrmals wiederholen oder dass zwischen zwei Phasen auch andere Methoden zwischen den abgebildeten Klassen aufgerufen werden.

Die vier abgebildeten Phasen sind die Folgenden:

- **Bereitstellung für die AEP-Kommunikation**: Die `AEPMain`-Klasse startet die SSH2-Sicherheitsschicht, und nachdem ein Kanal geöffnet wurde, wird zuerst das zugehörige `AEPSSH2ChannelManager`-Objekt konstruiert, danach das `AEPMessenger`-Objekt, schließlich wird das `AEPMessenger`-Objekt mit der passenden Methode (`startAS`, `startSP` bzw. `startAA`) gestartet.

- **Bereitstellung der gewählten Anonymitätsmethode**: Nachdem das gewünschte Anonymitätsverfahren vereinbart wurde, wird das entsprechende `AnonymityMethod`-Objekt (z. B. `Chaum`), und von diesem wiederum das zugehörige `AnonymitySide`-Objekt (`AAChaum`, `ASChaum` oder `SPChaum`) konstruiert.

- **Versand einer Nachricht**: Falls eine Nachricht zum Partner verschickt werden muss, so wird dies mittels der `sendMessage`-Methode des `AEPMessenger`-Objekts erledigt. Die Nachricht wird (nach eventuellen Transformationen bzw. Speicherung relevanter Daten) zum `AEPSSH2ChannelManager`-Objekt weitergeleitet (`send`-Methode). Dieses transformiert die Nachricht der Präsentationssprache mittels einem `CLangCompiler` der *CLang-API* in die Kommunikationssprache (`toCLang`-Methode), und schließlich wird der resultierende Text durch den Kanal verschickt (`write`-Methode).

- **Empfang einer Nachricht**: Nachdem der Kanal Daten empfangen hat, wird das entsprechende `AEPSSH2ChannelManager`-Objekt mit der `readEvent`-Methode benachrichtigt. Mittels der `getArrived`-Methode wird der empfangene Text abgefragt. Nachdem das `CLangCompiler`-Objekt den Text der Kommunikationssprache erfolgreich in eine Nachricht der Präsentationssprache umgewandelt hat, wird das entsprechende `AnonymitSide`-Objekt mittels `messageArrived` benachrichtigt.

  Zwischen `AEPChannel` und `AEPSSH2ChannelManager` ist der Datentransfer asynchron ereignisbasiert (*asynchronous event*), das `CLangCompiler`-Objekt muss vom `AEPSSH2ChannelManager` abgefragt werden, ob Daten abgeholt werden können (*polling*), und zwischen `AEPSSH2ChannelManager` und `AEPMessenger` bzw. `AEPMessenger` und `AnonymitSide` ist die Kommunikation synchron ereignisbasiert (*synrchronous event*).

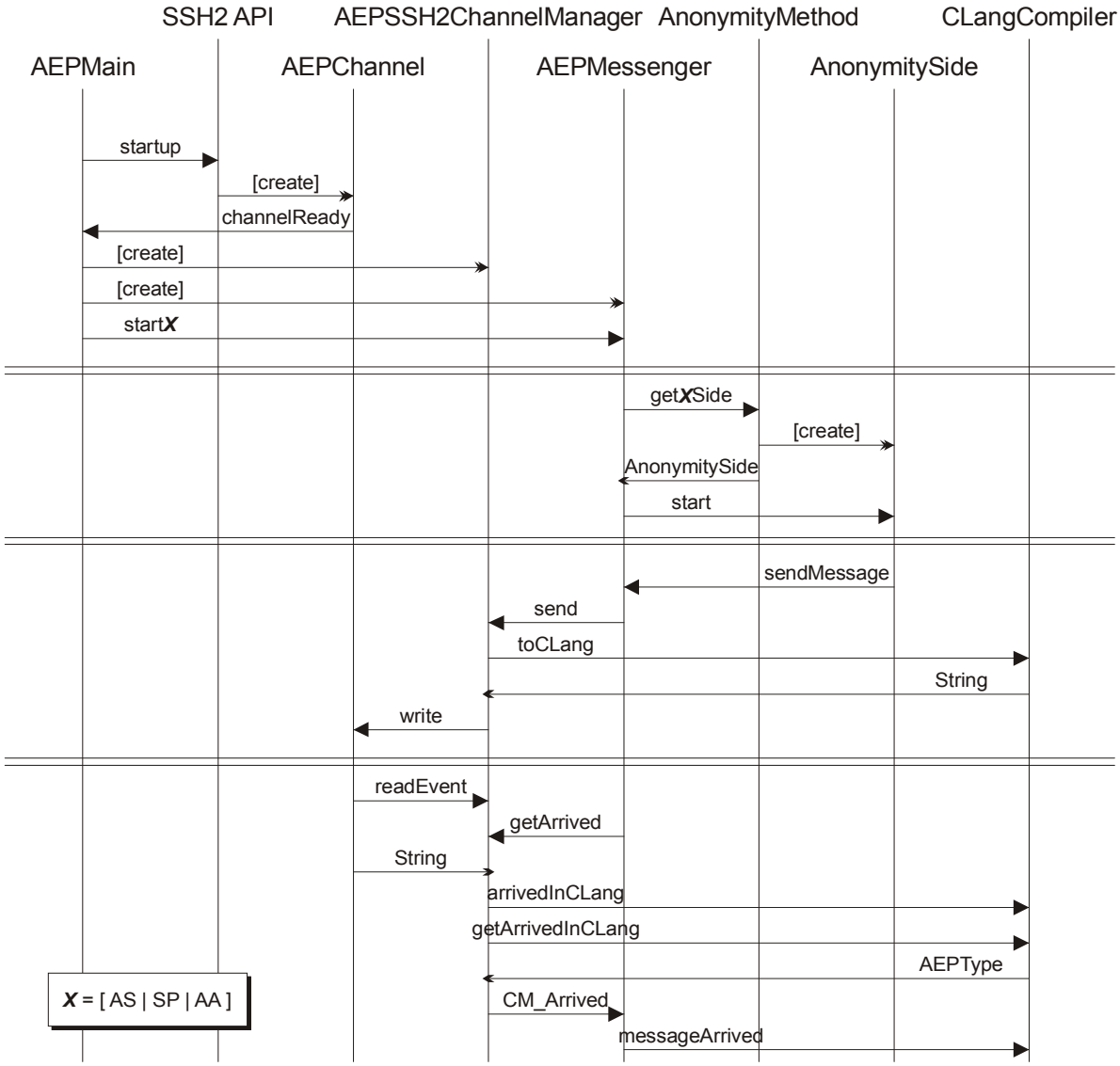


Abbildung 18 – **Methodenaufrufe und Rückkehrwerte innerhalb einer AEP-Instanz**

5.3.4 Implementierung von Anonymitätsverfahren

Damit das Protokoll auch benutzt werden kann, mussten neben der Implementierung der Kernfunktionen von AEP auch Anonymitätsverfahren in seinem Rahmen implementiert werden.

Deshalb wurden die schon ausführlich beschriebenen Anonymitätsmethoden der Pseudoidentität (`PseudoIdentity`, siehe 2.2.3.1.1 und 4.3.3.1), der einmaligen Pseudoidentität (`OTPseudoIdentity`, siehe 4.3.3.2) und der blinden Unterschrift (`Chaum`, siehe 2.2.3.2.1 und 4.3.3.3) in AEP implementiert. Die dabei erstellten Klassenbezeichnungen und ihre Relationen folgen dem allgemeinen Schema in Abbildung 19.

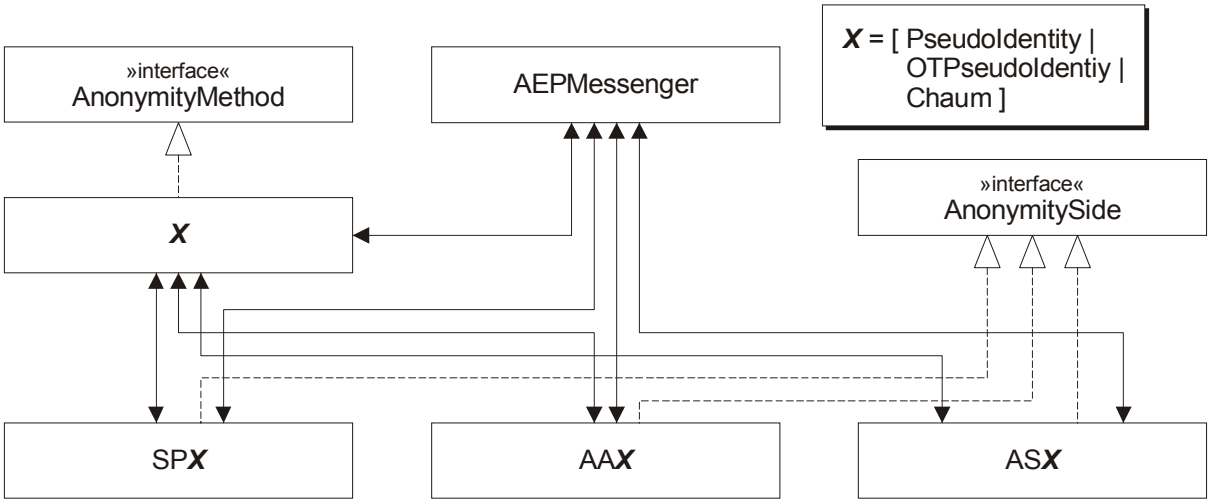


Abbildung 19 – **Allgemeines Schema für die Klassenhierarchie der implementierten Anonymitätsverfahren**

5.3.5 Interaktion mit dem Benutzer

Um das Protokoll ausprobieren zu können, wurden verschiedene, nicht streng AEP-bezogene Programmteile geschrieben. Einerseits wurde für das Subjekt eine graphische Oberfläche entwickelt, mit deren Hilfe sowohl der Initialisierungsprozess als auch die anonyme Autorisierung abgewickelt werden können. Andererseits wurden für die Server-Instanzen von AEP (Anonymitätsinstanz und Dienstanbieter) graphische Oberflächen für die Administration erstellt. Diese werden im aktuellen Abschnitt behandelt.

5.3.5.1 Inanspruchnahme eines Dienstes

Die Inanspruchnahme eines Dienstes mittels AEP wurde im Abschnitt 3.2.2 ausführlich behandelt und in Abbildung 8 dargestellt. In diesem Abschnitt werden zuerst einige Implementierungsdetails behandelt und dann die vom Subjekt während der Inanspruchnahme eines Dienstes benutzte graphische Oberfläche vorgestellt.

5.3.5.1.1 Implementierungsdetails

Nachdem das Subjekt die Anonymitätsdokumente erfolgreich erworben hat, kann die eigentliche Inanspruchnahme des Dienstes gestartet werden. Dazu muss jedoch das Subjekt zuerst anonym autorisiert werden.

