

Internet Security 1

Prof. Dr. Roland Schmitz
WS 2005 - 2006



nl013 - isec1@psbase.com

Dieses Dokument wurde anhand der Folien und Übungen der Vorlesung
Internet Security 1 (Prof. Dr. Roland Schmitz) zusammengestellt.

Keine Gewährleistung auf Richtigkeit ist gegeben.

Inhaltsverzeichnis

1 Grundlagen	11
1.1 Aspekte der IT - Security	11
1.2 Allgemeine Definitionen	11
1.3 Sicherheitsdienste	12
1.4 Sicherheitsmechanismen	13
2 Mobiler Code	15
2.1 Anwendungen	15
2.2 Bedrohung für mobilen Code	15
2.3 Bedrohung durch mobilen Code	16
2.4 Schutz des mobilen Codes	16
2.5 Schutz des Gastrechners	16
2.6 Java Security	16
2.7 Applets	17
2.8 ActiveX Controls	18
3 Bösartige Software und Spam	19
3.1 Übersicht	19
3.2 Viren	19
3.3 Trojanische Pferde	22
3.4 Würmer	22
3.5 Anderes Ungeziefer	23
3.6 Useful Links	24
3.7 Spam	24
3.8 Phishing	25
4 Kommunikation im Internet	27
4.1 Internet-Historie	27
4.2 OSI Schichtenmodell	27
4.3 Internet Protokoll	28
4.4 Transmission Control Protocol (TCP)	29
4.5 Sicherheitsprobleme bei TCP/IP	30
4.5.1 Probleme mit IP	30
4.5.2 Probleme mit TCP	30
4.6 User Datagram Protocol (UDP)	33
4.6.1 UDP-Flood (Echo)	34
4.7 Internet Control Message Protocol (ICMP)	34

5	Sicherheitsmechanismen	35
5.1	Firewalls	35
5.1.1	Packet Filter	35
5.1.2	Application-Level Gateway	38
5.1.3	Circuit-Level Gateway	39
5.1.4	Firewall Konfigurationen	39
5.1.5	Risiken und Grenzen von Firewalls	41
5.1.6	Rechtliche Fragen beim Betrieb von Firewalls	41
5.1.7	Personal Firewalls	41
5.1.8	Organisatorische Fragen	42
5.1.9	Zukünftige Trends	42
6	Symmetrische Kryptographie	43
6.1	Allgemeines	43
6.1.1	Begriffe	43
6.1.2	Symmetrische Kryptosysteme	43
6.1.3	Beispiel: Caesar-chiffre	43
6.1.4	Kerckhoffsches Prinzip	44
6.1.5	Starke Algorithmen	44
6.1.6	Angriffsarten	44
6.1.7	Beispiel Angriffe	44
6.2	Generelle Klassen symmetrischer Chiffren	45
6.3	Historische Chiffren	46
6.3.1	Skytale	46
6.3.2	Vigenère-Chiffre	46
6.3.3	Die ENIGMA	47
6.4	Exkurs: Wahrscheinlichkeitstheorie	47
6.4.1	Perfekte Sicherheit	49
6.4.2	Die Wahrscheinlichkeit eine Nachricht zu entschlüsseln	50
6.4.3	Perfektion von symmetrisches Kryptosystem	51
6.5	Stromchiffren	51
6.5.1	RC4	52
6.5.2	A5	53
6.5.3	Einsatz von Stromchiffren	54
6.6	Blockchiffren	54
6.6.1	Designprinzipien für Blockchiffren (SHANNON)	54
6.6.2	Feistel Chiffren	55
6.6.3	Data Encryption Standard: DES	56
6.6.4	More Feistel-Ciphers	57
6.6.5	Non-Feistel Blockciphers	58
6.6.6	Advanced Encryption Standard: AES	58
6.6.7	Betriebsmodi (Modes of Operation) für Blockchiffren	60
6.6.8	PKCS5 Padding	61
6.7	Java Cryptographic Architecture (JCA)	61
6.8	Message Authentication Codes (MACS)	62
6.8.1	Allgemeiner Ablauf	63
6.8.2	Why not encrypt?	63

6.8.3	1. Methode zur MAC-Bildung	64
6.8.4	2. Methode zur MAC-Bildung	65
6.8.5	Common Hash-Functions	65
6.9	Schlüssellängen bei symmetrischen Kryptoverfahren	65
7	Asymmetrische Kryptographie	67
7.1	Allgemeines	67
7.2	Das RSA-Verfahren	67
7.2.1	Schlüsselerzeugung	69
7.2.2	Exkurs: Primzahlen und modulare Arithmetik	69
7.2.3	Verschlüsselung	71
7.2.4	Entschlüsselung	72
7.2.5	Sicherheit des RSA-Verfahrens	72
7.2.6	Beispiel	73
7.3	Diskrete Logarithmen	73
7.3.1	Diffie-Hellman Schlüsseltausch-Protokoll	74
7.3.2	Man-in-the Middle Angriff auf Diffie-Hellman	75
7.3.3	Sicherheit des Diffie-Hellman-Verfahrens	75
7.3.4	ElGamal-Verfahren	76
7.3.5	Berechnung Diskreter Logarithmen	76
7.4	Digitale Signaturen und Zertifikate	76
7.4.1	Allgemeines Szenario	77
7.4.2	RSA-Signaturen	77
7.4.3	Signaturen auf ElGamal Basis	78
7.4.4	Zertifikate	78
7.4.5	Aufgaben von Certificate Authorities (CAs)	79
7.4.6	Certificate Policy und Certification Practice Statement	79
7.5	Authentifikationsprotokolle	79
7.5.1	Password-Based Authentication	80
7.5.2	Challenge-Response Protocols	80
7.5.3	Kerberos	83
8	Sicherheit auf der Anwendungsschicht	85
8.1	S/MIME	85
8.1.1	Simple Mail Transfer Protocol (SMTP)	85
8.1.2	Multipurpose Internet Mail Extension (MIME)	86
8.1.3	S/MIME Client Features	87
8.1.4	Algorithms Used in S/MIME Version 3	87
8.2	Pretty Good Privacy: PGP	88
8.2.1	PGP History	88
8.2.2	Why Is PGP Popular?	88
8.2.3	Operational Description	89
8.2.4	Algorithms used by PGP	89
8.2.5	Format of PGP Message	89
8.2.6	PGP Message Generation	89
8.2.7	PGP Message Reception	89
8.2.8	PGP "Web of Trust"	89
8.3	Secure Electronic Transactions: SET	92

8.3.1	Sicherheitsdienste von SET	92
8.3.2	SET Beteiligte	92
8.3.3	SET Vorbereitungsphase	93
8.3.4	SET Transaction	93
8.3.5	SET Dual Signature	94
8.3.6	Building the Purchase Request	95
8.3.7	Verifying a Dual Signature: Merchant's Site	96
8.3.8	SET Nachteile	96
9	Sicherheit auf der Transportschicht	97
9.1	SSL und TLS	97
9.1.1	SSL Protokollstapel	98
9.1.2	SSL Record Protocol	98
9.1.3	SSL Record Format	98
9.1.4	SSL Handshake	98
9.1.5	Schwächen von SSLv2.0	100
9.1.6	Vor- und Nachteile von SSLv3.0	101
9.2	SSH (Secure Shell)	101
9.2.1	SSH-2 Protokollstapel	101
9.2.2	SSH Transport Layer Protocol	102
9.2.3	SSH Protokollablauf	102
9.2.4	SSH Parameter für Diffie-Hellmann Schlüsseltausch	102
9.2.5	SSH Transport Layer Datenformat	103
9.2.6	SSH User Authentication Protocol	103
9.2.7	SSH Connection Protocol	103
9.3	Web Ressourcen	104
10	Sicherheit auf der Internetschicht	105
10.1	Übersicht	105
10.1.1	IPSec Services	105
10.1.2	Security Associations (SA)	106
10.1.3	Datenfelder einer Security Association	106
10.1.4	Datenbanken in IPSec	106
10.1.5	Beispiel: Sende IP-Paket von A nach B im ESP-Transport Mode	106
10.1.6	Unterschied zu SSH/SSL	106
10.2	Encapsulating Security Payload (ESP)	107
10.2.1	ESP Format	107
10.2.2	ESP Transport Mode	107
10.2.3	ESP Tunnel Mode	108
10.2.4	Transport Mode vs Tunnel Mode	108
10.2.5	ESP and IP	108
10.3	Authentication Header (AH)	109
10.3.1	AH and IP	109
10.3.2	Encryption and Authentication Algorithms	110
10.3.3	Anti-Replay Mechanism	111
10.4	Combinations of Security Associations	111
10.5	IPSec Key Management	114
10.5.1	Oakley	114

10.5.2	STS Protokollablauf	114
10.5.3	Von STS zu OAKLEY	115
10.5.4	ISAKMP	115
10.6	IPSec und IPv6	116
10.7	IPSec und NAT	116
10.7.1	NAT Formen	117
10.7.2	Konflikte mit IPSec	117
10.7.3	IPSec Overview	118
10.7.4	IPSec RFCs	118
11	Übungen	121
11.1	Übungsblatt 1	121
11.1.1	Aufgabe 1	121
11.1.2	Aufgabe 2	121
11.2	Übungsblatt 2	122
11.2.1	Aufgabe 1	122
11.2.2	Aufgabe 2	122
11.2.3	Aufgabe 3	123
11.2.4	Aufgabe 4	124
11.3	Übungsblatt 3	124
11.3.1	Aufgabe 1	124
11.3.2	Aufgabe 2	125
11.3.3	Aufgabe 3	125
11.3.4	Aufgabe 4	125
11.3.5	Aufgabe 5	126
11.3.6	Aufgabe 6	126
11.4	Übungsblatt 4	127
11.4.1	Aufgabe 1	127
11.4.2	Aufgabe 2	128
11.4.3	Aufgabe 3	128
11.4.4	Aufgabe 4	128
11.4.5	Aufgabe 5	129
11.4.6	Aufgabe 6	129
11.5	Übungsblatt 5	130
11.5.1	Aufgabe 1	130
11.5.2	Aufgabe 2	130
11.5.3	Aufgabe 3	131
11.5.4	Aufgabe 4	131
11.5.5	Aufgabe 5	132
11.6	Übungsblatt 6	133
11.6.1	Aufgabe 1	133
11.6.2	Aufgabe 2	134
11.6.3	Aufgabe 3	135
11.7	Übungsblatt 7	136
11.7.1	Aufgabe 1	136
11.7.2	Aufgabe 2	137
11.7.3	Aufgabe 3	137

11.7.4 Aufgabe 4	138
11.7.5 Aufgabe 5	139
11.8 Übungsblatt 8	140
11.8.1 Aufgabe 1	140
11.8.2 Aufgabe 2	140
11.8.3 Aufgabe 3	144
11.8.4 Aufgabe 4	144
11.8.5 Aufgabe 5	145
11.9 Übungsblatt 9	145
11.9.1 Aufgabe 1	145
11.9.2 Aufgabe 2	146
11.9.3 Aufgabe 3	147
11.9.4 Aufgabe 4	147
11.10 Übungsblatt 10	148
11.10.1 Aufgabe 1	148
11.10.2 Aufgabe 2	149
11.10.3 Aufgabe 3	149
12 Testfragen	151
13 Fragen zu diesem Dokument	153
13.1 Fragen zu Kapitel 1 (Seite 11)	153
13.2 Fragen zu Kapitel 2 (Seite 15)	153
13.3 Fragen zu Kapitel 3 (Seite 19)	153
13.4 Fragen zu Kapitel 4 (Seite 27)	154
13.5 Fragen zu Kapitel 5 (Seite 35)	154
13.6 Fragen zu Kapitel 6 (Seite 43)	155
Abkürzungsverzeichnis	163
Index	167

Einstimmung

Definition: Safety vs Security

Safety Ausgelegt für zufällig auftretende Fehler/Ereignisse

Security Schutz gegen gewolltem Eingreifen von einem intellektuellen Angreifer

Internet verschlimmert häufig Bedrohungen

- Trennung von Information und Informationsträger (z.B. Beim Klauen von vertraulichen Daten wird die Information geklaut, nicht aber der Informationsträger.
- Räumliche Distanzen spielen keine Rolle
- Kein persönlicher Kontakt von Geschäftspartnern möglich
- Automatisierbarkeit und Wiederholbarkeit von Angriffen
- Schnelle Verbreitung gelungener Hacks
- Beim Design des Internet spielte Sicherheit keine Rolle

Internetspezifische Bedrohungen

- Unerlaubtes Mithören von wichtigen Informationen (Passwörter, Kreditkarten-Nummern, etc.)
- Verfälschen von Informationen
- Absenden von Nachrichten unter falschem Namen
- Denial-of-Service Angriffe
- Session Hijacking
- Viren, Würmer, Trojanische Pferde: Angriffe auf den eigenen Rechner
- Spam: Unerwünschte e-mails
- Phishing
- Hinterlassen von Datenspuren

Kapitel 1

Grundlagen

1.1 Aspekte der IT - Security

- *Information Security*
Schutz vor unbefugtem Zugang zu Informationen jeglicher Art
- *Computer Security*
Schutz digitaler Informationen auf dem Computer
- *Network (Access) Security*
Schutz von Informationen, die zwischen Computern über ein Netz ausgetauscht werden; Schutz vor unbefugtem Zugang in ein Netz
- *Internet Security*
Schutz von Informationen, die über das Internet ausgetauscht werden; Schutz vor Angriffen aus dem Internet

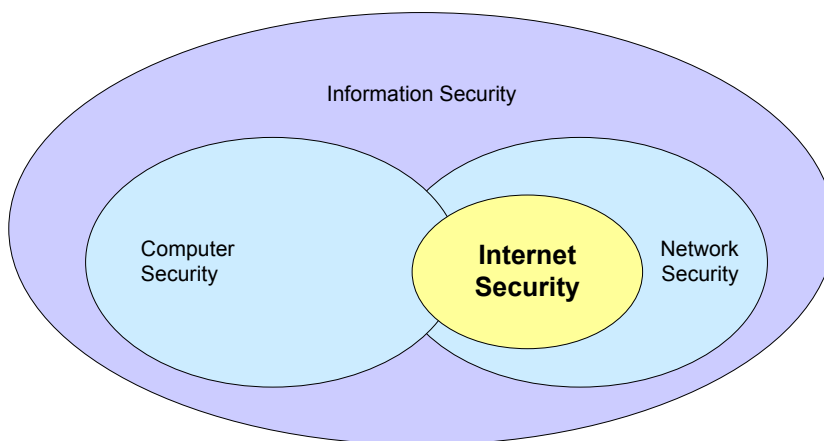


Abbildung 1.1: Internet-Security liegt im Zentrum der Begriffe

1.2 Allgemeine Definitionen

- Eine *Security Policy* teilt die Zustände eines IT - Systems in erlaubte und unerlaubte Zustände ein.
- Ein *sicheres System* nimmt nur erlaubte Zustände an.

Security Policy

Sichere Systeme

- Eine Verletzung der Security Policy (*Security Breach*) liegt vor, wenn das System einen unerlaubten Zustand annimmt.

1.3 Sicherheitsdienste

Sicherheitsdienste sind Eigenschaften IT-basierter Systeme, die die Sicherheit der verarbeiteten Informationen erhöhen und Bedrohungen von außen abwehren.

Spezielle Sicherheitsdienste

Vertraulichkeit (Confidentiality) Schutz vor unberechtigtem Mitlesen von Nachrichten und anderen nicht-öffentlichen Informationen

Integritätsschutz (Integrity Protection) Ermöglicht das Erkennen unberechtigter Modifikationen von Nachrichten

Authentifikation (Authentication) Verifikation der Identität von Personen (Identifikation) oder des Ursprungs von Nachrichten (Nachrichtenauthentifikation)

Authorisierung (Authorization) Zuweisung geeigneter Rechte an authentifizierte Personen/-Maschinen/Anwendungen

Nichtabstreitbarkeit (Nonrepudiation) Dieser Dienst verhindert, dass vorhergehende Verpflichtungen oder Aktionen von den Akteuren im Nachhinein verneint werden können.

Verfügbarkeit (Availability) stellt die Verfügbarkeit und das korrekte Funktionieren von Diensten oder Rechnern sicher.

Zugangskontrolle (Access Control) schränkt den Zugang zu Ressourcen (Dienste, Rechner, Informationen) auf einen autorisierten Personenkreis ein.

Sicherheitsdienste - Vergleich mit dem Alltag

Sicherheitsdienst	Internet	Alltag
<i>Zugriffsschutz</i>	Eigener Rechner	Haus
<i>Zugangskontrolle</i>	Password	Tür / Schloss
<i>Vertraulichkeit</i>	Verschlüsselte Email	Brief in Umschlag
<i>Integritätsschutz</i>	Signierte Email	Siegel
<i>Authentifikation</i>	Authentifikationsprotokolle	Ausweis
<i>Identifikation</i>	Digitales Zertifikat	Personalausweis
<i>Nicht fälschbar, Anonym</i>	Elektronisches Geld	Bargeld
<i>Verbindlichkeit Nichtabstreitbarkeit</i>	Digitale Signatur	Verträge / Unterschrift
<i>Verfügbarkeit</i>	Internet-Dienst	ÖPNV

Tabelle 1.1: Sicherheitsdienste - Vergleich mit dem Alltag

1.4 Sicherheitsmechanismen

Sicherheitsmechanismen sind Algorithmen, Protokolle oder Geräte, die einen oder mehrere Sicherheitsdienste realisieren.

Sicherheitsmechanismen

Z.B. realisiert ein (guter) Verschlüsselungsalgorithmus den Dienst "Vertraulichkeit", eine digitale Signatur den Dienst "Authentifikation".

Beispiel

- Security Policy: Nur Alice darf die Files lesen, die ihr gehören
- Security Breach: Bob loggt sich als Alice ein und liest ihre Files
- Sicherheitsdienst: Authentifikation der User
- Sicherheitsmechanismus: Eingabe von Username und Passwort

Kapitel 2

Mobiler Code

Mobiler Code ist Code, der auf einem entfernten Rechner generiert wurde und auf einem Gastrechner (Host) ausgeführt wird. (Z.B. Java-Applets, ActiveX Controls)

Mobiler Code

2.1 Anwendungen

- *Code-on-Demand (Cod)*:
 - Client fordert den Code an (z.B. Java Applets)
- *Remote Evaluation (Rev)*: server "pusht" den Code zum client
 - Ausführung wird an Zielrechner (mehr Rechenkapazität) delegiert (z.B. SETI@home)
- *Mobile Agenten*
 - Sucht sich Informationen im Internet zusammen
- *Dialerprogramme (ActiveX-Controls)*
 - Stellen neue Verbindung ins Internet über 0190-Zugangsnummer her
 - Kann sinnvolles Abrechnungstool für kostenpflichtige Websites sein.
 - Häufig zu Betrugszwecken missbraucht!

2.2 Bedrohung für mobilen Code

- Passive Angriffe
 - Unberechtigte Ausführung des Codes
 - Stehlen sensibler Daten
 - Erstellen von Bewegungsprofilen
- Aktive Angriffe
 - Unberechtigte Modifikation des Codes
 - Vorzeitiges Beenden (DoS)

2.3 Bedrohung durch mobilen Code

- Passive Angriffe
 - Lesender Zugriff auf Daten des Gastrechners
- Aktive Angriffe
 - Schreibender Zugriff auf Gastrechner
 - Monopolisieren der CPU (DoS)
 - Aufbau unzulässiger Verbindungen

2.4 Schutz des mobilen Codes

- Verschlüsseln
 - Schutz auf dem Transportweg
- Signieren
 - Schutz vor Modifizierung auf dem Transportweg (Code bleibt gleich)

2.5 Schutz des Gastrechners

- Ausschliessliche Nutzung von signiertem Code
 - Code nicht verändert
 - Problem: welche Signatur vertrauenswürdig
- Sandboxing
 - Satz von Regeln für den Zugriff ausführbaren Codes auf lokale Ressourcen

2.6 Java Security

- Sprachsicherheit
 - Keine Pointer-Arithmetik
 - Keine direkten Zugriffe auf Hauptspeicher
- Byte-Code Verification vor dem Laden von Applets
- Java Sandbox
 - Sicherheitsmodell für .class Files
 - JDK 1.0: "All-or-Nothing"
 - * Standard-Sandbox für alle Applets aus dem Internet

- * Globaler Zugriff für lokale class Files
- JDK 1.1: "Signed Applets"
 - * Signierte (vertrauenswürdige) Applets erhalten globale Rechte
- JDK 1.2: "Individual Access"
 - * Individuelle Security Policy für jede Art von Code möglich
 - * Durchsetzung der Policy durch Security Manager bzw. Access Controller

2.7 Applets

Sicheres Ausführen eines "gefährlichen" Applets

1. Erzeugen eines Schlüsselpaars mit dem keytool
2. Signieren des jar-Files
3. Erzeugen eines Zertifikats für den öffentlichen Schlüssel
4. Exportieren des Zertifikats auf das Zielsystem
5. Anpassen der Security Policy auf dem Zielsystem

Applet Beispiel

Ein Applet `DangerousApplet.jar` soll beim Ausführen eine log-Datei auf der Client-Festplatte anlegen und dort Informationen aus den System Properties reinschreiben.

Wird `DangerousApplet.jar` auf dem Client Browser ausgeführt wirft das Applet eine `AccessControlException`, da es keine Berechtigung für Schreibzugriffe hat und die System Properties nicht auslesen darf. Dies wird durch die Standard Policy gewährleistet, die dann benutzt wird, wenn das Applet nicht signiert ist.

Darum: Signieren des Applets:

Signieren des Applets mit keytool und jarsigner

- Erstellen des keystores:

```
1 keytool -genkey -keystore newKeyStore -alias roland
```

`newKeyStore` ist der Name des keystore und `roland` das alias, über den man das Schlüsselpaar (public key und private key) ansprechen kann.

Beim Ausführen der `keytool`-Command kann man anschliessend noch ein Password für das keystore und sonstige Angaben zur Signatur machen (Name, Organisation, Country usw.).

- Ist der keystore erstellt, kann das Applet nun signiert werden:

```
1 jarsigner -keystore newKeyStore -signedjar SignedApplet.jar DangerousApplet.jar
```

Die Command `jarsigner` hat eine signierte Kopie `SignedApplet.jar` erstellt.

- Beim Ausführen des neuen Applet `SignedApplet.jar`, kann man nun entscheiden ob man dem Zertifikat vertraut.
 - Wenn ja, kann das Applet problemlos auf die Festplatte schreiben und System Properties auslesen (vorausgesetzt der User kann dies auch).
 - Wenn nicht, wird die Standard Policy benutzt.

Individuelle Policy mit dem `policytool`

Das `policytool` erlaubt dem User dem Applet nur bestimmte Rechte zu geben.

Geben wir dem Applet z.B. nur `Filepermission`, wird das Applet die `log-Datei` erstellen können, aber nicht die `System Properties` auslesen können.

2.8 ActiveX Controls

ActiveX Control = ausführbare Datei

Kapitel 3

Bösartige Software und Spam

Angriff oder Bug?

Die FORTRAN-Zeile:

Listing 3.1: Die Fortran-Zeile

```
1 DO 20 I = 1.100
```

(statt DO 20 I = 1,100) führte zum Verlust einer Viking Raumsonde.

3.1 Übersicht

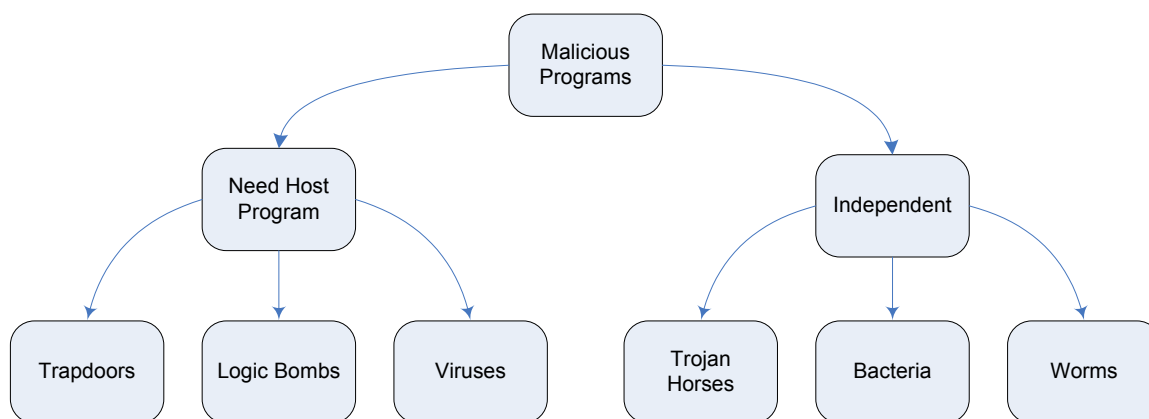


Abbildung 3.1: Übersicht bössartiger Software

3.2 Viren

Ein *Computervirus* ist eine Befehlsfolge, die ein Wirtsprogramm zur Ausführung benötigt. Die Ausführung eines Virus bewirkt, dass eine (evtl. modifizierte) Kopie (Reproduktion) in ein Programm, welches diese Befehlsfolge noch nicht enthält, geschrieben wird. Zusätzlich enthalten Viren zumeist einen Schadensteil, der durch einen Auslöser aktiviert werden kann.

Computerviren

Typen von Viren

- Bootviren
 - Befallen Ladeprogramm im Bootsektor bzw. Master Boot Record (MBR) von formatierten Datenträgern
 - Werden vor Start des Betriebssystems ausgeführt
- Dateiviren
 - Befallen ausführbare Dateien (*.exe, *.bat, *.vbs usw.)
- Makroviren
 - Befallen Dokumente mit Makroeigenschaften¹ (MS-Office oder StarOffice)

Struktur eines Dateivirus

Listing 3.2: Dateivirus Pseudocode

```
1 program V :=
2
3 {goto main;
4   1234567;
5
6   subroutine infect-executable :=
7     {loop:
8       file := get-random-executable-file;
9       if (first-line-of-file = 1234567)
10        then goto loop
11        else prepend V to file;}
12
13   subroutine do-damage :=
14     {whatever damage is to be done}
15
16   subroutine trigger-pulled :=
17     {return true if some condition holds}
18
19 main: main-program :=
20   {infect-executable;
21     if trigger-pulled then do-damage;
22     goto next;}
23
24 next:
25 }
```

Zeile 6: Reproduktionsteil , Zeile 13: Schadensteil , Zeile 16: Auslöser

¹Makros: ausführbare Programme

Lebensphasen

- Schlafphase
 - Warten auf ein bestimmtes Ereignis (z.B. Freitag, der 13.)
- Verbreitungsphase
 - Suche nach noch nicht infizierten Programmen
 - Infektion mit einer Kopie des Virencodes
- Schadphase
 - Ausführen der Schadfunktion

Infektionswege

- Disketten
 - 1996: 74% aller Infektionen
 - 2000: 7% aller Infektionen
- Download
 - 1996: 10% aller Infektionen
 - 2000: 5% aller Infektionen
- E-mail
 - 1996: 9% aller Infektionen
 - 2000: 87% aller Infektionen

Virens Scanner

- *First Generation*
 - Suche nach "Signaturen" (bekannten Bitmustern)
 - Nur bereits bekannte Viren werden entdeckt
 - Entdecken polymorpher Viren schwierig
- *Second Generation*
 - Heuristiken: Suche nach virentypischen Befehlsfolgen (z.B. Rücksprünge in Dateimitte, Schreibzugriff auf *.exe Dateien)
 - Integritätscheck von *.exe Dateien
- *Third Generation (Blocker)*
 - Überprüfe Rechner im Betrieb auf virentypische Aktionen und blockiere diese

- *Fourth Generation*
 - Kombination der ersten drei Generationen
 - Möglichkeit der Einschränkung der Rechte einzelner Programme (Sandboxing)

"There are no safe Virus tests"

Satz (Dowling, 1989)

"Es ist unmöglich, ein sicheres Programm (d.h. ein Programm, welches das Betriebssystem nicht verändert) zu schreiben, das von jedem beliebigem Programm entscheiden kann, ob es sicher ist oder nicht."

(Deutung: Der Virenschanner könnte selbst mit einem Virus infiziert sein)

Einfache Virenschutzmaßnahmen nach BSI

- Auf jedem Computer muß ein speicherresidentes Viren-Schutzprogramm laufen.
- Zum Schutz vor neuen Boot-Viren ist die Boot-Reihenfolge auf C::A: einzustellen.
- Zum Schutz vor neuen Makro-Viren bei den Programmen WINWORD, EXCEL und POWERPOINT den Makro-Viren-Schutz aktivieren.
- Bei Internet-Browsern und mail-clients Script-Sprachen (Java, Java-Script, ActiveX) ausschalten.
- Beim Versand oder der anderweitigen Bereitstellung von WORD-Dokumenten Rich-Text-File-Format (Extend .RTF) verwenden. RTF enthält keine Makros.
- Versand / Empfang von ausführbaren Programmen (Extend .COM, .EXE, .BAT) oder anderer Dateien, die Programmcode enthalten können (Extend .DO*; XL*, PPT, VBS...) vorher telefonisch abstimmen.
- Anzeige vollständiger Dateinamen einschalten

3.3 Trojanische Pferde

Ein *Trojanisches Pferd* ist ein eigenständiges Programm, dessen implementierte Ist-Funktionalität nicht mit der angegebenen Soll-Funktionalität übereinstimmt, da es über die Soll-Funktionalität hinaus noch eine zusätzliche, verborgene und potentiell schädliche Funktionalität besitzt.

3.4 Würmer

Ein *Wurm* ist ein selbstständiges Programm, das sich vervielfältigt und eigenständig über ein Netzwerk weiterverbreitet.

Buffer Overflow Attacks

- Viele Würmer nutzen sog. *Buffer Overflows* zum Eindringen in Server
- Problem: Viele Web-Anwendungen checken nicht die Größe der übergebenen Argumente, der Puffer "läuft über". Wichtige Speicherbereiche werden überschrieben.
- Sog. Exploit scripts veranlassen den Server, die Daten im Puffer als Instruktionen auszuführen
- Mehr dazu in Internet-Sec 2 (Prof. Kriha)

Buffer Overflows

Beispiel: Code Red (2001)

- Angriffspunkt: Microsoft Internet Information Server (IIS)
- Firewall lässt HTTP-Request an Webserver durch
- Übertrage Wurmcode durch verlängerten HTTP-Request
Wurmcode wird vom Webserver ausgeführt (Buffer Overflow Attack)
- Wirkung des Wurmcodes:
Ändere IP des Webserver leicht ab und versuche Wurmcode an geratene IP im Intranet zu übertragen
- Nach und nach gesamtes Intranet infiziert
- Rechner tut nichts anderes als Wurmcode auszuführen

3.5 Anderes Ungeziefer

Trap doors (dt: Falltüren) sind undokumentierte Einstiegspunkte in ein gesichertes System, die von einem Programmierer hinterlassen werden.

Logic Bombs Logische Bomben sind vom Programmierer versteckte Schadfunktionen, die bei einem bestimmten Ereignis ausgelöst werden.

Bakterien Bakterien sind sich selbst vervielfältigende Programme, die ausser dieser Vervielfältigung keine Schadwirkung besitzen.

Hoax Ein Hoax ist ein sog. Pseudovirus, eine Falschmeldung über angebliche Virengefahr.

