Authentifikations- und Schlüsselverteilsysteme

Rolf Oppliger
Andreas Greulich
Dieter Hogrefe

IAM-94-006

Juni 1994

Authentifikations- und Schlüsselverteilsysteme

Rolf Oppliger
Andreas Greulich
Dieter Hogrefe

Institut für Informatik und angewandte Mathematik
Universität Bern
Länggassstr. 51
CH-3012 Bern

Tel. +41 31 631 49 03
Fax. +41 31 631 39 65
{oppliger,greulich,hogrefe@iam.unibe.ch}

CR Category and Subject Descriptor: C.2.0 [Computer Systems Organization]: Computer-Communication Networks — general; K.6.5 [Computing Milieux] Management of Computing and Information Systems — security and protection

General Terms: Security

Additional Key Words: Network Security, Authentication, Key Distribution

Zusammenfassung

Für den Einsatz in Computer- und Kommunikationsnetzen stehen heute eine ganze Reihe von Authentifikations- und Schlüsselverteilsystemen zur Auswahl. In diesem Bericht werden Kerberos, NetSP, SPX, TESS und SESAME vorgestellt, diskutiert und miteinander verglichen.

1 Einleitung

In den meisten Computer- und Kommunikationsnetzen, die heute im Einsatz stehen, werden Benutzer über Passwörter authentifiziert. Angreifer können die in diesen Netzen übertragenen Passwörter passiv abhorchen und später für Identitätstäuschungen missbrauchen.

Mit Hilfe eines Authentifikations- und Schlüsselverteilsystemes kann diese Verwundbarkeit überwunden werden. Verschiedene Systeme mit unterschiedlichen Eigenschaften sind bis heute entwickelt und auf den Markt gebracht worden. In den folgenden Abschnitten werden Kerberos, NetSP, SPX, TESS und SESAME vorgestellt und diskutiert. Ein Vergleich findet sich im letzten Abschnitt.

2 Kerberos

Kerberos [SNS88] ist ein Authentifikations- und Schlüsselverteilsystem, das im Rahmen des Projektes *Athena* [Cha91] am Massachusetts Institute for Technology (MIT) entwickelt worden ist. Der Systemname wurde der griechischen Mythologie entnommen; "Kerberos" war ein dreiköpfiger Hund, der den Eingang zur Unterwelt bewachte.

Kerberos ist zentral organisiert. Ein Authentifikationsserver (AS) ist in einem "Realm" für die Authentifikation von Benutzern und für die Verteilung von Schlüsseln zuständig. Verläuft die Authentifikation eines Benutzers gegenüber dem AS erfolgreich, dann erhält der Benutzer ein Ticket, das seine Identität beglaubigt, und das er bei einem *Ticket Granting Server* (TGS) einlösen kann, um ein Ticket für die Beanspruchung einer Dienstleistung zu erhalten, die von einem Server angeboten wird. Durch den Einsatz von dezentralen TGS soll verhindert werden, dass der AS durch die Ausstellung von Tickets permanent überlastet wird.

Ein *Ticket* (ticket) bezeichnet einen Datenblock, mit dem sich ein Klient gegenüber einem Server ausweisen kann, und der bestimmte Informationen über den Klienten und den Server, sowie zeitliche Rahmenbedingungen enthält. Das Ticket $Tick_S = \{C, S, K, T, L\}_{K_S}$ kann vom Klienten C verwendet werden, um sich gegenüber dem Server S auszuweisen. K ist ein für die Kommunikation zwischen C und S einzusetzender Sitzungsschlüssel. Der Zeitstempel T und die maximale Lebensdauer L definieren die zeitlichen Rahmenbedingungen für das Ticket. *Authentikatoren* sind Datenblöcke, die Integritätsprüfungen von Tickets ermöglichen.

Die in Kerberos eingesetzten Authentifikations- und Schlüsselverteilprotokolle gehen auf Vorschläge von Needham und Schroeder zurück [NS78], und wurden später um global synchronisierte Zeitstempel erweitert [DS81, NS87]. Sie basieren auf einem symmetrischen Kryptosystem.

Im folgenden sind zwei der in Kerberos eingesetzten Protokolle beschrieben. Das

Protokoll (1) hat ein Benutzer U zu durchlaufen, wenn er sich gegenüber dem AS authentifizieren will, um ein Ticket für einen TGS zu erhalten (vgl. Abbildung 1):

$$
\begin{array}{lllll}
1: \mathrm{U} & \longrightarrow & \mathrm{C} & : \mathrm{U} \\
2: \mathrm{C} & \longrightarrow & \mathrm{AS} & : \mathrm{U,TGS} \\
3: \mathrm{AS} & & & : Tick_{TGS} := \{U, TGS, K, T, L\}_{K_{tgs}} \\
4: \mathrm{AS} & \longrightarrow & \mathrm{C} & : \{TGS, K, T, L, Tick_{TGS}\}_{K_u} \\
5: \mathrm{C} & \longrightarrow & \mathrm{U} & : \text{“Password?”} \\
6: \mathrm{U} & \longrightarrow & \mathrm{C} & : p \\
7: \mathrm{C} & & & : K_u := f(p) \\
8: \mathrm{C} & & & : \text{recover } \{TGS, K, T, L, Tick_{TGS}\}
\end{array}
$$

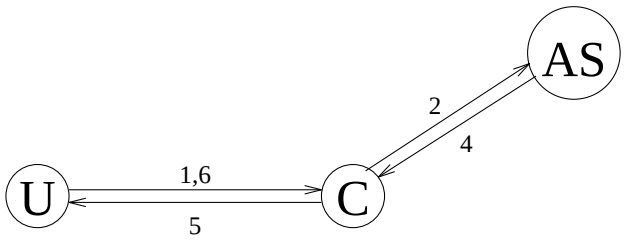

Abbildung 1: Kerberos Protokoll (1)

Im ersten Schritt übergibt U dem Klienten C seine Identifikation. C stellt diese Identifikation zusammen mit der Identifikation eines TGS dem AS zu. Der AS konstruiert im dritten Schritt das Ticket $Tick_{TGS} := \{U, TGS, K, T, L\}_{K_{tgs}}$ und gibt dieses Ticket, zusammen mit der Identifikation des TGS, dem zwischen C und dem TGS einzusetzenden Sitzungsschlüssel K, sowie den gleichen zeitlichen Rahmenbedingungen T und L, wie sie in $Tick_{TGS}$ festgelegt worden sind, im vierten Schritt C zurück. Weil die ganze Nachricht mit Hilfe des geheimen Schlüssels von U chiffriert worden ist, kann C die Nachricht vorerst nicht dechiffieren. U wird im fünften Schritt von C aufgefordert, sein Passwort einzugeben. Aus dem eingegebenen Passwort p und einer bekannten Einwegfunktion f kann C im siebenten Schritt den Schlüssel $K_u := f(p)$ rekonstruieren und damit im achten Schritt $\{TGS, K, T, L, Tick_{TGS}\}_{K_u}$ dechiffrieren.

Wenn die Dechiffrierung nicht gelingt, oder wenn das Ticket aufgrund von T und L nicht mehr gültig ist, dann gilt das Protokoll als nicht erfolgreich durchlaufen. Anderenfalls kann C aus der letzten Nachricht den für die Kommunikation mit dem TGS zu verwendenden Sitzungsschlüssel K extrahieren. Weil $Tick_{TGS}$ aber mit dem geheimen und nur AS und TGS bekannten Schlüssel K_{tgs} chiffriert worden ist, kann es von C nicht aufgelöst werden.