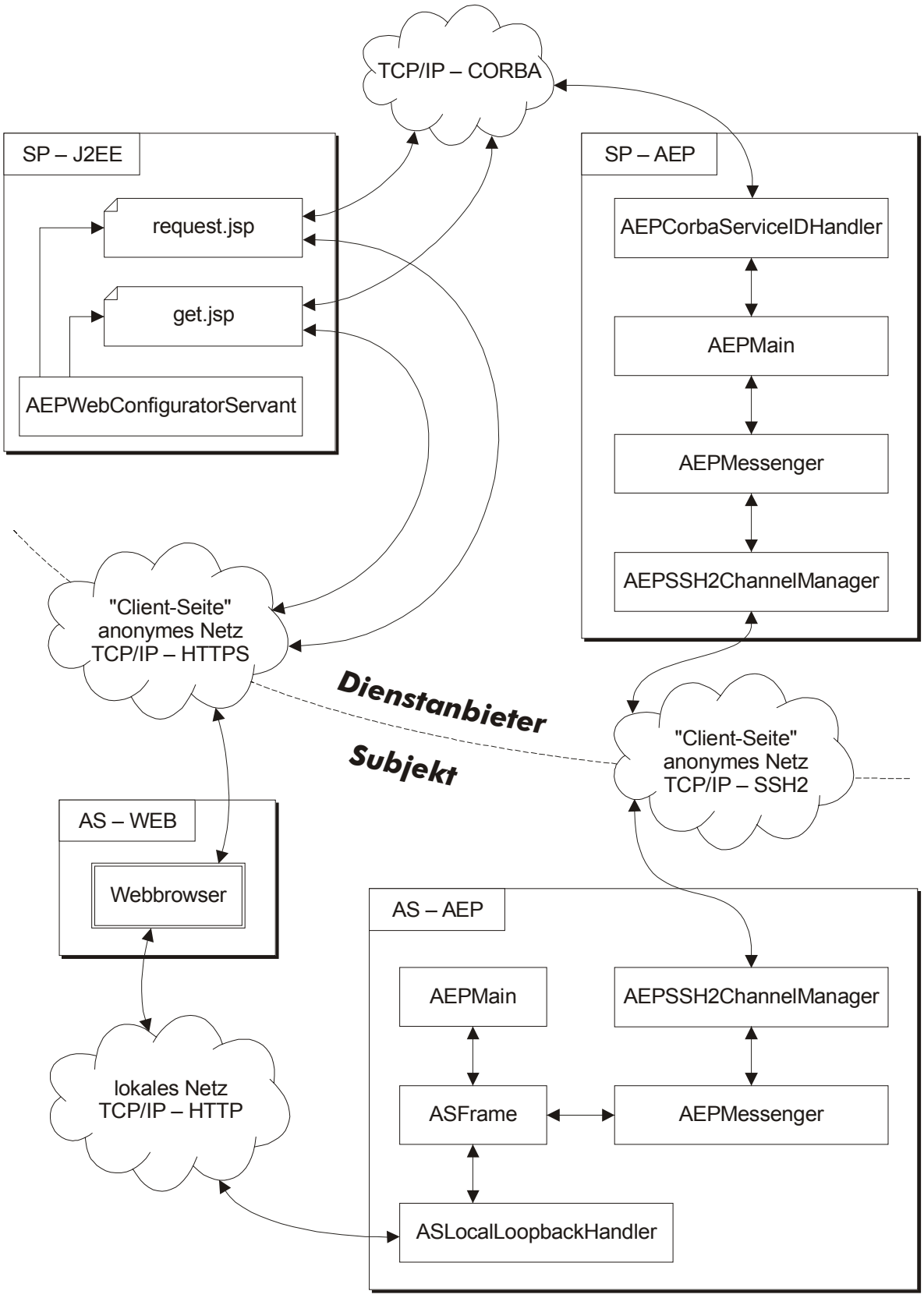


Abbildung 20 – **Klassenhierarchie, die während der Inanspruchnahme eines Dienstes benutzt wird**

Abbildung 20 zeigt die hier beschriebene Klassenhierarchie, die für die erfolgreiche Inanspruchnahme eines Dienstes verantwortlich ist.

Der Dienst wird vom Webbrowser des Subjekts angefordert. Dies geschieht durch das Abrufen einer HTML-Seite mittels HTTPS. Das Subjekt fordert die *request.jsp*-Seite vom Webserver des Dienstanbieters an. Diese Seite generiert eine entsprechende *serviceID*, teilt Daten über die erforderliche Autorisierung mittels CORBA dem `AEPCorbaServiceIDHandler`-Objekt mit und schickt schließlich dem Subjekt eine HTML-Antwort.

> Die Adresse des Webservers des Dienstanbieters sei *sp-web*, und dieser benutze den Port *port* benutzen. Dann sieht die Adresse, die in der Testimplementierung benutzt werden kann, folgendermaßen aus:
>
> *https://sp-web:port/aep_final_request/request.jsp?st=EUR_1&al=NON_TRACEABLE*
>
> Damit wird später die anonyme Autorisierung so durchgeführt, dass dafür Anonymitätsdokumente benötigt werden, die das Anonymitätsniveau des nicht zurückverfolgbaren Subjekts ermöglichen und für den Diensttyp EUR_1 ausgestellt wurden.

Nachdem der Webbrowser des Subjekts die Antwort-HTML-Seite erhalten hat, wird das `ASLocalLoopbackHandler`-Objekt benachrichtigt (in der derzeitigen Implementierung mit Hilfe von *meta refresh* mittels HTTP). Hier wird dann die anonyme Autorisierung des Subjekts gemäß der erhaltenen Daten gestartet. Wie dies technisch abläuft, wurde im Abschnitt 5.3.3 schon behandelt.

Nachdem der Dienstanbieter die Bewilligung von der Anonymitätsinstanz erhalten hat, wird zuerst die entsprechende *serviceID* beim `AEPCorbaServiceIDHandler` freigeschaltet, und erst dann wird die Bewilligung zum Subjekt weitergeleitet.

Der Webbrowser des Subjekts wird zu einer neuen Adresse geleitet, wo dann mittels der freigeschalteten *serviceID*, die hier als eine Art Einmalschlüssel funktioniert, der Dienst in Anspruch genommen werden kann.

> Die neue Adresse sieht folgendermaßen aus (falls *serviceID* = 1234).
>
> *https://sp-web:port/aep_final_get/get.jsp?serviceID=1234*

Beim Webserver des Dienstanbieters überprüft noch die *get.jsp*-Seite, ob die angeforderte *serviceID* freigeschaltet wurde (mittels CORBA beim `AEPCorbaServiceIDHandler`) und leistet dann den Dienst.

5.3.5.1.2 Initialisierungsprozess

Um die AEP-Funktionalität in Anspruch nehmen zu können, muss das Subjekt zuerst die entsprechende AEP-Anwendung starten.

Das Hauptfenster (siehe Abbildung 21) ist zweigeteilt, links befinden sich die gültigen Anonymitätsdokumente des Subjekts, rechts werden die verschiedenen Systemnachrichten, die während der AEP-Operationen generiert werden, angezeigt.

Von hier aus können weitere Anonymitätsdokumente angefordert werden (im *Token*-Menü) oder Einstellungen verändert werden (im *Configure*-Menü).

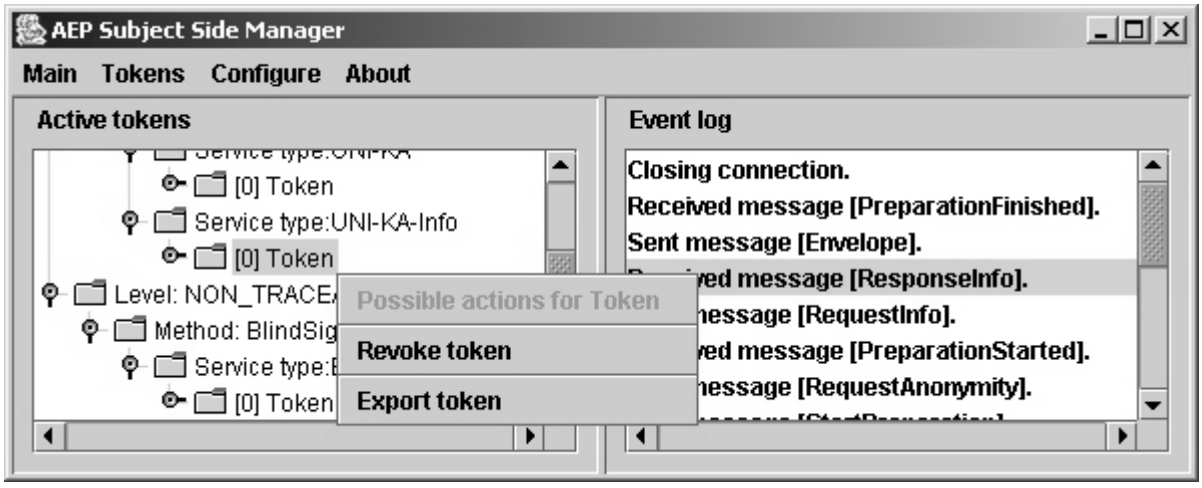


Abbildung 21 – **Hauptfenster der graphischen Oberfläche des Subjekts mit einem aktiviertem Popup-Menü**

Abbildung 22 zeigt das Dialogfenster, in dem die für den Initialisierungsprozess benötigten Daten vom Subjekt eingegeben werden können. Diese Daten werden einerseits für die SSH2-Verbindung gebraucht (Adresse und Port der Anonymitätsinstanz, sowie die Datei mit dem öffentlichen Schlüssel), andererseits wird hier angegeben, für welchen Diensttyp die Anonymitätsdokumente ausgestellt werden sollen und welches Anonymitätsniveau später erreicht werden soll.

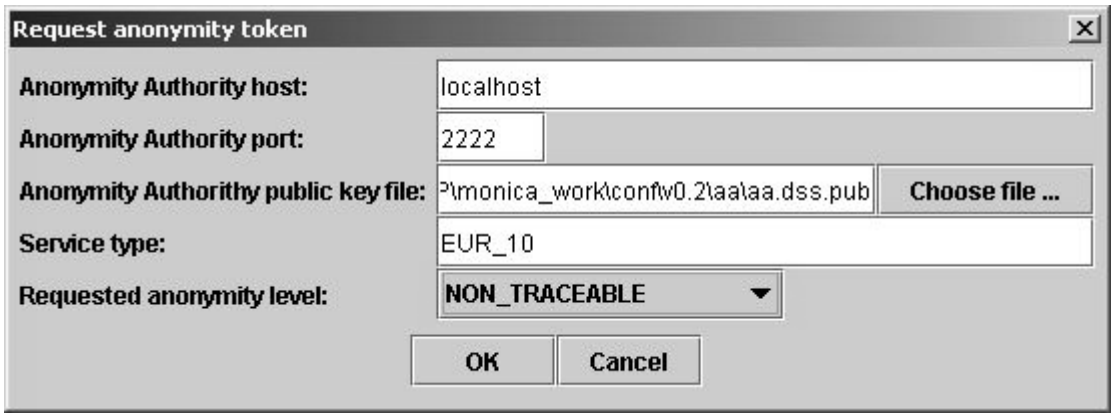


Abbildung 22 – **Dialogfenster zur Angabe der verschiedenen Daten, benötigt für den Initialisierungsprozess**

Nachdem der Initialisierungsprozess gestartet wurde, wird die AEP-Instanz des Subjekts die spezifizierten Anonymitätsdokumente anfordern. Werden diese empfangen, so werden sie in der Liste des Hauptfensters angezeigt. Die während des Initialisierungsprozesses generierten Systemnachrichten werden in der rechten Hälfte des Hauptfensters aufgelistet.

5.3.5.1.3 Anonyme Autorisierung

Die anonyme Autorisierung fängt mit einer Dienstanforderung im Webbrowser des Subjekts an. Nachdem der lokale *"loopback"*-Server die für die Autorisierung erforderlichen Daten erhalten hat, wird in einem Dialogfenster die Liste der möglichen Anonymitätsdokumente angezeigt (siehe Abbildung 23).