3.6 Useful Links

- Vireinfo des HdM - Rechenzentrums
http://www2.hdm-stuttgart.de/rz/index.cgi?kn=2_3
- Antivirus - Seite des Bundesamts für Sicherheit in der Informations-technik (BSI)
<http://www.bsi.bund.de/av/index.htm>
- Computer Emergency Response Team (CERT), Uni Pittsburgh
<http://www.cert.org>
- Hoax - Info Service der TU Berlin
<http://www.hoax-info.de>

3.7 Spam

Spam

Spam is unsolicited, unwanted, bulk, advertising email, which may or may not harm users

Techniken

- "Abgrasen" von e-mail Adressen aus News-Foren und Websites
- Einsetzen echter Adressen in das "From"-Feld (führt zu "Bounces" bei der betroffenen Domain)

Bedrohungen durch Spam

- Individuelle
 - Zeitverschwendung
 - Möglicher Betrug bei angebotenen Diensten
- Organisatorische
 - Produktivitätsverlust
 - Spambekämpfung ist teuer und zeitaufwändig
- Infrastrukturelle
 - Verschwendung von Ressourcen

Gegenmaßnahmen

- Individuelle
 - Sparsames Veröffentlichen der geschäftlichen e-mail Adresse
 - * Ersetze @ durch "at"
 - * Nutze evtl. einfache Verschlüsselung (z.B. ROT13)
 - Keine Postings in Newsgroups unter der geschäftlichen e-mail Adresse
 - Niemals auf Spam antworten
- Organisatorische
 - Content Checking: Suche nach Schlüsselwörtern
 - Blacklists: Mails von diesen Absendern werden verworfen
 - Whitelists: Mails von diesen Absendern sind vom Content Checking ausgenommen.
- Globale
 - "Hash Cash": Erschwere Versenden von Millionen e-mails durch vorgeschriebene Bildung von Prüfsummen (Hashing)

Links

- www.antispam.de
- www.spamflam.de
- www.goodmail.com
- www.ironworks.com/comedy/python/spam.htm

3.8 Phishing

- Spam-artiges Senden von Mails aus scheinbar legitimer Quelle
- Aufforderung, sich mit einer Website zu verbinden und dort Passwörter etc. einzugeben
- Oft gekoppelt mit gefälschten Links und gefakten Websites/Zertifikaten
- DNS-Poisoning

Phishing-Test-Seite:

www.sit.fraunhofer.de/_SIT-Projekte/fragebogen_spk/DA/test.php

Kapitel 4

Kommunikation im Internet

4.1 Internet-Historie

- Entwicklung in den 60er Jahren vom amerikanischen Verteidigungsministerium initiiert
- Entwickelt von Advanced Research Project Agency (ARPA)
- 1969: ARPA-Net: Erstes Paketvermitteltes Netz verbindet vier Rechner
- ab 1973: Entwicklung von TCP/IP
- 1983: Nutzung von TCP/IP durch das ARPA-Net
- 1991: Veröffentlichung von HTTP
- Heute: allein über 20 Millionen Web Server!

More Info: www.isoc.org/internet/history/

4.2 OSI Schichtenmodell

1. *Physikalische Schicht (Physical Layer)*
 - Festlegen von Übertragungsparametern zwischen zwei Netzknoten:
 - Medium (Kupferkabel, Glasfaser, etc.)
 - Übertragungsgeschwindigkeit
 - Bit-Kodierung
2. *Sicherungsschicht (Data Link Layer)*
 - Fehlerfreiheit der Bitübertragung zwischen zwei Netzknoten z.B. durch Prüfsummen
3. *Vermittlungsschicht (Network Layer) - IP*
 - u.a. Adressierung und Routing der Daten zwischen Sender und Empfänger
4. *Transportschicht (Transport Layer) - TCP, UDP*
 - Verbindungsaufbau zwischen den beteiligten Prozessen
 - Qualität der Übertragung
5. *Sitzungsschicht (Session Layer)*

- Kontrolle der Verbindung (Abbruch/ Wiederaufnahme)

6. *Darstellungsschicht (Presentation Layer)*

- Rechnerunabhängige Darstellung der Anwendungsdaten
- Komprimierung
- Verschlüsselung

7. *Anwendungsschicht (Application Layer)* - **FTP, SMTP, HTTP**

- Kommunikationsorientierte Anwendersoftware

4.3 Internet Protokoll

IPv4

- **Aufgaben:**
 - Routing der Daten
 - Fragmentierung (Aufteilung in Pakete)
- **Header-Informationen (unter anderem):**
 - Länge des Headers und des Pakets
 - Time-to-Live Parameter
 - Typ des folgenden Headers (TCP oder UDP)
 - 32-Bit IP-Adresse des Quellrechners
 - 32-Bit IP-Adresse des Zielrechners
- **Maximale IP-Paketlänge: 65 kByte**

IPv6

- Entwicklung seit 1995
- Größerer Adreßraum (16 Byte statt 4 Byte bei IPv4)
- Unterstützung "bevorzugter Weiterleitung" von Paketen
- enthält Sicherheitsmechanismen zur Sicherstellung von Vertraulichkeit (Encapsulating Security Payload, ESP) und Integrität (Authentication Header, AH) von Header und Payload
- Siehe Abschnitt [10.1.1](#).

4.4 Transmission Control Protocol (TCP)

- Verbindungsaufbau
- Verbindungsabbau
- Kontrolle der Reihenfolge der Pakete durch 32 Bit Sequence Numbers
- Prozessadressierung über 16 Bit Portnummern (z.B. HTTP über Port 80)

Abbildung 4.1 zeigt den Aufbau eines TCP Headers.

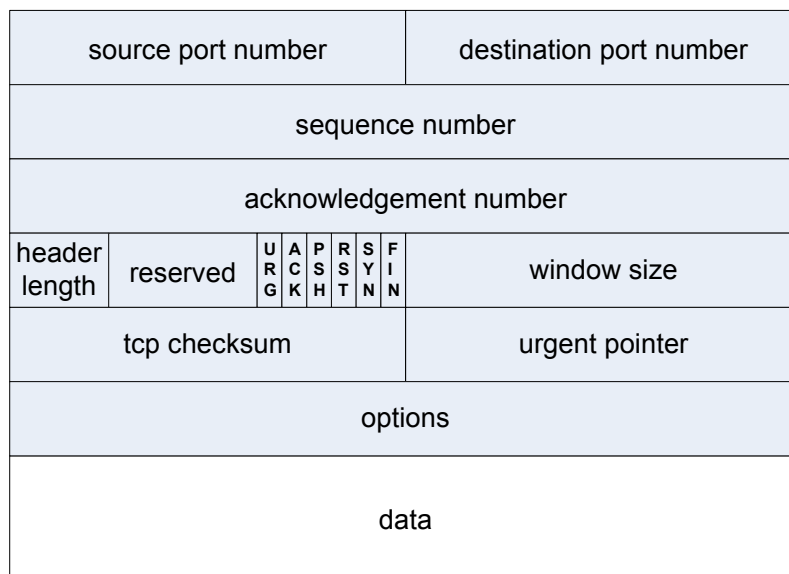


Abbildung 4.1: TCP Header

TCP Verbindungsaufbau (Handshake)

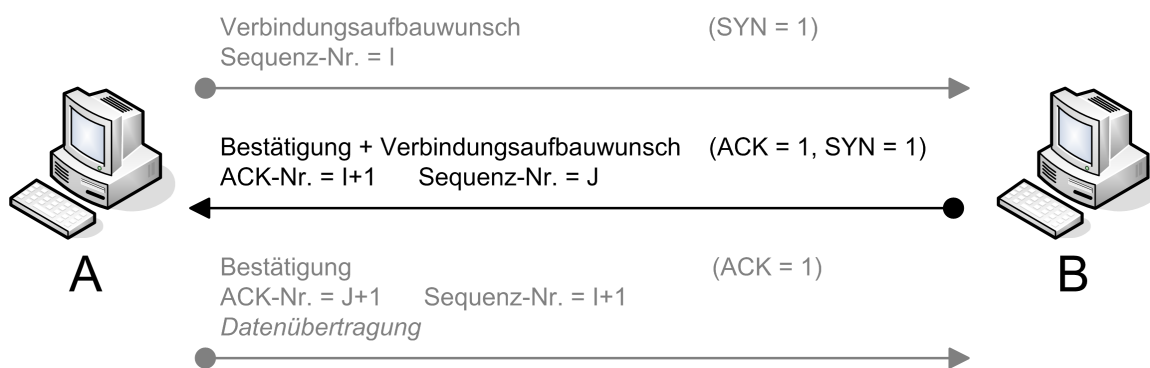


Abbildung 4.2: 3-Way handshake

Verbindungsaufbau geschieht in 3 Schritten:

1. Verbindungsaufbauwunsch vom Rechner A

2. Bestätigung und Verbindungsaufbauwunsch vom Rechner **B**
3. Bestätigung Und Datenübertragung vom Rechner **A**

Siehe Abbildung 4.2.

4.5 Sicherheitsprobleme bei TCP/IP

4.5.1 Probleme mit IP

- **Aktiv:** *IP-Payload* (Paket ohne Header) wird im Klartext und nicht integritätsgeschützt übertragen.
Gefahr: *Sniffing* (abhören)
- **Passiv:** *IP-Header* wird im Klartext und nicht integritätsgeschützt übertragen.
Gefahr: *IP-Spoofing* (sich für jemanden anderen ausgeben), *Traffic Analysis*

Sniffing

IP-Spoofing

Sniffing

- Zugriff auf Client-PC
- Zugriff auf Kupferkabel/Glasfaser/WLAN etc.
- Zugriff auf Router
- Zugriff auf umgeleitete Nachricht (z.B. auf einen umgeleiteten Router auf dem man Zugriff hat)

4.5.2 Probleme mit TCP

- TCP Header wird im Klartext und nicht integritätsgeschützt übertragen
- **Angriffe:**
 - Portscanning
 - SYN-Flooding
 - Session Hijacking
 - DNS-Spoofing
 - RST-Attack

Portscanning

- Ziel: Erhalte Informationen über installierte Dienste, Betriebssysteme etc.
- Installierte Dienste am Zielrechner antworten auf Verbindungsaufbauwünsche am entsprechenden Port
- Scan-Methoden:
 - Half-Open Scan
 - SYN-ACK Scan

Half-Open Scan

- Führe nur die ersten beiden Teile des 3-Way Handshakes durch
- Zweiter Teil vom Zielrechners enthält alle nötigen Infos:
 - ACK-Message: Port geöffnet, Dienst installiert
 - RST: Port geschlossen, Dienst nicht installiert
- Da Verbindung unvollständig, wird sie nicht mitprotokolliert

SYN-ACK Scan

- Schicke SYN-ACK Message an Zielpart (SYN-ACK darf normalerweise nur auf SYN folgen!)
- Antwort RST: Port geschlossen
- Keine Antwort: Port geöffnet (Zielrechner sucht nach vorhergehendem Handshake)

Denial of Service durch TCP-SYN-Flooding

1. Angreifer produziert viele TCP-Verbindungsaufbauwünsche an Opfer:
SYN-Flag gesetzt. Falsche IP-Source-Adresse in IP-Header (am besten nicht existierende)
2. Opfer antwortet mit Bestätigung:
SYN/ACK-Flag gesetzt, aber an falsche IP-Adresse
3. Opfer wartet vergeblich auf 3. Nachricht des TCP-Handshake legt unerledigte Aufbauwünsche in Speicher, bis er überläuft. Konsequenz: Opfer kann nicht mehr erreicht werden.

Session Hijacking

Ziel: Übernahme bestehende (autorisierte) Verbindung, trete unbemerkt an die Stelle eines der beiden Partner

Voraussetzung: Rechner **X** kann Seq.-Nr. i und Seq.-Nr. j mitlesen oder raten.

Siehe Abbildung 4.3.

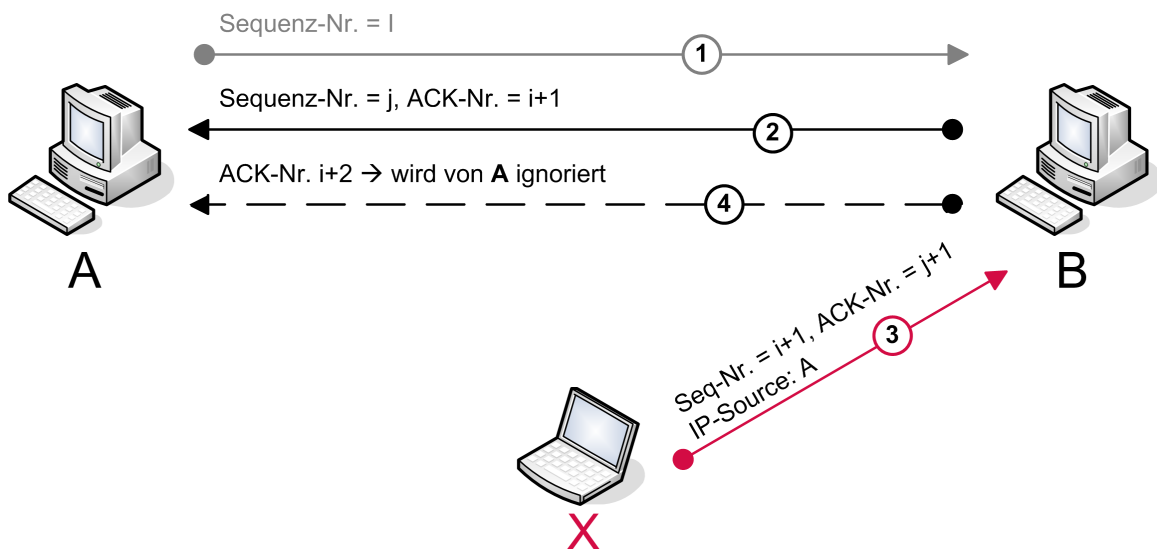


Abbildung 4.3: Session Hacking

Denial-of-Service durch RST-Angriff

- Schleuse in bestehende Verbindung Paket mit RST-Flag ein
- Verbindung wird sofort abgebrochen

Sequenznummern-Erzeugung unter TCP

- **Aus RFC 793 (TCP Protocol Specification):**
"When new connections are created, an initial sequence number (ISN) generator is employed which selects a new 32 bit ISN. The generator is bound to a (possibly fictitious) 32 bit clock whose low order bit is incremented roughly every 4 microseconds."
- **Realität (häufig):**
 - Verwende 32-Bit Zähler, der jede Sekunde um 128 erhöht wird.
 - Bei neuen Verbindungen: Erhöhe alten Zählerstand um 64 oder 65536
- **Auch die verwendeten Pseudozufallsgeneratoren sind angreifbar!**

DNS-Spoofing/Poisoning

- Anfrage an Server **A**: weiß Adresse nicht
- Angreifer **X** schickt Server **A** mit gefälschter Adresse von **B** andere IP-Adresse
- Damit liegt im DNS-Server **A** eine falsche IP-Adresse von **www.telekom.de**! Der Angreifer **X** kann unter dieser Adresse einen gefälschten Telekom-Web-Server einrichten.

Siehe Abbildung 4.4.

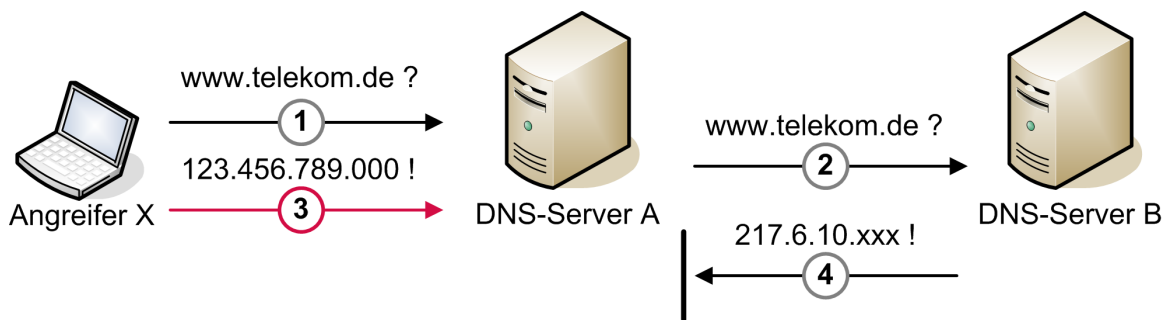


Abbildung 4.4: DNS-Spoofing/Poisoning

Zusammenfassende Bemerkung

Diese Angriffe (Portscanning, Half-Open Scan, SYN-ACK Scan, Session Hijacking, ...) sind alle möglich, weil bei TCP/IP *keine Authentifizierung* stattfindet.



4.6 User Datagram Protocol (UDP)

Der UDP-Header enthält nur den Source und Destination Port, die Länge des Pakets, die Prüfsumme und die Daten. (Siehe Abbildung 4.5)

source port	destination port
UDP length	UDP checksum
data	

Abbildung 4.5: UDP Header

Da die Sequenz-Nummer wegfällt, kann der Angreifer problemlos Pakete einschleusen.

4.6.1 UDP-Flood (Echo)

Beim UDP-Flood schickt der Angreifer (**X**) ein Paket an ein zufälligen Port des Opferrechners (**B**) und gibt dabei die Source-IP eines anderen Rechner (**A**) an. TODO: Weiter ausführen!

Siehe Abbildung 4.6.

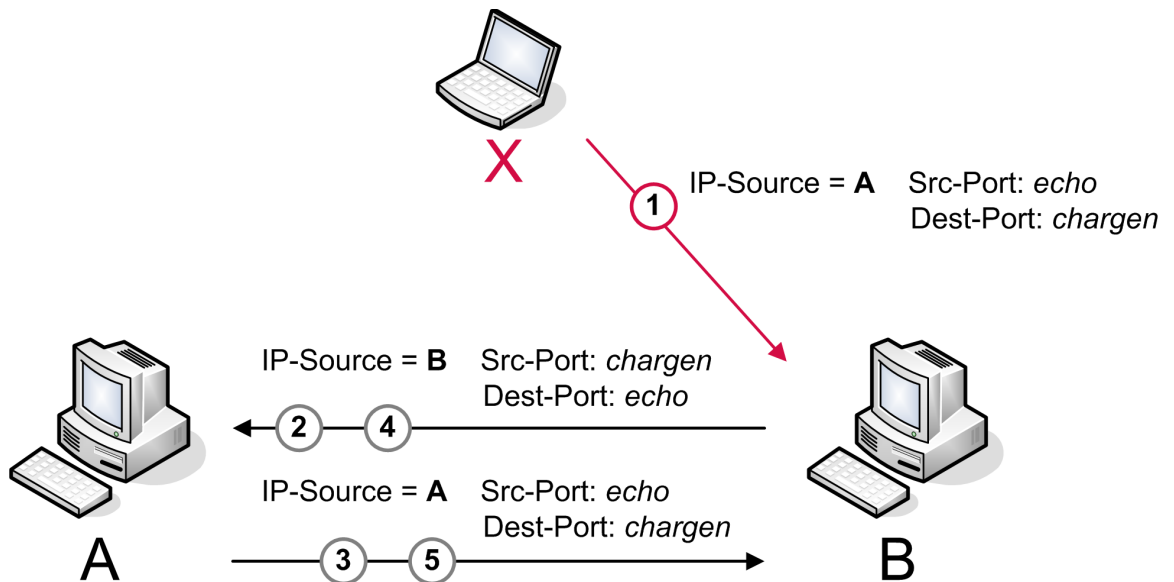


Abbildung 4.6: UDP-Flood (Echo)

4.7 Internet Control Message Protocol (ICMP)

Das ICMP wird normalerweise zwischen Routern benutzt. Ein solches Paket sollte bei einem PC ignoriert werden.

Kapitel 5

Sicherheitsmechanismen

5.1 Firewalls

Ein *Firewall* besteht aus einer oder mehreren Hard- und Software- Komponenten, die zwei Netzwerke koppeln und sicherstellen, dass jeglicher Verkehr zwischen den Netzen durch den Firewall geleitet wird. Er realisiert eine Sicherheitsstrategie, die Zugriffsrestriktionen und ggf. Authentifikationsanforderungen umfasst. Er leitet nur solche Datenpakete weiter, die diese Strategie erfüllen.

Firewalls

Firewall Typen

- In Hard- oder Software realisierbar
 - Vorteil der HW-Lösung: Vom lokalen Betriebssystem unabhängig
- Drei typische Ausprägungen von Firewalls:
 - Packet-filtering routers
 - Application-level gateways
 - Circuit-level gateways

5.1.1 Packet Filter

Siehe Abbildung 5.1.

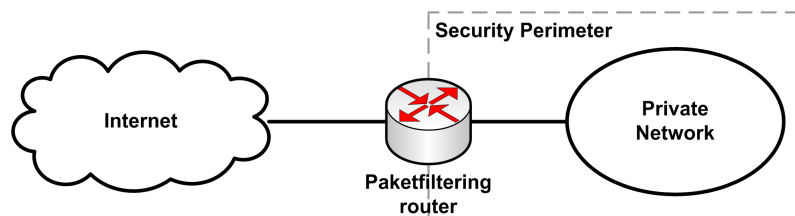


Abbildung 5.1: Paketfiltering Router

- Applies a set of rules (Access Control List, ACL) to each incoming IP packet and then forwards or discards the packet (layer 2-3)
- Filter packets going in both directions

- The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header
- Two default policies (discard or forward)
 - discard:
 - * block everything that is not explicitly allowed
 - * block everything for which there is no rule
 - * Problem: Every new protocol is blocked per default: Security policy must be revised
 - forward:
 - * forward everything that is not explicitly forbidden
 - * forward everything for which there is no rule
 - * Problem: Every new protocol is allowed. May cause Security problems

Default Policies

Aktion	Sende-IP	Sendeport	Ziel-IP	Zielpport	Flags	Bedeutung
Block	*	*	*	*	*	Default rule

Tabelle 5.1: Paketfilter default policy: Discard

Aktion	Sende-IP	Sendeport	Ziel-IP	Zielpport	Flags	Bedeutung
Allow	*	*	*	*	*	Default rule

Tabelle 5.2: Paketfilter default policy: Forward

Example Filtering Rules

Aktion	Sende-IP	Sendeport	Ziel-IP	Zielpport	Flags	Bedeutung
Block	Evil hosts	*	*	*	*	Don't trust them
Allow	Our hosts	*	*	*	*	Outgoing traffic

Tabelle 5.3: Paketfilter example Filtering Rule 1

Aktion	Sende-IP	Sendeport	Ziel-IP	Zielpport	Flags	Bedeutung
Block	Evil	*	intern	*	*	Don't trust them
Allow	Intern	*	*	25	*	Outgoing E-mail traffic
Allow	*	25	*	*	*	Responses to our mail

Tabelle 5.4: Paketfilter example Filtering Rule 2

Weitere einfache Regeln

- **Egress-Filtering:**
Ausgehende Pakete müssen Source-Adresse aus eigenem Bereich haben
- **Ingress-Filtering:**
Hereinkommende Pakete dürfen keine Source-Adresse aus eigenem Bereich haben

Angriff auf Paketfilter

Der Angreifer **X** kann bösartige Pakete zu **B** schicken, indem er sich für einen anderen Rechner **A** ausgibt. Siehe Abbildung 5.2.

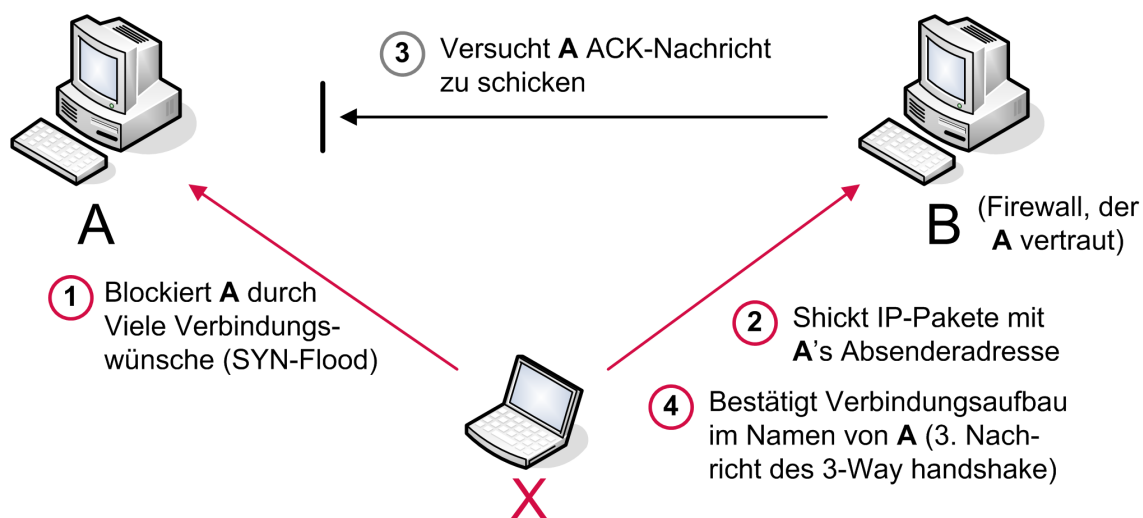


Abbildung 5.2: Angriff auf Paketfilter

Schwierigkeit für Angreifer: Er muss Sequence-Numbers raten.

Tiny Fragment Attack

- Erzeuge so kleine IP-Pakete, dass keine TCP-Header Infos mehr auslesbar sind. (Paket besteht also nur aus IP-Header)
- Filter, die nach TCP Headern filtern und mit Default-Forward arbeiten, können so ausgehebelt werden.

Vor- und Nachteile von Paketfiltern

- **Vorteile:**
 - Schnell
 - Einfach zu konfigurieren

- **Nachteile:**

- Unflexibel (keine Unterscheidung zwischen Anwendungen. Nur Layer 2-3 werden geprüft)
- Einfach zu konfigurieren
- Schwer, genügend korrekte Regeln aufzustellen

Paketfilter: Neuere Entwicklungen

- **Dynamische Filtertabellen:**

- Alle Ports sind per default geschlossen
- Ports werden nur auf Anfrage geöffnet und wieder geschlossen (je nachdem wer fragt)

- **Stateful Inspection:**

- Teile Verbindungen in Zustände (States) ein (schaut dabei auf TCP-Flags)
- Lege State-Tabelle mit Status der ein- und ausgehenden Verbindungen an
- Werte zusätzlich Informationen aus Anwendungs- und Transportschicht aus

5.1.2 Application-Level Gateway

Siehe Abbildung 5.3.

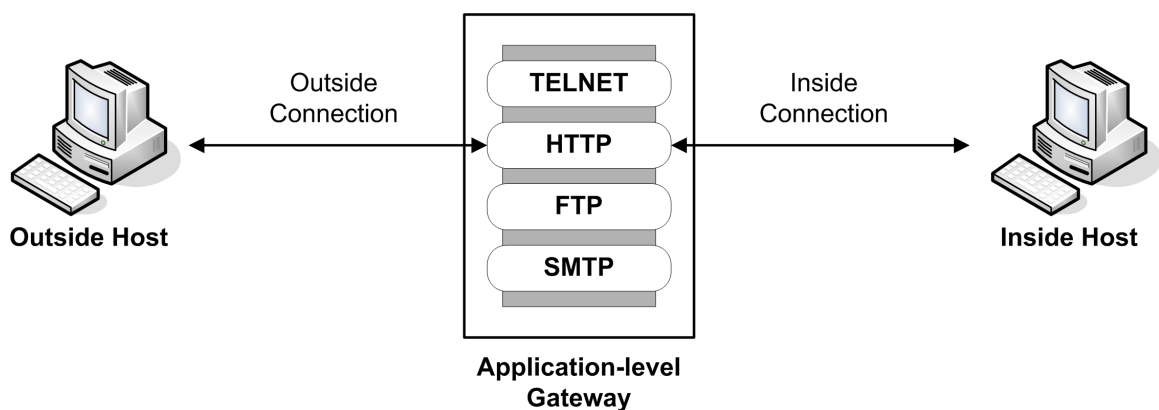


Abbildung 5.3: Application-level Gateway

- Kann Informationen bis OSI Schicht 7 auswerten
- Vorteil: Gateway hat Kontrolle über alle Verbindungen
- Nachteil: User muss immer über Gateway gehen (mühsam)

5.1.3 Circuit-Level Gateway

Siehe Abbildung 5.4.

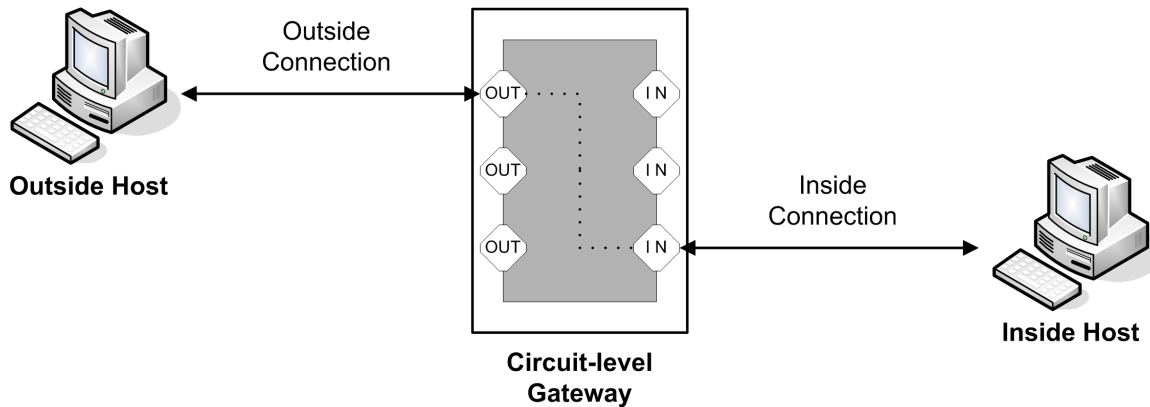


Abbildung 5.4: Circuit-level Gateway

- Keine strikte Kontrolle der Anwendung, aber der Verbindungen
- Gateway baut eigene TCP-Verbindungen nach aussen auf

5.1.4 Firewall Konfigurationen

Problem: Wie kann man einen Server betreiben, der vom Internet aus erreichbar sein soll, ohne das restliche Intranet zu gefährden?

Bastion Hosts

- Besonders gesicherter Rechner mit minimaler Funktionalität, der als sichere Plattform für Application-Level und Circuit-Level Gateways dient
- Häufig mit Paketfiltern kombiniert

Single-Homed Bastion Host

Siehe Abbildung 5.5.

- Alle Verbindungen von innen müssen über Bastion gehen
- Paketfilter so konfigurieren, dass alle Pakete zunächst zum Bastion-Host gehen. Bastion leitet nur weiter wenn Verbindung erlaubt
- Ist der Paketfilter gehackt, kann der Bastion Host umgangen werden!

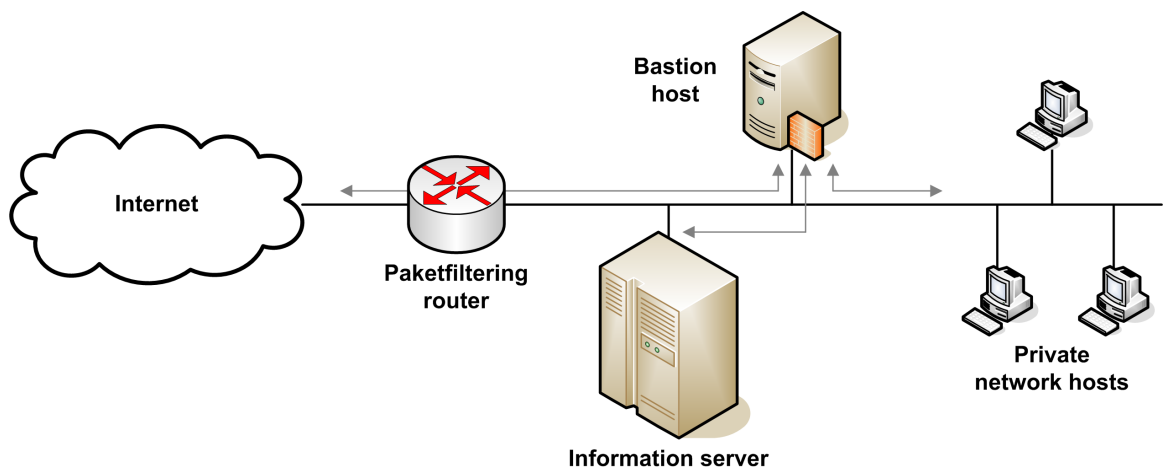


Abbildung 5.5: Single-Homed Bastion Host

Dual-Homed Bastion Host

Siehe Abbildung 5.6.