$Tick_{TGS}$ kann im Wesentlichen nur dazu verwendet werden, um beim spezifizierten TGS ein Ticket für einen Server S zu beantragen. Dazu ist das Protokoll (2) zu durchlaufen (vgl. Abbildung 2):

$$
\begin{array}{rlll}
9: & \text{C} & \longrightarrow \text{TGS}: & S, Tick_{TGS}, \{C, T_1\}_K \\
10: & \text{TGS} & & : \text{recover } K, T_1 \\
11: & \text{TGS} & & : \text{check } T_1 \\
12: & \text{TGS} & & : Tick_S := \{C, S, K', T', L'\}_{K_s} \\
13: & \text{TGS} & \longrightarrow \text{C} & : \{S, K', T', L', Tick_S\}_K \\
14: & \text{C} & & : \text{recover } K', Tick_S \\
15: & \text{C} & \longrightarrow \text{S} & : Tick_S, \{C, T_2\}_{K'} \\
16: & \text{S} & & : \text{recover } K', T_2 \\
17: & \text{S} & & : \text{check } T_2 \\
18: & \text{S} & \longrightarrow \text{C} & : \{T_2 + 1\}_{K'}
\end{array}
$$

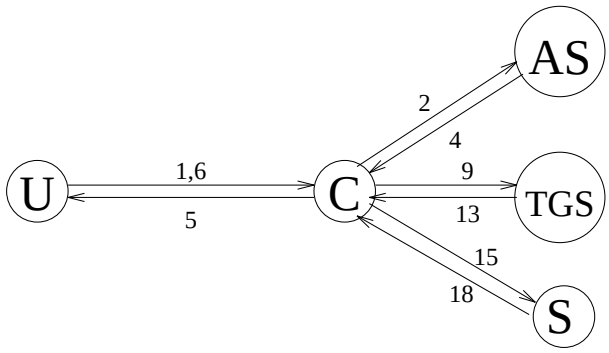

Abbildung 2: Kerberos Protokoll (2)

Im neunten Schritt übergibt C dem TGS die Identifikation von S, das Ticket $Tick_{TGS}$, sowie einen mit K chiffrierten Authentikator $\{C, T_1\}_K$. Mit Hilfe von K_{tgs} kann der TGS $Tick_{TGS}$ dechiffrieren, K extrahieren, damit den Authentikator dechiffrieren und in seine Komponenten C und T_1 zerlegen. Nun kann er die Prüfsumme C nachrechnen und den Zeitstempel T_1 mit seiner lokalen Zeit vergleichen. Fallen beide Prüfungen erfolgreich aus, dann wird $Tick_{TGS}$ als gültig angenommen. Der TGS konstruiert dann ein Ticket $Tick_S = \{C, S, K', T', L'\}_{K_s}$ für den Server S und stellt dieses Ticket C zu. Ebenfalls eingebunden in diese mit K chiffrierte Nachricht sind die Identifikation von S, ein zwischen C und S einzusetzender Sitzungsschlüssel K', sowie die gleichen zeitlichen Rahmenbedingungen T' und L', wie sie in $Tick_S$ enthalten sind. Weil auch C den Schlüssel K kennt, kann er im vierzehnten Schritt die Nachricht wieder in die Komponenten zerlegen. Insbesondere kann er K' und $Tick_S$ extrahieren. Das Ticket $Tick_S$ stellt er zusammen mit dem Authentikator $\{C, T_2\}_{K'}$ dem Server S zu. Mit Hilfe von K_s kann dieser $Tick_S$ dechiffrieren, K' extrahieren, den Authentikator dechiffrieren und in seine Komponenten C und T_2 zerlegen. Wiederum können Prüfsumme und Zeitstempel überprüft und das Ticket entsprechend angenommen oder verworfen werden. Nimmt S das Ticket an, dann kann er sich gegenüber C ausweisen, indem er $\{T_2 + 1\}_{K'}$ zurückgibt.

Mit Hilfe von K' können Datenvertraulichkeits- und -integritätsdienste realisiert werden. Für die Bildung der Hashwerte sind CRC-32, MD4, MD5 und DES im

CBC-Modus vorgesehen.

Der Begriff "Kerberos" hat sich in den letzten Jahren zu einem Synonym für sichere Netze entwickelt. Viele staatliche und private Institutionen scheinen sich heute in zunehmendem Mass dafür zu interessieren. Trotz der Vermeidung von Passwortübertragungen im Netz, sowie der zeitlich beschränkten Lebensdauer von Tickets, bleiben ein paar Kritikpunkte offen [BM90]:

- Die Vermeidung von Passwortübertragungen im Netz beseitigt z.B. nicht die Probleme, die mit der Passwortwahl behaftet sind. Der AS antwortet in Schritt vier von Protokoll (1) mit einer Nachricht, die unter einem passwortabhängigen Schlüssel chiffriert worden ist. Liegt diesem Schlüssel ein schlechtes Passwort zugrunde, dann kann ein Eindringer die Nachricht aufgreifen und Off-line solange Wörter ausprobieren, bis die Dechiffrierung der Nachricht Sinn ergibt. Beim betreffenden Wort wird es sich dann mit grosser Wahrscheinlichkeit um ein Passwort handeln [MLG+89, GLNS93, TVH93b].
- Auf der anderen Seite löst die zeitlich beschränkte Lebensdauer von Tickets nicht die Frage nach einer sinnvollen Lebensdauer. Hier muss ein Kompromiss zwischen Sicherheit vor Wiedereinspielungen und Benutzerkomfort gefunden werden.

Weitere Kritikpunkte betreffen die Ausrichtung von Kerberos ausschliesslich am Client-Server-Modell, d.h. die gemachten Annahmen in Bezug auf die Einsatzumgebungen. Zudem besitzt der AS mit seiner Fähigkeit, grundsätzlich alle Kommunikationsbeziehungen überwachen zu können, eine "Big Brother"-Eigenschaft, die entweder gewünscht oder nicht gewünscht sein kann.

Kerberos ist als Public Domain Software verfügbar und kann z.B. aus dem Verzeichnis `/pub/kerberos` des FTP-Servers `athena-dist.mit.edu` heruntergeladen werden. Während die Version 4 von Kerberos relativ stabil läuft, befindet sich die Version 5 noch in einem Betastadium; sie wurde von inzwischen als RFC 1510 vorgeschlagen [KN93].

Die Kerberos zugrunde liegenden Authentifikations- und Schlüsselverteilprotokolle basieren auf einem symmetrischen Kryptosystem. Die Entwickler von Kerberos hatten sich seinerzeit für DES entschieden. Aufgrund der amerikanischen Export-Beschränkungen ergeben sich aber rechtliche Probleme bei der Installation von Kerberos ausserhalb der USA und Kanada. Um diesen Problemen zu begegnen, wurde eine Version von Kerberos entwickelt, in der die DES-Bibliothek von Kerberos durch eine andere Bibliothek ersetzt ist. Diese Version von Kerberos ist unter dem Namen *Bones* verfügbar und kann z.B. aus dem Verzeichnis `/computing/systems/athena/eBones` des FTP-Servers `doc.ic.ac.uk` heruntergeladen werden.

3 NetSP

Die Tatsache, dass aus den USA Kryptosysteme nur dann ohne spezielle Ausfuhrbewilligung exportiert werden können, wenn sie ausschliesslich für Authentifikationszwecke und nicht für Datenchiffrierungen eingesetzt werden, hat IBM veranlasst, eine Familie von Authentifikations- und Schlüsselverteilprotokollen zu entwickeln, die auf dem Einsatz von Nachrichtenauthentifikationscodes (MACs) beruhen [BGH+92a, BGH+92b, BGH+93]. Für die Bildung von MACs können z.B. MD4 oder MD5 eingesetzt werden.

Auf der Basis dieser Protokolle wurde zuerst ein Prototyp namens *KryptoKnight* [MTVHZ92] und dann ein kommerziell verfügbares Produkt namens *NetSP* (Network Security Program) entwickelt.