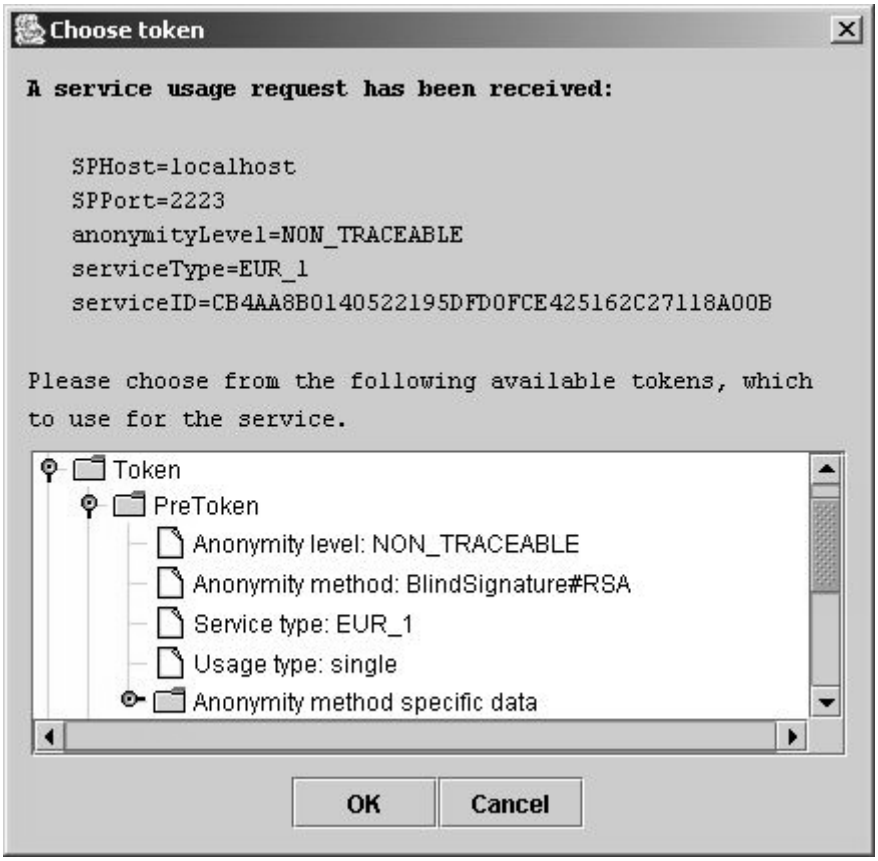


Abbildung 23 – **Dialogfenster, in dem das Subjekt die zu benutzenden Anonymitätsdokumente auswählen kann**

Nachdem das Subjekt die Anonymitätsdokumente ausgewählt hat, wird die anonyme Autorisierung gestartet. Wurde sie erfolgreich ausgeführt, so wird der Webbrowser benachrichtigt und somit kann dann das Subjekt den Dienst in Anspruch nehmen (siehe Abbildung 24).

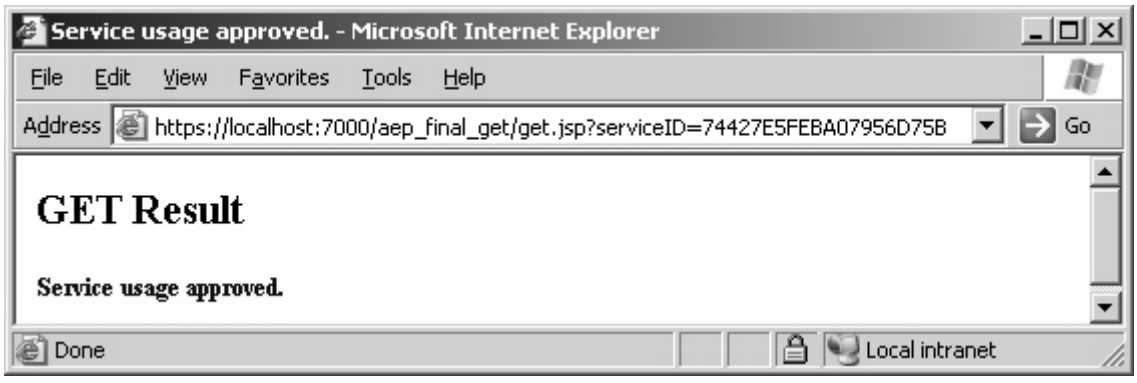


Abbildung 24 – **Ergebnis einer erfolgreichen anonymen Autorisierung**

Falls das Subjekt die für die anonyme Autorisierung erforderlichen Anonymitäts-dokumente nicht besitzt, so wird es benachrichtigt (siehe Abbildung 25) und die Autorisierung wird gar nicht gestartet.

Abbildung 25 – **Dialogfenster, das erscheint, falls das Subjekt nicht die zur anonymen Autorisierung benötigten Anonymitätsdokumente besitzt**

5.3.5.2 Administration von AEP-Serverinstanzen

Damit die Aktivitäten der zwei Serverinstanzen in AEP besser verfolgt werden können, wurden für beide Administrationsanwendungen erstellt. In diesem Abschnitt werden zuerst allgemeine Implementierungsdetails behandelt und dann die für die jeweiligen Anwendungen spezifischen Details erklärt.

5.3.5.2.1 Implementierungsdetails

Während der Administration einer Serverinstanz in AEP wird zwischen der graphischen Oberfläche des Administrators und der Serverinstanz eine SSH2-Verbindung aufgebaut und die eigentliche Administration erfolgt mittels spezieller Nachrichten (`AdminMessage`, siehe Abschnitt D.5 des Anhangs) der Präsentationssprache. Die allgemeine Klassenhierarchie, die für die Administration benutzt wurde, wird in Abbildung 26 gezeigt.

Die graphische Oberfläche wird von Objekten der Klassen `AAFrame` (für die Anonymitätsinstanz) bzw. `SPFrame` (für den Dienstanbieter) zur Verfügung gestellt. Diese Objekte sind auch verantwortlich für die SSH2-Verbindung.

Nachdem ein Administrator sich erfolgreich mit einer Serverinstanz verbunden hat, werden – gemäß der Aktivitäten – `AdminMessage`-Nachrichten in beide Richtungen verschickt. Auf der Seite des Administrators werden diese gleich von der graphischen Oberfläche behandelt. Auf der Seite des Servers erhält zuerst ein `AEPSSH2ServerAdministrator`-Objekt die Nachrichten, dann ein `AEPServerSideCommandProcessor`-Objekt. Das letztere ist für das Interpretieren der Nachrichten verantwortlich. Nachdem die Nachricht verstanden wurde, wird der Befehl von `AEPMain` ausgeführt.

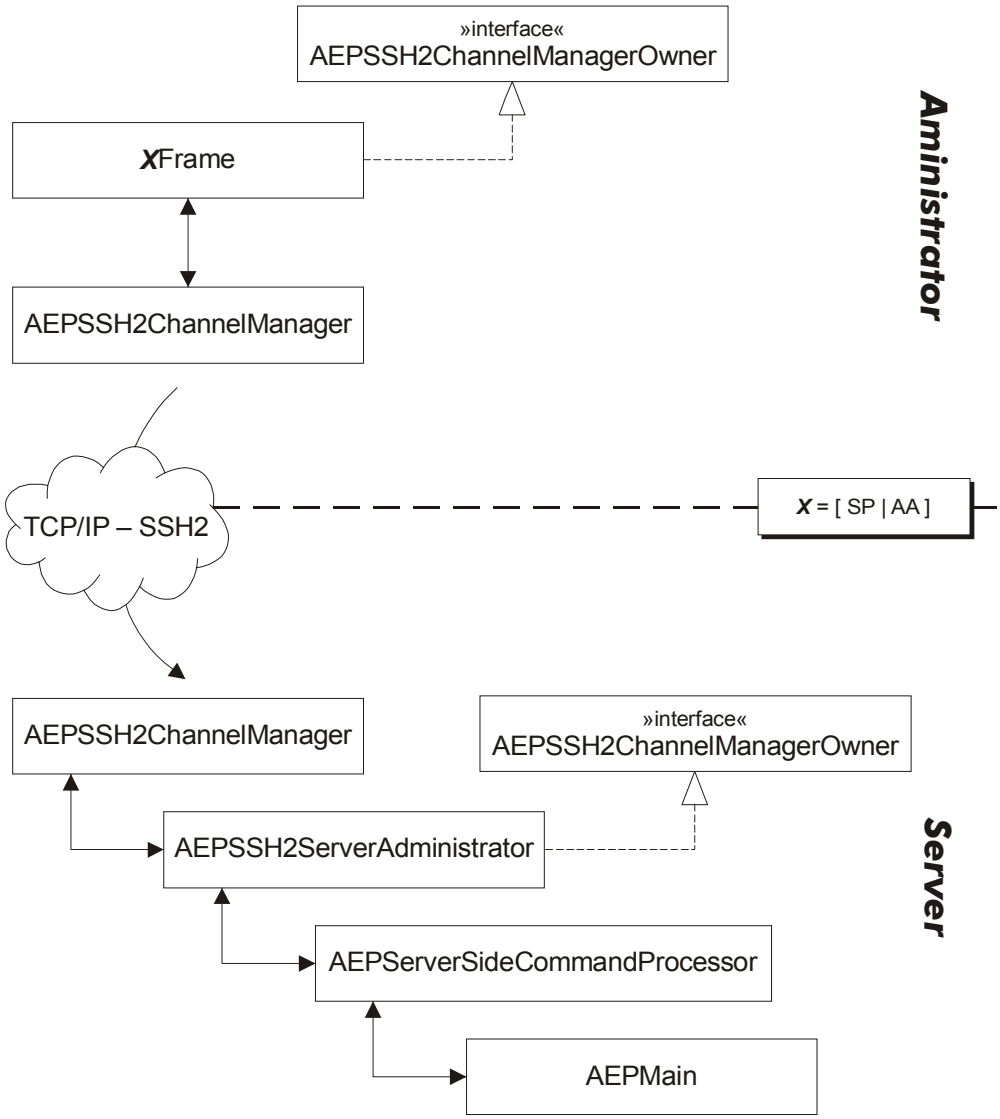


Abbildung 26 – **Allgemeine Klassenhierarchie, die bei der Adinistration der zwei Serverinstanzen in AEP benutzt wurde**

5.3.5.2.2 Administration der Anonymitätsinstanz

Während der Administration der Anonymitätsinstanz werden im Hauptfenster (siehe Abbildung 27) alle Systemnachrichten angezeigt. Es werden sowohl Nachrichten bezüglich SSH2-Verbindungen als auch bezüglich AEP-Operationen aufgelistet.

Mittels Popup-Menüs lassen sich verschiedene Aufgaben erledigen (SSH2-Verbindungen, Kanäle können geschlossen werden, Informationen über Ereignisse können angefordert und angezeigt werden, oder man kann auch Anonymitätsdokumente widerrufen, falls die Anonymitätsmethode es ermöglicht). Bestimmte Aufgaben bezüglich der verschiedenen Anonymitätsverfahren können auch nur hier erledigt werden (z. B. das Generieren eines RSA-Schlüssels eines bestimmten Diensttyps für die Methode der blinden Unterschrift).

Abbildung 27 – **Hauptfenster der Administratoranwendung einer Anonymitätsinstanz mit einem aktivierten Popup-Menü**

Datenstrukturen, die gemäß der Präsentationssprache interpretiert werden, können auch in einem Dialogfenster (siehe Abbildung 28) angezeigt werden.