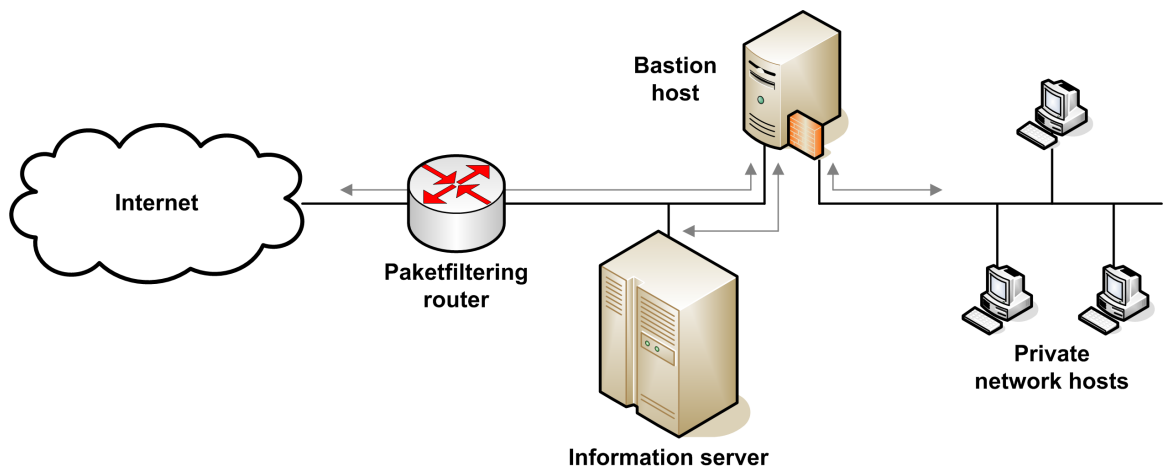


Abbildung 5.6: Dual Homed Bastion Host

Hier: Verbindung ins Internet physikalisch nur über Bastion Host möglich! (selbst wenn Paketfilter geknackt)

Screened-subnet Firewall System

Siehe Abbildung 5.7.

Kreiere Subnet, das nur aus Bastion Host und Information Server besteht. Von "innen" und "außen" ist nur das Subnet sichtbar!

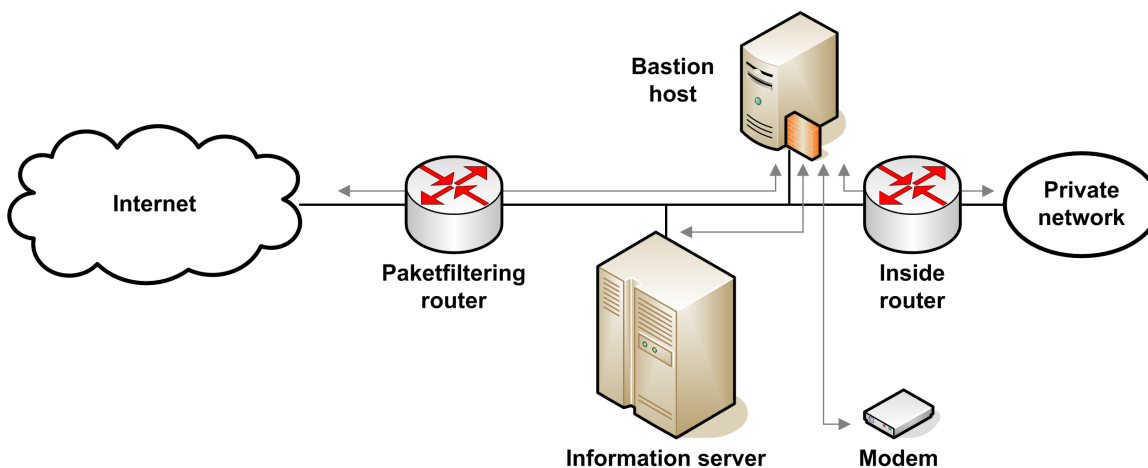


Abbildung 5.7: Screened-subnet Firewall System

5.1.5 Risiken und Grenzen von Firewalls

- Umfangreiche Netzkenntnisse zur Konfiguration erforderlich
- Kontinuierliche Administration nötig
- Trügerisches Sicherheitsgefühl der User führt zu Leichtsin
- Firewalls schützen nur rudimentär gegen aktive Inhalte
- Probleme beim Tunneling von Paketen

5.1.6 Rechtliche Fragen beim Betrieb von Firewalls

- Private e-mails unterliegen dem Fernmeldegeheimnis!
- Auch automatisches Content-Scanning stellt im Prinzip eine Verletzung dar
- Einfachste Lösung: Explizites Verbot privater e-mails am Arbeitsplatz

5.1.7 Personal Firewalls

- Ein *Personal Firewall* ist ein Stück SW, die auf dem Arbeitsplatzrechner installiert ist und FW-Funktionalitäten übernimmt
- Häufig kombiniert mit Virenscannern und Sandbox - Funktionalität

Personal Firewalls

Some freely available Personal Firewalls

- Windows Firewall (included in Service Pack SP2 for Windows XP)
- Linux iptables
- Zonealarm (www.zonelabs.com)
- Kerio Personal Firewall (www.kerio.com)

5.1.8 Organisatorische Fragen

- Kosten
 - Kosten/Nutzen Analyse
- Aufwand für Betrieb und Unterhalt

5.1.9 Zukünftige Trends

- **Managed Security Provider (MSP)**
 - Vergabe an Fremdfirma
 - Vorteil: keine neue Mitarbeiter/HW nötig
 - Nachteil: "man liefert sich aus" (Problem: Konkurs des Sec. Prov.)
- **Intrusion Detection Systems (IDS)**
 - Melden/Loggen verdächtige Aktivitäten von außen (Netzsensoren)
 - Versuchen Angreifer im Netz zu finden

Kapitel 6

Symmetrische Kryptographie

6.1 Allgemeines

6.1.1 Begriffe

- *Kryptographie*
"Verborgenes Schreiben"
- *Symmetrisch*
Sender und Empfänger benutzen den gleichen Schlüssel (Key)
- *Klartext (Plaintext)*
Was man verschlüsselt
- *Chiffretext (Ciphertext)*
Output der Verschlüsselung
- *Chiffren brechen*
Versuch (mit Mitteln der Kryptoanalyse), Chiffren zu brechen: d.h Klartext und/oder Schlüssel zu ermitteln

6.1.2 Symmetrische Kryptosysteme

Definition

Σ_1, Σ_2 seien zwei Alphonbete.
 $M \subseteq \Sigma_1^*$ sei eine Menge von Klartexten,
 $C \subseteq \Sigma_2^*$ sei eine Menge von Chiffretexten.

Ein Tripel $S = (M, F, C)$ wobei F eine Menge $F = \{f_0, f_1, f_2, \dots, f_n\}$ ist, bestehend aus umkehrbaren Funktionen $f_i : M \rightarrow C$ heißt *symmetrisches Kryptosystem*.



Symmetrische
Kryptosysteme

6.1.3 Beispiel: Caesar-chiffre

c = YHQL YLGL YLFL
 m = VENI VINI VICI

f_k = "Verschiebe Klartext um k Stellen nach vorne"
 f_k^{-1} = "Verschiebe Klartext um k Stellen nach hinten"
 M = $\{w \in \Sigma^* \mid |w| \leq 12\} = C$
 F = $\{f_0, f_1, \dots, f_{26}\}$

6.1.4 Kerkhoffsches Prinzip



Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus (d.h. der Funktionenmenge F) abhängen, sondern allein von der *Geheimhaltung des Schlüssels* (d.h. von der Geheimhaltung der ausgewählten Funktion f_i).

6.1.5 Starke Algorithmen



Ein Kryptoalgorithmus heißt "stark",

- wenn es keinen Angriff gibt, der schneller als eine vollständige Schlüsselsuche (Brute-Force-Attack) zum Ziel führt, UND
- wenn es so viele Schlüssel gibt, dass eine vollständige Schlüsselsuche unmöglich ist.

6.1.6 Angriffsarten

- Aktive Angriffe
 - Aktives Eingreifen in die Kommunikation, Verändern/Einschleusen/Unterdrücken von Nachrichten
- Passive Angriffe
 - Reines "Beluschen" ohne Störung der Kommunikation

Passive Angriffe

- Ciphertext-Only
- Known-Plaintext
- Known-Ciphertext
- Adaptive-Chosen Plaintext
- Adaptive-Chosen Ciphertext

6.1.7 Beispiel Angriffe

Brute-Force-Attack

- Angreifer kennt (M, F, C) und $c = f_k(m)$
Schlüssel k und Nachricht m sind unbekannt.
- Für alle $k \in \text{Schlüsselraum}$ bilde $f_k^{-1}(c)$
- Gegenmaßnahme: Schlüsselraum muss sehr gross sein!
(Heutige Chiffren: 2^{128} Schlüssel)

Known-Plaintext-Attack

c = YHQL YLGL YLFL
 m = VENI.....

Dadurch dass die ersten Buchstaben bekannt sind, lässt sich der Schlüssel leicht herleiten.

Ciphertext-Only-Attack

c = YHQL YLGL YLFL
 m = ...I...I...I...I

in Ciphertext häufig auftretende Buchstaben sind auch in Plaintext häufig auftretende Buchstaben. Auf diese Weise lässt sich der Schlüssel ebenfalls leicht herleiten.

6.2 Generelle Klassen symmetrischer Chiffren

- *Monoalphabetische Substitutionschiffren* :
Ersetze (schlüsselabhängig) jedes Symbol des Klartextraums durch festes Chiffresymbol
 - Problem: Häufigkeiten der Klartextsymbole gleichen denen der Chiffresymbole
- *Polyalphabetische Substitutionschiffren* :
Ersetze (schlüsselabhängig) jeden Klartextblock durch festen Chiffreblock
 - Problem: Selbes Problem wie bei monoalphabetische Substitutionschiffren, nur dass sich nicht Zeichen wiederholen, sondern Zeichenblöcke. Aber je länger die Blöcke, desto wirksamer gegen Ciphertext-Only-Attacken.

Monoalphabetische
Substitutionschiffren

Polyalphabetische
Substitutionschiffren

Buchstabenhäufigkeiten der deutschen Sprache (bezogen auf 1000 Zeichen)

Siehe Abbildung 6.1.

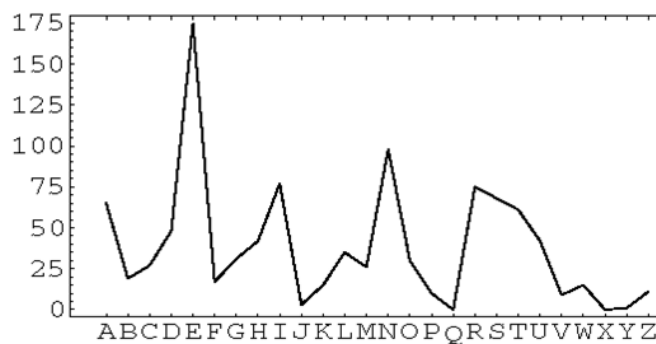


Abbildung 6.1: Häufigste Buchstaben (ENIRSAT)

6.3 Historische Chiffren

- Antike Chiffren
 - Caesar (Siehe Abschnitt 6.1.3)
 - Skytale (Siehe Abschnitt 6.3.1)
- Neuzeitliche Chiffren
 - Maria-Stuart-Chiffre (1586)
 - Vigenère-Chiffre (1586) (Siehe Abschnitt 6.3.2)
 - ENIGMA (1923-1945) (Siehe Abschnitt 6.3.3)

6.3.1 Skytale

- Permutationschiffre (Polyalphabetische Chiffre)
- Algorithmus: wickle Papierband um ein Stab
Schlüssel: Durchmesser des Stabs

Siehe Abbildung 6.2.

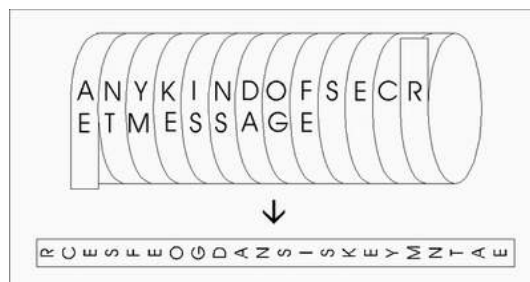


Abbildung 6.2: Skytale

6.3.2 Vigenère-Chiffre

- Erfunden von Blaise de Vigenère (1523-1596) und anderen
- Polyalphabetische Chiffre basierend auf Schlüsselwort
- Galt als "chiffre indechiffable" bis zur Kryptoanalyse durch Kasiski (1863)
- Funktionsweise:
 - Bestimme Periodenlänge p
 - Zeichenfolgen im Abstand p sind Caesar-verschlüsselt.

Methoden zur Bestimmung von p (nach Kasiski)

- Raten (und Vergleich der sich ergebenden Häufigkeiten mit Klartext-Häufigkeiten)
- Mustersuche nach Kasiski

– Beispiel:

c = DERMA NNIST DERBA UMBLU EHT..
 m = CDQLZ MMHRS CDQAZ TLAMT DGS..

Die beiden Stellen CDQ starten mit 175 Zeichen Versetzung. $175 = 5 \cdot 5 \cdot 5$
 $\Rightarrow$ Mögliche Blocklängen sind 5 und 7.

- Koinzidenzindex nach Friedman

6.3.3 Die ENIGMA**ENIGMA-Geschichte**

- 1923 Erfindung durch Arthur Scherbius
- 1925 Deutsche Marine kauft erste Exemplare
- 1928 Landstreitkräfte benutzen Enigma. Erste krypto-analytische Erfolge durch polnische Mathematiker
- 1930 Verbesserung durch Hinzufügen des Steckbretts
- 1933 Einsatz von elektromechanischen Computern (Bombas) zum Knacken der Enigma durch die Polen
- 1938 Statt 3 fester Walzen nun 3 beliebige aus 5
- 1939 Polen geben ihr Wissen an englische Kryptoanalytiker weiter.
- Kurz darauf Grossangriff auf die Enigma in Bletchley Park, unter Leitung von Alan Turing. Schon bald routinemäßige Entschlüsselung der 3-Walzen Enigma
- 1942 Einführung der 4-Walzen Enigma für die deutsche U-Boote
- 1943 Auch die 4-Walzen Enigma wird geknackt, u.a. durch Colossus, den ersten englischen programmierbaren Computer

6.4 Exkurs: Wahrscheinlichkeitstheorie**Definition**

Eine endliche, nichtleere Menge S heißt *Ereignismenge*.

Die Elemente von S heißen *Elementarereignisse*.

Die Teilmengen von S heißen *Ereignisse*.

Die Menge aller Teilmengen von S (= Menge aller Ereignisse) heißt *Potenzmenge* $P(S)$.



Beispiel

Würfeln mit einem Würfel

$$S = \{1, 2, 3, 4, 5, 6\}$$

Ereignis $E = \{2, 4, 6\} \in S$ ("Würfele eine gerade Zahl")

Wahrscheinlichkeitsverteilungen

Eine Abbildung $p : P(S) \mapsto [0, 1]$ heißt *Wahrscheinlichkeitsverteilung* $:\Leftrightarrow$

$$(i) \quad p(S) = 1$$

$$(ii) \quad p(E_1 \cap E_2) = p(E_1) + p(E_2) \text{ für alle } E_1, E_2 \in P(S) \text{ mit } E_1 \cap E_2 = \emptyset$$

Definition



S Ereignismenge. $E_1, E_2 \subseteq S$ Ereignisse.

Dann heißt $p(E_1|E_2) := \frac{p(E_1 \cap E_2)}{p(E_2)}$ *bedingte Wahrscheinlichkeit* von E_1 (*unter der Bedingung* dass E_2 eintritt).

bedingte
Wahrscheinlichkeit

Bemerkung: Haben E_1 und E_2 "nichts miteinander zu tun", dann gilt:

$$p(E_1|E_2) = p(E_1) = \frac{p(E_1 \cap E_2)}{p(E_2)}$$

Beispiele

Würfeln mit 2 Würfeln:

$$S = \{(1, 1), (1, 2), \dots, (6, 5), (6, 6)\}$$

Da $p(S) = 1$ und alle Elementarereignisse gleich wahrscheinlich $\Rightarrow p = \frac{1}{36}$ für alle Elementarereignisse.

- $E_1 = \text{"Augensumme} = 7" = \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\}$
 $\Rightarrow p(E_1) = 6 \cdot \frac{1}{36} = \frac{1}{6}$
- $E_2 = \text{"Augensumme} = 7, \text{ aber einer der beiden Würfel muss } 3 \text{ zeigen}" = \{(3, 4), (4, 3)\}$
 $\Rightarrow p(E_2) = \frac{2}{36} = \frac{1}{18}$

Fasse E_2 auf als Schnittmenge von E_1 (Augensumme = 7) und E_3 (Ein Würfel zeigt 3), d.h. gleichzeitiges Auftreten von 2 Ereignissen entspricht der Schnittmenge.

Hier: $E_3 = \{(3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (1, 3), (2, 3), (4, 3), (5, 3), (6, 3)\}$

$$\Rightarrow p(E_3) = \frac{11}{36}, p(E_1 \cap E_3) = \frac{1}{18} \neq \frac{11}{36} \cdot \frac{1}{6} \Rightarrow E_3 \text{ ist nicht unabhängig von } E_2.$$

- $E_4 = \text{"Ein Würfel zeigt 3, unter der Bedingung, dass Augensumme} = 7\text{"}$
 $= p(E_3|E_1)$

$\Rightarrow$ Schränke Ereignismenge ein auf $E_1 = \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\}$

$$\Rightarrow p(E_4) = \frac{\text{Anzahl Elemente in } E_3 \cap E_1}{\text{Anzahl Elemente in } E_1} = \frac{\frac{\text{Anzahl Elemente in } E_3 \cap E_1}{\text{Anzahl Elemente in } S}}{\frac{\text{Anzahl Elemente in } E_1}{\text{Anzahl Elemente in } S}} = \frac{p(E_3 \cap E_1)}{p(E_1)} = \frac{2}{6} = \frac{1}{3} = \text{bedingte Wahrscheinlichkeit.}$$

Definition

Zwei Ereignisse E_1, E_2 heissen **unabhängig** $\Leftrightarrow p(E_1 \cap E_2) = p(E_1) \cdot p(E_2)$



Beispiele

Unabhängige Ereignisse

Würfeln mit 2 Würfeln:

$$S = \{(1, 1), \dots, (6, 6)\}$$

- $E_1 = \text{"1. Würfel zeigt 1"} = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6)\} = \frac{1}{6}$
 $E_2 = \text{"2. Würfel zeigt 6"} = \{(1, 6), (2, 6), (3, 6), (4, 6), (5, 6), (6, 6)\} = \frac{1}{6}$

$$E_1 \cap E_2 = \{(1, 6)\} \Rightarrow p(E_1 \cap E_2) = \frac{1}{36} = p(E_1) \cdot p(E_2)$$

$\Rightarrow E_1$ und E_2 sind unabhängig!

- $E_3 = \text{"Augensumme} = 7" = \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\}$
 $\Rightarrow p(E_3) = \frac{1}{6}$

$$E_1 \cap E_3 = \{(1, 6)\} \Rightarrow p(E_1 \cap E_3) = \frac{1}{36} = p(E_1) \cdot p(E_3)$$

$\Rightarrow E_1$ und E_3 sind unabhängig!

- $E_4 = \text{"Augensumme} = 8" = \{(2, 6), (3, 5), (4, 4), (5, 3), (6, 2)\}$
 $\Rightarrow p(E_4) = \frac{5}{36}$

$$E_1 \cap E_4 = \emptyset \Rightarrow p(E_1 \cap E_4) = 0 \neq p(E_1) \cdot p(E_4) = \frac{1}{6} \cdot \frac{5}{36} = \frac{5}{216}$$

$\Rightarrow E_1$ und E_4 sind *nicht* unabhängig!

6.4.1 Perfekte Sicherheit

Definition (Shannon, 1948):

Ein Kryptosystem $S = (M, F, C)$ heisst perfekt $\Leftrightarrow$

Für alle Chiffretexte $c \in C$ und alle Klartexte $m \in M$ gilt:

$$p(m|c) = p(m)^1$$

(Also, wenn die Kenntnis der Chiffre keinerlei Rückschlüsse auf den Klartext gibt)

¹ $p(m|c)$: m ist Klartext unter der Bedingung dass c Chiffre ist = Erfolgswahrscheinlichkeit bei Analyse von c . $p(m)$: Erfolgswahrscheinlichkeit bei Raten

6.4.2 Die Wahrscheinlichkeit eine Nachricht zu entschlüsseln

Angreifer kennt Klartextraum:

Rate Klartext: $p(m)$ Erfolgswahrscheinlichkeit ($m = \text{Klartextraum}$)

Analysiere c : $p(m|c)$ Erfolgswahrscheinlichkeit bei Analyse von c ($c = \text{Chiffretext}$)

Shannon: $p(m|c) = p(m) \Rightarrow$ Perfekte Chiffre. Analyse von c hilft nicht weiter.

Beispiele

1. Caesar-Chiffre (Nicht perfekt nach Shannon)

$$\Sigma = \{A, B, C, \dots, Z\}, M = \Sigma^{12} = C \Rightarrow |M| = 26^{26}$$

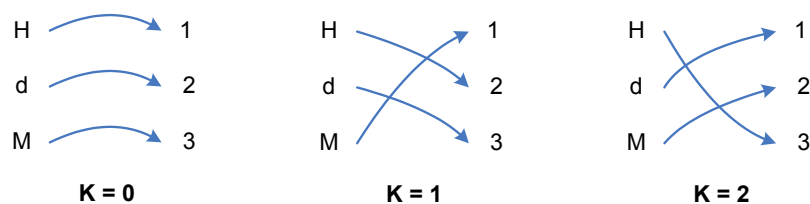
$$\Rightarrow p(m) = \frac{1}{26^{12}} \text{ (Raten des Klartextes)}^2$$

c	=	SBKFSFAFSFZF
m_0	=	SBKFSFAFSFZF
m_1	=	TCIFTGBGTGAG
m_2	=	UDJGUHCHUHBH
m_3	=	VENIVIDIVICI
$\dots$		
m_{25}	=	RADEREZEREZE

$\Rightarrow$ 26 mögliche Klartexte bei Analyse von c und der Bekanntheit des verschlüsselten Codes

$$p(m|c) = \frac{1}{26} \neq \frac{1}{26^{12}}$$

2. $\Sigma_1 = \{H, d, M\}$, $\Sigma_2 = \{1, 2, 3\}$
Klartextraum $M = \Sigma_1$, Schlüsselraum $C = \Sigma_2$, Abbildungen $K = \{0, 1, 2\}$



(Angreifer kennt diese Abbildungen)

Übertragen: $c = 2 \Rightarrow$ Mögliche Klartexte: d, H, M

$$\Rightarrow p(m|c) = \frac{1}{3} = p(m) \Rightarrow \text{perfekte Chiffre}$$

Warum? Schlüsselraum ist so gross wie Klartextraum

² $M = \Sigma^{12}$: Alle Wörter mit 12 Buchstaben. $|M| = 26^{26}$: Es gibt 26^{26} Klartexte

6.4.3 Perfektion von symmetrisches Kryptosystem

(M, F, C) sei ein symmetrisches Kryptosystem mit folgenden Eigenschaften:

- (i) $|M| = |F| = |C|$
(Es gibt so viele Schlüssel wie es Klartexte und Verschlüsselungsfunktionen gibt)
- (ii) Alle Schlüssel kommen mit gleicher Wahrscheinlichkeit vor.
- (iii) Zu jedem Klartext $m \in M$ und jeder Chiffre $c \in C$ gibt es genau $t_k \in F$ mit $f_k(m) = c$

Dann ist (M, F, C) perfekt!

Beispiel: One-Time-Pad (OTP)

$M = \{0, 1\}, C = \{0, 1\}, F = \{f_0, f_1\}$ mit $f_0(x) = x \oplus 0$ und $f_1(x) = x \oplus 1$

($\oplus$ bedeutet XOR (exclusive OR). Dh. $f_0 : 0 \rightarrow 0; 1 \rightarrow 1$ und $f_1 : 0 \rightarrow 1; 1 \rightarrow 0$).

Verallgemeinerung für längeren Klartext:

m	$=$	01100011	
k	$=$	11101001	(generiert)
$k \oplus m$	$=$	c	$=$ 10001010
$k \oplus c$	$=$	m	$=$ 01100011

Beispiel

m	$=$	ONETIMEPAD
k	$=$	DJPDOTCJHX
c	$=$	RWTWEGYHA
c	$=$	RWTWEGYHA
k	$=$	FIJMWESRVY (geraten)
m	$=$	MOKKABOHNE

$\Rightarrow$ Man kann über den Chiffretext nicht auf den Klartext kommen.

6.5 Stromchiffren

Idee: Verschlüssele wie beim One-Time-Pad, aber nicht mit Zufallsbits, sondern Pseudozufallsbits aus einem Algorithmus.

Siehe Abbildung 6.3.

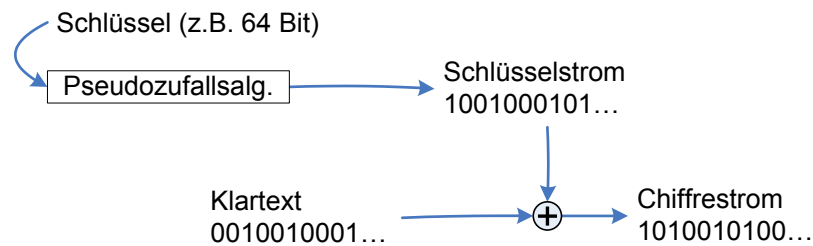


Abbildung 6.3: Stromchiffren

Java-Zufallsgeneratoren

- `java.util`:
 - Linearer Kongruenzgenerator:

$$x_{i+1} = (ax_i + b) \bmod m$$

$$x_0 = \text{"initial seed"}$$
 - Wenn x_i bekannt ist, dann weiss man die Folge für alle nächsten generierten Zahlen.
- `java.util`:
 - Bietet die Methode `secureRandom()`

6.5.1 RC4

- Developed 1987
- Can be easily implemented in software
- Encrypts cleartext bitwise
- Used in Netscape Browser and for WLAN encryption

RC4 Key Schedule

- Starts with an array `S` of numbers: 0 .. 255
- `S` forms internal state of the cipher
- given a key `k` of length `l` bytes

Listing 6.1: RC4 Key Schedule Pseudocode

```

1 for i = 0 to 255 do
2   {S[i] = i}
3
4 j = 0
5 for i = 0 to 255 do
6   {
7     j = (j + S[i] + k[i mod l]) (mod 256)
8     swap (S[i], S[j])
9   }
  
```

RC4 Stream Generation

- Encryption continues shuffling array values

Listing 6.2: RC4 Stream Generation Pseudocode

```

1 i = j = 0
2 for each message byte Mk:
3 {
4   i = (i + 1) (mod 256)
5   j = (j + S[i]) (mod 256)
6   swap(S[i], S[j])
7   t = (S[i] + S[j]) (mod 256)
8   Ck = Mk XOR S[t]
9 }

```

6.5.2 A5

- Used for voice encryption in GSM cellphones (150 million customers in europe)
- Hardware-oriented:
 - Cheap
 - Fast
 - Based on Linear Feedback Shift Registers (LFSR)

Linear Shift Registers

Deutsch: Lineare Schiebe-Register

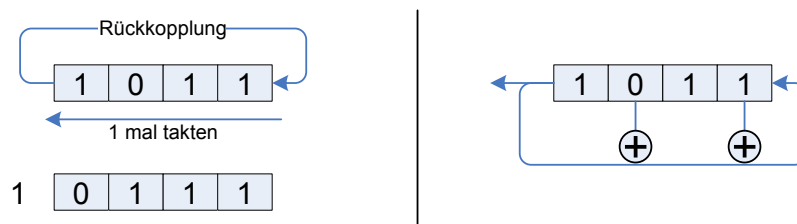


Abbildung 6.4: Linear Shift Register

Damit nach vier Takten nicht wieder die gleichen Outputbits ausgegeben werden, wird LFSR angewandt (siehe Abbildung 6.4)

Mit LFSR: $2^4 = 24$ Zustände = Maximale Periode.

A5 Stream Generation

Siehe Abbildung 6.5

Die Nichtlinearität ist gewährleistet durch C1, C2 und C3.

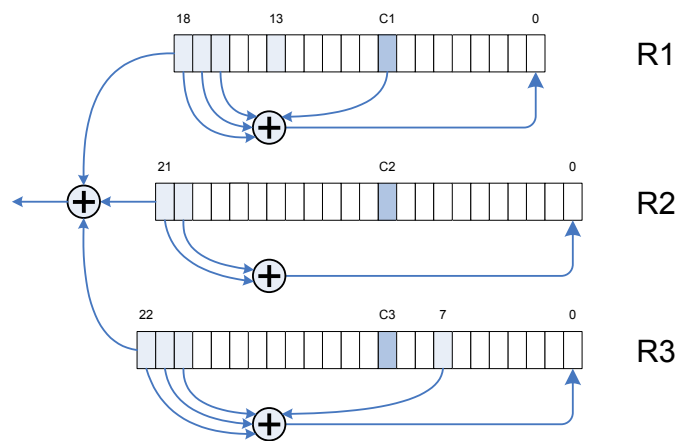


Abbildung 6.5: A5 Stream Generation

6.5.3 Einsatz von Stromchiffren

Einsatz ist hier z.B. bei GSM sinnvoll: Dadurch dass Stromchiffren bitweise verbinden, kann nur ein Bit verloren gehen, wenn bei einem Übertragungsfehler ein Bit schlecht übertragen wird.

6.6 Blockchiffren

Definition



Ein Kryptosystem (M, C, F) heißt Blockchiffre mit Blocklänge b , falls $M = C = (\sum^b)^N$

Bemerkung: Typische Werte: $M = C = (\{0, 1\}^{64})^N$ (Heute: 128 Bit)

Beachte: Jede polyalphabetische Substitutionschiffre ist Blockchiffre (z.B. Vigenère) aber nicht umgekehrt!

6.6.1 Designprinzipien für Blockchiffren (SHANNON)

- **Konfusion**

Der Chiffretext sollte von den Klartextbits in so komplizierter Weise abhängen, dass der Angreifer aus dem Chiffretext keine Info über den Klartext erhält.

- **Diffusion**

Jedes Klartextbit sollte möglichst viele Chiffrebits beeinflussen.

6.6.2 Feistel Chiffren

- Horst Feistel (IBM, ca. 1975):
- Teile Klartextblock B auf in 2 Hälften L_0, R_0
- Leite vom Schlüssel k genau r Teilschlüssel (Rundenschlüssel) k_i ab ($1 \leq i \leq r$).