Alle Protokolle basieren auf einem kanonischen 2-Parteien Authentifikationsprotokoll (2PAP), dessen kryptanalytische Stärke formal gezeigt werden kann. Obwohl sich das 2PAP sowohl mit einem symmetrischen als auch mit einem asymmetrischen Kryptosystem betreiben lässt, wird im folgenden nur der symmetrische Fall beschrieben.

Mit Hilfe eines gemeinsamen Sitzungsschlüssels K_{ab} können sich A und B gegenseitig authentifizieren, indem sie das folgende Protokoll durchlaufen:

$$\begin{array}{lcl} 1 : \text{A} & \longrightarrow & \text{B} : A, N_a \\ 2 : \text{B} & \longrightarrow & \text{A} : B, N_b, MAC_{K_{ab}}(N_a, N_b, N_a \oplus B) \\ 3 : \text{A} & \longrightarrow & \text{B} : MAC_{K_{ab}}(N_a, N_b) \end{array}$$

Im ersten Schritt übergibt A seine Benutzerkennung und eine 64 Bit lange Zufallsmarke N_a an B. Dieser antwortet im zweiten Schritt mit seiner Benutzerkennung, einer Zufallsmarke N_b und einem ebenfalls 64 Bit langen MAC, den er mit Hilfe des Sitzungsschlüssel K_{ab} aus N_a, N_b und $N_a \oplus B$ errechnet. Weil A die MAC-Komponenten auch kennt, kann er dessen Echtheit prüfen. Verläuft die Prüfung erfolgreich, dann kann A im dritten Schritt mit K_{ab} einen MAC über (N_a, N_b) rechnen und B zurückgeben. Wurde dieser MAC korrekt berechnet, dann kann auch B davon ausgehen, dass A authentisch sei. Die von 2PAP geleistete Authentifikation ist also gegenseitig.

Die Zufallsmarken N_a und N_b haben sicherzustellen, dass bei jedem Protokolllauf die zwischen A und B ausgetauschten Nachrichten verschieden sind. Ansonsten könnte ein Angreifer die Kommunikationsbeziehung passiv abhorchen und die dabei ausgetauschten Nachrichten aufzeichnen, um sie später für eine Identitätstäuschung zu missbrauchen.

Damit 2PAP korrekt arbeiten kann, müssen A und B einen Schlüssel K_{ab} bereits teilen. Dieser ist ihnen ebenfalls über das Netz bekanntzugeben. Dazu werden

Tickets eingesetzt; ein Ticket ist eine Nachricht, mit der ein geheimer Schlüssel von einem Ticketherausgeber (TH) einem Ticketempfänger (TE) zugestellt werden kann. Der Ticketherausgeber ist typischerweise ein Authentifikationsserver; als Ticketempfänger treten A und B auf. Das Ticket besteht aus Teilen, die im Klartext übertragen werden, und einem 64 Bit langen Token, das mit dem Schlüssel K_m chiffriert wird, den der Ticketherausgeber mit dem Ticketempfänger teilt. Die Teile, die im Klartext übertragen werden, sind

- eine vom TE erzeugte Zufallsmarke N_r,
- eine vom TH erzeugte Marke N_i, die bei der Anmeldung zufällig und sonst in Funktion zu N_r gewählt wird,
- der Name eines Teilnehmers T, mit dem der neue Schlüssel zu teilen ist, sowie
- einer Verfallzeit für den Schlüssel.

Das 64 Bit lange Token ist dann folgendermassen zu berechnen:

$MAC_{K_m}(N_i \oplus T, N_r, N_i \oplus TH, Verfallzeit) \oplus K_{ab}$

Offenbar kann der Schlüssel K_{ab} nicht ohne Kenntnis von K_m aus dem Ticket extrahiert werden.

Verfügen A und B über einen gemeinsamen Schlüssel K_{ab} und haben sich mit Hilfe von 2PAP gegenseitig authentifiziert, dann können sie bilateral einen neuen Sitzungsschlüssel K_{new} aushandeln. Das entsprechende 2-Parteien-Schlüsselverteilprotokoll (2PKDP) sieht dann folgendermassen aus:

$$\begin{array}{lcl} 1 : \text{A} & \longrightarrow & \text{B} : A, N_a \\ 2 : \text{B} & \longrightarrow & \text{A} : N_b, MAC_{K_{ab}}(N_a, N_b, B) \oplus K_{new} \end{array}$$

Wiederum übergibt A im ersten Schritt B seine Benutzerkennung und eine 64 Bit lange Zufallsmarke N_a. B generiert dann den neuen Schlüssel K_{new}, addiert ihn bitweise modulo 2 mit einem MAC, den er mit Hilfe von K_{ab} aus (N_a, N_b, B) errechnet, und stellt das Resultat zusammen mit einer neuen Zufallsmarke N_b A zu. A wird zuerst den MAC und dann K_{new} rekonstruieren können.

Wollen sich A und B auf einen Schlüssel einigen, ohne bereits einen Schlüssel zu teilen, dann haben sie sich eines 3-Parteien-Schlüsselverteilprotokolles (3PKDP) zu bedienen. Ein 3PKDP setzt die Existenz einer Schlüsselverteilzentrale (KDC) voraus, die als AS mit jedem Benutzer einen Schlüssel teilt. Ein dem Mixed Model folgendes 3PKDP erhält man aus zwei 2PKDP-Läufen:

$$
\begin{array}{lcll}
1 : \text{A} & \longrightarrow & \text{AS} & : A, B, N_a \\
2 : \text{AS} & \longrightarrow & \text{A} & : N_{sa}, MAC_{K_a}(N_a, N_{sa}, B) \oplus K_{ab} \\
3 : \text{B} & \longrightarrow & \text{KDC} & : B, A, N_b \\
4 : \text{KDC} & \longrightarrow & \text{B} & : N_{sb}, MAC_{K_b}(N_b, N_{sb}, A) \oplus K_{ab}
\end{array}
$$

N_{sa} und N_{sb} stehen dabei für Zufallsmarken, die vom AS für A und B gewählt worden sind. Die 3PKDP für die Push und Pull Modelle finden sich in [TVH93a, JT93].

In einem vernetzten System wäre es wünschenswert, dass sich ein Benutzer für die Dauer einer Arbeitssitzung nur einmal zu authentifizieren hätte (Single Sign-On), und dass ihm dabei ein kryptanalytisch starker Schlüssel zugewiesen würde. Weil der Benutzer zu Beginn einer Sitzung mit dem AS keinen Schlüssel teilt, wird mit Hilfe einer Einwegfunktion aus dem Passwort des Benutzers ein Schlüssel errechnet, der dann sowohl für die gegenseitige Authentifizierung zwischen Benutzer und AS, als auch für die gesicherte Übertragung eines neu erzeugten Schlüssels eingesetzt wird. Das Protokoll sieht folgendermassen aus:

$$
\begin{array}{lcll}
1 : \text{A} & \longrightarrow & \text{AS} : & A, N_a, N'_a = MAC_{K_{pwa}}(N_a) \\
2 : \text{AS} & \longrightarrow & \text{A} & : T_{pwa}, T_{ka}, N_k, \\
\quad : & & & : MAC_{K_a}(N'_a, N'_k, N'_a \oplus AS), N'_a \\
3 : \text{A} & \longrightarrow & \text{AS} : & MAC_{K_a}(N'_a, N'_k)
\end{array}
$$

Im ersten Schritt stellt A dem AS seinen Namen und zwei Zufallsmarken zu. Die Tatsache, dass die Marke N'_a eine Funktion von N_a ist, ermöglicht dem AS eine Überprüfung der Authentizität von A; nur A kennt sein Passwort und kann damit N'_a berechnen. Nach einer erfolgreichen Prüfung stellt der AS dem Benutzer zwei Tickets zu. Das Format der Tickets entspricht dem oben diskutierten, wobei der AS als Ticketherausgeber und A als Ticketempfänger auftritt. Der dritte Teilnehmer ist ebenfalls der AS. T_{ka} kann A dann seinen Schlüssel K_a entnehmen und mit Hilfe des mitgelieferten MAC überprüfen. Schliesslich muss A die erfolgreiche Schlüsselübergabe bestätigen, indem er über (N'_a, N'_k) mit dem neuen Schlüssel einen MAC errechnet und im dritten Schritt dem AS zuspielt. A teilt nun mit dem AS den Schlüssel K_a.