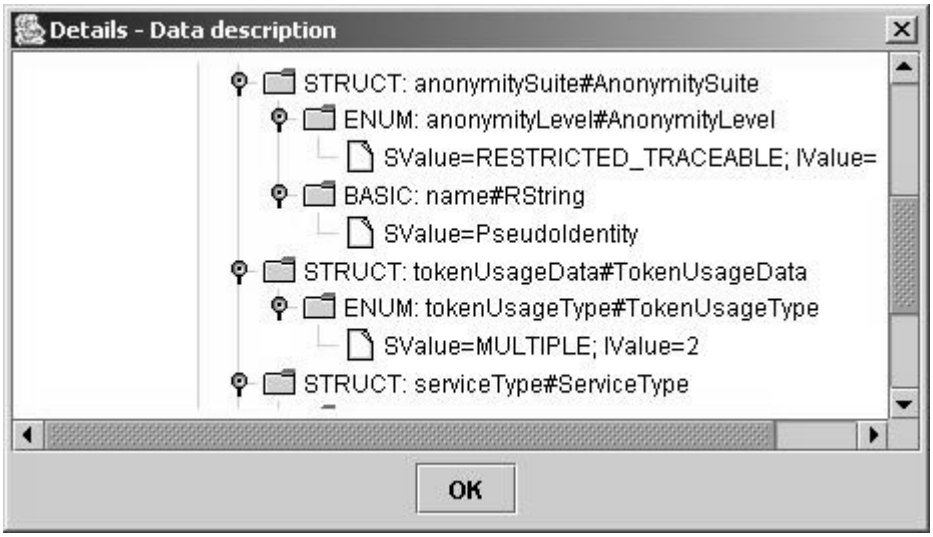


Abbildung 28 – **Graphische Darstellung einer Nachricht in der Präsentationssprache**

5.3.5.2.3 Administration des Dienstanbieters

Das Hauptfenster (siehe Abbildung 29) des Administrators eines Dienstanbieters ist zweigeteilt. In der linken Hälfte werden die Systemnachrichten (der SSH2-, und der AEP-Instanz bzw. auch des Webserver) angezeigt, in der rechten sieht man den Status verschiedener *serviceIDs*.

Mit Hilfe von Popup-Menüs kann der Administrator sich die Details der Nachrichten ansehen, mehr über die Dienstanforderungen, die abgewickelten Autorisierungen und die Dienstinanspruchnahme erfahren oder eine *serviceID* sperren.

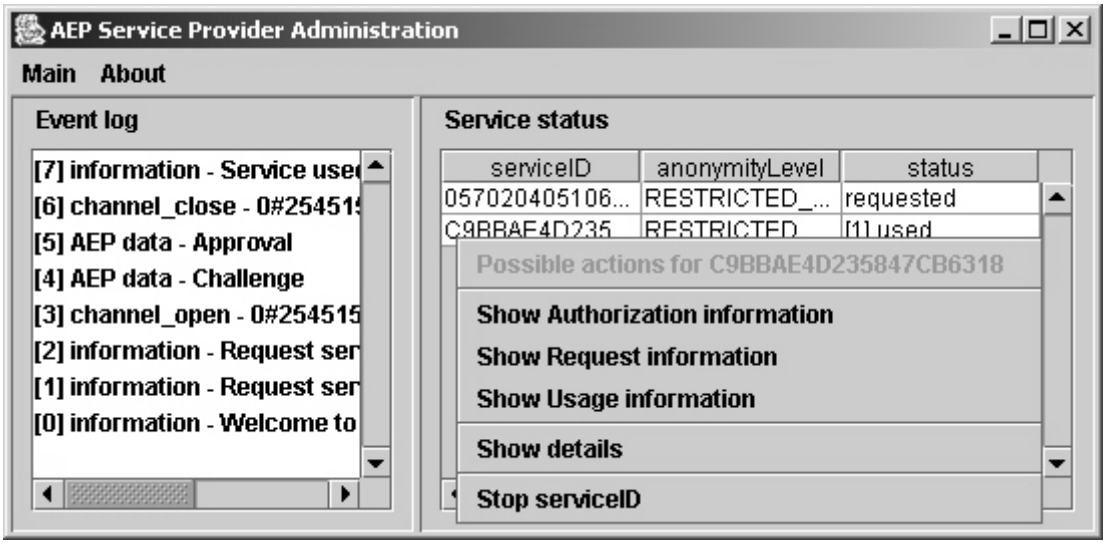


Abbildung 29 – **Hauptfenster der Administratoranwendung eines Dienstanbieters mit einem aktivierten Popup-Menü**

5.3.6 Verwaltung der Daten

Derzeit benutzt die Implementierung des Subjekts und der Anonymitätsinstanz SQL-Tabellen, um die verschiedene Daten effektiv zu verwalten.

- Die Anonymitätsinstanz speichert die ausgestellten Anonymitätsdokumente, Daten über die verschiedenen Pseudoidentitäten, die einmaligen Pseudoidentitäten und verschiedene kryptographische Schlüssel in SQL-Tabellen.
- Das Subjekt speichert die erhaltenen Anonymitätsdokumente, Daten über ihre Benutzung sowie verschiedene kryptographische Schlüssel in SQL-Tabellen.

6. Zusammenfassung

In diesem Kapitel werden zuerst verschiedene Ideen zur möglichen Weiterentwicklung der Spezifikation oder der Implementierung des Protokolls bzw. des Testsystems aufgelistet. Zum Schluss wird dann eine Bewertung der bisher geleisteten Arbeit gegeben.

6.1 Möglichkeiten für die Weiterentwicklung

Die vorliegende Arbeit ist das Ergebnis der Spezifizierung und Implementierung der Version 0.2 des *Anonymity Enhancing Protocol*. Während der Implementierung wurde das Fehlen bestimmter Funktionen entdeckt. Im Folgenden wird eine Liste von Ideen vorgestellt, die in zukünftigen Spezifikationen und Implementierungen berücksichtigt werden sollten:

- Es müssen mehr Anonymitätsverfahren in AEP implementiert werden und bereits implementierte Anonymitätsmethoden erweitert (siehe z. B. [PAYCASH-2] und [PAYCASH-3]) bzw. verbessert, damit das Protokoll getestet werden kann und eine breitere Anwendungsmöglichkeit geschaffen wird.
- Es soll analysiert werden, ob für die Kommunikationssprache vielleicht eine bessere Variante als XML gewählt werden kann.
- Die Datenbankverbindung des Protokolls und der verschiedenen Anonymitätsverfahren muss spezifiziert werden.
- Es müssen verschiedene Anonymitätsszenarien mit Hilfe von AEP implementiert und analysiert werden.
- Die verschiedenen graphischen Oberflächen nutzen nicht alle möglichen Funktionen des Protokolls.
- Die Konfigurierbarkeit des Protokolls und der verschiedenen zusätzlichen Anwendungen muss erhöht werden, die verschiedene Konfigurationsmöglichkeiten müssen dokumentiert werden.
- Die Verbindung von AEP und der Sicherheitsschicht muss spezifiziert werden.
- AEP muss mit anderen Sicherheitsschichten (z. B. SSL, TLS oder WTLS) verbunden werden können.
- Weitere sicherheitstechnische Aspekte (Datensicherheit bei den verschiedenen Serverinstanzen, Vertraulichkeit der Anonymitätsdokumente) müssen beachtet werden. [SEC-1] [SEC-2]

6.2 Bewertung

AEP versucht, eine Antwort auf das ständig wachsende Bedürfnis an Anonymität während der Netzwerkkommunikation zu geben. Die vorliegende Arbeit spezifiziert ein Protokoll, das verschiedene Anonymitätsniveaus und Anonymitätsverfahren unterstützt.

Die wichtigste Idee hinter dem Protokoll ist, eine Anonymitätsschicht oberhalb bewährter Sicherheitsschichten einzuführen. Die Sicherheitsschicht wird benötigt, da mit der Anonymität auch die Authentizität, die Vertraulichkeit und die restlichen Aspekte, die von Sicherheitsschichten behandelt werden, erwünscht werden. Das wichtigste Problem dabei ist,

die Balance zwischen der Identifizierung und der Anonymität zu finden, deshalb wurden die verschiedenen Anonymitätsniveaus eingeführt.

Die Zielsetzung dieser Arbeit war nicht, alle Anonymitätsverfahren zu implementieren, sondern eine allgemeine Umgebung, ein Protokoll zu spezifizieren, in dem die verschiedenen heutigen, aber auch zukünftige Anonymitätsverfahren einheitlich implementiert und benutzt werden können.

Vorteil des implementierten Protokolls ist, dass es eine einheitliche Umgebung für die Realisierung der verschiedenen Anonymitätsverfahren anbietet, es ermöglicht die Anwendung verschiedener Anonymitätsniveaus, die verschiedenen Anonymitätsmethoden können den Bedürfnissen entsprechend gewählt werden. All dies wird basierend auf eine Sicherheitsschicht bei höchster Sicherheit verwirklicht.

Es wurde als wichtig eingestuft, dass neben der theoretischen Spezifikation des Protokolls eine auch in der Praxis anwendbare Implementierung zustande kommt. Aufbauend auf einer bestehenden Implementierung der Sicherheitsschicht (SSH2) wurde die Anonymitätsschicht (AEP) in JAVA implementiert. Beide JAVA Implementierungen sind unter der GPL [GPL-1] veröffentlicht. Als Anonymitätsverfahren wurden die Methode der Pseudoidentität, die Methode der einmaligen Pseudoidentität und die Methode der blinden Unterschrift implementiert.

Als zukünftiges Ziel kann die Weiterentwicklung des Protokolls genannt werden (möglicherweise im Rahmen einer Doktorarbeit), damit die spezifizierte und implementierte Umgebung verbessert, weiter spezifiziert, standardisiert und evaluiert werden kann.

Mit der Anwendung des Protokolls sind die immer wichtiger werdenden Probleme der Anonymität lösbar. Somit können die persönlichen Daten des Individuums erfolgreicher geschützt werden.

Anhang A Abkürzungen

Das vorliegende Dokument benutzt die folgenden Abkürzungen:

3DES	Triple-DES
AA	Anonymity Authority
AEP	Anonymity Enhancing Protocol
AES	Advanced Encryption Standard
API	Application Programming Interface
AS	Anonymous Subject
ASCII	American Standard Code for Information Interchange
BNF	Backus Naur Form
CA	Certification Authority
DES	Data Encryption Standard
DN	Distinguished Name
DoS	Denial of Service
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
GPL	General Public License
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JDBC	Java DataBase Connectivity
LAN	Local Area Network
LSB	Least Significant Bit
MD5	Message Digest 5
MSB	Most Significant Bit
NAT	Network Address Translation
PET	Privacy Enhancing Technologies
PKI	Public Key Infrastructure
RSA	Rivest, Shamir & Adleman
SAP	Service Access Point
SHA1	Secure Hash Algorithm 1
SP	Service Provider
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	Trusted Third Party
WAP	Wireless Application Protocol
WTLS	Wireless Transport Layer Security
WWW	World Wide Web
XML	Extensible Markup Language

Anhang B Notationen für die Beschreibung eines SAP

In diesem Abschnitt werden die Notationen erklärt, die für die Beschreibung eines Dienstzugangspunkts (*SAP*, *service access point*) in der vorliegenden Arbeit benutzt werden. Die Notationen sind von WTLS [WTLS-1] übernommen worden.