Verschlüsseln und Entschlüsselung bei Feistel-Chiffren

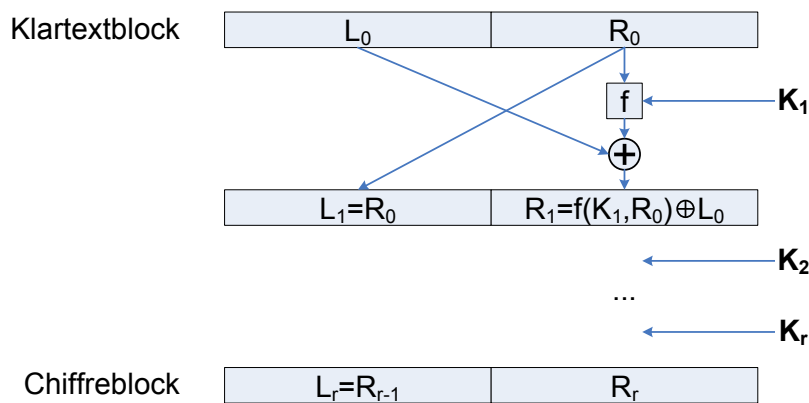


Abbildung 6.6: Designpattern zur Verschlüsselung bei Feistel-Chiffren

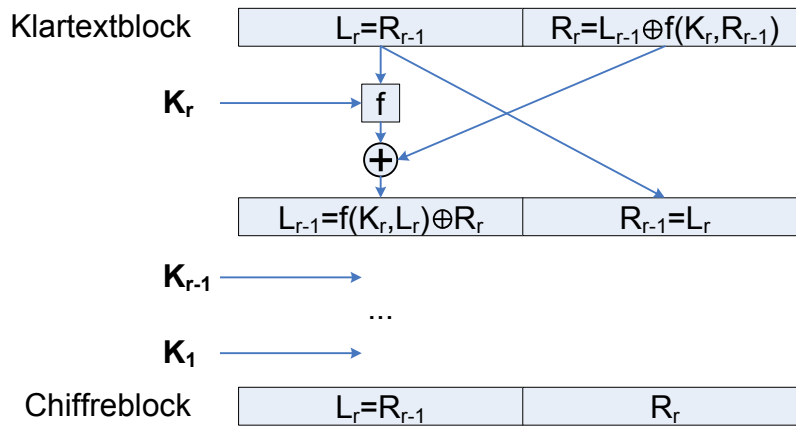


Abbildung 6.7: Designpattern zur Entschlüsselung bei Feistel-Chiffren

Bemerkung zum Entschlüsseln:

- Es wird nur ein Rundenschlüssel K_r gebraucht.
- Die Rundenfunktion f muss nicht Umkehrfunktion sein und kann beliebig gewählt werden.

6.6.3 Data Encryption Standard: DES

Paradebeispiel für Feistelchiffren

- Blocklänge 64 Bit
- Schlüssel: 64 Bit, davon 8 "Paritätsbits" also 56 Bit effektiv
- 16 Feistelrunden
- immer noch im Bankbereich stark vertreten, z.B. früher bei Prüfung der EC-Karten-PIN:
 - lese Nutzerdaten auf Magnetstreifen
 - überführe in 64-Bit Block
 - verschlüssele diesen mit DES und 56 Bit Bankschlüsse
 - Verwende nur erste 2 Byte des Ergebnisses, bestimme daraus die PIN-Ziffer

Triple-DES (3DES)

- Wird heute angewendet
- 3DES ist definiert durch:
 - $3DES(m) = DES_{K_1}(DES_{K_2}^{-1}(DES_{K_3}(m)))$
 - Weil wir 3 Schlüssel haben, spricht man auch vom *Three-Key-Triple-DES* (3 · 56 Bit keys, effektiv aber nur 112 Bits!)
 - Es gibt auch die 2-Schlüssel-Variante *Two-Key-Triple-DES* (2 · 56 = 112 Bit keys)
 $3DES(m) = DES_{K_1}(DES_{K_2}^{-1}(DES_{K_1}(m)))$
 - DES^{-1} entspricht dem *DES* im Entschlüsselungsmodus. Grund dafür: Abwärtskompatibilität zum einfachem *DES*.
 - *Three-Key-Triple-DES* und *Two-Key-Triple-DES* haben effektiv beide nur einen 112 Bit-Schlüsselraum.

Three-Key-Triple-DES

Warum kein 2DES?

$$2DES(m) = DES_{K_2}^{-1}(DES_{K_1}(m))$$

1. Keine Abwärtskompatibilität
2. Known-Plaintext-Attacke: (Auch *Meet-in-the-Middle*)

Meet-in-the-Middle

Angreifer kennt m und $c = 2DES(m)$

$$\begin{array}{lcl} c & = & DES_{K_2}^{-1}(DES_{K_1}(m)) \quad |DES_{K_2}() \\ DES_{K_2}(c) & = & DES_{K_1}(m) \end{array}$$

Nun kann der Angreifer alle DES durchprobieren mit allen 2^{56} möglichen k_1 und k_2 .
Wenn die Gleichung aufgeht, hat man die beiden Schlüssel k_1 und k_2 .

$\Rightarrow$ Aufwand: 2^{57}

6.6.4 More Feistel-Ciphers**• CAST**

- Developed in 1993 by Adams & Tavares
- 64 Bit Blocklength, 64 Bit Keylength
- 8 Rounds
- Patented

• Blowfish

- Developed in 1994 by B. Schneier
- 64 Bit Blocklength, Keylength variable (up to 448 Bits)
- 16 Rounds
- Fast, but large memory requirements
- Not patented

6.6.5 Non-Feistel Blockciphers

- **International Data Encryption Algorithm (IDEA)**
 - Developed in 1990 by J. Massey (ETH Zürich)
 - 64 Bit Blocklength, 128 Bit Keylength
 - 8 Rounds
 - Patented (but free for non-commercial use)
- **Advanced Encryption Standard (AES)**
 - Replacement for DES
 - Was found in a "beauty contest"

6.6.6 Advanced Encryption Standard: AES

AES-Geschichte

- 2.1.1997 Ankündigung der Entwicklung eines DES-Nachfolgers durch NIST (National Institute of Standards and Technology, USA)
- 12.9.1997 Formaler "Call for algorithms". Anforderungen:
 - Blockgröße (mindestens) 128 bit,
 - Schlüssellänge variabel 128, 192 oder 256 Bit
 - "As secure as Triple-DES, but much more efficient."
 - Referenz-Implementationen in C und Java
- 20.8.1998 Erste AES Candidate Conference (AES1). Die fünfzehn Kandidaten der ersten Runde werden von ihren Entwicklern vorgestellt.
- 15.4.1999 AES2.
 - Vorstellung der Analyse-Resultate für die 15 eingereichten Algorithmen.
 - Papers on Cryptoanalytic Attacks, Performance on Smart Cards and PCs.
- August 1999: Die fünf Kandidaten des Finales werden bekannt gegeben: MARS, RC6, Rijndael, Serpent, Twofish
- 13.4.2000 AES3.
 - Öffentliche Diskussion der bisherigen Analyse-Resultate der fünf Finalisten
 - Performance-Analysen für FPGAs und ASICs
- 15.5.2000 Abgabeschluss für öffentliche Kommentare
- 2.10.2000 Bekanntgabe des Gewinners: Rijndael (Entwickler: V. Rijmen & J. Daemen, Uni Leuven (Belgien))

- Juni 2001: AES = Rijndael wird offizieller Federal Information Processing Standard (FIPS)
- ISO (International Standards Organisation), IETF (Internet Engineering Task Force) und IEEE (Institute of Electrical and Electronic Engineers) übernehmen AES als Standard.

Auswahlkriterien

- Security
- Costs
 - Implementation Costs
 - Execution Costs
- Versatility: Performance on
 - smart cards
 - PCs and Servers
 - dedicated hardware
- Key Agility
- Simplicity
 - Einfache Chiffre führt zu weniger Fehler in der Implementation
 - Zudem muss man sie transparent halten

DES versus AES

Siehe Tabelle [6.1](#).

	DES	AES
Schlüssellänge	56 Bit	128/192/256 Bit
Blockgröße	64 Bit	128 Bit
Anzahl der Runden	16	10/12/14 (abhängig von Schlüssellänge)
Geschwindigkeit	80 Clock-cycles/byte	20 Clock-cycles/byte

Tabelle 6.1: DES-AES-Vergleich

More Info on AES

- <http://www.esat.kuleuven.ac.be/rijmen/rijndael/>
- <http://csrc.nist.gov/encryption/aes/>
- <http://www.esat.kuleuven.ac.be/rijmen/rijndael/>
(Official Rijndael Fanpage with all pronunciations)
- J. Daemen, V. Rijmen:
The Design of RIJNDAEL, Springer-Verlag (2002)

6.6.7 Betriebsmodi (Modes of Operation) für Blockchiffren

Problem: Verschlüssele viele Klartextblöcke $B_1, B_2, \dots, B_n$ mit gleicher Blockchiffre f_k

- **Electronic Codebook Mode (ECB)** (Einfachste Lösung)

- Encrypt: $C(B_i) = f_k(B_i)$
- Decrypt: $B_i = f_k^{-1}(C_i)$

Problem: Muster in der Verteilung der Klartextblöcke bleiben sichtbar.

- **Cipher Block Chaining Mode (CBC)** (wird am häufigsten verwendet)

- Encrypt:
 - * $C(B_i) = f_k(C(B_{i-1}) \oplus B_i), i > 0$
 - * $C(B_1) = f_k(IV \oplus B_1), IV$ fester Initialisierungsvektor
- Decrypt: $B_i = f_k^{-1}(C(B_i)) \oplus C(B_{i-1})$

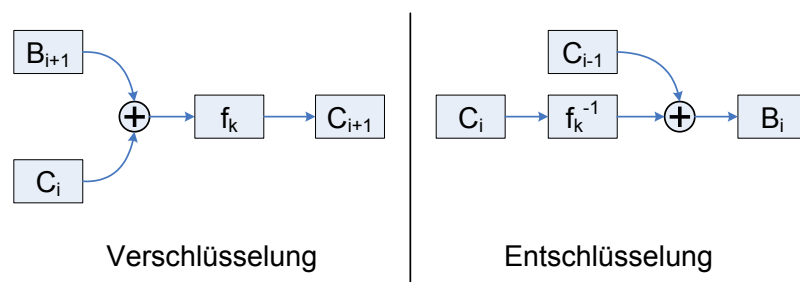


Abbildung 6.8: Ver- und Entschlüsselung bei CBC

- Nutze Blockchiffre als Schlüsselstromgenerator
- **Cipher Feedback Mode (CFB)**
 - Encrypt: $C_i = C(B_i) = B_i \oplus f_k(C_{i-1}), i > 0; \quad C_1 = C(B_1) = B_1 \oplus f_k(IV)$
 - Decrypt: $B_i = C(B_i) \oplus f_k(C_{i-1})$
- **Output Feedback Mode (OFB)**

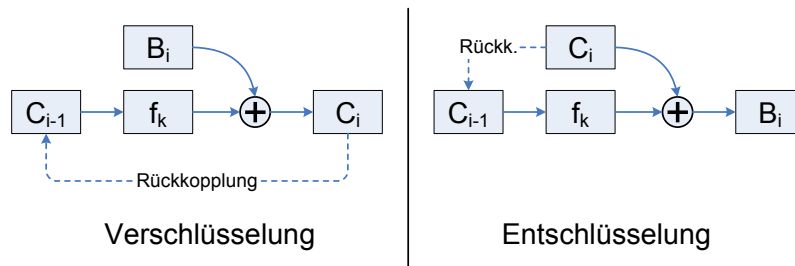


Abbildung 6.9: Ver- und Entschlüsselung bei CFB

- Encrypt: $C(B_i) = B_i \oplus f_k^i(IV)$
- Decrypt: $B_i = C(B_i) \oplus f_k^i(IV)$

- **Counter Mode (CTR)**

- Encrypt: $C(B_i) = B_i \oplus f_k(IV||i)$, $i > 0$
- Decrypt: $B_i = C(B_i) \oplus f_k(IV||i)$, $i > 0$

6.6.8 PKCS5 Padding

Because a block cipher works on units of a fixed size, but messages come in a variety of lengths, some modes (mainly CBC) require that the final block be padded before encryption.

Example: Three bytes missing in the last plaintext block (Siehe Abbildung 6.10)



Abbildung 6.10: PKCS5 Padding

6.7 Java Cryptographic Architecture (JCA)

Siehe Abbildung 6.11.

Example: Getting an Instance of the DES Algorithm

Listing 6.3: Getting an Instance of the DES Algorithm

```

1 Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
2
3 /*
4  getInstance() is a so-called factory method.
5  It is also possible to specify a cryptographic service provider (CSP).
6  */

```

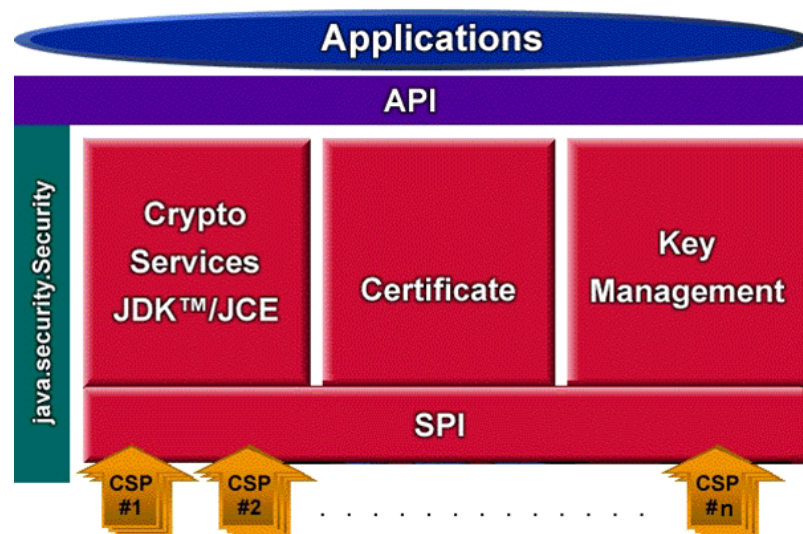


Abbildung 6.11: Java Cryptographic Architecture

More Providers

- <http://www.cryptix.org>
- <http://jce.iaik.tugraz.at>

More Info

- J. Knudsen: Java Cryptography, O'Reilly

6.8 Message Authentication Codes (MACS)

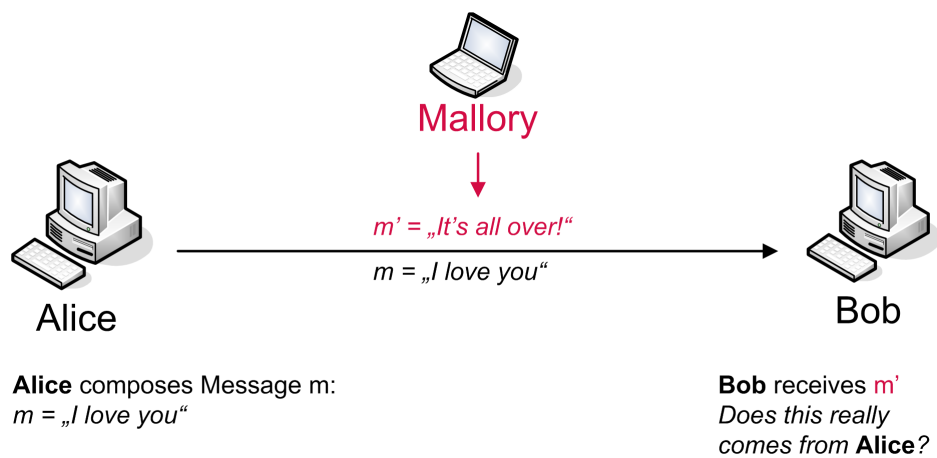


Abbildung 6.12: Manipulation einer Nachricht

- Motivation: Schutz vor
 - unbefugter Manipulation von Daten (Integritätsschutz)
 - unbefugten Senden von Nachrichten im Namen anderer (Nachrichtenauthentifikation)
- Idee:
 - Berechne kryptografische Prüfsumme (MAC) mit folgenden Eigenschaften:
 - * Nicht fälschbar
 - * Ändert sich bei kleinsten Veränderungen der Nachricht

6.8.1 Allgemeiner Ablauf

Siehe Abbildung 6.13.

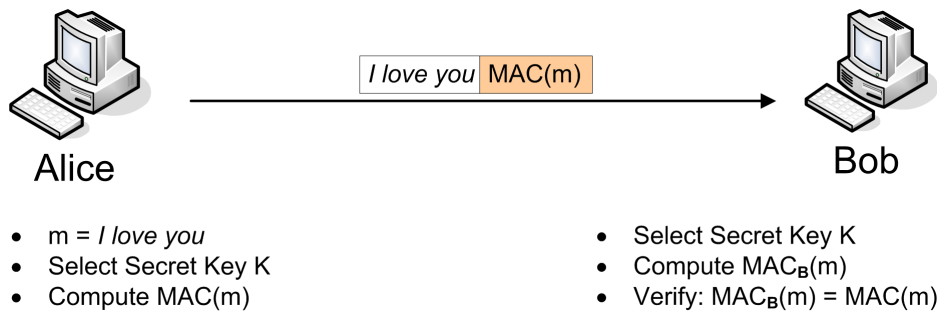


Abbildung 6.13: MACs: Allgemeiner Ablauf

6.8.2 Why not encrypt?

- Encrypted messages may be changed without being noticed
- Sometimes confidentiality is not wanted
- Encryption may cause unnecessary overhead

MACS vs Encryption

MAC: Integritätsschutz
 Verschlüsselung: Vertraulichkeit

Definition

Eine Funktion $hash : \{0, 1\}^* \rightarrow \{0, 1\}^n$ heisst *Hashfunktion*.
 (* = String beliebiger Länge; n = String fester Länge. Z.B. $n = 128$ bei MD5 oder $n = 160$ bei SHA-1)



Hashfunktionen

Beispiel

$$\text{hash}(x_0, x_1, \dots) = \bigoplus_i x_i \quad (n = 1)$$

$$\left. \begin{array}{l} \text{hash}(0010) = 1 \\ \text{hash}(1000) = 1 \end{array} \right\} \text{Kollision}$$

Definition



$$S_1, S_2 \in \{0, 1\}^* \text{ mit } S_1 \neq S_2$$

Gilt $\text{hash}(S_1) = \text{hash}(S_2)$, spricht man von *Kollision*.

Kollisionen

6.8.3 1. Methode zur MAC-Bildung

m Nachricht, k geheimer Schlüssel, f Blockchiffre mit Blocklänge n ,

$$\text{hash} : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

$$\Rightarrow \text{MAC}(m) = f_k(\text{hash}(m))$$

Gelingt es dem Angreifer zu m ein $\tilde{m}$ zu finden mit $\text{hash}(m) = \text{hash}(\tilde{m})$, gilt $\text{MAC}(m) = \text{MAC}(\tilde{m})$.

Definition



Ist es "schwer", zu einem *gegebenen* $x \in \{0, 1\}^*$ ein $y \in \{0, 1\}^*$ zu finden mit $\text{hash}(x) = \text{hash}(y)$, so heisst hash *schwach Kollisionsresistent*.

schwach
Kollisionsresistent

Beispiel

$$\text{hash}(x_0, x_1, \dots) = x_i \cdot \left(\bigoplus_i x_i \cdot \bigoplus_i x_{i+1} \right)$$

$$\text{hash}(111101) = 1 \cdot (1 \cdot 1 \oplus 1 \cdot 1 \oplus 1 \cdot 1 \oplus 1 \cdot 0 \oplus 0 \cdot 1) = 1 \cdot (1) = 1$$

Aufgabe:

- (i) Finde x mit $\text{hash}(x) = \text{hash}(111101) = 1$
- (ii) Finde x_1, x_2 mit $\text{hash}(x_1) = \text{hash}(x_2) \leftarrow$ Einfacher als (i)!

Definition



$$\text{hash} : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

ist es "schwer", *irgendwelche* $x, y \in \{0, 1\}^*$ zu finden mit $\text{hash}(x) = \text{hash}(y)$, heisst hash *stark Kollisionsresistent*.

stark Kollisionsresistent

6.8.4 2. Methode zur MAC-Bildung

1. Mache Nachricht m von Schlüssel k abhängig.
2. Hashe Nachricht gemeinsam mit k .

⇒ Falls Hash-Bildung rückgängig gemacht werden kann, kennt Angreifer die schlüsselabhängige Nachricht ⇒ Schlüssel k !

Definition

$hash : \{0, 1\}^* \rightarrow \{0, 1\}^n$

Falls es zu gegebenen $y \in \{0, 1\}^n$ "schwer" ist, ein x zu finden mit $y = hash(x)$, dann hat $hash$ die *One-Way-Eigenschaft*.



One-Way-Eigenschaft

Beispiel

$HMAC(m) = hash(k) || hash(k || m)$ (Vgl. IPSec $hash$ = MD5 oder SHA-1)

6.8.5 Common Hash-Functions

- MD5:
 - 128 Bit Hash
 - Collisions found!
- SHA-1:
 - 160 Bit Hash
 - North American Standard Algorithm
- RIPEMD-160:
 - 160 Bit Hash
 - Developed in Joint European Research Project RIPE

6.9 Schlüssellängen bei symmetrischen Kryptoverfahren

- 56 Bit DES Schlüssel können in < 24 h gebrochen werden (Stand 1999)
 - Ca. 100000 Internet-PCs
 - Einzelmaschine ("Deep Crack") brauchte 1998 noch 56 h
- Heutiger Stand: 128 Bit erscheinen "mittelfristig" sicher
- S. Keylength-Paper auf MI-Server

Kapitel 7

Asymmetrische Kryptographie

7.1 Allgemeines

- Bisher: *symmetrische* Verfahren:
Sender + Empfänger benutzen den gleichen Schlüssel zum Ver- und Entschlüsseln
- Jetzt: *asymmetrische* Verfahren:
Sender benutzt zum Verschlüsseln anderen Schlüssel als der Empfänger zum Entschlüsseln.
- Der Verschlüsselungsschlüssel (Public Key) ist öffentlich zugänglich, deshalb auch *Public-Key-Verfahren*.

Public-Key-Verfahren

Public-Key-Kryptosystem

M Klartextmenge, C Chiffretextmenge.

Gegeben seien:

- Verschlüsselungsfunktionen $E = \{E_{PK_1}, E_{PK_2}, \dots, E_{PK_n}\}$ mit $E_{PK_i} : M \rightarrow C$
- Entschlüsselungsfunktionen $D = \{D_{SK_1}, D_{SK_2}, \dots, D_{SK_n}\}$ mit $D_{SK_i} : C \rightarrow M$
- Menge von Schlüsselpaaren $K = \{(PK_1, SK_1), (PK_2, SK_2), \dots, (PK_n, SK_n)\}$



Dann heißt (M, C, E, D, K) *Public-Key-Kryptosystem*, falls für alle $m \in M$ gilt:

$$D_{SK_i}(E_{PK_i}(m)) = m$$

Public-Key-Kryptosysteme

Allgemeines Szenario

Siehe Abbildung 7.1.

7.2 Das RSA-Verfahren

RSA = Rivest / Shamir / Adleman (1977) (Siehe Bild 7.2)

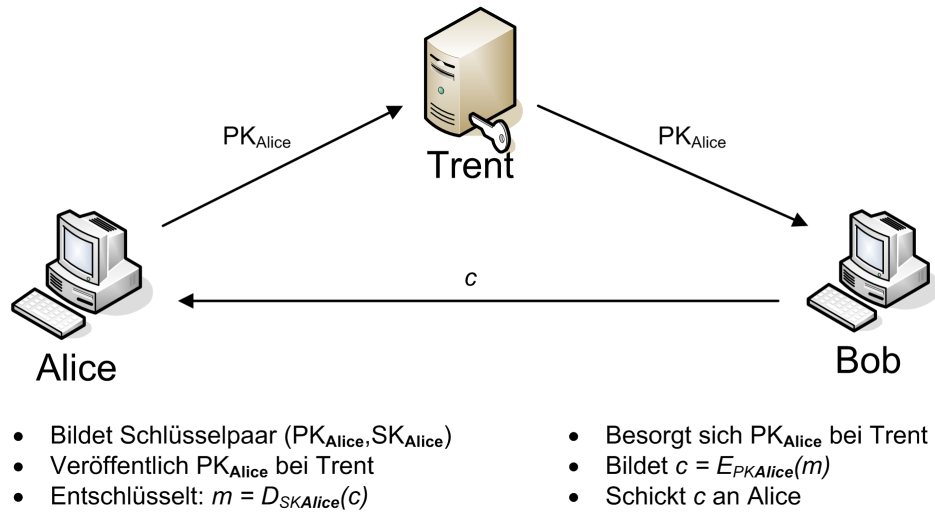


Abbildung 7.1: MACs: Public Key Kryptosystem: Allgemeiner Ablauf



Abbildung 7.2: Adi Shamir, Ron Rivest, Len Adleman

7.2.1 Schlüsselerzeugung

1. Alice wählt 2 große Primzahlen p, q und bildet
 $n = p \cdot q, \phi(n) = (p-1)(q-1)$
2. Alice wählt Zahl $e \in \mathbb{N}$ mit $1 < e < \phi(n)$, die zu $\phi(n)$ teilerfremd ist.
3. Alice berechnet Zahl $d \in \mathbb{N}$ mit $1 < d < \phi(n)$ und $d \cdot e \bmod \phi(n) = 1$
(also $d = e^{-1}$ in $Z_{\phi(n)}$)
4. Alice veröffentlicht seinen Public Key $PK_{Alice} = (e, n)$. Sein Secret Key $SK_{Alice} = d$ hält sie geheim.

p, q	geheim gehalten. Für Faktorisierung nötig.
n	public module (bekannt)
e	Encryption Exponent

7.2.2 Exkurs: Primzahlen und modulare Arithmetik

Bezeichnungen

- b **teilt** a (oder auch b ist **Teiler** von a) $\Leftrightarrow$ Es gibt ein m mit $a = m \cdot b$.
In Zeichen: $a|b$ Teiler
- $p > 1$ heisst **Primzahl** $\Leftrightarrow$ Die einzigen Teiler von p sind 1 und p . Primzahl
- c heisst **grösster gemeinsamer Teiler** (ggT) von a und $b \Leftrightarrow$ grösster gemeinsamer Teiler
 - (i) c ist Teiler von a und b .
 - (ii) Jeder andere gemeinsame Teiler von a und b teilt c .
- a und b heissen **teilerfremd** $\Leftrightarrow ggT(a, b) = 1$ teilerfremd
- Die Anzahl der Zahlen $1 \leq x < n$, die zu n teilerfremd sind, heisst $\phi(n)$.
 ϕ heisst Euler'sche ϕ -Funktion
Beispiel: $\phi(12) = 4, \phi(7) = 6$

Satz

1. p Primzahl $\Rightarrow \phi(p) = p - 1$
2. p, q Primzahl $\Rightarrow \phi(p, q) = (p-1)(q-1), p \neq q$



Beweis

1. klar ;-)
2. Es gibt $(p \cdot q) - 1$ Zahlen $< p \cdot q$
Nicht teilerfremd zu $p \cdot q$ sind:
 - $p, 2p, 3p, \dots, (q-1) \cdot p = (q-1)$ Stück

- $q, 2q, 3q, \dots, (p-1) \cdot q = (p-1)$ Stück

$$\Rightarrow \phi(p \cdot q) = p \cdot q - 1 - (q-1) - (p-1) = p \cdot q - q - p + 1 = (p-1)(q-1)$$

(Siehe Punkt 1 aus Abschnitt 7.2.1)

Bemerkung: $n = p^2 \Rightarrow \phi(p^2) = p^2 - 1 - (p-1) = p^2 - p = n - p$

Definition



Lässt a bei Division durch n den Rest r ($r < n$), schreibt man $a \bmod n = r$

Bemerkung: $a \bmod n = r \Leftrightarrow a = k \cdot n + r \Leftrightarrow n | (a - r)$

Satz: Euklid'scher Algorithmus



$a, b \in \mathbb{N}$ mit $a > b$

$ggT(a, b)$ lässt sich mit folgendem Algorithmus bestimmen:

Listing 7.1: Euklid'scher Algorithmus Pseudocode

```

1 while (b != 0)
2 {
3   r := a mod b
4   a := b
5   b := r
6 }
7
8 ggT = a

```

Beispiel

$a = 42, b = 15$

a	b	r	
42	$\bmod 15$	$=$	12
15	$\bmod 12$	$=$	3
12	$\bmod 3$	$=$	0
3	$\bmod 0$		$\leftarrow b = 0 \Rightarrow$ Abbruch der Schleife

$$\Rightarrow ggT(42, 15) = 3$$

Satz: Erweiterter Euklid'scher Algorithmus



Es sei $c = ggT(a, b)$. Dann gibt es Zahlen $x, y \in \mathbb{Z}$ mit $c = xa + yb$

Beispiel

$3 = ggT(42, 15)$. Suche x, y mit $3 = x \cdot 42 + y \cdot 15$

$$42 \bmod 15 = 12 \Leftrightarrow 12 = 42 - 2 \cdot 15 \quad \text{Setze } (42 - 2 \cdot 15) \text{ in } 12$$

$$15 \bmod 12 = 3 \Leftrightarrow 3 = 15 - 1 \cdot 12$$

$$\Rightarrow 3 = 15 - 1(42 - 2 \cdot 15) = 3 \cdot 15 - 1 \cdot 42$$

$$\Rightarrow y = 3, x = -1$$

Anwendung des erweiterten Euklid auf RSA-Schlüsselerzeugung

Siehe Schritt 3 aus Abschnitt 7.2.1:

Alice sucht d mit $d \cdot e \bmod \phi(n) = 1$

$$\Leftrightarrow d \cdot e - 1 = k \cdot \phi(n)$$

$$\Leftrightarrow d \cdot e - k \cdot \phi(n) = 1 = ggT(e, \phi(n)) \quad (\text{nach Schritt 2})$$

$\Rightarrow$ Finde d mittels erweiterten Euklid, angewandt auf e (als a) und $\phi(n)$ (als b).

Bemerkung: Der Angreifer müsste $\phi(n)$ herausfinden um SK_{Alice} heraus zu finden.

Beispiel

$$p = 7, q = 13$$

$$1. n = p \cdot q = 91, \phi(n) = (p - 1) \cdot (q - 1) = 72$$

$$2. \text{ Wähle } e = 5$$

$$3. \text{ Suche } d \text{ mit } 5 \cdot d \bmod 72 = 1 \Leftrightarrow 5 \cdot d - k \cdot 72 = 1$$

Wende erweiterten Euklid auf 72 und 5 an:

$$72 \bmod 5 = 2 \Leftrightarrow 2 = 72 - 14 \cdot 5 \quad \text{Setze } (72 - 14 \cdot 5) \text{ in } 2$$

$$5 \bmod 2 = 1 \Leftrightarrow 1 = 5 - 2 \cdot 2$$

$$\Rightarrow 5 - 2 \cdot (72 - 14 \cdot 5) = 29 \cdot 5 - 2 \cdot 72$$

$$\Rightarrow d = 29$$

$$4. PK_{Alice} = (5, 91), SK_{Alice} = 29$$

7.2.3 Verschlüsselung

$$1. \text{ Bob besorgt sich } PK_{Alice} = (e, n)$$

$$2. \text{ Bob wählt Klartext } m \in M = 1, 2, \dots, n - 1$$

$$3. \text{ Bob bildet den Ciphertext } c = E_{PK_{Alice}}(m) = m^e \bmod n$$

$$4. \text{ Bob schickt } c \text{ an Alice}$$

Beispiel

Verschlüsse $m = 4$ mit $PK_{Alice} : c = 4^5 \bmod 91 = 2024 \bmod 91 = 23$

Sicherheit der Verschlüsselung

Das Problem:

- gegeben: $e, n, c = m^e \bmod n$
- gesucht: m

wird auch als RSA-Problem bezeichnet

7.2.4 Entschlüsselung

- Alice erhält die Nachricht m aus $m = D_{SK_{Alice}}(c) = c^d \bmod n$

Beispiel

Entschlüssele $c = 23$ mit $SK_{Alice} : m = 23^{29} \bmod 91$

Zerlege Exponenten in 2er Potenzen; $29 = 2^4 + 2^3 + 2^2 + 2^0$

$$\Rightarrow 23^{29} \bmod 91 = \left((23^{2^4} \bmod 91) \cdot (23^{2^3} \bmod 91) \cdot (23^{2^2} \bmod 91) \cdot (23 \bmod 91) \right) \bmod 91$$

$$23^{2^{i+1}} = 23^{2^i} \cdot 23^{2^i} = \left(23^{2^i} \right)^2$$

$$\left. \begin{array}{l} 23^{2^1} \bmod 91 = 529 \bmod 91 = 74 \\ 23^{2^2} \bmod 91 = 74^2 \bmod 91 = 16 \\ 23^{2^3} \bmod 91 = 16^2 \bmod 91 = 74 \\ 23^{2^4} \bmod 91 = 74^2 \bmod 91 = 16 \end{array} \right\} \Rightarrow 23^{29} \bmod 91 = (16 \cdot 74 \cdot 16 \cdot 23) \bmod 91 = 4$$

Bemerkung: e kann also so gewählt werden, dass die Binärdarstellung viele 0er hat, also Schneller gerechnet werden kann $\Rightarrow$ Verschlüsselung mit PK ist schneller als Entschlüsselung mit SK

7.2.5 Sicherheit des RSA-Verfahrens

- Beruht auf drei Säulen:
 1. Schwierigkeit, den Modul n zu faktorisieren
 2. Schwierigkeit des RSA-Problems
 3. Authentizität des Public Keys (e, n)
- Keine der Säulen darf wegbrechen!