Bemerkenswert an KryptoKnight und NetSP ist die Tatsache, dass sie sich als IBM-Produkte nicht auf die Programmierschnittstelle der Common Cryptographic Architecture (CCA-API) beziehen, sondern der GSS-API (Generic Security ServiceI Application Programming Interface) folgen [Lin93, Wra93].

NetSP ist seit Januar 1994 für IBM AIX/6000-, OS/2- und DOS-Systeme verfügbar; für andere Systemplattformen und Protokolle liegen Portierungspläne vor. Untersucht werden zurzeit auch Möglichkeiten, NetSP um Datenvertraulichkeitsdienste und für den Einsatz in mobilen Systemen zu erweitern [MST94].

Unklar ist zurzeit noch die marktpolitische Abgrenzung von NetSP gegenüber der am Ende dieses Kapitels behandelten OSF DCE. Innerhalb von OSF DCE könnte NetSP zum Einsatz kommen, wenn Authentifikationsdienste auf der Transportschicht (oder unterhalb der Transportschicht) anzubieten sind.

4 SPX

Mit dem *Distributed Authentication Security Service* (DASS) hat die Digital Equipment Corporation (DEC) einen Authentifikationsdienst spezifiziert [GKL+92, Kau93] und als Teil der Digital *Distributed System Security Architecture* (DSSA) [GGKL89] im Prototypen *SPX* umgesetzt [TA91].

Im Unterschied zu Kerberos und NetSP basiert SPX auf einem hybriden Ansatz, d.h. es werden sowohl ein symmetrisches (DES), als auch ein asymmetrisches Kryptosystem (RSA) eingesetzt.

Die Namensgebung in SPX folgt der ITU-T Empfehlung X.500. Bis im Rahmen eines X.500-Verzeichnisdienstes öffentliche Schlüssel aber zertfiziert und auf globaler Ebene zur Verfügung stehen, werden in SPX öffentliche Schlüssel noch von *Certificate Distribution Centers* (CDCs) verwaltet und verteilt.

Für die Initialisierung und Installierung von Benutzerbeglaubigungen (credentials initialization) stellt SPX einen Dienst zur Verfügung, der als *Login Enrollment Agent Facility* (LEAF) bezeichnet wird. Um seine Beglaubigungen auf dem Klienten C zu installieren, hat der Benutzer U das folgende Protokoll zu durchlaufen (vgl. Abbildung 3):

$$
\begin{array}{lcll}
1: \text{U} & \longrightarrow & \text{C} & : \text{U},p \\
2: \text{C} & \longrightarrow & \text{LEAF} & : \{U, T, N, H_1(p)\}_{K_{leaf}} \\
3: \text{LEAF} & \longrightarrow & \text{CDC} & : \text{U} \\
4: \text{CDC} & \longrightarrow & \text{LEAF} & : \{\{K_u^{-1}\}_{H_2(p_U)}, H_1(p_U), U\}_K, \{K\}_{K_{leaf}} \\
5: \text{LEAF} & \longrightarrow & \text{C} & : \{\{K_u^{-1}\}_{H_2(p_U)}, U\}_N \\
6: \text{C} & \longrightarrow & \text{CDC} & : \text{U} \\
7: \text{CDC} & \longrightarrow & \text{C} & : \{U, TA_u, L, K_{TA_U}\}_{K_u^{-1}}
\end{array}
$$

Im ersten Schritt hat der Benutzer U dem Klienten C seinen Namen und sein Passwort p zu übergeben. C wendet dann eine Hashfunktion H_1 auf p an und chiffriert das Resultat $H_1(p)$ zusammen mit dem Namen von U, einem Zeitstempel T und einer frischen Zufallsmarke N mit dem öffentlichen Schlüssel des LEAF-Dienstes, d.h. mit K_{leaf}. Das Resultat spielt C im zweiten Schritt der LEAF zu. Die LEAF kontaktiert im dritten Schritt das CDC, um mit dem privaten Schlüssel von U und mit dem für U registrierten Passwort versorgt zu werden. Das CDC verwaltet diese Angaben in Form eines Tripels, das sich aus den Komponenten $\{K_u^{-1}\}_{H_2(p_U)}$, $H_1(p_U)$

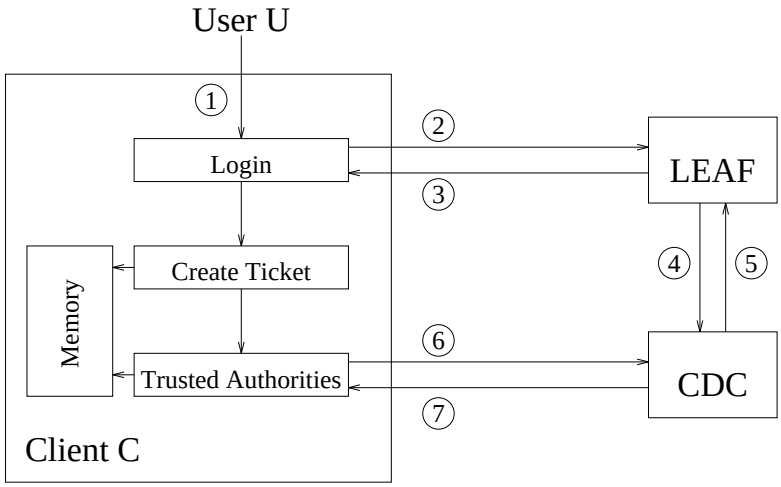

Abbildung 3: Initialisierung

und U zusammensetzt. Dabei stellt p_U das Passwort von U und H_2 eine zweite Hashfunktion dar. Das CDC wählt nun einen zufälligen DES-Schlüssel K aus, chiffriert damit das Tripel für U und stellt den Chiffretext im vierten Schritt der LEAF zu. Ergänzt wird die Nachricht noch um den mit dem öffentlichen Schlüssel des LEAF-Dienstes, d.h. mit K_{leaf}, chiffrierten DES-Schlüssel K. Die LEAF kennt ihren privaten Schlüssel K_{leaf}^{-1} und kann damit K dechiffrieren. Mit Hilfe von K kann die LEAF das chiffrierte Tripel entschlüsseln und in die einzelnen Komponenten auflösen; sie erhält $\{K_u^{-1}\}_{H_2(p_U)}$, $H_1(p_U)$ und U. Nun kann der mit H_1 errechnete Hashwert für das von U eingegebene Passwort mit dem vom CDC erhaltenen Wert verglichen werden. Sind $H_1(p)$ und $H_1(p_U)$ verschieden, dann wird das Protokoll als nicht erfolgreich durchlaufen betrachtet. Anderenfalls gilt die Authentifizierung von U als erfolgreich. Die LEAF kann dann im fünften Schritt das mit Hilfe der Zufallsmarke N DES-chiffrierte Paar $\{\{K_u^{-1}\}_{H_2(p_U)}, U\}_N$ C zurückgeben. Mit Hilfe von N und $H_2(p_U)$, bzw.$H_2(p)$ kann C aus diesem Chiffretext den privaten Schlüssel K_u^{-1} von U rekonstruieren.