B.1 Allgemeines

An einem Dienstzugangspunkt werden vom Diensterbringer Dienste angeboten. Diese können mit Hilfe von Dienstelementen in Anspruch genommen werden. Dienste werden durch den Dienstnamen gekennzeichnet. Der Dienstelementstyp kennzeichnet die Funktion des Dienstelements eines Dienstes.

Dienstelemente sind nicht identisch mit einem API (*application programming interface*) und aus Dienstelementen kann nicht auf die Struktur von Implementierungen geschlossen werden. Dienstelemente stellen die abstrakte Funktionsweise des spezifizierten Protokolls dar.

Die allgemeine Syntax ist die Folgende:

```
SAPBezeichner-DienstName.Dienstelementtyp (Parameter)
```

In der vorliegenden Arbeit wird "`AEP`" als Bezeichner des SAP benutzt.

B.2 Dienstelementtype

Die vorliegende Spezifikation von AEP benutzt die folgenden vier Dienstelementtypen:

Dienstelementtyp	Abkürzung	Beschreibung
request	*req*	Die darüberliegende Schicht fordert einen Dienst an.
indication	*ind*	Die Schicht, die den Dienst erbringt, benutzt diesen Dienstelementtyp, um die darüberliegende Schicht über die Dienstanforderung zu benachrichtigen.
response	*res*	Dieser Dienstelementtyp wird vom Empfänger des *indication*-Dienstelementtyps dem Empfänger des *request*-Dienstselementtyps geschickt, um den Empfang des *indication*-Dienstelementtyps zu bestätigen.
confirmation	*cnf*	Mit diesem Dienstelementtyp signalisiert die Schicht, die den Dienst erbringt, derjenigen, die die Anforderung gestellt hat, dass der Dienst erfolgreich erbracht wurde.

B.3 Dienstparametertabellen

Dienstelemente werden durch Tabellen definiert, in denen angezeigt wird, wie die verschiedenen möglichen Parameter bei den Dienstelementtypen benutzt werden. Eine Tabelle sieht folgendermaßen aus:

	Dienstelement AEP-Dienstelement			
Parameter	req	ind	res	cnf
Parameter1	M	M(=)		
Parameter2			O	C(=)

Ist ein Dienstelementtyp bei einem Dienstelement nicht möglich, so wird die entsprechende Spalte der Tabelle weggelassen.

In einer Dienstparametertabelle werden die folgenden Bezeichnungen benutzt:

M	*mandatory*: Die Präsenz dieses Parameters ist obligatorisch.
C	*conditional*: Die Präsenz dieses Parameters hängt von den Werten anderer Parameter ab.
O	*optional*: Die Präsenz dieses Parameters hängt vom Dienstnutzer ab.
	absent: Dieser Parameter wird nicht benutzt.
(=)	*equal*: Der Wert des Parameters ist gleich mit dem Wert des entsprechenden Parameters des vorigen Dienstelementtyps.

Im obigen Beispiel ist Parameter1 im *AEP-Dienstelement.request* immer präsent. Im *AEP-Dienstelement.indication* ist er auch immer präsent und hat den gleichen Wert, wie beim *AEP-Dienstelement.request*. Im *AEP-Dienstelement.response* und *AEP-Dienstelement. confirmation* wird Parameter1 nicht benutzt. Ist Parameter2 im *AEP-Diestelement.response* präsent, so muss er mit dem gleichen Wert auch im *AEP-Dienstelement.confirmation* präsent sein. Wird er im *AEP-Dienstelement.response* weggelassen, so darf er im *AEP-Dienstelement. confirmation* auch nicht präsent sein. Im *AEP-Dienstelement.request* und *AEP-Dienstelement. indication* kann Parameter2 nie präsent sein.

Anhang C Sprachdefinitionen

C.1 Präsentationssprache

Dieser Abschnitt stellt die Präsentationssprache von AEP vor. Die für AEP benutzte Präsentationssprache ist ähnlich zu der, die bei der Programmiersprache C [C-1] und bei TLS [TLS-1] benutzt wurde.

C.1.1 Allgemeines

C.1.1.1 Bemerkungen

In der hier vorgestellten Sprache wird **Groß- und Kleinschreibung unterschieden** (*case-sensitivity*).

Die folgenden Wörter können nicht (weder in Groß- noch in Kleinschreibung, noch gemischt) als Name eines Datentyps oder einer Variable benutzt werden (*reserved words*):

```
basic, case, enum, default, extension, list, select, struct,
type, value
```

Die Grunddatengröße der Präsentationssprache ist ein **Byte** (ein Oktett, 8 Bits).

Bemerkungen fangen mit // am Beginn der Zeile an.

C.1.1.2 Allgemeine BNF Notationen

Hier werden die BNF Definitionen (*Backus Naur Form* [BNF-1]) aufgezählt, die als Grundlage anderer Definitionen später benutzt werden.

```
<kleinbuchstabe> ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" |
  "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" |
  "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

<grossbuchstabe> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" |
  "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" |
  "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

<zahl> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |
   "9" | "0"

<hex> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
    "8" | "9" | "A" | "B" | "C" | "D" | "E" | "F"

<sonstiges> ::= "_"

<kleinanfang> ::= <kleinbuchstabe> | <sonstiges>

<grossanfang> ::= <grossbuchstabe>  | <sonstiges>

<alles> ::= <kleinbuchstabe> | <grossbuchstabe> | <sonstiges> |
  <zahl>

<variablenname> ::= <kleinanfang> {<alles>}

<datentypname> ::= <grossanfang> {<alles>}

<variablendeklaration> ::= <datentypname> " " <variablenname> ";"
```

C.1.2 Sorten von Datentypen

C.1.2.1 Datentyp *Basis*

Ein *Basis*-Datentyp ist ein Datentyp, der nicht aus anderen Datentypen zusammengesetzt ist.

```
<basic> ::= basic " " <datentypname> ";"
```

Die Präsentationssprache benutzt die im Folgenden definierten Datentypen.

```
basic RString;
basic HEXBlob;
basic DECNumber;
basic HEXNumber;
```

C.1.2.1.1 RString

Variablen des Datentyps `RString` können beliebig lange ASCII-Zeichensequenzen speichern, allerdings nur Zeichen des Intervalls ASCII 32-127, außer den Zeichen '\', '"', '<' und '>'.

C.1.2.1.2 HEXBlob

Variablen des Datentyps `HEXBlob` können beliebig lange ASCII Zeichensequenzen in hexadezimaler Repräsentation speichern. Jedem ASCII-Zeichen werden zwei hexadezimale Zahlen zugeordnet, die gemäß der "*network byte order*" kodiert sind, das heißt, sowohl innerhalb eines Bytes als auch insgesamt steht MSB links und LSB rechts.

```
<hex2> ::= <hex><hex>
<HEXBlob> ::= <hex2> {<hex2>}
```

Beispiel:

```
"41"
```

Dies ist das ASCII-Zeichen 65, das den Buchstaben "A" repräsentiert.

C.1.2.1.3 DECNumber

Variablen des Datentyps `DECNumber` speichern beliebig lange ganze Zahlen in dezimaler Repräsentation.

```
<DECNumber> ::= ["+" | "-"] <zahl> {<zahl>}
```

Beispiel:

```
"-52"
```

C.1.2.1.4 HEXNumber

`HEXNumber` ist eine weiter interpretierte Form von `HEXBlob`. Variablen des Datentyps `HEXNumber` können beliebig lange ganze Zahlen in Zweierkomplementform speichern. Wäre das erste Bit einer positiven Zahl 1, so muss an die Zahl vorne `"00"` angehängt werden.

```
<HEXNumber> ::= <HEXBlob>
```

Beispiel:

```
"00"    ⇨   (dezimal)    0
"80"    ⇨   (dezimal) -128
"0080"  ⇨   (dezimal)  128
"FF"    ⇨   (dezimal)   -1
```

C.1.2.2 Datentyp *Erweiterung*

In bestimmten Fällen kann es erwünscht sein, Datentypen des Protokolls mit zusätzlichen Daten zu erweitern. Diesen Zweck erfüllt der Datentyp *Erweiterung* (`Extension`). Die konkrete Form hängt von der Kommunikationssprache ab, bei XML enthält eine solche Variable eine syntaktisch korrekte (*well-formed*) XML-Zeichensequenz.

C.1.2.3 Datentyp *Aufzählung*

Variablen des Datentyps *Aufzählung* können nur Werte einer vorher schon definierten Menge annehmen.

```
<enumwert> ::= <grossbuchstabe> {<grossbuchstabe>}
<enum> ::= enum "{" <enumwert>
  {"," <enumwert>} "}" <datentypname> ";"
```

Bispiel:

```
enum {RED, GREEN, BLUE} Color;
```

C.1.2.3.1.1 Operationen

- Deklaration

  ```
  Color c1; Color c2;
  ```

- Wertzuweisung

  ```
  c1 = RED; c2 = BLUE;
  ```

C.1.2.4 Datentyp *Liste*

Variablen des Datentyps *Liste* enthalten andere, schon vorher definierte Variablen in einer bestimmten Reihenfolge.

```
<list> ::= list "<" <datentypname> ">" <datentypname> ";"
```

Beispiel:

```
list<Color> ColorList;
```

Demnach enthalten Variablen des Typs `ColorList` Variablen des Typs `Color`.

C.1.2.4.1.1 Operationen

- Deklaration

  ```
  ColorList colorList;
  ```

- Wertzuweisung: Die Elemente müssen zwischen "{" und "}" mit "," getrennt aufgezählt werden.

```
colorList = {c1,c2};
```

- Wertabfrage: Elemente können mit dem "[]"-Operator abgefragt werden. Die Nummerierung der Elemente beginnt mit 0.

```
c2 = colorList[1];
```

- Längenabfrage: Mit dem ".length"-Operator kann die Länge der Liste (Anzahl der Elemente) abgefragt werden. Das letzte Element einer der Liste ist demnach:

```
colorList[colorList.length-1]
```

C.1.2.5 Datentyp *Struktur*

Eine *Struktur* ist die Zusammenfassung mehrerer, schon vorher definierter Datentypen zu einem komplexeren Datentyp. Die Syntax ist mehr oder weniger ähnlich zu der der C `struct`.

Der Datentyp *Struktur* hat einen Spezialfall, der *variante Struktur* genannt wird. Jeder *variante Struktur* enthält mindestens einen *varianten Teil*. Zu jedem *varianten Teil* gehört ein Datentyp *Aufzählung*. Jedem möglichen Wert des Datentyps *Aufzählung* kann eine Menge von Variablendeklarationen zugewiesen werden. Demnach können vom *varianten Teil* nur diejenigen Variablen erreicht werden, die durch den Wert des Datentyps *Aufzählung* bestimmt werden.

```
<selectorende> ::= <variabledeklaration>
  {<variabledeklaration>
<caseselector> ::= case <enumwert> ":" <selectorende>
<defaultselector> ::= default ":" <selectorende>
<varianterteil> ::= <variablendenklaration>
  select "(" <variablenname> ")" "{"
  {<caseselector>}
  [<defaultselector>]
  "}"
```

Mit dem `case`-Ast kann der Wert des Datentyps *Aufzählung* spezifiziert werden, der `default`-Ast bedeutet alle noch nicht bestimmten Werte. Somit ist die Definition des Datentyps *Struktur* die Folgende:

```
<struct> ::= struct "{"
  <variabledenklaration>
  {<variabledenklaration>}
  {<varianterteil>}
  "}" <datentypname> ";"
```

Beispiel:

```
struct {
  Color c;
  select (c) {
    case RED :
      ColorList cL;
    default :
      Color c2;
  }
} VariantStruct;
```

C.1.2.5.1.1 Operationen

- Deklaration

```
VariantStruct v;
```

- Wertabfrage: Interne Variablen können mit dem "."*Operator abgefragt oder eingestellt werden.

```
v.c = RED;
```

C.2 XML-Kommunikationssprache

Dieser Abschnitt erklärt, wie Variablen der Präsentationssprache in die Kommunikationssprache übersetzt werden können.