Satz: Bestimmung von Anzahl möglicher Primzahlen

$\pi(n)$ = Anzahl Primzahlen $< n$

$\pi(n) \sim \frac{n}{\log(n)}$ ($\sim$ bedeutet Asymptotisch gleich)

**Beispiel**

$$2^{511} < n < 2^{513}$$

$$\left. \begin{array}{l} \pi(2^{511}) \sim \frac{2^{511}}{511} \\ \pi(2^{513}) \sim \frac{2^{513}}{513} \end{array} \right\} \Rightarrow \text{Es gibt } \frac{2^{513}}{513} - \frac{2^{511}}{511} = \frac{3 \cdot 2^{511}}{513} \text{ Primzahlen mit Bitlänge 512 Bit}$$

7.2.6 Beispiel

Zentraler Server teilt allen Teilnehmern gleichen Modul n , aber unterschiedliche e_i und d_i zu.

Problem: Ein Teilnehmer kennt sein d_i

$$\begin{aligned} \Rightarrow e_i \cdot d_i \bmod \phi(n) &= 1 \\ e_i \cdot d_i - 1 &= k \cdot \phi(n) \quad \text{für ein } k \end{aligned}$$

$$\phi(n) = (p-1)(q-1) = \underbrace{p \cdot q}_n - p - q + 1 \approx n - 2\sqrt{n}$$

$\Rightarrow k$ bekannt $\Rightarrow \phi(n)$ bekannt (also Faktorisierung bekannt) $\Rightarrow$ alle d_i bekannt.

7.3 Diskrete Logarithmen**Definition**

Sei p Primzahl, $g = p - 1$ sei *primitives Element mod p*, d.h. zu jedem $A \in \{1, 2, \dots, p-1\}$ existiert ein a , so dass $A = g^a \bmod p$.

Gilt also $A = g^a \bmod p$, heisst a *diskreter Logarithmus* von A zur Basis g (modulo p).

Die Aufgabe:

- gegeben: A, g, b
- gesucht: $d \leq p-1$ mit $A = g^d \bmod p$

heisst *diskretes Logarithmusproblem*.



primitives Element

diskreter Logarithmus

diskretes
Logarithmusproblem

Bemerkung: Diskretes Logarithmusproblem wird unlösbar wenn $p > 2^{1024}$

Beispiel

$$g = 5, p = 23, A = 17$$

Finde $x \in 1, 2, \dots, 22$ mit $5^x \bmod 23 = 17$

$$x = 1: 5^1 \bmod 23 = 5$$

$$x = 2: 5^2 \bmod 23 = 2$$

$$x = 3: 5^3 \bmod 23 = 10$$

$$x = 4: 5^4 \bmod 23 = 4$$

$$x = 5: 5^5 \bmod 23 = 20$$

$$x = 6: 5^6 \bmod 23 = 8$$

$$x = 7: 5^7 \bmod 23 = 17$$

$\Rightarrow x = 7$ ist diskreter Logarithmus von 17 zur Basis $\bmod 23$.

7.3.1 Diffie-Hellman Schlüsseltausch-Protokoll

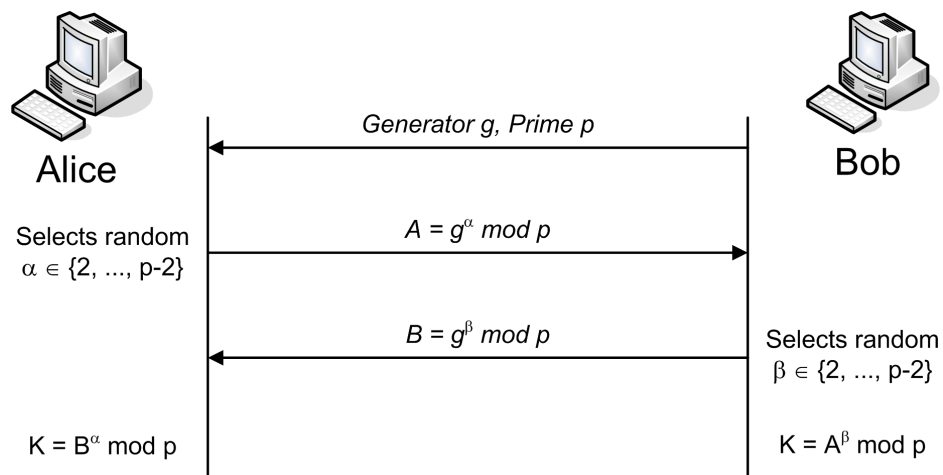


Abbildung 7.3: Diffie-Hellman Schlüsseltausch-Protokoll

1. Alice und Bob einigen sich öffentlich auf Primzahl p und primitives Element $g = p - 1$
2. Alice wählt $\alpha \in \{2, \dots, p - 2\}$ zufällig und bildet $A = g^\alpha \bmod p$
3. Bob wählt $\beta \in \{2, \dots, p - 2\}$ zufällig und bildet $B = g^\beta \bmod p$
4. Alice und Bob tauschen öffentlich A und B aus.
5. Alice und Bob besitzen gemeinsames Geheimnis:

$$\begin{aligned} K &= A^\beta \bmod p = g^{\alpha\beta} \bmod p \\ &= B^\alpha \bmod p = g^{\beta\alpha} \bmod p \end{aligned}$$

Siehe Abbildung 7.3.

Beispiel

$$p = 7, g = 3$$

Alice wählt $\alpha = 3$ und bestimmt $A = g^\alpha \bmod p = 6. \Rightarrow K = B^\alpha \bmod p = 6$

Bob wählt $\beta = 5$ und bestimmt $B = g^\beta \bmod p = 5. \Rightarrow K = A^\beta \bmod p = 6$

In beiden Fällen erhält man den Schlüssel $K = 6$.

Die Sicherheit beruht darauf, dass α und β geheim sind.

7.3.2 Man-in-the Middle Angriff auf Diffie-Hellman

Siehe Abbildung 7.4.

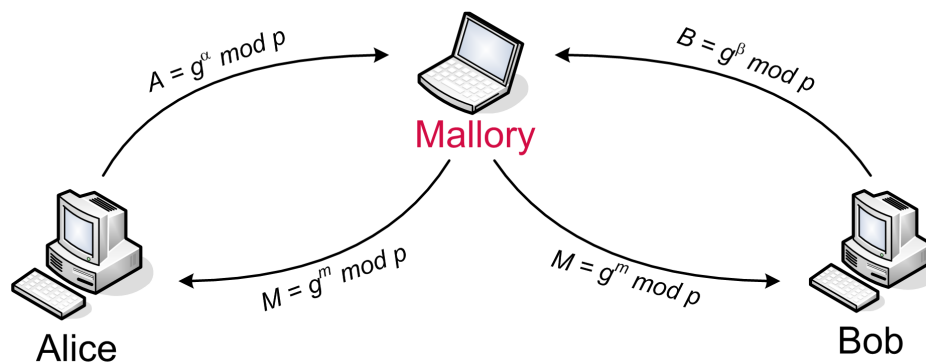


Abbildung 7.4: Man-in-the Middle Angriff auf Diffie-Hellman

Deshalb wichtig: Bei Diffie-Hellman ausgetauschte Nachrichten müssen authentisch sein! (gleiches Problem wie bei RSA)

7.3.3 Sicherheit des Diffie-Hellman-Verfahrens

- Beruht auf drei Säulen:
 1. Diskretes Logarithmus-Problem
 2. Authentizität der ausgetauschten Nachrichten
 3. Schwierigkeit des Diffie-Hellman-Problems:
 - a) Gegeben: $g, p, A = g^x \bmod p, B = g^y \bmod p$
 - b) Gesucht: $K = g^{xy} \bmod p$
- Keine der Säulen darf wegbrechen!

7.3.4 ElGamal-Verfahren

- Wandle DH-Protokoll um in asymmetrisches Kryptoverfahren.
- Idee:
 1. Alice veröffentlicht $g, p, A = g^\alpha \bmod p$ als ihren Public-Key (wobei $\alpha = SK_{Alice}$)
 2. Bob erzeugt $B = g^\beta \bmod p$ und bildet $k = A^\beta \bmod p$ (wobei $\beta = SK_{Bob}$)
 3. Bob schickt B und mit k verschlüsselte Nachricht m an Alice
 4. Alice kann aus B und α den Schlüssel k berechnen und die Nachricht entschlüsseln.

Bemerkung: Die ElGamal-Formeln wurden ausgelassen.

7.3.5 Berechnung Diskreter Logarithmen

- Gegeben: a, p, b .
- Berechne x mit $a^x \bmod p = b$.
- Aufzählen aller Potenzen $a^x \bmod p$
 - Zeitbedarf $O(p)$
- Baby-Step/Giant-Step Algorithmus
 - Zeitbedarf $O(\sqrt{p})$
 - Speicherbedarf $O(\sqrt{p})$

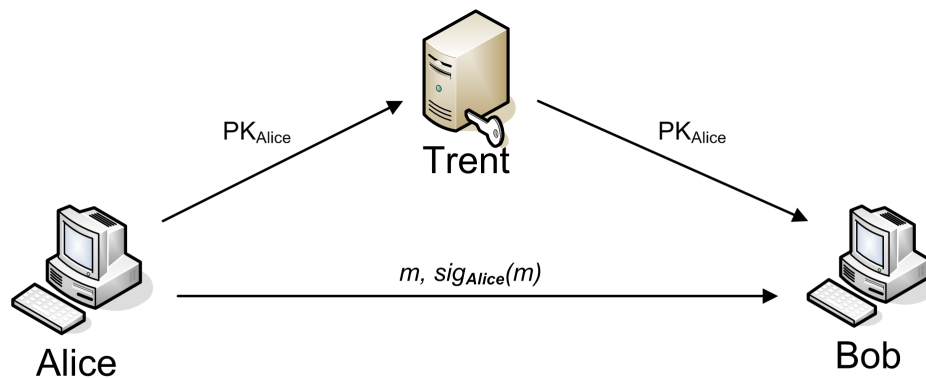
7.4 Digitale Signaturen und Zertifikate

- Anforderungen an Digitale Signaturen:
 - Fälschungssicher
 - Eindeutige Zuordnung der Unterschrift zur unterschreibenden Person
 - Untrennbarer Zusammenhang der Signatur mit dem unterschriebenen Dokument.
 - Nichtabstreitbarkeit der Unterschrift.
- Idee:
 - Verschlüssele Dokument mit Secret Key
 - Verifikation der Unterschrift mit Public Key
- Probleme:
 - Verschlüsseln langer Dokumente ist aufwändig

- Unterschrift ist genau so lang wie Dokument
- Dokument m muss im Klartextraum des asymmetrischen Verfahren liegen.
- Lösung:
 - Hashe Dokument vor der eigentlichen Signatur!

7.4.1 Allgemeines Szenario

Siehe Abbildung 7.5.



- Bildet Schlüsselpaar (PK_{Alice}, SK_{Alice})
- Veröffentlicht PK_{Alice} bei Trent
- Bildet: $sig_{Alice}(m) = E_{SK_{Alice}}(hash(m))$
- Besorgt sich PK_{Alice} bei Trent
- Bildet $hash(m)$
- Prüft, ob $hash(m) = D_{PK_{Alice}}(sig(m))$

Abbildung 7.5: Digitale Signaturen: Allgemeiner Ablauf

7.4.2 RSA-Signaturen

- Voraussetzungen:
 - $PK_{Alice} = (e, n), SK_{Alice} = d$
 - m die zu signierende Nachricht
 - h eine Hashfunktion
- Signaturerzeugung:
 - $sig_{Alice}(m) = (h(m))^d \bmod n$
- Signaturverifikation:
 - Bob erhält $(m, sig_{Alice}(m))$
 - Bob bildet $h(m)$ und $(sig_{Alice}(m))^e \bmod n$
 - Falls beides übereinstimmt, ist die Signatur verifiziert.
Falls beides *nicht* übereinstimmt, wurde entweder die Nachricht nicht signiert, oder sie wurde auf dem Weg verändert.

Beispiel

$$p = 3, q = 11 \Rightarrow \dots \Rightarrow PK_{Alice} = (7, 33), SK_{Alice} = d = 3$$

m sei Nachricht mit $hash(m) = 6 \Rightarrow sig_{Alice}(m) = 6^3 \bmod 33 = 18$

- Schicke $(m, 18)$ an Bob.
- Bob bildet $hash(m) = 6$ und prüft: $18^7 \bmod 33 = 6$

7.4.3 Signaturen auf ElGamal Basis

- Public Key: $PK_{Alice} = (p, g, A)$
- Secret Key: $SK_{Alice} = \alpha \in \{1, \dots, p-2\}$
- Signieren von $m \in \{1, \dots, p-2\}$:
 - Alice wählt Zufallszahl $k \in \{1, \dots, p-2\}$ teilerfremd zu $(p-1)$ und bildet $a = g^k \bmod p$
 - Alice berechnet b mit $(\alpha a + kb) \bmod (p-1) = m$
 - $sig_{Alice} = (a, b)$
- Verifikation:
 - Bob prüft ob $g^m \bmod p = A^a a^b \bmod p$

7.4.4 Zertifikate

- Bestätigung, dass ein Public Key zu einer Person gehört
- digital signiert von speziellen Dienstleistern oder Behörden
- **X.509** Zertifikate:
 - ITU (International Telecommunications Union) Standard
 - X. 509 Zertifikate enthalten mindestens folgende Elemente:
 - * Version (aktuell: V3)
 - * Seriennummer
 - * Signatur-Algorithmus (der zum Signieren des Zert. benutzte Algorithmus)
 - * Issuer Name (Name der ausstellenden Certificate Authority)
 - * Gültigkeitsperiode
 - * Subject Name (Inhaber des Zertifikats)
 - * Subject Public Key Info (Public Key des Inhabers und Public Key Algorithmus)

- * Signature Value (Digitale Signatur der ausstellenden CA über das gesamte Zertifikat)

7.4.5 Aufgaben von Certificate Authorities (CAs)

- Registrierung von Nutzern
- Schlüsselerzeugung
- Zertifizierung von Public Keys
- Archivierung öffentlicher Schlüssel (zur Verifizierung alter Signaturen)
- Verzeichnisdienst
- Rückruf von Zertifikaten
- Betreiben einer Certificate Revocation List (CRL)

7.4.6 Certificate Policy und Certification Practice Statement

- A *Certificate Policy*, as defined in X.509, is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.
- A *Certification Practice Statement (CPS)* is a statement of the practices that a CA employs in managing the certificates that it issues.

Certificate Policy

Certification Practice
Statement

7.5 Authentifikationsprotokolle

What's a "cryptographic protocol"?

A *cryptographic protocol* consists of a sequence of steps precisely specifying the actions required of two or more parties to achieve a specific security objective (here: entity authentication)

cryptographic protocol

Actors in a cryptographic protocol

Alice: First participant. Normally initiates protocol

Bob: Second participant. Normally responds to Alice

Eve: Passive (eavesdropping) attacker

Mallory: Active (malicious) attacker

Trent: Third Party trustworthy for Alice and Bob.

7.5.1 Password-Based Authentication

- Basic Scheme:
 - Alice and Bob agree on a shared secret (password)
 - Later on, if Alice wants to authenticate herself, she gives her name and password to Bob.
 - Bob checks if password is correct

Attacks on Passwords

- Eavesdropping (Password belauschen)
- Stealing password files
- Dictionary Attacks
- Virus Attacks
- Brute Force

7.5.2 Challenge-Response Protocols

Basic Challenge-Response Schemes

- Bob sends Random Number (*challenge*) to Alice
- Alice uses a secret K (shared symmetric key or private asymmetric key) to compute *response*
- Alice sends response to Bob
- Bob uses shared key or Alice's public key to verify response

Advantage: No need to send secret over insecure channel.

Siehe Abbildung 7.6.

Example: HTTP Digest Authentication ([RFC 2617](#))

Siehe Abbildung 7.7.

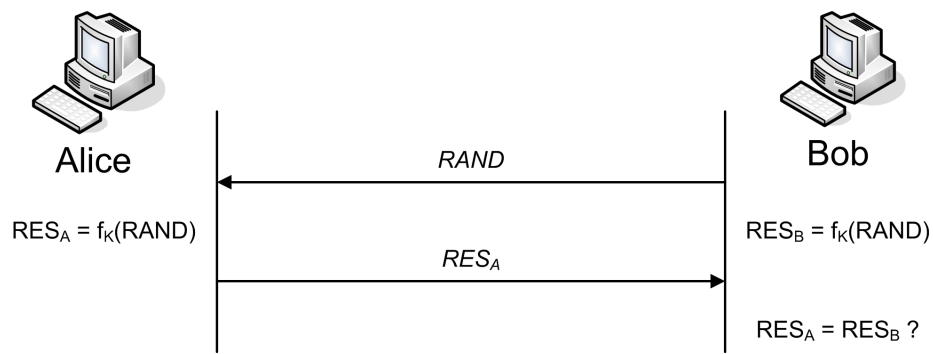


Abbildung 7.6: Basic Challenge-Response

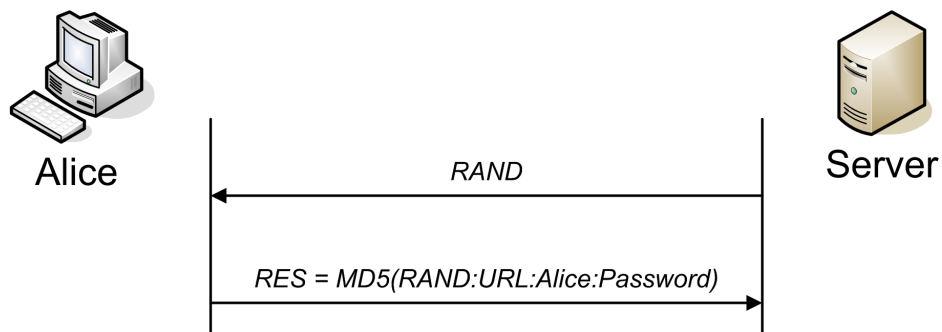


Abbildung 7.7: HTTP Digest Authentication

Asymmetric Challenge-Response Schemes

- Based on Public-Key Decryption or
- Digital Signatures
- Schemes based on Digital Signatures are non-repudiable!

Siehe Abbildung 7.8, 7.9, 7.10 und 7.11.

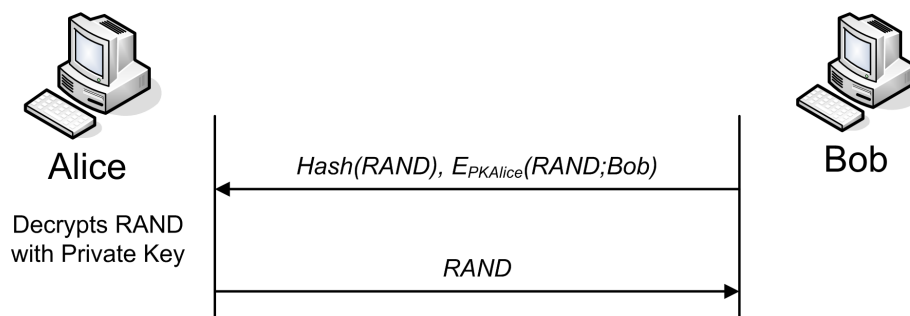


Abbildung 7.8: Challenge-Response Scheme based on PK Decryption

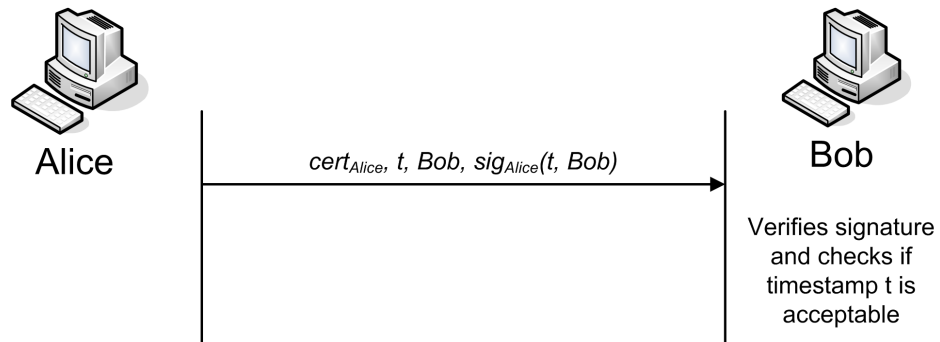


Abbildung 7.9: X.509 Challenge-Response based on Dig. Sig.: 1-Way with Timestamp

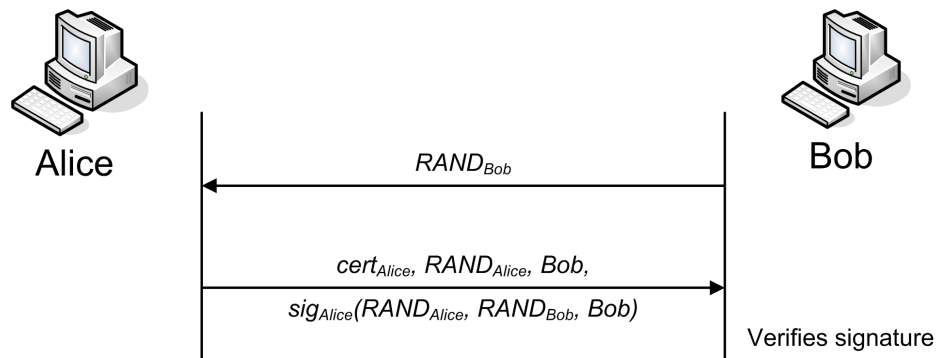


Abbildung 7.10: X.509 Challenge-Response based on Dig. Sig.: 2-Way with Rand. Challenge

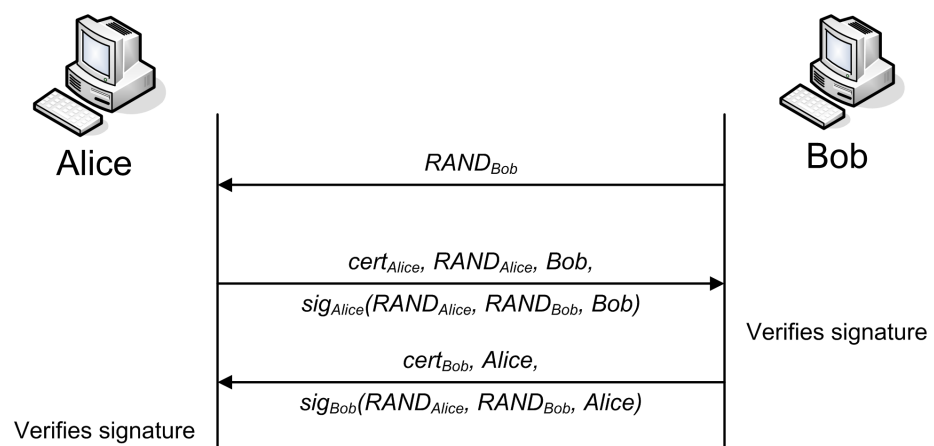


Abbildung 7.11: X.509 Challenge-Response based on Dig. Sig.: 3-Way Mutual Auth.

7.5.3 Kerberos

- Trusted Third Party Authentication Protocol for TCP/IP Networks
- Based on symmetric cryptography
- Goals:
 - Allow a person access to different machines/services on a network
 - Allow multiple access to a service without having to authenticate more than once (*Single-Sign-On*)
- Assumptions:
 - All person/machines X share their own secret keys T_X with the Trusted Third Party T.
 - All have a synchronized clock

Single-Sign-On

Basic Kerberos

Siehe Abbildung 7.8.

$$E_{K_{AB}}(\text{Alice}, t) = \text{Authenticator} \\ E_{TB}(t, L, K_{AB}, \text{Alice})$$

Abbildung 7.12: Basic Kerberos

Erläuterungen zur Abbildung 7.8:

1. Alice teilt Trent mit, dass sie sich mit Bob verbinden möchte.
2. E_{TA} : Verschlüsselt mit dem Geheimnis von Trent und Alice.
 E_{TB} : Sog. *Ticket Granting Ticket (TGT)*. Nur für Bob relevant. Für Alice ist es nur ein Token.
3. $E_{K_{AB}}$: Sog. *Authenticator*. Benutzt den Sessionkey K_{AB} um "Alice" und t zu verschlüsseln.
 Der TGT (E_{TB}) wird von Bob entpackt um den Sessionkey zu erhalten.
4. Optional kann sich Bob auch noch bei Alice authentifizieren (als Nachweis dass er den Sessionkey kennt)

Ticket Granting Ticket (TGT)

Authenticator

Bemerkung: Trent ist in diesem Beispiel nur verantwortlich für die Verteilung der Schlüssel.

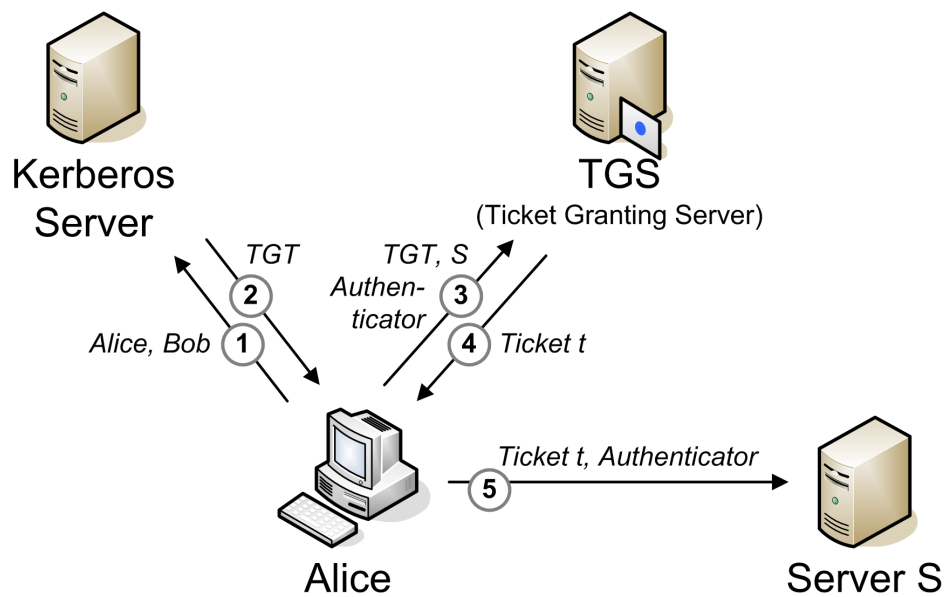


Abbildung 7.13: Kerberos in a network

Kerberos in a Network

Siehe Abbildung 7.13.

Erläuterungen zur Abbildung 7.13:

1. Alice braucht Ticket für Bob.
2. Alice bekommt den TGT (somit also auch den Sessionkey K_{ATGS}).
3. Alice teilt dem TGS unter anderem den gewünschten Server S und den Authenticator mit. (Der Authenticator beweist, dass Alice den Sessionkey kennt).
4. Alice erhält das Ticket t für den Server S. Dieser enthält unter anderem den mit K_{ATGS} verschlüsselten Sessionkey K_{AS} .
5. Alice meldet sich bei S mit dem Ticket t und dem aus K_{AS} gebildeten Authenticator an.

Attacks on Kerberos

- Replay of tickets/authenticators during ticket lifetime (up to 8 hours)
- Attack on trusted time in network
- Password guessing/dictionary attacks
- Compromising Kerberos client software to record passwords

Kapitel 8

Sicherheit auf der Anwendungsschicht

8.1 S/MIME

- Secure/Multipurpose Internet Mail Extension
- Standardisation Effort driven by RSA
- S/MIME will probably emerge as an Industry Standard
- Supported by older versions of Netscape Communicator (up to V4.7) and MS-Outlook
- More Info: <http://www.rsasecurity.com/standards/smime/index.html>

8.1.1 Simple Mail Transfer Protocol (SMTP)

- Defined by IETF in 1982 ([RFC 821](#))
- Simple ASCII-based protocol for exchange of e-mails between (UNIX -) servers with an SMTP demon running
- SMTP alternatives for PCs:
 - Post Office Protocol (POP, [RFC 1725](#))
 - Internet Message Access Protocol (IMAP, [RFC 1730](#))

SMTP Message Parts

- Envelope
- SMTP Header
- Body
 - Pure ASCII
 - Separated from header empty line
- SMTP Header Fields
 - Date
 - From
 - Subject

– To

SMTP Limitations

SMTP can not transmit, or has problems with:

- executable files, or other binary files (e.g. jpeg images)
- "national language" characters (non-ASCII)
- messages over a certain size
- lines longer than a certain length (72 to 254 characters)
- E-mail gateways may interpret ASCII control characters differently

8.1.2 Multipurpose Internet Mail Extension (MIME)

- Defined in RFCs [2045](#), [2046](#), [2047](#), [2048](#) and [2049](#)
- Addition of five new header fields in order to describe transferred content
- Definition of standardised content formats
- Standardised transfer encodings

Header fields in MIME

- **MIME-Version:** Must be "1.0" according to [RFC 2045](#), [RFC 2046](#)
- **Content-Type:** describes attached content with type/subtype (e.g. Image/JPEG or Application/word)
- **Content-Transfer-Encoding:** How message has been encoded (e.g. 7 Bit (only ASCII) or Base64)
- **Content-ID:** Unique character string identifying content
- **Content Description:** Needed when content is not readable text (e.g.,mpeg)

Secure MIME: S/MIME

- defined in RFCs [2630](#), [2632](#), [2633](#)
- defines additional content (sub-) types:
 - **Multipart/signed:**
 - * Cleartext message and signature as independent parts
 - * Message readable also for non-S/MIME clients
 - **Application/pkcs7-mime/signedData:** signed S/MIME message according to PKCS#7 encoding
 - **Application/pkcs7-mime/envelopedData:** Encrypted S/MIME message according to PKCS#7
 - **Application/pkcs7-signature:** Content type of signature part of multipart/-signed message
 - **Application/pkcs10-mime:** Request for generating a Certificate according to PKCS#10.

8.1.3 S/MIME Client Features

- S/MIME uses Public-Key Certificates - X.509 version 3 signed by Certification Authority
- Functions:
 - Key Generation:** Diffie-Hellman, DSS, and RSA key-pairs.
 - Registration Request:** Public keys must be registered with X.509 CA.
 - Certificate Storage:** Local (as in browser application) for different services.
 - Create Signed and/or Enveloped Data:** Sign and/or encrypt messages

8.1.4 Algorithms Used in S/MIME Version 3

- **Message Digesting:** SHA-1 (mandatory) , MD5 (optional)
- **Digital Signatures:** DSA (mandatory), RSA (optional)
- **Secret-Key Encryption:**
 - Encryption: Triple-DES, RC2/40 (optional)
 - Decryption: RC2/40 (mandatory), Triple-DES (optional)
- **Public Key Encryption:** Diffie-Hellman according to RC 2631 (mandatory), RSA with key sizes of 512 and 1024 bits (optional).