Es ist nun die Aufgabe von C, für U eine Beglaubigung zu installieren. Dazu werden mit (K_d, K_d^{-1}) ein zeitlich befristetes RSA-Delegationsschlüsselpaar und mit $Tick_U = \{L, U, K_d\}_{K_u^{-1}}$ ein mit K_u^{-1} signiertes Ticket für U instanziert. L legt die Lebensdauer des Tickets fest. Schliesslich muss C beim CDC noch Zertifiakte von *vertrauenswürdigen Autoritäten* (trusted authorities, TAs) TA_u für U einholen. Diese Zertifikate werden C im siebenten Schritt zugestellt; sie sind mit K_u^{-1} signiert und können mit K_u geprüft werden. Alle TA_u bilden eine TA-Liste für U. Die Beglaubigung für U besteht nun aus $K_d, K_d^{-1}, Tick_U$ und einer TA-Liste. Diese Daten sind lokal zu verwalten.

Will sich C im Auftrag von U gegenüber einem Server S authentifizieren, dann tritt C als Beansprucher und S als Prüfer auf. Beide benötigen einen Zugang zu einem CDC, um in den Besitz des öffentlichen Schlüssels des jeweiligen Partners zu

kommen. Die Authentifiaktion von C gegenüber S erfolgt über ein dreiteiliges Authentifikationstoken, auf dessen genauen Aufbau an dieser Stelle nicht eingegangen werden kann. Das Token enthält unter anderem einen mit DES chiffrierten Zeitstempel. Der für die Chiffrierung verwendete Schlüssel wird von C zufällig gewählt, mit dem öffentlichen Schlüssel von S chiffriert und in das Authentifikationstoken miteingebunden. S kann mit seinem privaten Schlüssel den DES-Schlüssel dechiffrieren und die Aktualität des Zeitstempels überprüfen. Will sich S auch gegenüber C authentifizieren, dann retourniert er einfach einen mit demselben DES-Schlüssel chiffrierten Zeitstempel.

Interessant an DASS und SPX ist die Tatsache, dass sie sich zum Teil ebefalls auf die Programmierschnittstelle GSS-API beziehen. Für SPX eher negativ zu werten ist die Tatsache, dass sich DEC strategisch auf OSF DCE ausgerichtet hat, und dass die Entwicklungsarbeiten an SPX zwischenzeitlich eingestellt worden sind.

SPX ist heute frei verfügbar und kann aus dem Verzeichnis `pub/DEC/SPX` des FTP-Servers von DEC heruntergeladen werden. Eine Mailing-Liste SPX-bezogene Mailing-Liste wird über `sphinx@crl.dec.com` moderiert; Anmeldungen und Kündigungen werden von `sphinx-request@crl.dec.com` entgegengenommen.

5 TESS

Auf der Basis des diskreten Logarithmusproblemes wurde am Europäischen Institut für Systemsicherheit (E.I.S.S.) der Universität Karlsruhe E.I.S.S. mit *TESS* (The Exponential Security System) ein Sicherheitssystem für heterogene Netze entwickelt [Bet91, Bet94]. TESS umfasst ein Authentifikations- und Schlüsselverteilsystem *SELANE* (Secure Local Area Network Environment) und ein digitales Unterschriftensystem *EES* (Exponential Electronic Signature).

5.1 SELANE

Das Authentifikations- und Schlüsselverteilsystem SELANE bezieht sich auf ein Familie von kryptographischen Protokollen, die unter dem Begriff "KATHY" zusammengefasst werden [BK90, HK91b, HK91a]. "K" und "Y" stehen für "KeY exchange" und "ATH" für "embedded AuTHentication"; KATHY steht damit für "key exchange with embedded authentication", also Schlüsselverteilung mit integrierter Authentifizierung.

Die KATHY-Protokolle sehen eine *Secure Key Issuing Authority* (SKIA) vor, die zwar für die Registrierung der Benutzer und für die Generierung, Unterzeichnung und Verteilung von Merkmalslisten zuständig ist, die aber nicht On-line zur Verfügung stehen muss. Die Protokolle laufen in drei Phasen ab:

1. Initialisierung der SKIA

2. Registrierung der Benutzer

3. Authentifikation der Benutzer

Die Initialisierungsphase muss genau einmal durchlaufen werden, nämlich bei Inbetriebnahme des Systems. Als Basisparameter sind eine grosse Primzahl p und eine Primitivwurzel α modulo p zu bestimmen. Die SKIA wählt dann eine geheime Zufallszahl $X \in GF(p)$ und berechnet $Y := \alpha^X \bmod p$. Während p, α und Y öffentlich bekanntgegeben werden, bleibt X geheim.

Für die Registrierung eines Benutzers i muss die SKIA eine eindeutige Merkmalsliste $m_i \in GF(p)$ konstruieren, die sich z.B. aus dem Namen, dem Geburtsdatum und der Anschrift des Benutzers, sowie einer Gültigkeitsdauer zusammensetzt. Beim KATHY-Basisprotokoll wählt die SKIA dann ein zufälliges $k_i \in_R Z^*_{p-1}$ und errechnet für $r_i := \alpha^{k_i} \bmod p$ ein s_i, das die Kongruenz $Xr_i + s_i k_i \equiv m_i \bmod (p-1)$ erfüllt. Das Paar (r_i, s_i) stellt dann eine ElGamal-Signatur der Merkmalsliste m_i dar. Die Parameter m_i, r_i und s_i werden dem Benutzer z.B. in Form einer nicht auslesbaren Speicher- oder Chipkarte ausgehändigt. An dieser Stelle muss mit besonderem Nachdruck auf die Notwendigkeit hingewiesen werden, s_i auch gegenüber dem Benutzer geheimzuhalten. Im Gegensatz zu den anderen Parametern wird k_i nicht weiter benötigt und kann gelöscht werden.

Will sich A mit B auf einen Sitzungsschlüssel einigen, dann teilt er ihm seine Parameter m_A und r_A mit. B wählt eine zufällige Zahl $z_B \in_R Z^*_{p-1}$ und sendet $v_B := r_A^{z_B} \bmod p$ an A. Beide sind nun in der Lage, einen gemeinsamen Schlüssel K_A zu errechnen; A rechnet $K_A := v_B^{s_A} \bmod p$ und B $K'_A := (\alpha^{m_A} Y^{-r_A})^{z_B} \bmod p$. Das Protokoll lässt sich folgendermassen zusammenfassen:

$$
\begin{array}{lcll}
1: \text{A} & \longrightarrow & \text{B}: & m_A, r_A \\
2: \text{B} & & : & z_B \in_R Z^*_{p-1} \\
3: \text{B} & \longrightarrow & \text{A}: & v_B := r_A^{z_B} \bmod p \\
4: \text{A} & & : & K_A := v_B^{s_A} \bmod p \\
5: \text{B} & & : & K'_A := (\alpha^{m_A} Y^{-r_A})^{z_B} \bmod p
\end{array}
$$

Um zu zeigen, dass K_A und K'_A identisch sind, muss man zeigen dass $v_B^{s_A}$ und $(\alpha^{m_A} Y^{-r_A})^{z_B}$ in $GF(p)$ gleich sind. Durch Einsetzen und Umformen erhält man $v_B^{s_A} = (r_A^{z_B})^{s_A} = (r_A^{s_A})^{z_B}$. Es bleibt zu zeigen, dass $(r_A^{s_A})^{z_B}$ und $(\alpha^{m_A} Y^{-r_A})^{z_B}$ gleich sind. Weil als Exponent in beiden Ausdrücken z_B auftritt, kann dieser auch weggelassen werden, d.h. es muss gezeigt werden, dass $r_A^{s_A}$ und $\alpha^{m_A} Y^{-r_A}$ gleich sind. Aus der Konstruktion von r_A folgt $r_A^{s_A} = \alpha^{k_A s_A}$ (1). Auf der anderen Seite kann $\alpha^{m_A} Y^{-r_A}$ folgendermassen aufgelöst und umgeformt werden: $\alpha^{m_A} Y^{-r_A} = \alpha^{X r_A + s_A k_A} Y^{-r_A} = \alpha^{X r_A} \alpha^{s_A k_A} Y^{-r_A} = \alpha^{X r_A} \alpha^{s_A k_A} \alpha^{-X r_A} = \alpha^{s_A k_A}$ (2). Ein Vergleich von (1) und (2) zeigt, dass K_A und K'_A wirklich identisch sind.