C.2.1 Bemerkungen

Sei ***v*** eine Variable, dann ist ihr Datentyp `T(`***v***`)` und ihr Wert `V(`***v***`)`. XML-Zeilen werden mit dem Zeichen # eingeleitet. Beginnt eine Zeile mit den Zeichen ##, dann muss diese Zeile interpretiert und in XML umgesetzt werden.

In den meisten Fällen kann man innerhalb einer XML Struktur vom Namen einer Variable auf den Datentyp der Variablen schließen, deshalb ist die Angabe des Datentyps überflüssig. Kann jedoch aus irgendeinem Grund der Datentyp nicht ermittelt werden, so muss der, der Variablen entsprechende XML-Block (*tag*) um ein `type`-Attribut (*attribute*) erweitert werden, das den Datentyp der Variablen angibt. Das muss z. B. bei *Erweiterungen* oder bei der äußersten Variablen einer Struktur gemacht werden.

C.2.2 Konversionsregeln

C.2.2.1 Datentyp *Basis*

Nach der Übersetzung ist die resultierende XML-Struktur die folgende:

```
#<v value="V(v)" />
```

Beispiel:

```
RString v1 = "a1";
#<v1 value="a1" type="RString" />
```

Der Wert der Variable muss gemäß der Präsentationssprache interpretiert werden.

C.2.2.2 Datentyp *Erweiterung*

Nach der Übersetzung ist die resultierende XML-Struktur die folgende:

```
#<v>
## Die syntaktisch korrekte Zeichensequenz der Variable
#</v>
```

Beispiel:

```
Extension ext;
ext = "<message type=\"Approval\">\n</message>\n"
```

```
#<ext type="Extension">
#  <message type="Approval">
#  </message>
#</ext>
```

C.2.2.3 Datentyp *Aufzählung*

Nach der Übersetzung ist die resultierende XML-Struktur die folgende:

```
#<v value="V(v)" />
```

Beispiel:

```
Color c;
c = RED;
```

```
#<c value="RED" type="Color" />
```

C.2.2.4 Datentyp *Liste*

Nach der Übersetzung ist die resultierende XML-Struktur die folgende:

```
# <v>
## 1. Element
## 2. Element
## ...
# </v>
```

Elemente der Liste werden zwischen <***v***> und </***v***> so in XML übersetzt, als wären sie Variablen mit den Namen `element_i`, wobei `i` die Position des Elements in der Liste ist.

Beispiel:

```
list<DECNumber> DecList;
DECNumber d1, d2, d3;
d1 = 1; d2 = 2; d3 = 3;
DecList decList;
decList = {d1, d2, d3};
```

```
#<decList type="DecList">
#  <element_0 value="1" />
#  <element_1 value="2" />
#  <element_2 value="3" />
#</decList>
```

C.2.2.5 Datentyp *Struktur*

Nach der Übersetzung ist die resultierende XML-Struktur die folgende:

```
# <v>
## XML Representation der internen Variablen
# </v>
```

Enthält der Datentyp *Struktur* auch *variante Teile*, so müssen nur diejenigen internen Variablen aufgezählt werden, die durch den Wert des Datentyps *Aufzählung* des *varianten Teils* bestimmt werden.

Beispiel:

```
VariantStruct v;
v.c = BLUE;
v.c2 = RED;
```

```
#<v type="VariantStruct">
#  <c value="BLUE" />
#  <c2 value="RED" />
#</v>
```

Anhang D Datenstrukturen

In diesem Kapitel werden die in der AEP-Kommunikation benutzten Datenstrukturen (*langspec.in*, siehe Abschnitt 5.3.1 und Abbildung 16) gemäß der Präsentationssprache definiert.

D.1 Allgemeine Datenstrukturen

```
// ------------------------------------------------------------------------
// BASIC
// ------------------------------------------------------------------------

basic RString;
basic DECNumber;
basic HEXNumber;
basic HEXBlob;
basic Extension;

// ------------------------------------------------------------------------
// Pair and Pairlist
// ------------------------------------------------------------------------

struct {
  RString key;
  RString value;
} Pair;

list<Pair> PairList;
list<PairPair> PairListList;

// ------------------------------------------------------------------------
// AnonymityLevel & AnonymitySuite
// ------------------------------------------------------------------------

enum {NONE, TRACEABLE, RESTRICTED_TRACEABLE, NON_TRACEABLE, FULL} AnonymityLevel;

struct {
  AnonymityLevel anonymityLevel;
  RString name;
  Extension parameters;
  Extension ext;
} AnonymitySuite;

list<AnonymitySuite> AnonymitySuiteList;

// ------------------------------------------------------------------------
// Date & Time & DateTime
// ------------------------------------------------------------------------

struct {
   DECNumber year;
   DECNumber month;
   DECNumber day;
   Extension ext;
} Date;

struct {
   DECNumber hour;
   DECNumber minute;
   DECNumber second;
   RString timeZone;
   Extension ext;
} Time;

struct {
   Date date;
   Time time;
   Extension ext;
} DateTime;
```

```
// ------------------------------------------------------------------------------
// Service & ServiceType & ServiceUsage
// ------------------------------------------------------------------------------

struct {
  RString serviceID;
  Extension ext;
} Service;

struct {
  RString name;
  Extension ext;
} ServiceType;

struct {
  Service service;
  DateTime usageTime;
  Extension ext;
} ServiceUsage;

// ------------------------------------------------------------------------------
// InfoType & InfoDesc
// ------------------------------------------------------------------------------

enum {PUBKEY} InfoType;

struct {
  DECNumber infoID;
  InfoType infoType;
  Extension ext;
} InfoDesc;

list<InfoDesc> InfoDescList;

// ------------------------------------------------------------------------------
// TokenUsage
// ------------------------------------------------------------------------------

enum {SINGLE, FIXEDNUMBER, MULTIPLE} TokenUsageType;

struct {
  TokenUsageType tokenUsageType;
  select (tokenUsageType) {
    case FIXEDNUMBER :
      DECNumber numberOfUsages;
    default :
  }
} TokenUsageData;

// ------------------------------------------------------------------------------
// ANONYMITYLEVEL SPECIFIC
// NON-TRACEABLE
// ------------------------------------------------------------------------------

enum {ENVELOPE, SIGNED_ENVELOPE, SIGNED_FINAL} NonTraceableStatus;

struct {
  NonTraceableStatus status;
  HEXBlob hexData;
  Extension ext;
} NonTraceableData;

// ------------------------------------------------------------------------------
// ANONYMITYLEVEL SPECIFIC
// RESTRICTED-TRACEABLE
// ------------------------------------------------------------------------------

struct {
  Extension data;
  Extension ext;
} RestrictedTraceableData;

struct {
  RString pseudoIdentity;
  Extension ext;
} PseudoIdentity;
```

```
// ------------------------------------------------------------------------------
// PublicKey
// ------------------------------------------------------------------------------

struct {
  RString keyType;
  HEXBlob keyData;
  Extension ext;
} PublicKey;

// ------------------------------------------------------------------------------
// IdentifiedEntity & Signature
// ------------------------------------------------------------------------------

struct {
  RString entityCN;
  RString entityDN;
  RString entityUID;
  Extension ext;
} IdentifiedEntity;

struct {
  RString signatureType;
  IdentifiedEntity signer;
  HEXBlob signatureCore;
  Extension ext;
} Signature;

// ------------------------------------------------------------------------------
// PreToken
// ------------------------------------------------------------------------------

struct {
  AnonymityLevel anonymityLevel;
  AnonymitySuite anonymitySuite;
  TokenUsageData tokenUsageData;
  ServiceType serviceType;
  select (anonymityLevel) {
    case RESTRICTED_TRACEABLE :
      RestrictedTraceableData restrictedTraceableData;
    case NON_TRACEABLE :
      NonTraceableData nonTraceableData;
    default :
  }
  Extension ext;
} PreToken;

// ------------------------------------------------------------------------------
// TokenSecurity
// ------------------------------------------------------------------------------

struct {
  RString tokenID;
  DateTime validityStart;
  DateTime validityEnd;
  Signature signature;
  Extension ext;
} TokenSecurity;

// ------------------------------------------------------------------------------
// Token
// ------------------------------------------------------------------------------

struct {
  RString version;
  PreToken preToken;
  DateTime issuedOn;
  TokenSecurity security;
  Extension ext;
} Token;

list<Token> TokenList;
```

```
// ------------------------------------------------------------------------------
// UsageConnection
// ------------------------------------------------------------------------------

enum {SUBJECT_PROVIDER, PROVIDER_ANONYMITY_AUTHORITY} UsageConnection;
```

D.2 Datenstrukturen des Initialisierungsprozesses

```
// ------------------------------------------------------------------------------
// StartPreparation: AS->AA -- I/1
// ------------------------------------------------------------------------------

struct {
  RString version;
  AnonymitySuiteList supportedAnonymitySuites;
  Extension ext;
} StartPreparation;

// ------------------------------------------------------------------------------
// RequestAnonymity: AS->AA -- I/2
// ------------------------------------------------------------------------------

struct {
  AnonymityLevel anonymityLevel;
  ServiceType serviceType;
  Extension ext;
} RequestAnonymity;

// ------------------------------------------------------------------------------
// PreparationStarted: AA->AS -- I/3
// ------------------------------------------------------------------------------

struct {
  AnonymitySuite chosenAnonymitySuite;
  Extension ext;
} PreparationStarted;

// ------------------------------------------------------------------------------
// RequestInfo: AS->AA -- I/4
// ------------------------------------------------------------------------------

struct {
  InfoDescList requests;
  Extension ext;
} RequestInfo;

// ------------------------------------------------------------------------------
// ResposnseInfo: AA->AS -- I/5
// ------------------------------------------------------------------------------

struct {
  InfoDescList responses;
  Extension ext;
} ResponseInfo;

// ------------------------------------------------------------------------------
// Envelope: AS->AA -- I/6
// ------------------------------------------------------------------------------

struct {
  NonTraceableData envelope;
  Extension ext;
} Envelope;

// ------------------------------------------------------------------------------
// PreparationFinished: AA->AS -- I/7
// ------------------------------------------------------------------------------

struct {
  Token token;
  Extension ext;
} PreparationFinished;
```

D.3 Datenstrukturen der anonymen Autorisierung

```
// ---------------------------------------------------------------------------
// RequestService: AS->SP / SP->AA -- II/1 & II/2
// ---------------------------------------------------------------------------

struct {
  UsageConnection usageConnection;
  select (usageConnection) {
    case SUBJECT_PROVIDER :
      Service service;
    default :
  }
  TokenList tokenList;
  Extension ext;
} RequestService;

// ---------------------------------------------------------------------------
// Challenge: AA->SP / SP->AS -- II/3 & II/4
// ---------------------------------------------------------------------------

struct {
  HEXBlob challenge;
  Extension ext;
} Challenge;

// ---------------------------------------------------------------------------
// Response: AS->SP / SP->AA -- II/5 & II/6
// ---------------------------------------------------------------------------

struct {
  Signature signature;
  Extension ext;
} Response;

// ---------------------------------------------------------------------------
// Approval: AA->SP -- II/7
// ---------------------------------------------------------------------------

struct {
  Service service;
  Extension ext;
} Approval;
```