8.2 Pretty Good Privacy: PGP

- Created by Philip Zimmerman in 1991
- PGP provides a confidentiality and authentication service that can be used for file storage applications but also for e-mail.
- "Quasi-Standard" for File Encryption
- PGP-Plugins available for various e-mail clients
- PGP is Freeware: Download at <http://www.pgpi.org>

8.2.1 PGP History

- **1991:** PGP V1.0 developed by P. Zimmerman
 - freely distributed via Internet
 - Algorithms: RSA, MD4, Bass-O-Matic (self developed symmetric algorithm)
- **1992:** PGP V2.0
 - Bass-O-Matic and MD4 replaced by IDEA and MD5
 - Version 2.1 - 2.3a follow till July 1993
 - Version 2.4 in November 1993 is first commercial version
- **1995:** PGP 2.6.2i first legally available version of PGP outside USA
- **June 1997:** PGP 5.0 (commercial version) with
 - ElGamal Public Key Encryption
 - DSS Signatures
 - SHA-1
 - CAST128, 3DES
- **October 1997:** "PGP 5.5 for Business Security" includes "Additional Decryption Key" (ADK) functionality (see section [11.9.2](#))

8.2.2 Why Is PGP Popular?

- It is available free on a variety of platforms.
- Based on well known algorithms.
- Wide range of applicability
- Not developed or controlled by governmental or standards organizations

8.2.3 Operational Description

Siehe Abbildung 8.1.



Abbildung 8.1: PGPTools Screenshot

- Signing/ Verification
- Encryption/ Decryption
- Secure File Deletion ("Wiping")
- Support for Public Key Management
- Key Pair Generation
- Secure Storage of Private Keys

8.2.4 Algorithms used by PGP

- **Digital Signature:** DSS/SHA or RSA/SHA
- **Message Encryption:** CAST or IDEA or three-key triple DES or AES or Twofish
- **Key transport:** ElGamal or RSA

8.2.5 Format of PGP Message

Siehe Abbildung 8.2.

8.2.6 PGP Message Generation

Siehe Abbildung 8.3.

8.2.7 PGP Message Reception

Siehe Abbildung 8.4.

8.2.8 PGP "Web of Trust"

Siehe Abbildung 8.5.

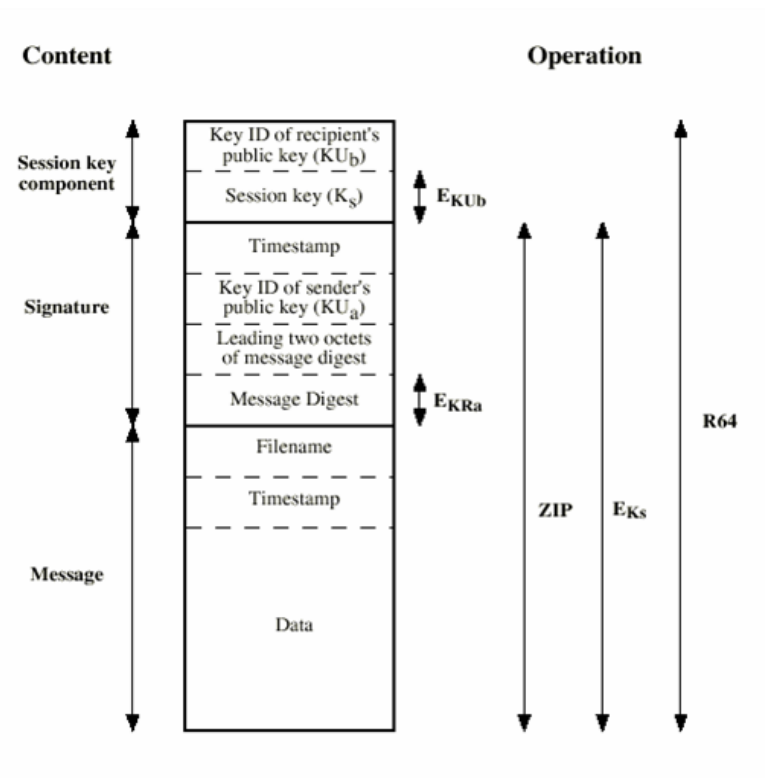


Abbildung 8.2: Format of PGP Message (From User A to User B, signed by A)

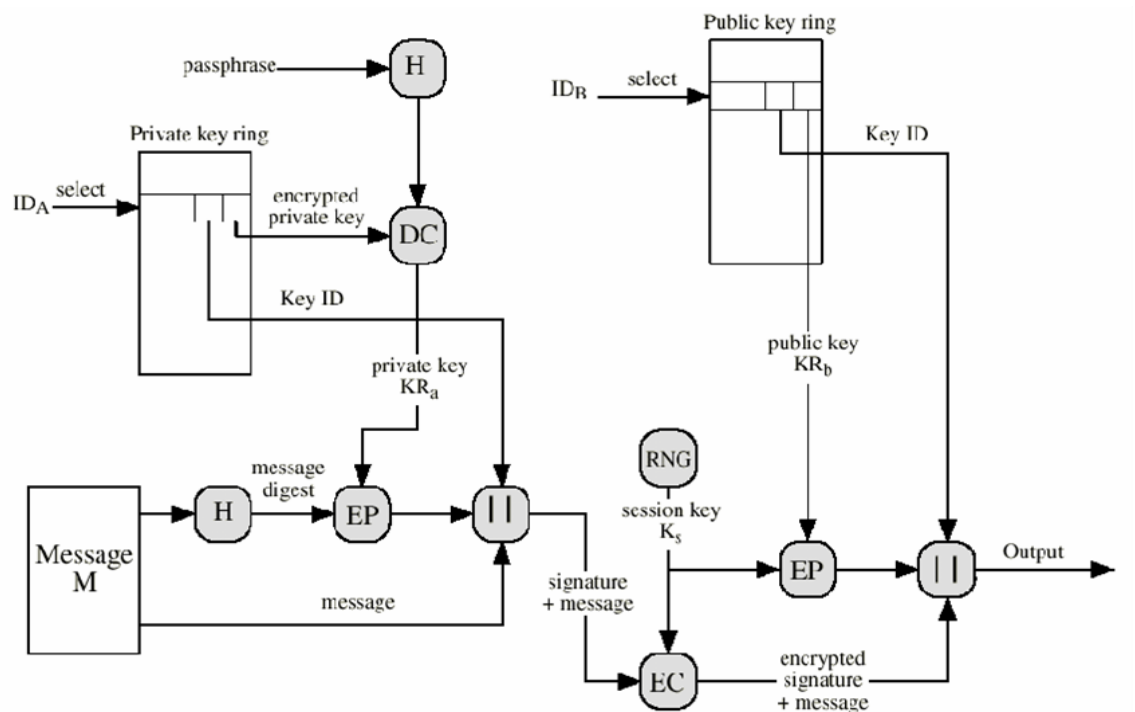


Abbildung 8.3: PGP Message Generation (User A to User B)

8.3 Secure Electronic Transactions: SET

- 1996 - 1998 von VISA und MasterCard entwickelt
- Ziele:
 - Verhindere Kreditkartenbetrug sowohl Händler- als auch Kundenseitig
 - Stärke Vertrauen der Kunden in Kreditkartenzahlung via Internet
- Wegen zu geringer Verbreitung im Oktober 2003 eingestellt.
- Nachfolgesysteme (z.B. "Mastercard SecureCode") werden z.Zt. Entwickelt.

8.3.1 Sicherheitsdienste von SET

- Vertraulichkeit der Bestelldaten gegenüber ausführender Bank
- Vertraulichkeit der Kontendaten gegenüber ausführendem Händler
- Authentifikation aller beteiligten Parteien: Kunden, Händler, Banken
- Nichtabstreitbarkeit aller durchgeführten Transaktionen

8.3.2 SET Beteiligte

Siehe Abbildung 8.6.

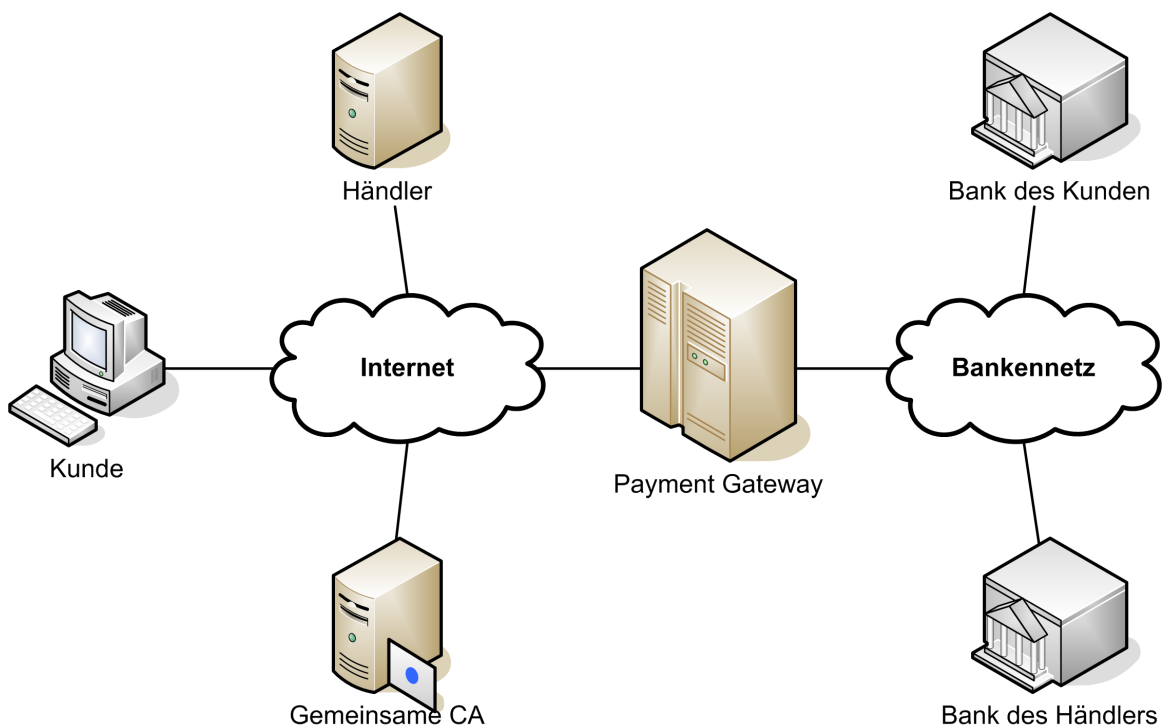


Abbildung 8.6: SET Beteiligte

8.3.3 SET Vorbereitungsphase

- **Kunde:**
 - beantragt Kreditkarte bei einer Bank, die SET unterstützt und installiert das SET-Wallet
 - lässt sich bei CA registrieren und erhält X.509 Zertifikat
- **Händler:**
 - lässt sich bei CA registrieren und erhält X.509 Zertifikat
 - Besorgt sich das Zertifikat des Payment-Gateway

8.3.4 SET Transaction

Siehe Abbildung 8.7.

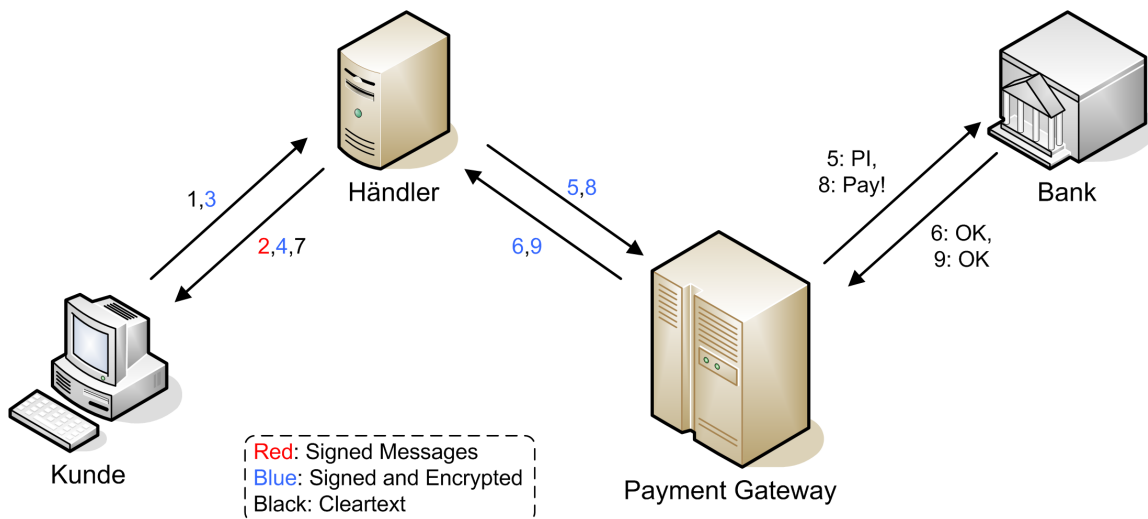


Abbildung 8.7: SET Transaction

(1) Initiate Request:

- Zufallszahl $RAND_K$
- Typ der Kreditkarte

(2) Initiate Response:

- $RAND_K$
- Transaktions ID
- Händlerzertifikat
- Zertifikat des Payment Gateway (wird gebraucht um die Kreditkartendaten zu verschlüsseln. Somit bekommt der Händler diese nicht mit)

(3) Purchase Request:

- Payment Info (PI) verschlüsselt mit PK_{PGW}
 - Kreditkartennummer
 - Bank-Details
 - Betrag
 - $RAND_K$, Transaktions ID
- Order Info (OI) verschlüsselt mit PK_H
 - Bestellformular, Rechnungsbetrag
 - $RAND_K$, Transaktions ID
- Dual Signature über PI und OI (siehe Abschnitt 8.3.5)
- Zertifikat des Kunden

(4) Purchase Response:

- Digitalsignierte Transaktions ID + $RAND_K$ (Bestellbestätigung)

(5) Payment Authorisation Request:

- $E_{PK_{PGW}}(PI)$
- DualSig(PI,OI)
- Transaktions ID
- Betrag
- $RAND_{PK}$ TODO: CHECKME (PK????)
- Zertifikat des Kunden

(6) Payment Authorisation Response:

- Digitalsigniertes OK der Bank

(7) Warenlieferung (z.B. per Post)

(8) Capture Request:

- $RAND_{PK}$, Transaktions ID, Betrag

(9) Capture Response:

- Digitalsigniertes OK

8.3.5 SET Dual Signature

Siehe Abbildung 8.8.

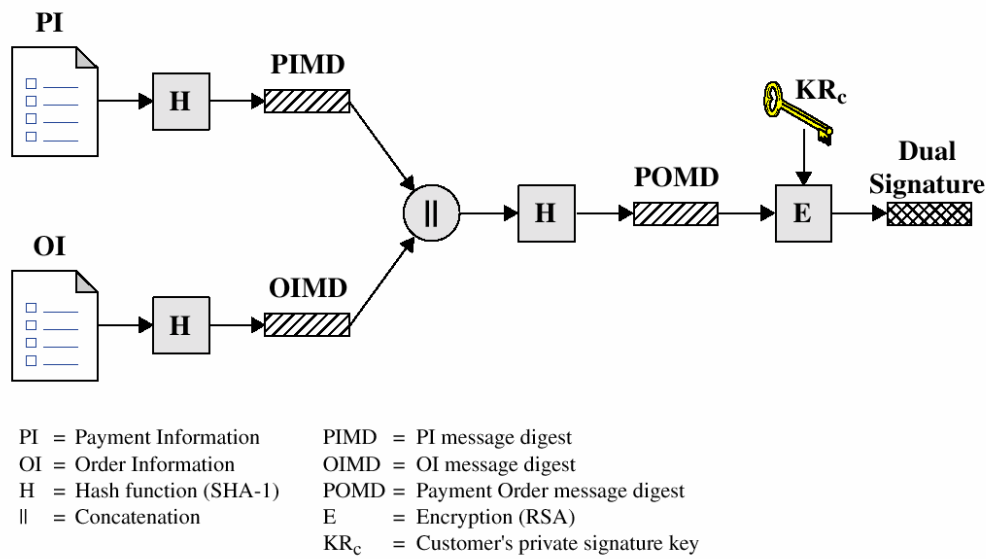


Abbildung 8.8: SET Dual Signature

8.3.6 Building the Purchase Request

Siehe Abbildung 8.9.

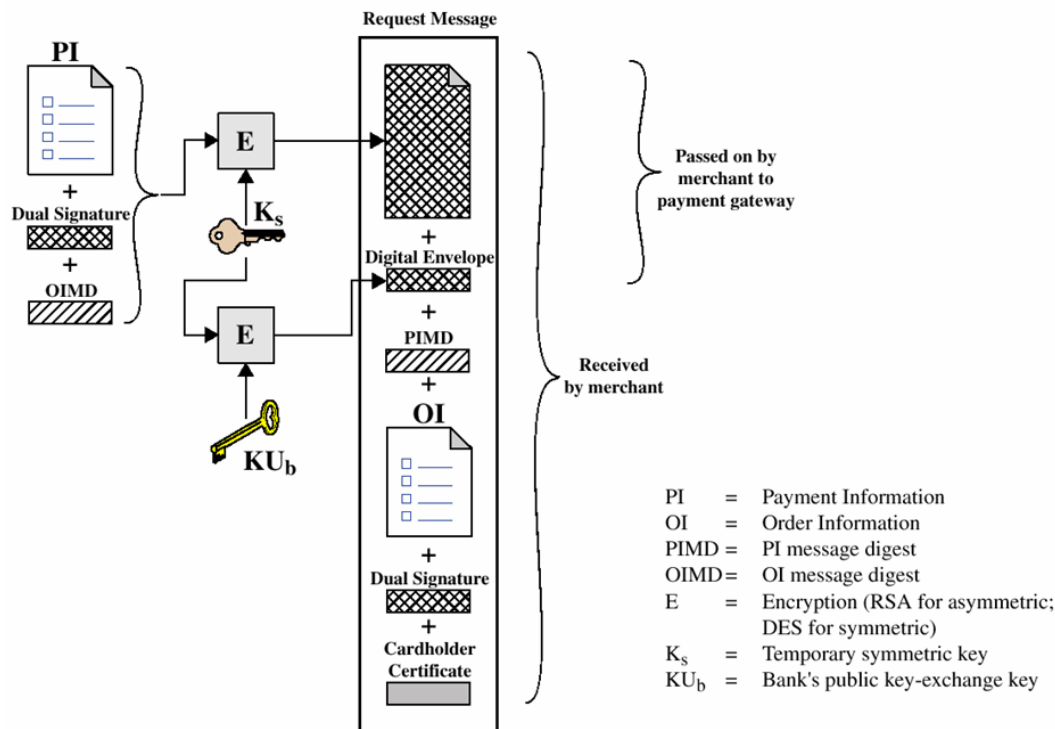


Abbildung 8.9: SET: Building the Purchase Request

8.3.7 Verifying a Dual Signature: Merchant's Site

Siehe Abbildung 8.10.

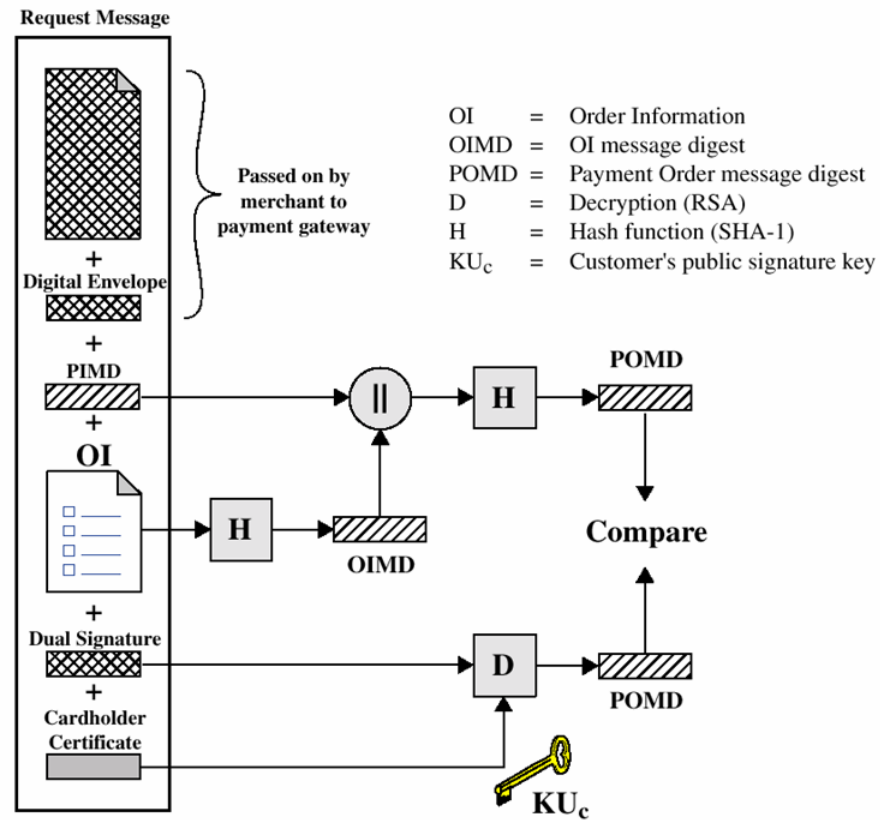


Abbildung 8.10: SET: Verifying a Dual Signature: Merchant's Site

8.3.8 SET Nachteile

- Kunden brauchen Zertifikate
- Kunden müssen Wallet-Software installieren
- SET Händler Software ist teuer (ca. 1000 Euro)
- Eine SET Transaction kostet ca. 50 Cent, deshalb nur für größere Beträge lohnend.

Kapitel 9

Sicherheit auf der Transportschicht

Lage im Protokollstapel

Siehe Abbildung 9.1.

HTTP	FTP	SMTP	Telnet
SSL/TLS	SSH		
TCP			
IP			

Abbildung 9.1: SSL/TLS, SSH: Lage von im Protokollstapel

9.1 SSL und TLS

- *Secure Socket Layer (SSL)*
 - 1994 von Netscape entwickelt und in Browser integriert. aktuelle Version: 3.0
- *Transport Layer Security (TLS)*
 - 1999 als Internet-Standard von IETF verabschiedet
 - Weiterentwicklung von SSL v 3.0 (nur sehr geringe Unterschiede zu SSL)
- Konkurrenzprotokolle (heute bedeutungslos)
 - PCT (Microsoft)
 - S-HTTP (nicht zu verwechseln mit https)
- Sicherheitsdienste von SSL/TLS
 - Vertraulichkeit
 - Integrität
 - Authentizität von Client (optional) und Server
 - Schlüssel- und Algorithmenvereinbarung

Secure Socket Layer
(SSL)

Transport Layer
Security (TLS)

9.1.1 SSL Protokollstapel

Siehe Abbildung 9.2.

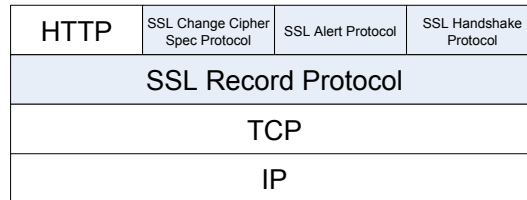


Abbildung 9.2: SSL Protokollstapel

9.1.2 SSL Record Protocol

Aufgaben:

- Fragmentierung
- Komprimierung (optional)
- MAC-Bildung (schlüsselabh. MD5 oder SHA-1)
- Verschlüsselung

9.1.3 SSL Record Format

Siehe Abbildung 9.3.

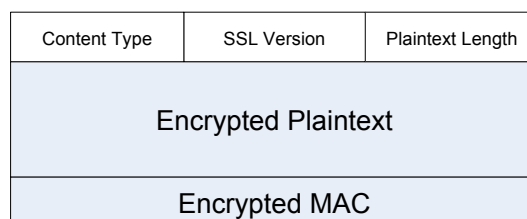


Abbildung 9.3: SSL Record Format

9.1.4 SSL Handshake

Aufgaben:

- Authentifizierung/Identifizierung von Server und (optional) Clients
- Aushandeln von Algorithmen
- Austausch von Schlüsseln

Siehe Abbildung 9.4.

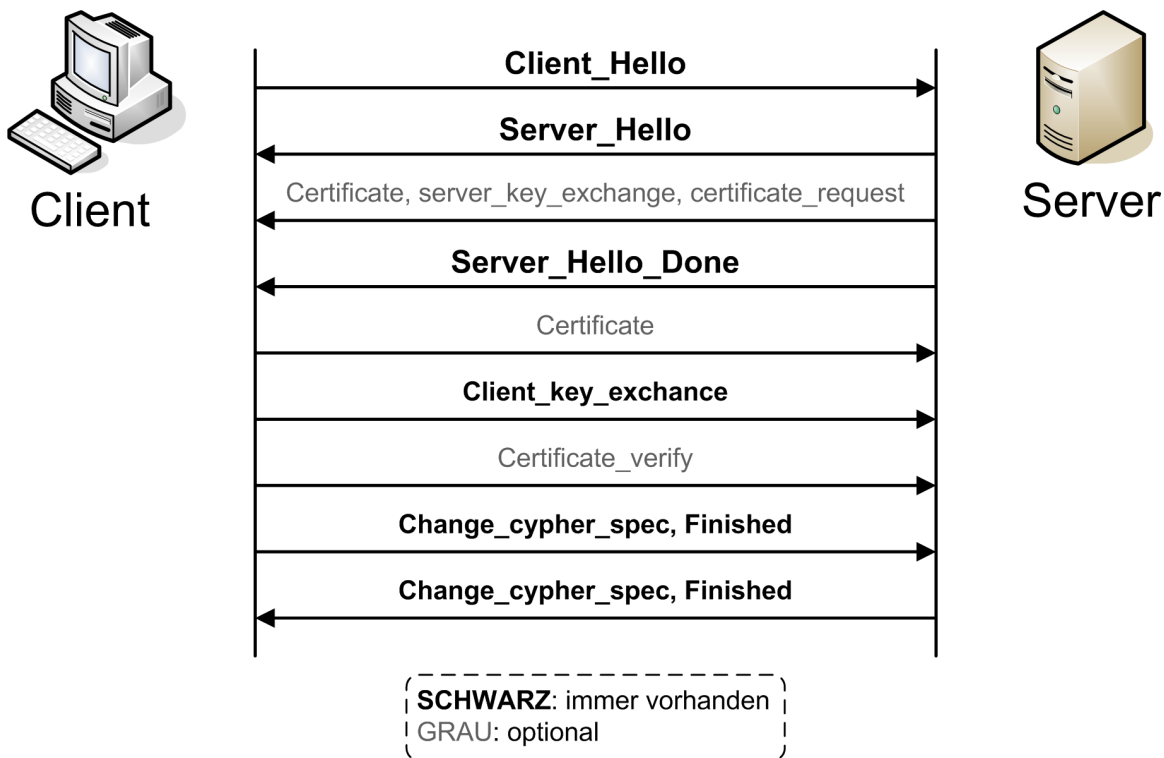


Abbildung 9.4: SSL Handshake

Nachrichten des Handshake

1. **ClientHello** (4 Bytes)
 - SSL Version
 - Zufallszahl
 - Session ID
 - Compression Method
 - Ciphersuite: Liste der unterstützten Krypto-Algorithmen
2. **ServerHello** (76 Bytes)
 - wie ClientHello, aber mit aus Ciphersuite ausgewähltem Algorithmen
3. **Certificate** (c.a. 500 Bytes)
 - X.509 Zertifikat des Servers + evtl. weitere Zertifikate aus der Hierarchie
4. **ServerKeyExchange** (entfällt, falls fixedRSA gewählt)
 - DH-Parameter $p, g, A = g^a \bmod p$ oder
 - RSA Public Key (speziell zum Schlüsseltausch)
 - Signatur über die Parameter + Client-Random
⇒ Server-Authentisierung

5. **CertificationRequest**

- Anforderung Client-Zertifikat, entfällt meistens

6. **ServerHelloDone** (4 Bytes)

7. **Certificate** (falls vorhanden und angefordert)

8. **ClientKeyExchange** (52 Bytes)

- DH-Parameter des Clients: $B = g^b \bmod p$ oder falls RSA gewählt:
- $E_{PK_{Server}}$ (PreMasterSecret): 48 Byte Random
⇒ Implizite Server Authentifikation im Fall fixedRSA

9. **CertificateVerify** (nur falls Client-Zertifikat vorhanden)

- Signatur über Hash aller vorhergehenden Nachrichten

10. **ChangeCipherSpec**

- 1 Byte mit Wert 1
- Aktiviert Verschlüsselung + Integritätsschutz Client
+ Server leiten Master-Secret MS aus PreMasterSecret ab:
$$MS = MD5(PMS \parallel SHA('A' \parallel PMS \parallel RAND_{Client} \parallel RAND_{Server}) \parallel$$
$$MD5(PMS \parallel SHA('BB' \parallel PMS \parallel RAND_{Client} \parallel RAND_{Server}) \parallel$$
$$MD5(PMS \parallel SHA('CCC' \parallel PMS \parallel RAND_{Client} \parallel RAND_{Server}))$$

(ges. 48 Byte)

11. **Finished** (36 Bytes)

- Bilde MAC aus allen vorhergehenden Nachrichten:
$$MD5(MS \parallel pad2 \parallel MD5(Messages \parallel Sender \parallel MS \parallel pad1)) \parallel$$
$$SHA(MS \parallel pad2 \parallel SHA(Messages \parallel Sender \parallel MS \parallel pad1))$$

12. **ChangeCipherSpec** (siehe 10)

13. **Finished** (siehe 11)

(seit SSL v3 um CypherSuiteRollback-Attacke entgegen zu wirken.

Siehe Abschnitt 9.1.5)

9.1.5 Schwächen von SSLv2.0

- Sicherheitsmängel:
 - CypherSuiteRollback-Attacke: Finaler "finished" Hash fehlt, damit ClientHello nicht gesichert: CipherSuite kann geändert werden.
 - In Export-Version: Nicht nur 40-Bit Verschlüsselung, sondern auch 40-Bit MAC
 - Während einer Session kein Schlüsselwechsel möglich
- Funktionale Mängel:
 - Keine Übertragung von Zertifikatsketten

- RSA einziges asymmetrisches Verfahren
- Keine Datenkompression

9.1.6 Vor- und Nachteile von SSLv3.0

- Vorteile:
 - einfach zu nutzen
 - Sicher
 - Weit verbreitet
- Nachteile:
 - SSL bietet nur sicheren Tunnel zwischen Client und Server (Kanalverschlüsselung)
 - Authentifizierung beruht auf X.509-Zertifikaten: User sind über Bedeutung im unklaren.

9.2 SSH (Secure Shell)

- 1995 von T. Ylönien (Uni Helsinki) entwickelt.
- Für Privatanutzer und Bildungseinrichtungen frei verfügbar
- Ziel: Biete sichere Alternativen für Anwendungsprogramme wie rlogin, telnet, ftp
- Sicherheitsdienste
 - Serverauthentifizierung
 - Userauthentifizierung
 - Kanalbasierte Vertraulichkeit und Integrität
- Aktuelle Version: SSH-2

9.2.1 SSH-2 Protokollstapel

Siehe Abbildung 9.5.