A und B kennen nun also den Schlüssel K_A und können damit Daten chiffrieren. Die Authentifikation erfolgt indirekt aufgrund der Tatsache, dass sich A und B in der Folge "verstehen". Dabei ist aber A nur gegenüber B authentifiziert; zur gegenseitigen Authentifikation muss das Protokoll noch einmal mit vertauschten Rollen durchlaufen werden:

$$\begin{array}{rcl} 6: \text{B} & \longrightarrow & \text{A}: m_B, r_B \\ 7: \text{A} & & : z_A \in_R Z_{p-1}^* \\ 8: \text{A} & \longrightarrow & \text{B}: v_A := r_B^{z_A} \bmod p \\ 9: \text{B} & & : K_B := v_A^{s_B} \bmod p \\ 10: \text{A} & & : K'_B := (\alpha^{m_B} Y^{-r_B})^{z_A} \bmod p \end{array}$$

Wiederum kann man zeigen, dass K_B und K'_B identisch sind. A und B kennen beide Schlüssel und können damit entweder Daten direkt chiffrieren, oder sich vorgängig mit Hilfe einer Funktion f auf einen Schlüssel $K := f(K_A, K_B)$ einigen.

Archiviert die SKIA bei der Registrierung des Benutzers A in Phase zwei den Wert für s_A, dann wird sie später den Schlüssel K_A auch rekonstruieren können. Die SKIA muss deshalb als vertrauenswürdig angenommen werden und das uneingeschränkte Vertrauen aller Benutzer geniessen. Kann dies nicht vorausgesetzt werden, kann das KATHY-Basisprotokoll dahingehend erweitert werden, dass sich ein Benutzer seine Merkmalsliste von der SKIA "verdeckt" unterschreiben lassen kann. Hierzu wählt der Benutzer i ein zufälliges $a_i \in_R Z_{p-1}^*$ und berechnet die Primitivwurzel $\beta_i := \alpha^{a_i} \bmod p$, die er der SKIA mitteilt. Die SKIA verwendet nun anstelle von α den Wert β_i, berechnet $r_i := \beta_i^{k_i} \bmod p$ und löst die Kongruenz $Xr_i + b_i k_i \equiv m_i \bmod (p-1)$. Die Parameter m_i, r_i und b_i werden dem Benutzer ausgehändigt, wobei das Paar (r_i, b_i) nun eine modifizierte ElGamal-Signatur der Merkmalsliste m_i darstellt. Der Benutzer kann die Kongruenz $s_i \equiv b_i a_i^{-1} \bmod (p-1)$ lösen, ohne dass dies die SKIA ebenfalls tun könnte. Damit kann s_i von der SKIA nicht mehr ermittelt werden und der Benutzer kann mit einem Partner vertraulich und authentisch einen Schlüssel vereinbaren: Während des Protokolllaufes berechnet A den Schlüssel $K_A = v_B^{s_A} = r_A^{z_B s_A} = \alpha^{a_A k_A z_B b_A a_A^{-1}} = \alpha^{k_A z_B b_A} \bmod p$ und B $K'_A = (\alpha^{m_A} Y^{-r_A})^{z_B} = \alpha^{b_A k_A z_B} \bmod p = K_A$. Die SKIA kann K_A nicht rekonstruieren, weil sie weder a_A noch z_B kennt.

Neben den verdeckten Unterschriften gibt noch eine andere Variante des KATHY-Basisprotokolls. Bei dieser Variante löst die SKIA für Benutzer i die Kongruenz $Xs_i + r_i k_i \equiv m_i \bmod (p-1)$, womit auch $\alpha^{m_i} = Y^{s_i} r_i^{r_i}$ gilt. Das Paar (s_i, r_i) stellt eine modifizierte ElGamal-Signatur der Merkmalsliste m_i dar. Die Authentifikation eines Benutzers A gegenüber einem Benutzer B verläuft analog zum KATHY-Basisprotokoll, während sich die beiden Benutzer mit Hilfe des folgenden Protokolls auf einen gemeinsamen Sitzungsschlüssel $K_{AB} = K_{BA}$ einigen können:

$$
\begin{array}{lcl}
1 : \mathrm{A} & \longrightarrow & \mathrm{B} : m_A, r_A \\
2 : \mathrm{B} & \longrightarrow & \mathrm{A} : m_B, r_B \\
3 : \mathrm{A} & & : K_{AB} := (\alpha^{m_B} r_B^{-r_B})^{s_A} \bmod p \\
4 : \mathrm{B} & & : K_{BA} := (\alpha^{m_A} r_A^{-r_A})^{s_A} \bmod p
\end{array}
$$

Kann man sicherstellen, dass die Parameter Y von verschiedenen SKIAs authentisch sind, dann können sich Benutzer gegenseitig auch dann authentifzieren, wenn sie von unterschiedlichen SKIAs registriert wurden. Wurde Benutzer A z.B. von SKIA1 und B von SKIA2 registriert, und sind Y_1 und Y_2 die entsprechenden öffentlichen Parameter, dann ergeben sich die Schlüssel K_A und K_B des KATHY-Basisprotokolls aus $K_A := (\alpha^{m_A} Y_1^{-r_A})^{z_B} \bmod p$ und $K_B := (\alpha^{m_B} Y_2^{-r_B})^{z_A} \bmod p$. Die KATHY-Protokolle scheinen geeignet zu sein, auch hierarchische Strukturen zu unterstützen.

Für die Implementierung der KATHY-Protokolle und die Integration in SELANE wurden am E.I.S.S. eine Reihe von Toolboxen entwickelt [KH92]. Dabei wird als *Toolbox* eine Sammlung von Funktionen verstanden, die in sich abgeschlossene (kryptographische) Basisdienste zur Verfügung stellt.

5.2 ESS

Wie die KATHY-Protokolle von SELANE basiert auch das elektronische Unterschriftensystem EES auf dem digitalen Unterschiftensystem von ElGamal.

Ein Benutzer A ist mit Hilfe der von der SKIA ausgegebenen und beglaubigten Merkmalsliste (m_A, r_A, s_A) in der Lage, eine Nachricht $m \in GF(p)$ digital zu unterschreiben. Er wählt dazu ein zufälliges $k \in_R Z_{p-1}^*$, berechnet $t := r_A^k \bmod p$ und löst die Kongruenz $m \equiv s_A t + ku \bmod (p-1)$. Das Paar (t, u) stellt dann eine ElGamal-Signatur von m dar. Die Signatur ist genau dann authentisch, wenn die Gleichung $r_A^m = (\alpha^{m_A} Y^{-r_A})^t t^u$ erfüllt ist.

6 SESAME

Im Hinblick auf den Einsatz in offenen Systemen hat auch die *European Computer Manufacturer Association* (ECMA) eine Sicherheitsarchitektur entwickelt [ECM88, ECM89, Par89, Pre90]. Im Zentrum dieser Architektur steht das Konzept einer Sicherheitsdomäne (security domain), sowie die in Tabelle 17.1 zusammengestellten Sicherheitsdienste. Sicherheitsdienste werden in der Terminologie der ECMA als "Security Facilities" bezeichnet.

Die Arbeiten der ECMA sind im Rahmen des Projektes SESAME (A Secure European System for Applications in a Multi-vendor Environment) konkretisiert und im Hinblick auf den Einsatz asymmetrischer Kryptosysteme auch fortgesetzt worden.