D.4 Datenstrukturen für Fehlverhalten

```
struct {
  DECNumber alertID;
  RString reason;
  Extension ext;
} Alert;

struct {
  DECNumber warningID;
  RString reason;
  Extension ext;
} Warning;
```

D.5 Datenstrukturen für die Administration

```
enum {INFORMATION, SECURITY, AEP, COMMAND, CONFIRMATION} AdminMessageType;

enum {CONNECTION_OPEN, CHANNEL_OPEN, CHANNEL_DATA, CHANNEL_CLOSE, CONNECTION_CLOSE}
AdminMessageSecurityType;

enum {SUCCESS, FAILURE} AdminMessageConfirmationSuccess;
```

```
struct {
  DECNumber requestMessageID;
  AdminMessageConfirmationSuccess success;
  select (success) {
    case SUCCESS :
      Extension result;
    case FAILURE :
      RString reason;
  }
  Extension ext;
} AdminMessageConfirmation;

struct {
  DECNumber connectionID;
  IdentifiedEntity subject;
  Extension ext;
} Connection;

struct {
  DECNumber channelID;
  Extension ext;
} Channel;

struct {
  AdminMessageSecurityType adminMessageSecurityType;
  select (adminMessageSecurityType) {
    case CONNECTION_OPEN :
      Connection connection;
    case CHANNEL_OPEN :
      Connection connection;
      Channel channel;
    case CHANNEL_DATA :
      Connection connection;
      Channel channel;
      Extension channelData;
    case CHANNEL_CLOSE :
      Connection connection;
      Channel channel;
    case CONNECTION_CLOSE :
      Connection connection;
  }
  Extension ext;
} AdminMessageSecurity;

struct {
  RString data;
  Extension ext;
} AdminMessageNormalData;

struct {
  DECNumber messageID;
  DateTime messageTimeStamp;
  AdminMessageType adminMessageType;
  select (adminMessageType) {
    case INFORMATION :
      AdminMessageNormalData information;
    case COMMAND :
      AdminMessageNormalData command;
    case SECURITY :
      AdminMessageSecurity security;
    case AEP :
      Extension aep;
    case CONFIRMATION :
      AdminMessageConfirmation confirmation;
  }
  Extension ext;
} AdminMessage;
```

Anhang E Benutzeranleitung

Im folgenden wird kurz erklärt, wie die auf der beigefügten CD befindliche Software installiert, konfiguriert und gestartet werden kann.

E.1 Allgemeines

Auf der CD befindet sich die Version 0.2 des *Anonymity Enhancing Protocols* (AEP), das die Version 0.31 der SSH2-Implementierung von Team MONICA [MONICA-1] benutzt. Beide sind unter der GNU GPL [GPL-1] anwendbar.

Da die ganze Implementierung in JAVA geschrieben wurde, sollten die Anwendungen unter allen Betriebssystemen, auf denen eine virtuelle Maschine für JAVA zur Verfügung steht, ohne Probleme laufen.

Im Folgenden wird eine Linux- (bzw. UNIX-) Shell angenommen, und die eventuellen Kommandos (eingeführt mit ==>) werden entsprechend angegeben.

E.2 Voraussetzungen

Das Folgende wird für das korrekte Laufen von AEP benötigt:

- **JAVA**
  - AEP wurde unter J2SE 1.4 geschrieben und getestet. Empfohlen wird, AEP unter J2SE 1.4 zu benutzen. Nach Erfahrung funktioniert fast alles auch unter J2SE 1.3.1, es können jedoch Fehlverhalten auftauchen, wenn frühere Versionen als 1.4 benutzt werden.
  - Die konkrete Implementierung des Dienstanbieters benötigt J2EE 1.3.1.

  Für J2SE siehe [J2SE-1], für J2EE siehe [J2EE-1].

- **SQL-Server**

  Die Implementierungen der Anonymitätsinstanz und des Subjekts benötigen einen SQL Server. Getestet wurde MySQL (sowohl unter Linux als auch unter Windows) und PostgreSQL (unter Linux). Theoretisch sollte ein beliebiger SQL-Server, der JDBC [JDBC-1] anbietet, für AEP ausreichen. Empfohlen wird MySQL [MYSQL-1], da es weitgehend getestet wurde. Im Folgenden werden SQL-bezogenen Informationen gemäß MySQL angegeben.

- **Rechner und Netzwerk**

  Für AEP ist ein funktionierendes TCP/IP-Netzwerk erforderlich. Die verschiedenen Anwendungen können auf acht Rechner verteilt werden, es kann jedoch alles auch auf einem Computer laufen (dafür wird jedoch ein ziemlich starker PC mit großem Hauptspeicher benötigt).

  Die acht verschiedenen Anwendungen sind die Folgenden: AEP-Server der Anonymitätsinstanz (AA_AEP), SQL-Server der Anonymitätsinstanz (AA_SQL), Administrator der Anonymitätsinstanz (AA_GUI), AEP-Server des Dienstanbieters (SP_AEP), Webserver des Dienstanbieters (SP_WEB), Administrator des

Dienstanbieters (SP_GUI), AEP-Client des Subjekts (AS_AEP) und SQL-Server des Subjekts (AS_SQL).

Empfohlen wird, das System auf 3 Rechner verteilt zu installieren. Einen Rechner soll die Anonymitätsinstanz bekommen (AA_AEP, AA_GUI, AA_SQL), einen der Dienstanbieter (SP_AEP, SP_WEB, SP_SQL) und einen das Subjekt (AS_AEP, AS_SQL).

Wie AEP letztendlich auf den verschiedenen Rechner installiert wird, ist die Sache der Tester.

- **Arbeitsverzeichnis**

  Im Folgenden wird angenommen, dass die Installation mit der Dekomprimierung der auf der CD befindlichen TGZ-Datei begonnen wurde. Im Weiteren wird angenommen, dass die Shell sich im so entstandenen Verzeichnis befindet.

E.3 Aufgaben beim Installieren

Die folgenden Aufgaben müssen beim Installieren erledigt werden.

E.3.1 AA_SQL

- Der ausgewählte SQL-Server sollte installiert sein und TCP/IP-Verbindungen akzeptieren.
- Verschiedene SQL-Tabellen müssen erstellt werden.

  In `mysql` muss die folgende Datei ausgeführt werden: (`./sql/aa/aa_full.sql.my`)
  `==> \. ./sql/aa/aa_full.sql.my`

E.3.2 AA_AEP

- J2SE sollte installiert sein, die `java` Anwendung sollte sich im `PATH` befinden.
- Die für die JDBC-Verbindung zu AA_SQL benötigten Klassen sollten sich im `CLASSPATH` befinden.
- Die folgenden Konfigurationsdateien und die angegebenen Variablen müssen entsprechend der Installation verändert werden:
  - `./conf/v0.2/aa/ccdm/ccdm.sql.conf`

    `databaseDriver, databasePath, databaseUsername, databasePassword`
  - `./conf/v0.2/aa/sqltokenmanager.conf`

    `databaseDriver, databasePath, databaseUsername, databasePassword`
  - `./conf/v0.2/aa/pi.conf`

    `databaseDriver, databasePath, databaseUsername, databasePassword`
  - `./conf/v0.2/aa/otpi.conf`

    `databaseDriver, databasePath, databaseUsername, databasePassword`

E.3.3 AA_GUI

- J2SE sollte installiert sein, die `java` Anwendung sollte sich im `PATH` befinden.
- Die folgenden Konfigurationsdateien und die angegebenen Variablen müssen entsprechend der Installation verändert werden:
  - `./conf/v0.2/aa/admin/client/aep.conf`

    `destHost, destPort`

E.3.4 SP_WEB

- J2SE sollte installiert sein, die `java` Anwendung sollte sich im `PATH` befinden.
- J2EE sollte installiert sein, die `j2ee`- bzw. `deploytool`- Anwendungen sollten sich im `PATH` befinden.
- Die folgenden Konfigurationsdateien und die angegebenen Variablen müssen entsprechend der Installation verändert werden:
  - `./conf/v0.2/sp/corba.conf`

    `ORBInitialHost, ORBInitialPort`
  - `./conf/v0.2/sp/web.conf`

    `SPHost, SPPort, serviceProto, serviceHost, servicePort, servicePage`

E.3.5 SP_AEP

- J2SE sollte installiert sein, die `java` Anwendung sollte sich im `PATH` befinden.
- Die folgenden Konfigurationsdateien und die angegebenen Variablen müssen entsprechend der Installation verändert werden:
  - `./conf/v0.2/sp/corba.conf`

    `ORBInitialHost, ORBInitialPort`
  - `./conf/v0.2/sp/aa/ssh2client.conf`

    `destHost, destPort`

E.3.6 SP_GUI

- J2SE sollte installiert sein, die `java` Anwendung sollte sich im `PATH` befinden.
- Die folgenden Konfigurationsdateien und die angegebenen Variablen müssen entsprechend der Installation verändert werden:
  - `./conf/v0.2/sp/admin/client/aep.conf`

    `destHost, destPort`

E.3.7 AS_SQL

- Der ausgewählte SQL-Server sollte installiert sein und TCP/IP-Verbindungen akzeptieren
- Verschiedene SQL-Tabellen müssen erstellt werden.

  In `mysql` muss die folgende Datei ausgeführt werden: (`./sql/as/as_full.sql.my`)
  `==> \. ./sql/as/as_full.sql.my`

E.3.8 AS_AEP

- J2SE sollte installiert sein, die `java` Anwendung sollte sich im `PATH` befinden.
- Die für die JDBC-Verbindung zu AS_SQL benötigten Klassen sollten sich im `CLASSPATH` befinden.
- Die folgenden Konfigurationsdateien und die angegebenen Variablen müssen entsprechend der Installation verändert werden:
  - `./conf/v0.2/as/ccdm/ccdm.sql.conf`

    `databaseDriver, databasePath, databaseUsername, databasePassword`
  - `./conf/v0.2/as/sqltokenmanager.conf`

    `databaseDriver, databasePath, databaseUsername, databasePassword`

E.4 Einstellungen

Bevor AEP benutzt werden kann, müssen noch die folgenden Anwendungen gestartet werden, um die Konfiguration abzuschließen.

E.4.1 AA_AEP

- Es müssen noch verschiedene Schlüssel generiert und registriert werden:

  ```
  ==> ./aasetup
  ```

E.4.2 AA_GUI

- Für die verschiedenen Diensttypen, für die die Methode der blinden Unterschrift benutzt werden soll, müssen RSA-Schlüssel generiert werden:

  ```
  [AA_SQL]
    Der SQL-Server muss laufen

  [AA_AEP]
    ==> ./aa

  [AA_GUI]
    ==> ./aa_gui
    "Methods" > "Blind signature [Chaum]" >
    "Generate new RSA key for given service type"
  ```

E.4.3 AS_AEP

- Es müssen noch verschiedene Schlüssel registriert werden:

```
==> ./assetup
```

- Die öffentlichen Schlüssel des Dienstanbieters und der Anonymitätsinstanz müssen registriert werden:

```
==> ./as
"Configure" > "Specify Server public key"
```

E.5 Starten des Systems

Zuerst müssen die verschiedenen Instanzen der Anonymitätsinstanz gestartet werden. Dann können die Instanzen des Dienstanbieters kommen, und schließlich die des Subjekts.