Telnet	rlogin	FTP
SSH User Authentication Protocol (optional)		SSH Connection Protocol
SSH Transport Layer Protocol		
TCP		

Abbildung 9.5: SSH-2 Protokollstapel

9.2.2 SSH Transport Layer Protocol

Voraussetzungen:

- Server besitzt Public Key (sog. *Host Key*)
- Client besitzt vertrauenswürdige Kopie des Host Key
- Zuverlässiges Transportprotokoll zwischen Client u. Server (d.h. TCP)

Aufgaben:

- Server-Authentifizierung
- Aushandeln von Algorithmen und Schlüsseln
- Vertraulichkeit und Integritätsschutz für Anwendungsdaten

9.2.3 SSH Protokollablauf

Siehe Abbildung 9.6.

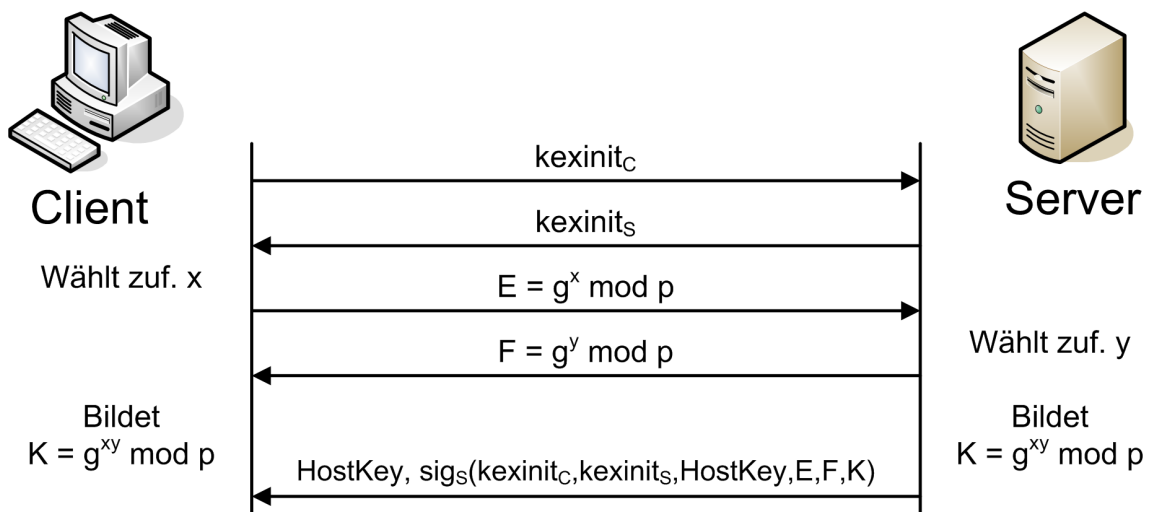


Abbildung 9.6: SSH-2 Protokollablauf

9.2.4 SSH Parameter für Diffie-Hellmann Schlüsseltausch

The prime p is equal to $2^{1024} - 2^{960} - 1 + 2^{64} \cdot \text{floor}(2^{894}\pi + 129093)$.
Its hexadecimal value is:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B
80DC1CD129024E08A67CC74020BBEA63B139B22514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
```

EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381
FFFFFFFF FFFFFFFF.

The generator used with this prime is $g = 2$. The group order q is $\frac{(p-1)}{2}$.

9.2.5 SSH Transport Layer Datenformat

Sequence Number (32 Bit)	Verschlüsselte Payload	MAC
-----------------------------	---------------------------	-----

- $MAC = HMAC(IntegrityKey, Seq.No. || UnencryptedPacket)$
- Algorithmen:
 - SHA-1 (required), MD5, RIPEMD-160 (optional)
 - in SSH-1: CRC-32 (unsicher, da linear!)
 - AES, 3DES, Twofish, IDEA, CAST (jeweils CBC-Mode)

9.2.6 SSH User Authentication Protocol

- Setzt auf Transport Layer Protocol auf
- Ziel: Authentifiziere User
- Server gibt Authentifizierungsmethode vor
- Unterstützte Varianten:
 - Public Key: User signiert UserID und vom Server vergebene SessionID
 - Passwort: Schicke UserID + Passwort an Server
 - Host-Based Public Key: Analog zu Public Key, diesmal aber mit Schlüssel des Client-Rechners
 - None

9.2.7 SSH Connection Protocol

Dienste:

- Sitzungskontrolle
- Datenkomprimierung
- Entfernte Kommandoausführung (Öffnen einer Shell auf dem entfernten System)
- Kapselung/Weiterleitung von TCP/IP Verbindungen

9.3 Web Ressources

- <http://www.openssl.org>
- <http://www.ssh.com>
- <http://www.openssh.org>

Kapitel 10

Sicherheit auf der Internetschicht

10.1 Übersicht

- IPv4: Keine Sicherheit
- IPv6: Support von IPSec zwingend vorgeschrieben. Bis dahin:
- *IPSec* : Zusätzliche optionale Protokollfamilie zur Absicherung von IPv4.
IPSec besteht aus folgenden Teilprotokollen:

IPSec

- *Encapsulating Security Payload ESP*

Encapsulating Security
Payload (ESP)

- * Verschlüsselung der IP-Payload

- * optional auch Integritätsschutz

- *Authentication Header AH*

Authentication Header
(AH)

- * Integritätsschutz für gesamtes IP-Paket

- *Internet Key Exchange IKE*

Internet Key Exchange
(IKE)

- * Schlüsseltausch

- * Algorithmenvereinbarung

- * Authentifizierung der Parteien

10.1.1 IPSec Services

- Access Control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
- Confidentiality (encryption)
- Limited traffic flow confidentiality

10.1.2 Security Associations (SA)

- A one way relationship between a sender and a receiver.
- It describes the way two IPSec Hosts communicate
- Identified by three parameters:

Security Parameter
Index (SPI)

- *Security Parameter Index (SPI)* (unique identifier)
- IP Destination address

Security Protocol
Identifier

- *Security Protocol Identifier (ESP/AH)*

10.1.3 Datenfelder einer Security Association

- Transport oder Tunnelmode?
- Algorithmen
 - Verschlüsselungsalgorithmen
 - Hashfunktionen
- Initialisierungsvektoren
- Schlüssel
- Lebensdauer der SA

10.1.4 Datenbanken in IPSec

Security Association
Database (SAD)

- *Security Association Database (SAD)*
 - Mapping von SPI und SA

Security Policy
Database (SPD)

- *Security Policy Database (SPD)*
 - Mapping von IP-Ziel-Adresse und SPI (falls bereits SA vereinbart)
 - Sonst: Mapping von IP-Ziel-Adresse und Security Policy für dieses Ziel

10.1.5 Beispiel: Sende IP-Paket von A nach B im ESP-Transport Mode

Siehe Abbildung 10.1.

10.1.6 Unterschied zu SSH/SSL

Bei IPSec wird ein neues IP-Paket erzeugt, wobei sich SSH/SSL ganz klar *über* der IP-Schicht befindet

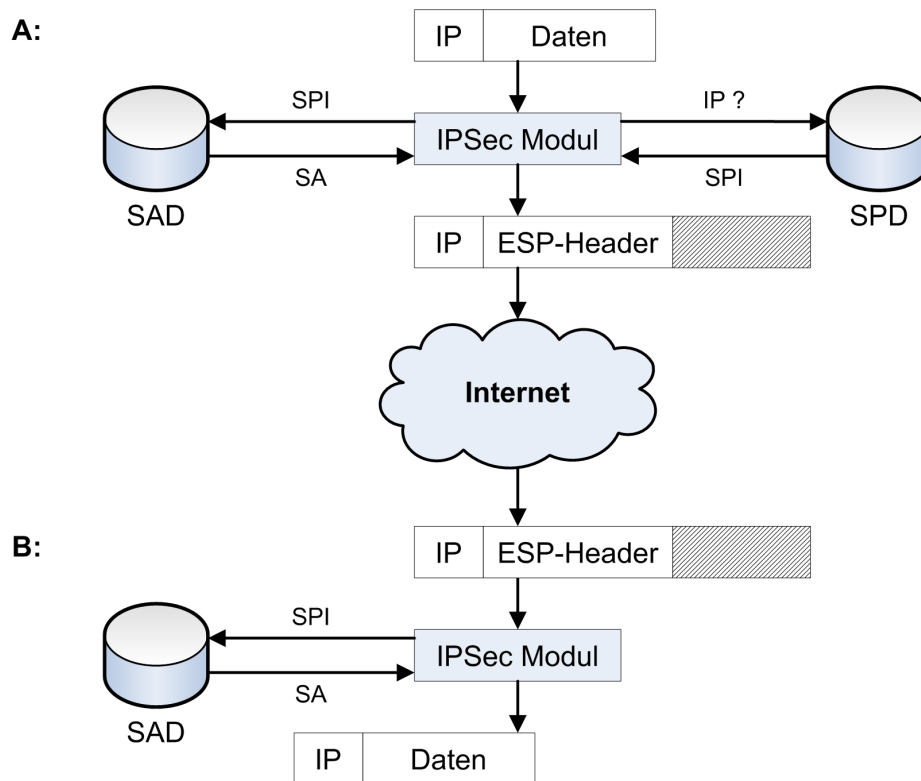


Abbildung 10.1: Beispiel für eine IPsec-Übertragung im ESP-Transport Mode

10.2 Encapsulating Security Payload (ESP)

10.2.1 ESP Format

Füge ESP-Paket hinter IP-Header ein:

- ESP-Header
 - SPI der verwendeten SA
 - Sequence Number
- ESP-Payload
 - verschlüsselte IP-Payload (Transport Mode)
 - verschlüsselter IP-Header + Payload (Tunnel Mode)
 - optional: Authentication Data

10.2.2 ESP Transport Mode

- IP-Header bleibt unverschlüsselt? Routing problemlos möglich
- Verkehrsfluss bleibt im Klartext!

10.2.3 ESP Tunnel Mode

- verschlüssele IP-Header als Teil der ESP-Payload mit
- generiere neuen IP-Header mit IPSec-Gateways als Source/Destinations.

10.2.4 Transport Mode vs Tunnel Mode

Beim Tunnel Mode ist der Verkehrsfluss vertraulich, da der IP-Header nicht mehr sichtbar ist.

Beim Transport Mode ist der Payload vertraulich, jedoch nicht mehr der Verkehrsfluss.

10.2.5 ESP and IP

ESP Header

Siehe Abbildung 10.2.

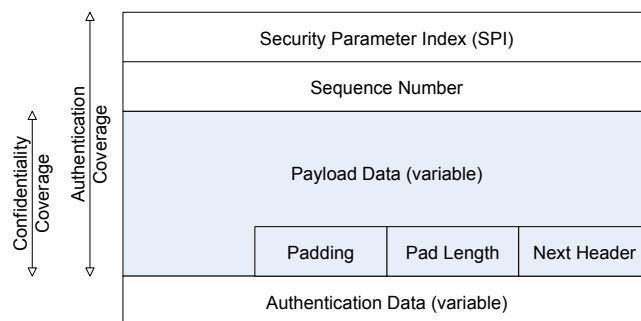


Abbildung 10.2: ESP Header

ESP Encryption and Authentication (Transport Mode)

Siehe Abbildung 10.3.

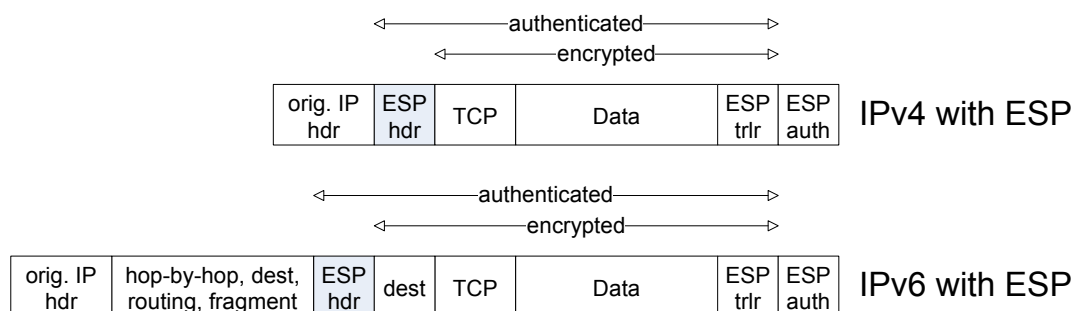


Abbildung 10.3: ESP Encryption and Authentication (Transport Mode)

ESP Encryption and Authentication (Tunnel Mode)

Siehe Abbildung 10.4.

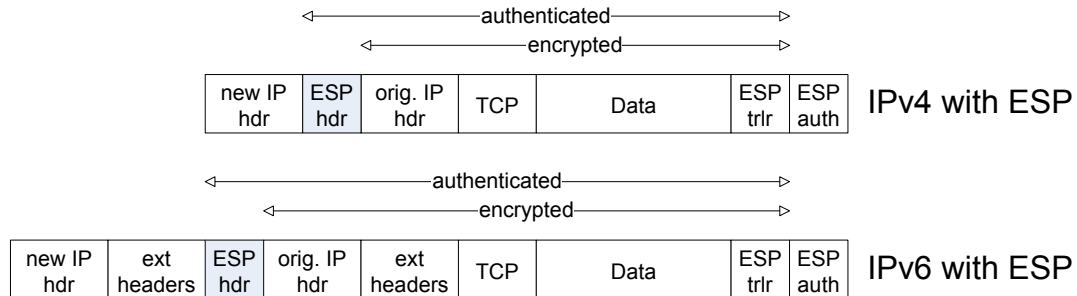


Abbildung 10.4: ESP Encryption and Authentication (Tunnel Mode)

10.3 Authentication Header (AH)

1. Rarely deployed
2. Provides support for data integrity and authentication (MAC code) of IP packets.
3. Guards against replay attacks by authenticated sequence numbers.

10.3.1 AH and IP

AH Header

Siehe Abbildung 10.5.

Next Header	Payload Length	reserved
Security Parameter Index (SPI)		
Sequence Number		
Authentication Data (variable)		

Abbildung 10.5: AH Header

Before applying AH

Siehe Abbildung 10.6.

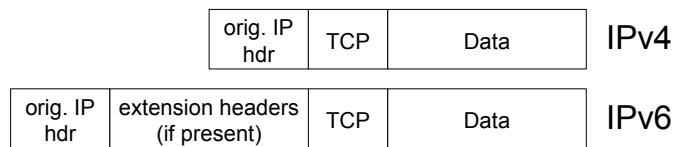


Abbildung 10.6: IP Headers before applying AH

AH Authentication (Transport Mode)

Siehe Abbildung 10.7.

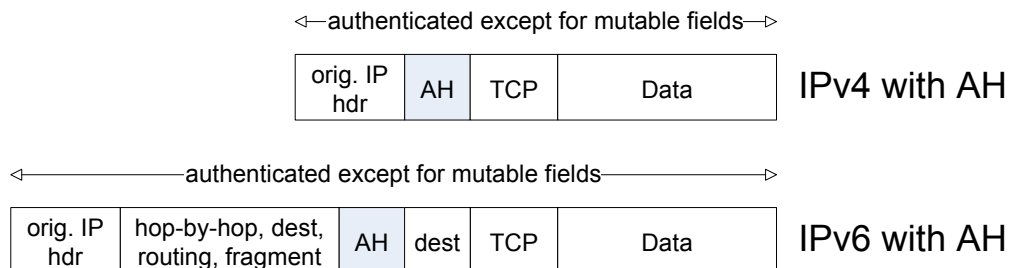


Abbildung 10.7: AH Authentication (Transport Mode)

AH Authentication (Tunnel Mode)

Siehe Abbildung 10.8.

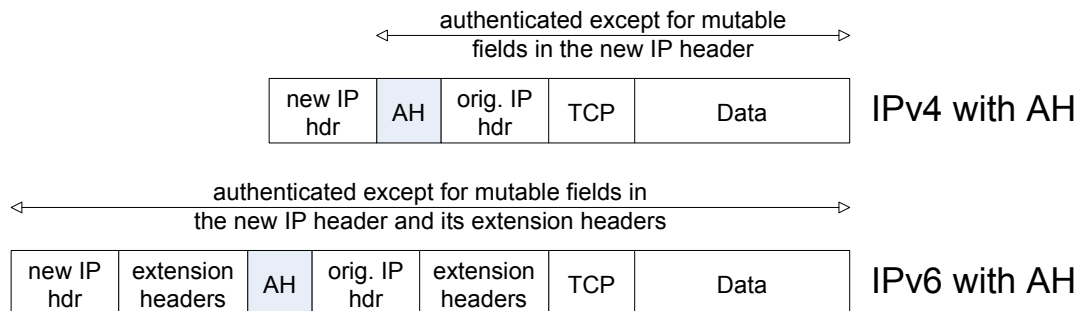


Abbildung 10.8: AH Authentication (Tunnel Mode)

10.3.2 Encryption and Authentication Algorithms

- Encryption:
 - AES
 - Three-key triple DES
 - RC5

- IDEA
- Three-key triple IDEA
- CAST
- Blowfish
- Authentication:
 - HMAC-MD5-96
 - HMAC-SHA-1-96

10.3.3 Anti-Replay Mechanism

- Replay means: An attacker tries to resend a valid packet
- For each new SA, the sender initializes the sequence number counter to 0
- For each packet for this SA, the counter is incremented by one, max. is $2^{32} - 1$
- Receiver accepts packets with:
 - An unused number that is
 - within a certain window or
 - Greater than all previously received packet numbers

10.4 Combinations of Security Associations

Two basic ways to combine SAs:

- Apply more than one protocol (ESP/AH) to the same IP Packet,
e.g. ESP in Transport Mode first and then AH in Transport Mode
- Iterated Tunneling:
Deploy if IPSec enabled routers are on the way between source and destination.

Siehe Abbildungen [10.9](#), [10.10](#), [10.11](#) und [10.12](#).

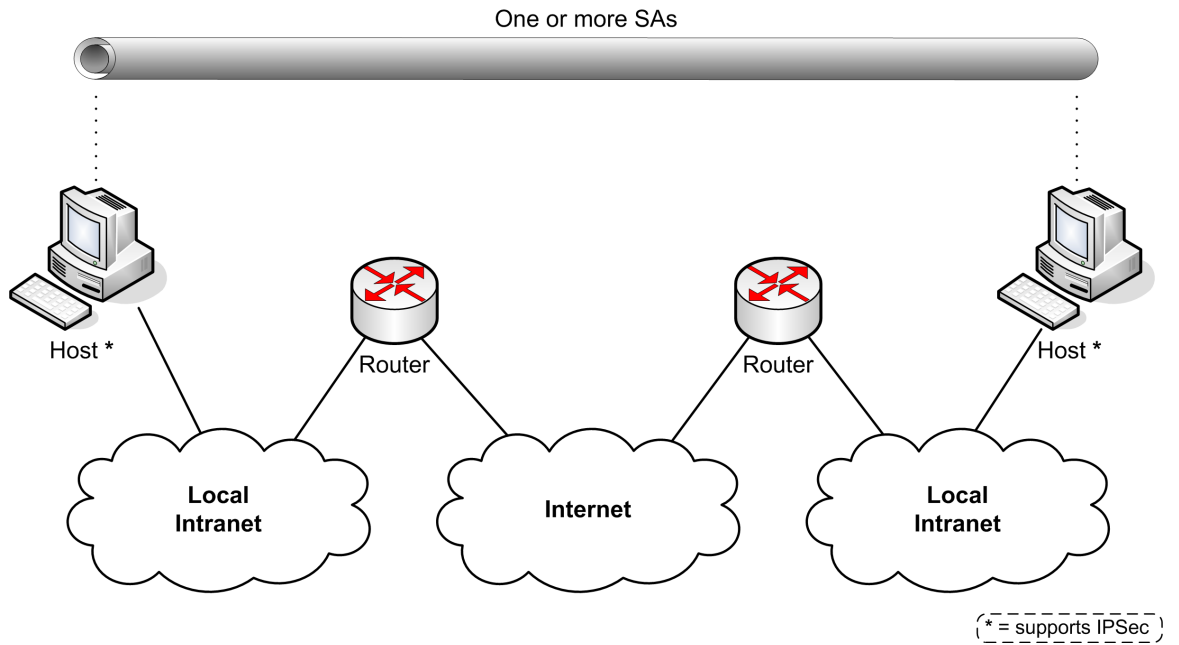


Abbildung 10.9: SA Combinations - case 1

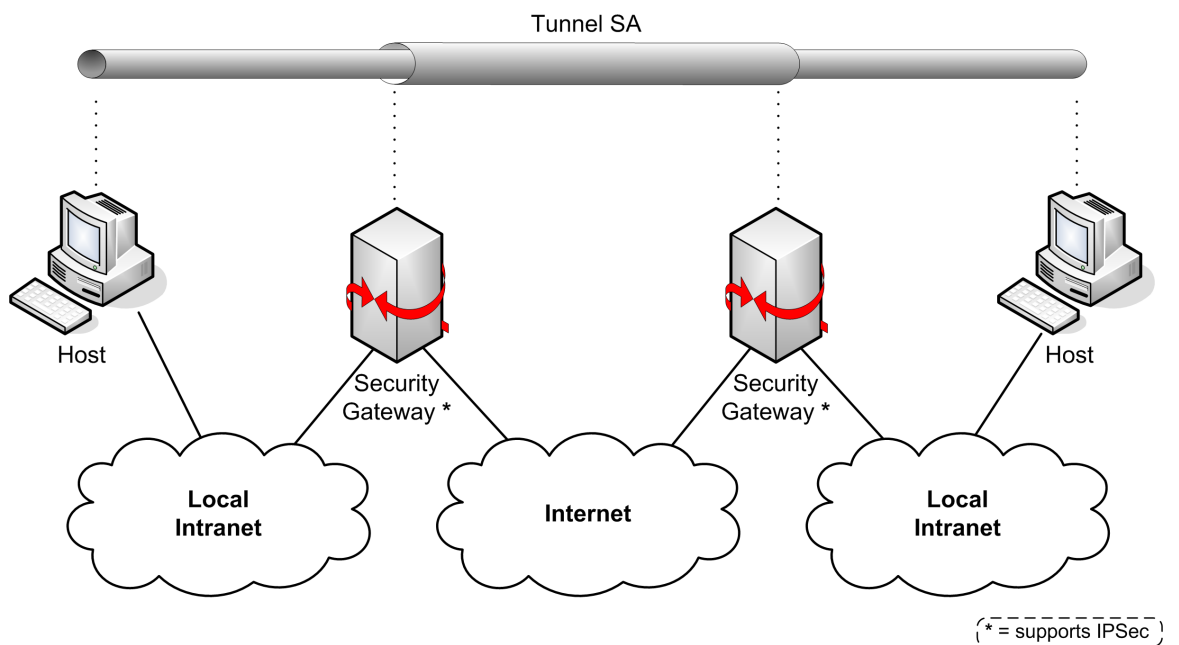


Abbildung 10.10: SA Combinations - case 2

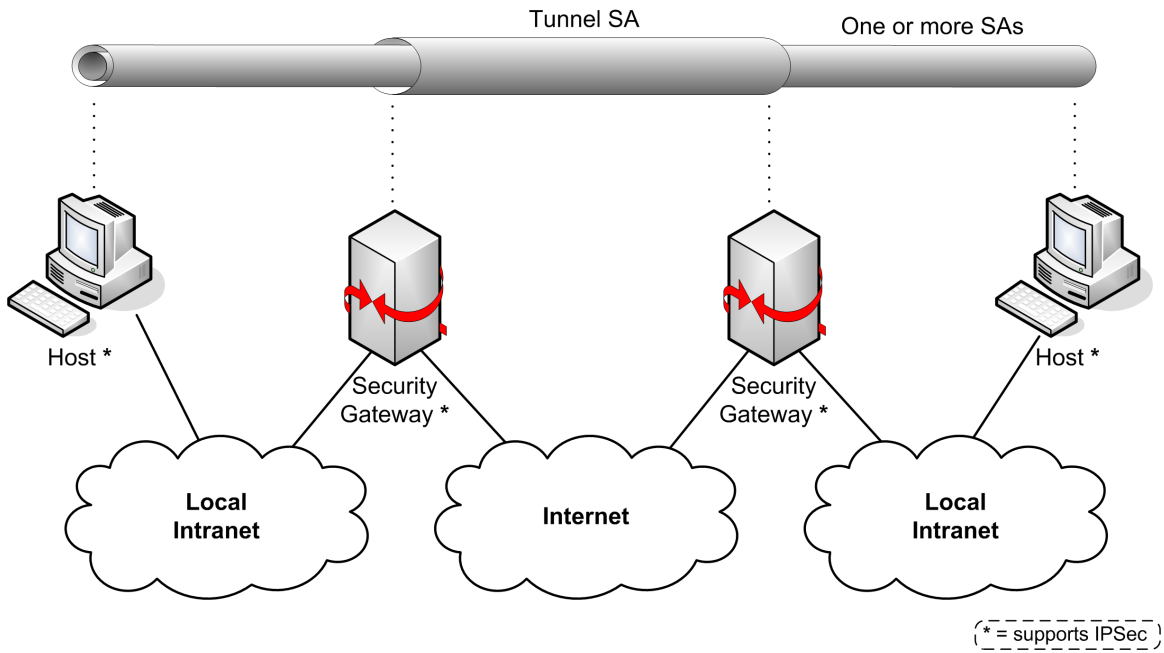


Abbildung 10.11: SA Combinations - case 3

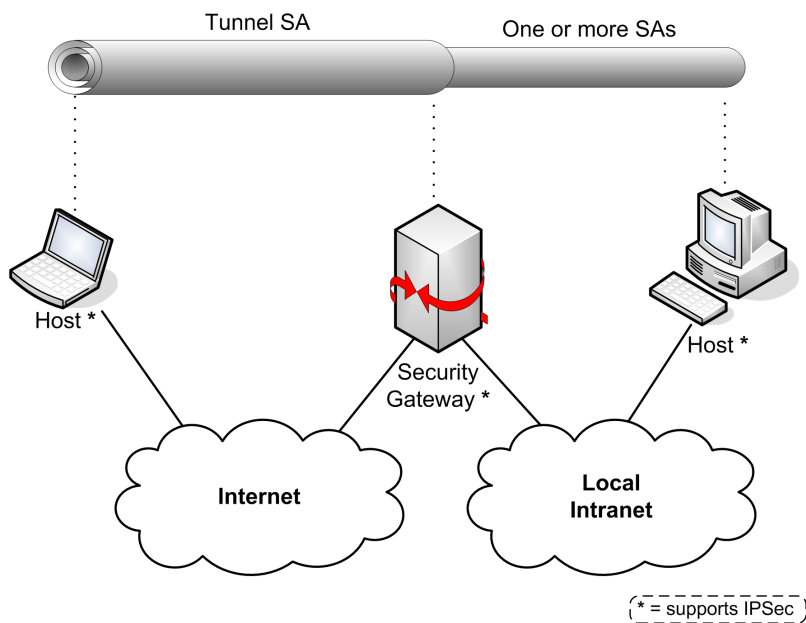


Abbildung 10.12: SA Combinations - case 4

10.5 IPSec Key Management

- Goal: Negotiate SA between two IPSec hosts
- Two types:
 - Manual (for small, static environments)
 - Automated via IKE (Internet Key Exchange). IKE has two parts:
 - * Oakley Key Determination Protocol
 - * Internet Security Association and Key Management Protocol (ISAKMP)

10.5.1 Oakley

- Authenticated Diffie-Hellman Key Exchange
- Three authentication methods:
 - Digital signatures
 - Public-key encryption
 - Symmetric-key encryption

10.5.2 STS Protokollablauf

Siehe Abbildung 10.13.

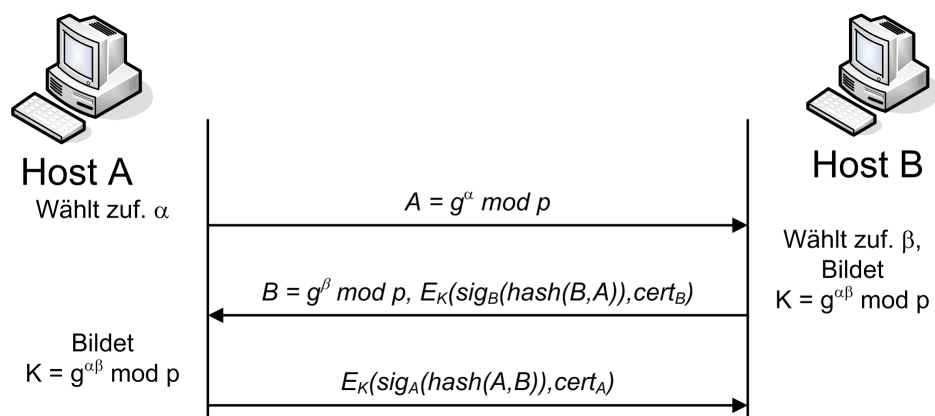


Abbildung 10.13: STS Protokollablauf

10.5.3 Von STS zu OAKLEY

- Photuris:
Tausche vor STS Zufallszahlen (Cookies) aus, die in folgenden Nachrichten enthalten sein müssen
- OAKLEY:
Photuris + Aushandlung weiterer Authentisierungsoptionen + Aushandlung der benutzten DH-Parameter

10.5.4 ISAKMP

- Transported via TCP/UDP
- Provides generic framework for Key Exchange Messages
- Three Modes:
 - Main Mode: 6 Messages, Hosts remain anonymous over the network (Es wird erst ein Schlüssel ausgehandelt, mit welchen die Zertifikate übertragen werden)
 - Aggressive Mode: Only 3 Messages, no anonymity (Zertifikate werden gleich übertragen)
 - Quick Mode: ...

ISAKMP Formats

Siehe Abbildungen 10.14 und 10.15.

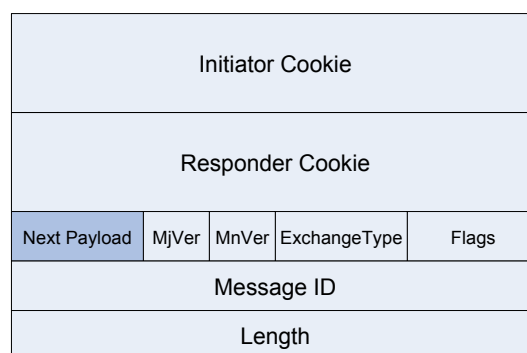


Abbildung 10.14: ISAKMP Header



Abbildung 10.15: Generic Payload Header

10.6 IPSec und IPv6

- Lagere wenig benutzte Header-Fields aus IPv4 und Optionen in sog. Erweiterungsheader aus.
- Fasse AH und ESP in IPv6 als Erweiterungsheader auf, die bei Bedarf eingefügt werden (AH und ESP können als "Extension Header" angesehen werden)
- IPSec fügt sich perfekt in IPv6 ein!

Siehe Abbildungen 10.16 und 10.17.

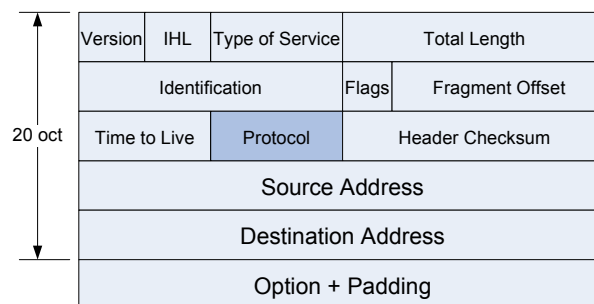


Abbildung 10.16: IPv4 Header

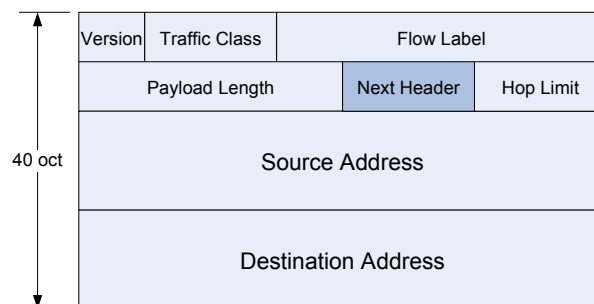


Abbildung 10.17: IPv6 Header

10.7 IPSec und NAT

Network Address
Translation (NAT)

Network Address Translation (NAT):

- Verberge interne Netzstruktur nach aussen
- Bekämpfe Adressknappheit nach IPv4
- Private Adressen sind im Intranet frei vergebbar, nach aussen nur NAT-Gateway mit registrierter Adresse sichtbar (siehe Abbildung 10.18)

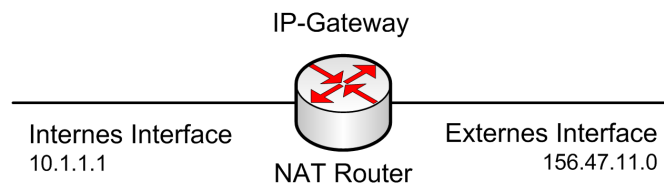


Abbildung 10.18: NAT-Gateway

10.7.1 NAT Formen

- *Statisches NAT (NAPT)*

Statisches NAT (NAPT)

- Mehrere interne Hosts ↔ eine externe Adresse
- Unterscheide Sessions anhand von dynamisch vergebenen Portnummern

- *Dynamisches NAT (DNAT)*

Dynamisches NAT (NAPT)

- Weise privater IP-Adresse temporär gültige externe Adresse aus einem Pool zu.
- Vor allem bei Dial-In Verbindungen genutzt.

10.7.2 Konflikte mit IPSec

- AH ↔ DNAT:
Konflikt, da NAT IP-Adressen ändert, die durch AH geschützt sind
- ESP ↔ NAPT:
Konflikt, da NAPT Portnummern ändert, die durch ESP geschützt sind
- ESP ↔ DNAT:
Kein Konflikt, da DNAT nur die ungeschützten IP-Adressen ändert

Lösungsansätze

- Generelle Lösung:
 - Verpacke IPSec Pakete vor Weiterleitung in UDP-Paket
 - Standardisierung bei IETF IPSec NAT Transversal
 - Proprietäre Lösung von Cisco
- Spezielle Lösung bei ESP ↔ NAPT:
 - Weise im privaten Netz einen einzigen Host als IPSec-Endpunkt aus: Keine Änderung der Portnr. nötig.
 - Bei mehreren IPSec-Hosts: Nutze nicht Portnr., sondern SPI als Unterscheidungsmerkmal

10.7.3 IPSec Overview

- IPSec Security Protocols
 - Encapsulated Security Payload ESP
 - * Confidentiality for IP payload
 - * Optional integrity protection
- Authentication Header AH
 - Integrity Protection for whole IP Packet
- Both sit on top of IP (siehe Abbildung 10.19)

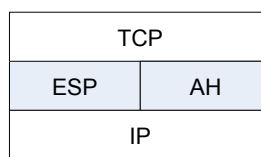


Abbildung 10.19: ESP und AH im Protokollstapel

- IPSec Key Management
 - Internet Key Exchange (IKE)
 - * Automated exchange of security associations (SAs)
 - Internet Security Association and Key Management Protocol ISAKMP
 - * Provides generic framework for various key exchange protocols, e.g. OAKLEY
 - * "IKE = ISAKMP + OAKLEY"
 - IKE is transported via UDP (Port 500) (siehe Abbildung 10.20)

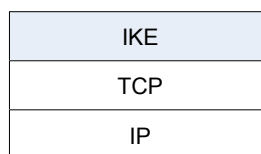


Abbildung 10.20: IKE im Protokollstapel

10.7.4 IPSec RFCs

- [RFC 2401](#): An overview of IP security architecture
- [RFC 2402](#): IP Authentication Header (AH)
- [RFC 2406](#): IP Encapsulating Security Payload (ESP)
- [RFC 2407](#): IPSec Domain of Interpretation (DoI)
- [RFC 2408](#): Internet Security Association and Key Management Protocol (ISAKMP)

- [RFC 2409](#): The Internet Key Exchange (IKE)
- [RFC 2412](#): The Oakley Key Determination Protocol

Kapitel 11

Übungen

11.1 Übungsblatt 1

11.1.1 Aufgabe 1

Welche Operationen darf ein nicht signiertes Java-Applet nicht ausführen, wenn die Default Security Policy in Kraft ist (s. <http://www.java.sun.com>)?

Lösung

Nicht-signierte Applets haben Rechte-Beschränkungen, da für diese die Standard Policy benutzt wird.

- Operationen auf Dateiebene
 - Lokale Dateien lesen/schreiben
 - Prüfen, ob Verzeichnisse/Dateien existieren
- Operationen auf Netzebene
 - Verbindungen zu anderen Hosts als dem Source-Host aufbauen
 - An lokalen Ports lauschen
- Öffnen von Top-Level Windows ohne Titel "Applet Window"
- Setzen oder Abfragen von System Properties
- Starten oder manipulieren anderer Programme
- Beenden der JVM
- Manipulation/Ersetzen des Security Managers
- Erweitern lokaler Pakete um eigene Klassen

11.1.2 Aufgabe 2

Warum sind ActiveX-Controls generell als gefährlicher als Java-Applets anzusehen? Welche Sicherheitsmaßnahmen sieht der Microsoft Internet-Explorer bei der Nutzung von ActiveX-Controls vor?

Lösung

Kann beliebige Programmiersprache sein, d.h. meistens C. Darum keine genauere Sicherheit wie bei Java

Bei Microsoft:

- signiert = sicher
- unsigned = unsicher

11.2 Übungsblatt 2

11.2.1 Aufgabe 1

Was sind Cookies? Welche Bedeutung haben sie aus Security-Sicht?

Lösung

”Ein HTTP-Cookie bezeichnet Informationen, die ein Webserver zu einem Browser sendet, die dann der Browser wiederum bei späteren Zugriffen auf denselben Webserver zurücksendet. Mit Cookies ist das zustandslose Hypertext Transfer Protocol um die Möglichkeit erweitert, Information zwischen Aufrufen zu speichern.” ([Wikipedia](#))

- **Probleme:**

- Wenn Cookie von Angreifer mitgelesen wird
- Wenn Cookie von Angreifer kopiert wird (z.B. durch Zugriff auf den Opferrechner)
- Es besteht die Möglichkeit Benutzerprofile anzulegen. Marketingfirmen, die bei vielen Websites Werbebanner haben, können mit sog. ”serverfremde” Cookies über einzelne Seiten hinweg den Benutzer verfolgen.

- **Lösungen:**

- Serverfremde Cookies nicht zulassen (z.B. solche von doubleclick)
- Session-Cookies zulassen, persistente Cookies allerdings nicht.

11.2.2 Aufgabe 2

Wie funktionierte der Morris-Wurm von 1988?

Lösung

- Verbreitung über sendmail, finger, rexec
- Keine echte Schadwirkung, sondern nur Demonstration dass man Schaden anrichten kann
- Wurm Routinen:
 - infect:
 - * try_finger: Buffer overflow Bug
 - * try_rsh: Versucht eine Shell aufzumachen
 - * try_sendmail: Bug der es erlaubt Mails auch an *Prozesse* zu senden.
 - crack_some:
 - * Suche nach Hosts zum infizieren (Hosts ohne Password)
 - * read /etc/passwords
 - * probiere einfache Passwörter (30% Erfolg)
 - * probiere Passwörter aus einer Dictionary

11.2.3 Aufgabe 3

Welche Sicherheitsprobleme ergeben sich bei Nutzung des User Datagram Protocol (UDP)?

Lösung

- Unzuverlässiger Datenaustausch von Datenpaketen (Datagrams)
 - Max. Paketlänge 65.5 kByte
 - Keine Sequenznummern
 - Keine Verbindungskontrolle durch Flags
 - Kein Three-Way Handshake
- UDP Header-Informationen
 - Quell-und Zielport
 - Gesamtlänge des UDP-Pakets
 - Prüfsumme
- Gleiche Bedrohungen wie bei TCP durch fehlende Vertraulichkeit und Integritätsschutz, aber:
 - Durch fehlende Sequenznummern: Spoofing, Replay und Session Hijacking werden deutlich erleichtert

- Spezieller Angriff: UDP Flood

11.2.4 Aufgabe 4

Warum ist bei Redirect-Nachrichten des ICMP-Protokolls besondere Vorsicht geboten?

Lösung

Mit dem Pakettyp REDIRECT kann man Pakete umleiten und dadurch mitlesen. Es ist auch möglich DOS-Attacken umzusetzen.

11.3 Übungsblatt 3

11.3.1 Aufgabe 1

Interpretieren Sie den in Tabelle 11.1 abgebildeten Regelsatz für einen Paketfilter:

Aktion	Sende-IP	Sendeport	Ziel-IP	Zielpport	Flags	Bedeutung
allow	intern	*	*	*	SYN	?
allow	intern	*	*	*	ACK	?
allow	extern	*	*	*	ACK	?
allow	*	*	Our Mail-GW	25	*	?
allow	*	*	Our Mail-GW	53	UDP	?
block	*	*	*	*	UDP	?
block	*	*	*	512-514	*	?
allow	*	*	*	22	*	?

Tabelle 11.1: Paketfilter Übungsaufgabe Filtering Rule

Lösung

- 1) allow: TCP-Verbindungsaufbau von internet
- 2) allow: TCP-Antwortpakete von internet
- 3) allow: TCP-Antwortpakete von extern nach intern
- 4) allow: SMTP-Pakete zum Mail-Gateway
- 5) allow: DNS-Anfragen an Mail-Gateway
- 6) block: Jeden anderen UDP-Traffic
- 7) block: rexec, rlogin, rsh
- 8) allow: ssh

Port 512: rexec (remote execution)

Port 513: rlogin (remote login)

Port 514: rsh (remote shell)

11.3.2 Aufgabe 2

Lesen Sie die Erzählung "The Gold Bug" (Der Goldkäfer) von Edgar Allan Poe. Was sind die entscheidenden Schritte bei der Entschlüsselung des Pergaments?

Lösung

Vermutungen

- monoalphabetische Chiffre (da simpel!)
- Sprache: englisch
- Häufigster Buchstabe: "8" $\Rightarrow 8 \leftrightarrow e$
- Bigramme und Trigramme anschauen: z.B. kommt das Trigramm ";48" häufig vor $\Rightarrow$;48 $\leftrightarrow$ the
- ;48;(88;4 $\leftrightarrow$ thet?eeth $\Rightarrow ? \leftrightarrow r$ (ergibt das Wort the tree)
- etc...

11.3.3 Aufgabe 3

Die folgende Nachricht ist mit der Caesar-Chiffre verschlüsselt. Entschlüsseln Sie sie, *ohne* eine vollständige Schlüsselsuche durchzuführen.

c = NSOCS CDOSX DIZSC MRNOE DCMRO BDOHD WSDFS OVOXO C

Lösung

Häufigster Buchstabe: O $\leftrightarrow$ E $\Rightarrow$ Schlüssel = 10

11.3.4 Aufgabe 4

Der folgende Text wurde mit einer Vigenère-Chiffre verschlüsselt. Die Länge des Schlüsselworts ist 3. Wie lautet der Text?

c = OLQYK MIHSD LDLLZ LQQAQ MZOMLQ SLUQU NXHUF LAFBP
UOQQU GULHZ AVOOO GLVEL OGUJL BHDSH UJKFL UZ

Lösung

Betrachte 1./4./7. usw Chiffrebuchstaben:

Häufigster: $L \leftrightarrow E \Rightarrow G$

2./5./8. Chiffrebuchstabe: Häufigster: $H \leftrightarrow E \Rightarrow C$

3./6./9. Chiffrebuchstabe: Gleichhäufig: Q und E (Vermute $Q \leftrightarrow E \Rightarrow L$)

Das Schlüsselwort ist GCL. Um eins verschoben ergibt das HDM

11.3.5 Aufgabe 5

Der folgende Chiffre - Text wurde mit einer Vigenère-Chiffre (Blocklänge 3) erstellt. Das Wort HOCHSCHULE kommt im Klartext vor. Wie lautet der vollständige Klartext und das Schlüsselwort?

$c =$ IWQ EGU ULD ITH IQF IUF IWO FFH SOH EKH O

Lösung

$m =$ HOCHSCHULE

$c =$ x..x..x...

Wo kommt 3 mal der gleiche Buchstabe vor im Abstand von 3?

$m =$ HOCHSCHULE

$c =$ ITHIQFIUFI $\leftarrow$ geht nicht (wegen dem I oder wegen dem C)

$m =$ HOCHSCHULE

$c =$ IQFIUFIWOF $\leftarrow$ geht! $\Rightarrow$ Schlüssel ABC

Lösung: "Hundert Jahre Hochschule der Medien"

11.3.6 Aufgabe 6

Schreiben Sie ein Java-Programm, das die Caesar-Chiffre implementiert. Übergeben Sie Klartext und Schlüssel dem Programm als Argumente.

Lösung

Listing 11.1: Caesar-Chiffre En-/Decoder