Subject Sponsor
Authentication Facility
Association Management Facility
Security State Facility
Security Attribute Facility
Authorization Facility
Interdomain Facility
Security Audit Facility
Security Recovery Facility
Cryptographic Support Facility

Tabelle 1: ECMA Security Facilities

Beteiligt am SESAME-Projekt sind die Firmen Bull (Frankreich), ICL (Grossbritannien) und Siemens Nixdorf Informationssysteme (SNI) AG (Deutschland).

Von allen in diesem Bericht beschriebenen Projekten ist SESAME das einzige, das geheim ist, und über das kaum Informationen zu finden sind. Die folgenden Ausführungen beziehen sich denn auch auf einen Bericht, der bereits vor drei Jahren publiziert worden ist [Hof91].

Demnach stellt in SESAME ein sogenannter *User Sponsor* die Schnittstelle des Benutzers zum System dar. Der User Sponsor befindet sich in jedem System und ersetzt dort die Login-Komponente. Bei jedem Authentifikationsversuch wird der Benutzer mit seinem User Sponsor verbunden; dieser nimmt die Authentifikationsinformation entgegen, chiffriert sie und übergibt sie einem zentralen *Authentifikationsserver* (AS). Bei der Authentifikationsinformation wird es sich im einfachsten Fall um ein Passwort handeln; der Einsatz einer Chipkarte ist aber ebenfalls möglich.

Der AS überprüft die Authentifikationsinformation und erstellt im positiven Fall ein Zertifikat, das die Authentizität des Benutzers bescheinigt, und das als *Initial-PAC* (Privilege Attribute Certificate) bezeichnet wird. Erreicht den AS ein Authentifikationswunsch eines Benutzers, der in einem anderen AS registriert ist, so leitet er die Anfrage an den entsprechenden AS weiter. Die Antwort dieses AS wird dann transparent dem User Sponsor zurückgegeben. Die Initial-PACs werden von den User Sponsoren verwaltet.

Um Anwendungen starten zu können, benötigt ein Benutzer anwendungsspezifische PACs. Diese können unter Vorweisung des Initial-PACs vom User Sponsor bei einem *Zugriffsserver* angefordert werden. Der Zugriffsserver wird die gewünschten PACs ausstellen, wenn der Benutzer die erforderlichen Zugriffsberechtigungen auch besitzt. Prinzipell können Authentifikations- und Zugriffsserver auch in einer Systemkomponente implementiert sein.

Mit den vom Zugriffsserver erhaltenen PACs kann der Benutzer die benötigten Anwendungen aufrufen. Dabei werden die PACs sowohl in den Ausgangs- als auch in den Zielsystemen überprüft. Beim Start einer Anwendung kann der User Spon-

sor einen *Association Manager* aufrufen, der für einen sicheren Verbindungsaufbau zuständig ist; er besitzt eine Komponente im lokalen System und eine im Zielsystem.

Für die Protokollierung sicherheitsrelevanter Ereignisse ist ein *Audit-Server* vorgesehen. Ereignisse können vom User Sponsor, vom Authentifikations- oder Zugriffsserver, bzw. vom Association Manager gemeldet werden.

7 Schlussfolgerungen

In diesem Bericht sind verschiedene Authentifikations- und Schlüsselverteilsysteme vorgestellt und diskutiert worden. Sie unterscheiden sich in vielerlei Hinsicht:

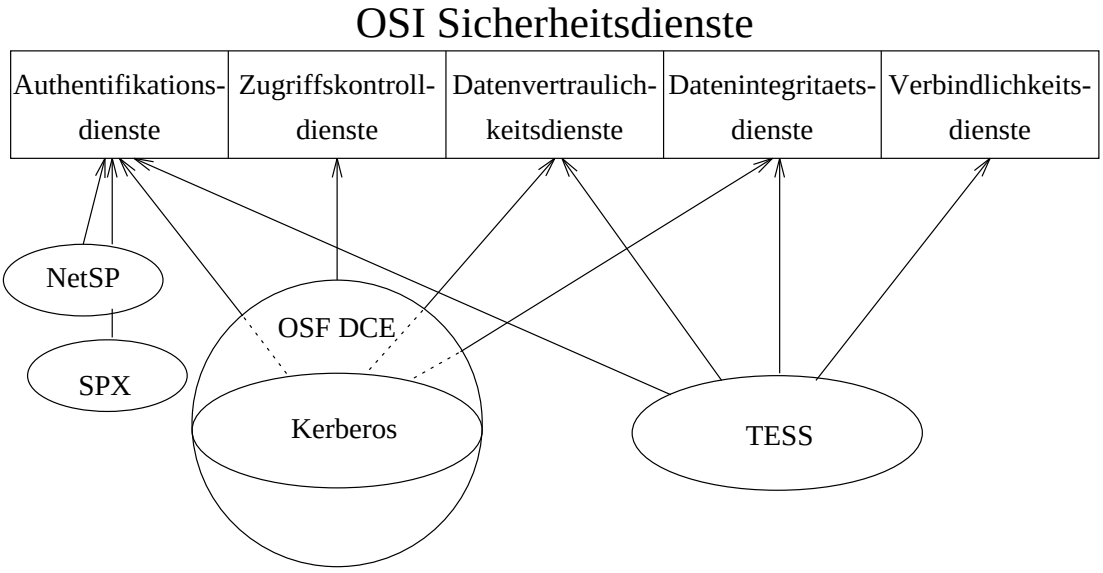

Abbildung 4: Angebotene OSI-Sicherheitsdienste

- Während sich Kerberos und NetSP auf den Einsatz symmetrischer Kryptosysteme beschränken, versuchen SPX, TESS und SESAME die Vorteile eines hybriden Ansatzes auszunutzen.
- Während Kerberos und SPX zur Verhinderung, bzw. Aufdeckung von Wiederholungsangriffen Zeitstempel als TVPs einsetzen, bauen TESS und NetSP auf Challenge-Response-Verfahren und den Einsatz frischer Zufallsmarken.
- Während NetSP und SPX ganz oder wenigstens teilweise der GSS-API folgen, bauen Kerberos, TESS und SESAME auf eigene Programmierschnittstellen.

Die verschiedenen Authentifikations- und Schlüsselverteilsysteme bieten auch unterschiedliche Sicherheitsdienste an. Abbildung 4 soll zeigen, welche Systeme welche

AT&T Global Information Systems
Bull Worldwide Information Systems
Digital Equipment Corporation (DEC)
Fujitsu Limited
Hewlett-Packard Company
Hitachi Limited
IBM Corporation
International Computers Limited (ICL)
NEC Corporation
Novell, Incorporated
Siemens Nixdorf Information Systems, Inc.
Silicon Graphics, Inc.
Sony Corporation
SunSoft Incorporated
Transarc Corporation

Tabelle 2: Mitglieder der OSF

OSI-Sicherheitsdienste anbieten. Dabei soll die Grösse der Ovale auch die strategische Bedeutung der Systeme darstellen.

Der Kerberos einschliessende Kreis ist mit OSF DCE betitelt, weil in der Version 1.0 der *Distributed Computing Environment* (DCE) die *Open Software Foundation* (OSF) als Authentfikationsdienst eben Kerberos Version 5 vorgesehen hat [Wel93]. In Tabelle 2 sind die wichtigsten Mitglieder der OSF zusammengestellt (Stand Mai 1994).

Ergänzt wird der Authentfikationsdienst in der DCE um Registratur-, Zugriffskontroll-, Datenvertraulichkeits- und -integritäts-, sowie um Schlüsselverwaltungsdienste. In Form einer *Application Environment Specification* (AES) hat die OSF DCE-spezifische Sicherheitsmodelle, -dienste und -protokolle spezifiziert [OSF94].