E.5.1 Anonymitätsinstanz

E.5.1.1 AA_SQL

Der SQL-Server sollte gestartet werden. Z. B. bei MySQL:

```
==> mysqld
```

E.5.1.2 AA_AEP

Die Serverinstanz der Anonymitätsinstanz kann folgendermaßen gestartet werden:

```
==> ./aa
```

E.5.1.3 AA_GUI

Die Administrationsoberfläche der Anonymitätsinstanz kann folgendermaßen gestartet werden:

```
==> ./aa_gui
```

E.5.2 Dienstanbieter

E.5.2.1 SP_WEB

Zuerst muss der Webserver des Dienstanbieters gestartet werden, dann die Konfigurationsinstanz. Schließlich müssen noch die Testseiten erreichbar gemacht werden:

```
==> j2ee
==> ./sps
==> deploytool
```

Die folgende Datei muss auf den J2EE Server heruntergeladen (`deploy`) werden:

```
./web/aep_final_stage3.ear
```

E.5.2.2 SP_AEP

Die Serverinstanz des Dienstanbieters kann folgendermaßen gestartet werden:

```
==> ./sp
```

E.5.2.3 SP_GUI

Die Administrationsoberfläche des Dienstanbieters kann folgendermaßen gestartet werden:

```
==> ./sp_gui
```

E.5.3 Subjekt

E.5.3.1 AS_SQL

Der SQL-Server soll gestartet werden. Z. B. bei MySQL:

```
==> mysqld
```

E.5.3.2 AS_AEP

Der AEP-Client des Subjekts kann folgendermaßen gestartet werden:

```
==> ./as
```

E.6 Testen des AEP

Wie das installierte und konfigurierte System benutzt werden kann, wird im Abschnitt 5.3.5 mit Abbildungen erklärt.

Anhang F Referenzen

[ANON-1] **Anonymizer.com – Online Privacy Services**
http://www.anonymizer.com

[AES-1] **FIPS 197 – Advanced Encryption Standard**, (November 26, 2001)
http://csrc.nist.gov/encryption/aes/

[ARCFOUR-1] K. Kaukonen, R. Thayer: **A Stream Cipher Encryption Algorithm "Arcfour"**, (July 14, 1999)

[BDSG-1] **Bundesdatenschutzgesetz (BDSG)**
http://www.datenschutz-berlin.de/recht/de/bdsg/bdsg1.htm

[BNF-1] Th. Estier: **About the BNF notation**, CUI – University of Geneva
http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html

[C-1] **Rationale for American National Standard for Information Systems – Programming Language C**
http://www.lysator.liu.se/c/rat/title.html

[CHAUM-1] D. Chaum, R.L.Rivest & A.T.Sherman: **Blind Signature for Untraceable Payments,** Advances in Cryptology, Proceedings of Crypto '82

[CHAUM-2] D. Chaum: **Blind Unanticipated Signature Systems**, U.S. Patent 4 759 064, (July 19, 1998)
http://www.uspto.gov

[CHAUM-3] D. Chaum: **Returned Value Blind Signature Systems**, U.S. Patent 4 949 380, (August 14. 1990)
http://www.uspto.gov

[CHAUM-4] D. Chaum: **Security without Identification: Transaction Systems to Make Big Brother Obsolete**, ACM 28, no. 10, p. 1030-1044, (October 1985)
http://www.chaum.com/articles/Security_Wthout_Identification.htm

[CYPHERP-1] E. Hughes: **A Cypherpunk's Manifesto**, (March 9, 1993)
http://www.eff.org/Privacy/Crypto/Crypto_misc/cypherpunk.manifesto

[DIGICASH-1] **eCash Technologies Inc.**
http://www.digicash.com

[DES-1] **FIPS 46-3 – Data Encryption Standard (DES)**, (October 25, 1995)
http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[DSS-1] **FIPS 186-2 – Digital Signature Standard**, (January 27, 2000)
http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf

[EC-1] **SEC 1: Elliptic Curve Cryptography**, Certicom Research (September 20, 2000)
http://www.secg.org/collateral/sec1.pdf

[ETHERNET-1] **IEEE 802.3 CSMA/CD (ETHERNET)**
http://www.ieee.org/groups/802/3/

[FROOMKIN-1] A. M. Froomkin: **Digital Signatures Today**, Proceedings of Financial Cryptography '97
http://www.law.miami.edu/~froomkin/articles/digsig1.pdf

[FROOMKIN-2] A. M. Froomkin: **Flood Control on the Information Ocean: Living with Anonymity, Digital Cash and Distributed Databases**, 15. U. Pittsburgh Journal of Law and Commerce 395 (1996)
http://www.law.miami.edu/~froomkin/articles/oceanno.htm

[GPL-1] **GNU General Public License**
http://www.gnu.org/copyleft/

[HAC-1] A. Menezes, P. van Oorschot, S. Vanstone: **Handbook of Applied Cryptography**, CRC Press, 1996
http://www.cacr.math.uwaterloo.ca/hac

[HMAC-1] H. Krawczyk, M. Bellare, R. Canetti: **RFC 2104 – HMAC: Key Hashing for Message Authentication**, IBM, UCSD, (February 1997)
http://www.ietf.org/rfc/rfc2104.txt

[IETF-1] **Internet Engineering Task Force**
http://www.ietf.org

[IP-1] **RFC 791 – Internet Protocol, Protocol Specification**, Information Sciences Institute, University of Southern California, (September 1981)
http://www.ietf.org/rfc/rfc0791.txt

[IPSEC-1] **IP Security Protocol**
http://www.ietf.org/ids.by.wg/ipsec.html

[J2EE-1] **JAVA 2 Platform, Enterprise Edition**
http://java.sun.com/j2ee

[J2SE-1] **JAVA 2 Platform, Standard Edition**
http://java.sun.com/j2se

[JAP-1] **JAP – Anonymity & Privacy**
http://anon.inf.tu-dresden.de

[JAVA-1] **The source for JAVA Technology**
http://java.sun.com

[JDBC-1] **JDBC Technology**
http://java.sun.com/products/jdbc

[KDE-1] **K Desktop Environment**
http://www.kde.org

[KRÜGER-1] G. Krüger, D. Reschke: **Lehr- und Übungsbuch Telematik**, Fachbuchverlag Leipzig, 2000

[MD5-1] R. Rivest: **RFC 1321 – The MD5 Message Digest Algorithm**, MIT Laboratory for Computer Science & RSA Data Security, Inc., (April 1992)
http://www.ietf.org/rfc/rfc1321.txt

[MONICA-1] **Team MONICA Home**
http://monica.sourceforge.net

[MYSQL-1] **MySQL**
http://www.mysql.com

[NETBEANS-1] **NetBeans**
http://www.netbeans.org

[PAYCASH-1] **PayCash Internet Bank**
http://www.paycash.ru

[PAYCASH-2] I. M. Khamitov: **PayCash Internet Payment System**
http://www.paycash.ru/eng/press/PayCash.zip

[PAYCASH-3] I. M. Khamitov, A. G. Moshonkin, A. L. Smirnov: **Blind unanticipated RSA-signature schemes**
http://www.paycash.ru/eng/press/blind.zip

[PET-1] R. Hes, J. Borking: **Privacy Enhancing Technologies - the path to anonymity**, Registratiekamer, The Hague, (September 1998)
http://www.registratiekamer.nl/bis/top-1-1-9-5-2.html

[PET-2] I. Goldberg, D. Wagner, E. Brewer: **Privacy-enhancing technologies for the Internet**, University of California, Berkeley
http://www.cs.berkeley.edu/~daw/papers/privacy-compcon97.ps

[PKI-1] Netscape Communication Inc.: **Introduction to Public-Key Cryptography**, 1998
http://developer.netscape.com/docs/manuals/security/pkin/contents.htm

[RSA-1] **RSA Security Inc.**
http://www.rsasecurity.com

[RSA-2] **PKCS #1: RSA Cryptography Standard**, RSA Laboratories
http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

[SCHIPSEY-1] R. Schipsey: **How Long ...?**, Royal Holloway, University of London, (April 19, 2001)
https://www.cosic.esat.kuleuven.ac.be/nessie/reports/rhuwp3-015.pdf

[SCHNEIER-1] B. Schneier: **Applied Cryptography: Protocols, Algorithms, and Source Code in C**, John Wiley&Sons, New York, 2nd edition, 1996.

[SEC-1] P. Holbrook, J. Reynolds: **RFC 1244 – Site Security Handbook**, CICNet, ISI, (July 1991)
http://www.ietf.org/rfc/rfc1244.txt

[SEC-2] E. Guttman, L. Leong, G. Malkin: **RFC 2504 – User's Security Handbook**, Sun Microsystems, COLT Internet, Bay Networks, (February 1999)
http://www.ietf.org/rfc/rfc2504.txt

[SHS-1] **FIPS 180-1 – Secure Hash Standard (SHS)**, (April 17, 1995)
http://www.itl.nist.gov/fipspubs/fip180-1.htm

[SSH-1] SSH IETF: **Secure Shell internet-drafts**,
http://www.ietf.org/ids.by.wg/secsh.html

[SSH-2] **SSH Communication Security**
http://www.ssh.com

[SSL-1] Alan O. Freier, Philip Karlton, Paul C. Kocher: **The SSL Protocol Version 3.0**, 1996
http://home.netscape.com/eng/ssl3/ssl-toc.html

[SSL-2] Netscape Communication Inc.: **Introduction to SSL**, 1998
http://developer.netscape.com/docs/manuals/security/sslin/index.htm

[SUSE-1] **SuSE Linux**
http://www.suse.com

[SZEKELY-1] I. Székely: **Jogi és technolgógiai szempontoka személyes adatokat tartalmazó adatbázisok kezeléséhez**, DAT '97/ASzJE Conference

[TCP-1] **RFC 793 – Transmission Control Protocol, Protocol Specification**, Information Sciences Institute, University of Southern California, (September 1981)
http://www.ietf.org/rfc/rfc0793.txt

[TELNET-1] J. Postel, J. Reynolds: **RFC 854 – TELNET Protocol Specification**, ISI, May 1983
http://www.ietf.org/rfc/rfc0854.txt

[TLS-1] T. Dierks, C. Allen: **RFC 2246 – The TLS Protocol Version 1.0**, Certicom, (January 1999)
http://www.ietf.org/rfc/rfc2246.txt

[WAP-1] **WAP – Wireless Application Protocol**
http://www.wapforum.org

[WTLS-1] **WAP – Wireless Transport Layer Security**
http://www.wapforum.org

[X509-1] **Information technology – Open systems interconnection – Public-key and attribute certificate frameworks, ITU-T Recommendation X.509**
http://www.itu.org

[XML-1] **Extensible Markup Language**
http://www.w3.org/xml

[XML-2] **JAVA API for XML Processing**
http://java.sun.com/xml/jaxp/index.html

[YAHOODDOS-1] **Yahoo Attributes a Lengthy Failure to an Attack**, The New York Times, February 8, 2000
http://www.nytimes.com/library/tech/00/02/biztech/articles/08yahoo.html

[ZLIB-1] P. Deutsch, J-L. Gailly: **RFC 1950 – ZLIB Compressed Data Format Specification version 3.3**, Aladdin Enterprises, Info-ZIP, May 1996
http://www.ietf.org/rfc/rfc1950.txt

[ZLIB-2] P. Deutsch: **RFC 1951 – DEFLATE Compressed Data Specification version 1.3,** Aladdin Enterprises, May 1996
http://www.ietf.org/rfc/rfc1951.txt