Für eine spätere Version der DCE werden zurzeit auch hybride Ansätze untersucht. Besonderes Augenmerk scheint dabei auf SESAME gerichtet zu sein. Mit ein Grund ist wohl auch die Tatsache, dass mit Bull, ICL und SNI alle am SESAME-Projekt beteiligten Firmen in der OSF Einsitz haben.

Danksagung

Dieser Bericht wurde ermöglicht dank freundlicher Unterstützung des Bundesamtes für Informatik (BFI). Namentlich werwähnen möchten wir P. Trachsel und M. Frauenknecht von der Sektion Informatiksicherheit.

Literaturverzeichnis

[Bet91] T. Beth. TESS — The Exponential Security System. Report 91/13, Europäisches Institut für Systemsicherheit (E.I.S.S), Universität Karlsruhe, 1991.

[Bet94] Th. Beth. Security Systems Based on Exponentiation Primitives, TESS. In *Proceedings of the IFIP TC11 Tenth International Conference on Information Security, IFIP SEC'94*, 1994. Curacao, May 23 – 27, 1994.

[BGH+92a] R. Bird, I. Gopal, A. Herzberg, P.A. Janson, S. Kutten, R. Molva und M. Yung. Systematic Design of Two-Party Authentication Protocols. In J. Feigenbaum, Hrsg., *Proceedings of CRYPTO '91*. Springer-Verlag, 1992.

[BGH+92b] R. Bird, I. Gopal, A. Herzberg, P.A. Janson, S. Kutten, R. Molva und M. Yung. A Modular Family of Secure Protocols for Authentication and Key Distribution. Technical report, IBM Zürich Research Laboratory, Rüschlikon, Switzerland, November 1992.

[BGH+93] R. Bird, I. Gopal, A. Herzberg, P.A. Janson, S. Kutten, R. Molva und M. Yung. Systematic Design of a Family of Attack-Resistent Authentication Protocols. *IEEE Journal on Selected Areas in Communications*, 11(5):679 – 693, June 1993.

[BK90] F. Bauspiess und H.J. Knoblauch. How to keep Authenticity Alive in a Computer Network. In J.J. Quisquater und J. Vadewalle, Hrsg., *Proceedings of EUROCRYPT '89*, Seiten 38 – 46. Springer-Verlag, 1990.

[BM90] S.M. Bellovin und M. Merritt. Limitations of the Kerberos Authentication System. *ACM Computer Communication Review*, 20(5):119 – 132, 1990.

[Cha91] G.A. Champine. *MIT Project Athena — A Model for Distributed Computing*. Digital Press, 1991.

[DS81] D. Denning und G. Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):533 – 536, 1981.

[ECM88] ECMA. Security in Open Systems — A Security Framework. ECMA TR/46, European Computer Manufacturer Association, Geneva, Switzerland, July 1988.

[ECM89] ECMA. Security in Open Systems — Data Elements and Service Definitions. ECMA TR/138, European Computer Manufacturer Association, Geneva, Switzerland, December 1989.

[GGKL89] M. Gasser, A. Goldstein, C. Kaufman und B. Lampson. The Digital Distributed System Security Architecture. In *Proceedings of the 12th National Computer Security Conference*, Seiten 305 – 319, 1989.

[GKL+92] M. Gasser, C. Kaufman, J Linn, Y. Le Roux und J. Tardo. DASS: Distributed Authentication Security Service. In R. Aiken, Hrsg., *Education and Society*, Seiten 447 – 456. Elsevier Science Publishers B.V., North-Holland, 1992. Information Processing 92, Volume II.

[GLNS93] L. Gong, M.A. Lomas, R.M. Needham und J.H. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648 – 656, June 1993.

[HK91a] P. Horster und H.J. Knobloch. Protocols for Secure Networks. In D.W. Davies, Hrsg., *Proceedings of EUROCRYPT '91*, Seiten 399 – 408. Springer-Verlag, 1991.

[HK91b] P. Horster und H.J. Knobloch. Protokolle zum Austausch authentischer Schlüssel. In A. Pfitzmann und E. Raubold, Hrsg., *VIS '91 — Verlässliche Informationssysteme*, Seiten 321 – 328. Springer-Verlag, 1991.

[Hof91] G. et. al. Hoffmann. Zentrale Authentifizierung und Zugriffskontrolle in verteilten Systemen. *Zeitschrift für Kommunikations- und EDV-Sicherheit (KES)*, 7(5):302 – 310, 1991.

[JT93] P. Janson und G. Tsudik. Secure and Minimial Protocols for Authenticated Key Distribution. Technical report, IBM Zürich Research Laboratory, Rüschlikon, Switzerland, November 1993.

[Kau93] C. Kaufman. DASS — Distributed Authentication Security Service. Request for Comments 1507, September 1993.

[KH92] H.J. Knoblauch und P. Horster. Eine Krypto-Toolbox für Smartcards. *Datenschutz und Datensicherung*, 16(7):353 – 361, Juli 1992.

[KN93] J. Kohl und C. Neumann. The Kerberos Network Authentication Service (V5). Request for Comments 1510, September 1993.

[Lin93] J. Linn. Generic Security Service Application Program Interface. Request for Comments 1508, September 1993.

[MLG+89] T. Mark, A. Lomas, L. Gong, J.H. Saltzer und R.M. Needham. Reducing Risks from Poorly Chosen Keys. *ACM Operating Systems Review*, 23(5):14 – 18, 1989. Special Issue — Proceedings of the Twelfth ACM Symposium on Operating Systems Principles.

[MST94] R. Molva, D. Samfat und G. Tsudik. Authentication of Mobile Users. *IEEE Network*, 8(2):26 – 34, 1994.

[MTVHZ92] R. Molva, G. Tsudik, E. Van Herreweghen und S. Zatti. KryptoKnight Authentication and Key Distribution System. In Y. Deswarte, G. Eizenberg und J.J. Quisquater, Hrsg., *Computer Security — ESORICS '92* , Seiten 155 – 174. Springer-Verlag, 1992. Second European Symposium on Research in Computer Security, Toulouse, France, November 23–25, 1992.

[NS78] R.M. Needham und M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993 – 999, December 1978.

[NS87] R.M. Needham und M.D. Schroeder. Authentication Revisted. *ACM Operating Systems Review*, 21(1):7, 1987.

[OSF94] OSF. AES/Distributed Computing — Security. RevisionA — Preliminary Version, 11 Cambridge Center, Cambridge, MA 02142, March 1994.

[Par89] T. Parker. ECMA Security Standards. In *Proceedings of the 5th Annual Computer Security Applications Conference*, December 1989.

[Pre90] K. Presttun. European Activities Part II — ECMA Open Systems Security Framework, Alive and Beyond. In *Proceedings of the 6th Annual Computer Security Applications Conference*, December 1990.

[SNS88] J. Steiner, C. Neuman und J. Schiller. Kerberos: An Authentication Service for Open Network Systems. Technical report, Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, Massachusetts, 1988.

[TA91] J.J. Tardo und K. Alagappan. SPX: Global Authentification Using Public Key Certificates. In *Proceedings of the IEEE Symposium on Security and Privacy*, Seiten 232 – 244, Los Alamitos, California, 1991. IEEE Computer Society Press.

[TVH93a] G. Tsudik und E. Van Herreweghen. On Simple and Secure Key Distribution. Technical report, IBM Zürich Research Laboratory, Rüschlikon, Switzerland, August 1993.

[TVH93b] G. Tsudik und E. Van Herreweghen. Some Remarks on Protecting Weak Keys and Poorly-Chosen Secrets from Guessing Attacks. Technical report, IBM Zürich Research Laboratory, Rüschlikon, Switzerland, 1993.

[Wel93] H. Welter. *Heterogene Netze: Einführung in Standardarchitekturen, Protokolle, Verwaltung und Sicherheit.* Addison-Wesley, 1993.

[Wra93] J. Wray. Generic Security Service API : C-bindings. Request for Comments 1509, September 1993.