```
1 class CaesarChiffre {
2
3     static int offset = (int) 'A';
4
5     public static String encrypt(String cleartext, int key) {
6         char[] c = cleartext.toCharArray();
7         char[] cipherchar = new char[c.length];
8         for (int i = 0; i < c.length; i++) {
9             int pos = (int) c[i] - offset;
```

```

10     int newpos = (pos + key) % 26;
11     cipherchar[i] = (char) (newpos + offset);
12 }
13
14 String cipher = new String(cipherchar);
15 return cipher;
16 }
17
18 public static String decrypt(String ciphertext, int key) {
19     char[] c = ciphertext.toCharArray();
20     char[] clearchar = new char[c.length];
21     for (int i = 0; i < c.length; i++) {
22         int pos = (int) c[i] - offset;
23         int newpos = (pos + (26 - key)) % 26;
24         clearchar[i] = (char) (newpos + offset);
25     }
26
27     String cleartext = new String(clearchar);
28     return cleartext;
29 }
30
31 public static void main(String[] args) {
32     String option = args[0];
33     String message = args[1];
34     int key = Integer.parseInt(args[2]);
35
36     if (option.equals("e") || option.equals("E")) {
37         System.out.println("This is your ciphertext:");
38         String ciphertext = encrypt(message, key);
39         System.out.println(ciphertext);
40     }
41     if (option.equals("d") || option.equals("D")) {
42         System.out.println("This is your cleartext:");
43         String cleartext = decrypt(message, key);
44         System.out.println(cleartext);
45     }
46     if (!(option.equals("e") || option.equals("E") || option.equals("d")
47         || option.equals("D")))) {
48         System.out.println("Use d or e for decryption or encryption!");
49     }
50 }
51
52 }

```

11.4 Übungsblatt 4

11.4.1 Aufgabe 1

Der folgende Chiffre-Text wurde mit einer Enigma-Maschine mit drei Walzen (Walzenlage 312), Anfangsstellung der Rotoren HDM, Ringstellung 1-1-1 und den Steckverbindungen AB, CD, EF, GH, IJ erstellt:

c = SDEWR ZISPF

Entschlüsseln Sie Ihn mit Hilfe des Enigma - Applets unter
http://homepages.tesco.net/~andycarlson/enigma/enigma_j.html

Lösung

$m = \text{ALANT URING}$

11.4.2 Aufgabe 2

Betrachten Sie die Enigma- Maschine mit Steckbrett, auf dem fünf Buchstabenpaare miteinander verkabelt werden, und drei Walzen, die aus fünf Walzen ausgewählt werden können. Wie viele mögliche Anfangskonfigurationen (d.h. Schlüssel) gibt es?

Lösung

Buchstabenpaare:

$$\binom{26}{2} \cdot \binom{24}{2} \cdot \binom{22}{2} \cdot \binom{20}{2} \cdot \binom{18}{2} = 6 \cdot 10^{11}$$

Walzen:

Wähle 3 aus 5 aus: $\binom{5}{3} = \frac{5 \cdot 4 \cdot 3}{3! \text{ Möglichkeiten}} = 10$

Mögliche Anordnungen: $3! = 6$ Möglichkeiten

Anfangspositionen: 26^3 Möglichkeiten

Zusammen insgesamt ca. $6 \cdot 10^{17}$ Schlüssel $\approx 2^{55}$ Schlüssel (55bit)

11.4.3 Aufgabe 3

Fassen Sie die Enigma-Chiffre mit drei Walzen als polyalphabetische Substitutionschiffre auf. Wie gross ist die Blocklänge?

Lösung

Die Blocklänge ist 26^3 , denn das einzige was sich bewegt sind die Walzen.

11.4.4 Aufgabe 4

Gegeben seien zwei Urnen. Urne1 enthält drei rote und vier schwarze Kugeln, Urne2 enthält zwei rote, drei schwarze und zwei weiße Kugeln. Man zieht je eine Kugel aus jeder Urne. Wie groß ist die Wahrscheinlichkeit, dass beide Kugeln dieselbe Farbe haben? Wie groß ist die *bedingte* Wahrscheinlichkeit, dass beide Kugeln dieselbe Farbe haben, unter der Bedingung, dass die Kugel aus Urne1 rot ist?

Lösung

Ereignisraum $E = \{(R, R), (R, S), (R, W), (S, R), (S, S), (S, W)\}$
 $E_1 = \{(R, R), (S, S)\} \Rightarrow p(E_1) = p(R, R) + p(S, S) = \frac{3}{7} \cdot \frac{2}{7} + \frac{4}{7} \cdot \frac{3}{7} = \frac{18}{49}$

Wir suchen die bedingte Wahrscheinlichkeit. Der Ereignisraum ist also verkleinert auf den Ereignisraum E_2 :

$E_2 = \{(R, R), (R, S), (R, W)\} \Rightarrow p(E_2) = \frac{3}{7}$
 $\Rightarrow p(E_1|E_2) = p(R, R) \in E_2 = p(\text{"Ziehe } R \text{ aus Urne2"}) = \frac{2}{7}$

Alternativer Ansatz: mit Formel: $p(E_1|E_2) = \frac{p(E_1 \cap E_2)}{p(E_2)} = \frac{p(R, R)}{\frac{3}{7}} = \frac{\frac{6}{49}}{\frac{3}{7}} = \frac{2}{7}$

11.4.5 Aufgabe 5

Warum darf beim One-Time Pad der Schlüssel nur ein einziges Mal verwendet werden?

Lösung

$$c_1 = m_1 \oplus k$$

$$c_2 = m_2 \oplus k$$

Ziel des Angreifers:

$$c_1 \oplus c_2 = m_1 \oplus m_2 \oplus k \oplus k = m_1 \oplus m_2 (\text{weil } k \oplus k = 0)$$

Wenn diese beiden Texte mit dem gleichen Schlüssel k verschlüsselt wurden, kann man nun einfach eine Known-Plaintext-Attacken durchführen.

11.4.6 Aufgabe 6

Die folgenden fünf Chiffretexte sind alle mit demselben One-Time-Pad erzeugt worden. Rekonstruieren Sie die Klartexte, unter der Voraussetzung, daß der erste Klartext mit INDEN NEUEN beginnt.

```

c1  =  ZVIHZ IMZIJ KLCXE ZPDPC BNVNP TPYLC GHQMM NMQLM HZEZK FOJRV
c2  =  IWMLS NMMIJ MZTXI LRVEA CRWTB RTCNW IVPGT YISER CLUDB TJNFI
c3  =  DQYHU IMRGK MVVYN LVDED POYEO DTUKJ KYXYN LEPVJ UYRMB BYJPU
c4  =  WCJUP DMXYY QVCUC OMQET PPELX KEYXL KLFEX GERND SKHMW ZGIYK
c5  =  DMILL DVYIY QEXEE YIULG ZNMTB NPYSK ULKML DSXVL GBHQB EUYPL

```

Lösung

- Man kann nun die ersten 2 Blöcke jeweils betimmen. Diese haben den Abstand Z zu I , bzw. V zu N usw.
- Anschliessend raten: z.B. MEDIZINTECHNIK. Dadurch kann man wieder die anderen Blöcke ermitteln, usw.

m_1 = INDEN NEUEN BUNDE SLAEN DERNs TEIGT DASST IMMUN GSBAR OMETE
 m_2 = ROSIG SEHEN DIEDI ENSTL EISTE RIMIN FORMA TIONS BEREI CHIHR
 m_3 = MITEI NEMCO DEGEN ERATO RFUER DIEFA HRZEU GINDU STRIE WERDE
 m_4 = FUERD IESUC HENAC HINTE RGALA KTISC HEMLE BENWE RDEND IEDAT
 m_5 = MEDIZ INTEC HNIKE RERAR BEITE NEINB REMSS YSTEM FUERI NSTRU

11.5 Übungsblatt 5

11.5.1 Aufgabe 1

Der DES-Algorithmus benötigt für die Verschlüsselung eines Bytes 80 Taktzyklen, der AES benötigt 20. Wie viele Megabyte pro Sekunde kann man mit dem DES bzw. AES auf einem 500 MHz-Rechner verschlüsseln?

Lösung

$$500 \text{ Mhz} = 5 \cdot 10^8 \frac{\text{Zyklen}}{\text{sec}}$$

$$\text{DES: } \frac{80 \text{ Zyklen}}{\text{Byte}}$$

$$\Rightarrow \frac{\text{Byte}}{\text{sec}} = \frac{5 \cdot 10^8}{80} = \frac{5}{8} \cdot 10^7 = 6.25 \cdot 10^6 = 6.25 \text{ MB}$$

$$\text{AES: } \frac{20 \text{ Zyklen}}{\text{sec}} \Rightarrow 25 \frac{\text{MB}}{\text{sec}}$$

11.5.2 Aufgabe 2

Gegeben sei der (hexadezimal geschriebene) Klartext $m = 0123456789ABCDEF$ und der (ebenfalls hexadezimal geschriebene) DES-Schlüssel $k = 133457799BBCDFF1$. Berechnen Sie das Ergebnis der Verschlüsselung von m nach der ersten DES-Runde.

Lösung

TODO

11.5.3 Aufgabe 3

Der Triple-DES (oder auch 3DES) ist folgendermaßen definiert:

$$c = 3DES(m) = DES_{K_1}(DES_{K_2}^{-1}(DES_{K_1}(m)))$$

Angenommen, ein Angreifer kennt ein zusammengehöriges Paar (c, m) (sog. Known-Plaintext Angriff). Wie hoch ist beim Triple-DES der Aufwand für einen Meet-in-the-Middle Angriff verglichen mit dem einer vollständigen Schlüsselsuche?

Lösung

$$\begin{aligned} c &= DES_{K_1}(DES_{K_2}^{-1}(DES_{K_1}(m))) & | DES_{K_1}^{-1}() \\ DES_{K_1}^{-1}(c) &= DES_{K_2}^{-1}(DES_{K_1}(m)) \end{aligned}$$

linke Seite	rechte Seite
$DES_{K_1}^{-1}(c)$	$DES_{K_2}^{-1} \cdot DES_{K_1}(m)$
$\vdots$	$\vdots$
(alle 2^{56} mögl. K_1)	(alle $2^{56} \cdot 2^{56}$ mögl. k_1 und k_2 Kombis)

Rechenaufwand: 2^{112}

Speicheraufwand: 2^{112}

Mit Three-key-Triple-DES:

$$\begin{aligned} c &= DES_{K_3}(DES_{K_2}^{-1}(DES_{K_1}(m))) & | DES_{K_3}^{-1}() \\ DES_{K_3}^{-1}(c) &= DES_{K_2}^{-1}(DES_{K_1}(m)) \end{aligned}$$

$\Rightarrow$ Aufwand für Meet-in-the-Middle ist 2^{112} , obwohl es 3 Schlüssel gibt.

11.5.4 Aufgabe 4

Betrachten Sie eine lange, in Blöcke aufgeteilte Nachricht $m = b_1b_2b_3\dots b_n$. Sie verschlüsseln diese Nachricht mit einer Blockchiffre in unterschiedlichen Betriebsmodi. Angenommen, bei der Übertragung eines Chiffreblocks c_i geschieht ein Fehler. Wie wirkt sich dieser Fehler auf die Klartextblöcke aus, die der Empfänger im CBC, CFB, OFB und ECB erhält?

Lösung

$$m = b_1b_2b_3\dots b_n$$

$$1. \text{ ECB-Mode: } c_i = f_K(b_i) \Rightarrow b_i = f_K^{-1}(c_i)$$

$$\tilde{b}_i = f_K^{-1}(\tilde{c}_i) \Rightarrow \text{ein gestörter Klartextblock}$$

$$\tilde{b}_{i+1} = f_K^{-1}(c_{i+1}) \Rightarrow \text{nächster Klartextblock ist wieder in Ordnung}$$

$$2. \text{ CBC-Mode: } c_i = f_K(c_{i-1} \oplus b_i) \Rightarrow b_i = f_K^{-1}(c_i) \oplus c_{i-1}$$

$$\tilde{b}_i = f_K^{-1}(\tilde{c}_i) \oplus c_{i-1} \Rightarrow \text{ein gestörter Klartextblock}$$

$$\tilde{b}_{i+1} = f_K^{-1}(\tilde{c}_{i+1}) \oplus \tilde{c}_i \Rightarrow \text{ein weiterer gestörter Klartextblock}$$

3. CFB-Mode: $b_i = c_i \oplus f_K(c_{i-1})$
 $\tilde{b}_i = \tilde{c}_i \oplus f_K(c_{i-1}) \Rightarrow$ ein gestörter Klartextblock
 $\tilde{b}_{i+1} = \tilde{c}_{i+1} \oplus f_K(\tilde{c}_i) \Rightarrow$ ein weiterer gestörter Klartextblock
4. OFB-Mode: $b_i = c_i \oplus f_K^i(IV)$
Ein Bitfehler in $c_i \Rightarrow$ Ein Bitfehler in b_i
5. CTR-Mode: $b_i = c_i \oplus f_K(IV || i)$
Ein Bitfehler in $c_i \Rightarrow$ Ein Bitfehler in b_i

11.5.5 Aufgabe 5

Schreiben Sie ein Java-Programm, welches einen gegebenen Klartext mit dem DES-Algorithmus verschlüsselt. Eine Implementation des DES finden Sie im Paket `javax.crypto` in der Klasse `Cipher`. Nutzen Sie zur Erzeugung eines DES-Schlüssels die Klasse `KeyGenerator`.

Lösung

1. Erzeuge Instanz der Klasse `Cipher` mit `factory-method`
2. Erzeuge DES-key mit Klasse `KeyGenerator`
 - Zufällige Erzeugung oder
 - Spezifikation des key in byte-Array
3. Transferiere Klartext in byte-Array und übergebe an erzeugte `Cipher`-Instanz.

Listing 11.2: Code Beispiel im Umgang mit DES

```

1 import java.security.*;
2 import javax.crypto.*;
3 import javax.crypto.spec.*;
4 import sun.misc.*; //needed for Base64 Encoding
5
6 class DESTest {
7     public static void main(String[] args) throws Exception {
8
9         //generate a random DES key:
10        /* KeyGenerator generator = KeyGenerator.getInstance("DES");
11           generator.init(new SecureRandom());
12           Key key = generator.generateKey(); */
13
14        //using a pre-specified key instead:
15        byte[] rawkeybytes = new byte[] { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55,
16                                           0x66, 0x77 };
17        SecretKey key = new SecretKeySpec(rawkeybytes, "DES");
18
19        //obtain a cipher:
20        Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
21        cipher.init(Cipher.ENCRYPT_MODE, key);
22
23        //Encrypt:
24        byte[] clear = args[0].getBytes();
25        byte[] cipherbytes = cipher.doFinal(clear);
26

```

```

27 //Encode cipherbytes in Base64:
28 BASE64Encoder encoder = new BASE64Encoder();
29 String cipherstring = encoder.encode(cipherbytes);
30
31 byte[] keybytes = key.getEncoded();
32 String keystring = encoder.encode(keybytes);
33
34 //Decrypt:
35 cipher.init(Cipher.DECRYPT_MODE, key);
36 byte[] clearbytes = cipher.doFinal(cipherbytes);
37 String cleartext = new String(clearbytes);
38
39 //Output:
40 System.out.println("Ciphertext: " + cipherstring);
41 System.out.println("Key used: " + keystring);
42 System.out.println("Key Length: " + keybytes.length * 8 + "Bits");
43 System.out.println("Raw Bytes of Key:");
44 for (int i = 0; i < keybytes.length; i++) {
45     System.out.print(keybytes[i] + " ");
46 }
47 System.out.println();
48 System.out.println("Deciphered Cleartext: " + cleartext);
49 }
50 }

```

11.6 Übungsblatt 6

11.6.1 Aufgabe 1

Geburtstagsparadoxon: Es befinden sich n Personen in einem Raum (z.B. einem Hörsaal). Wie groß muss n sein, damit mit Wahrscheinlichkeit $> 50\%$ zwei Personen am gleichen Tag Geburtstag haben? Betrachten Sie dabei die folgenden Fälle:

- a) Ein Jahr hat 365 Tage.
- b) Ein Jahr hat k Tage.

Was hat das Geburtstagsparadoxon mit Hashfunktionen zu tun?

Lösung

- a) Ein Paar am gleichen Tag Geburtstag: $p = \frac{1}{365}$

$$\text{Anzahl Paare: } \frac{n(n-1)}{2} = \frac{n}{2}$$

Wahrscheinlich, dass *kein* Paar am gleichem Tag Geburtstag hat: $p = \left(\frac{364}{365}\right)^{\frac{n(n-1)}{2}}$

$$\Rightarrow \text{Gesuchte Wahrscheinlichkeit ist } p = 1 - \left(\frac{364}{365}\right)^{\frac{n(n-1)}{2}} > \frac{1}{2}$$

$$\Leftrightarrow \begin{aligned} \left(\frac{364}{365}\right)^{\frac{n(n-1)}{2}} &< \frac{1}{2} & |\log() \\ \frac{n(n-1)}{2} \cdot \log\left(\frac{364}{365}\right) &< \log\left(\frac{1}{2}\right) \\ n(n-1) &> \frac{2 \cdot \log\left(\frac{1}{2}\right)}{\log\left(\frac{364}{365}\right)} \end{aligned}$$

$$n^2 \cdot c - 505 > 0$$

$$n_{0.5} = \frac{1}{2} \pm \sqrt{\frac{1}{4} + 505} \approx 22,98$$

$$\Rightarrow n \geq 23 \text{ Personen} \Rightarrow \text{Wahrscheinlichkeit} > 0,5$$

$$\text{b) Gesuchte Wahrscheinlichkeit ist } p = 1 - \left(\frac{k-1}{k}\right)^{\frac{n(n-1)}{2}} > \frac{1}{2}$$

$$\Leftrightarrow \begin{aligned} \left(\frac{k-1}{k}\right)^{\frac{n(n-1)}{2}} &< \frac{1}{2} & | \log() \\ \frac{n(n-1)}{2} \cdot \log\left(\frac{k-1}{k}\right) &< \log\left(\frac{1}{2}\right) \\ n(n-1) &> \frac{2 \cdot \log\left(\frac{1}{2}\right)}{\log\left(\frac{k-1}{k}\right)} \approx \frac{-1,39}{\log\left(\frac{1}{2}\right)} \end{aligned}$$

$$\text{Für kleine } x \text{ gilt: } \log\left(1 - \frac{1}{k}\right) \approx \frac{1}{2}x = \frac{1}{2}x$$

$$\text{Hier: } \log\left(\frac{k-1}{k}\right) = \log\left(1 - \frac{1}{k}\right)$$

$$\Rightarrow \text{Gesuchte Wahrscheinlichkeit: } n(n-1) > (-k) \cdot (-1,386) = 1,386 \cdot k$$

$$\Rightarrow n^2 - n - 1,386 > 0$$

$$n_{0.5} = \frac{1}{2} \pm \underbrace{\sqrt{\frac{1}{4} + 1,386k}}_{\approx \sqrt{k}} \Rightarrow n > \sqrt{k} \text{ für grosse } k$$

Zusammenhang mit Hashfunktionen:

Hashwert eines Strings = Geburtstag

Kollision zweier Hashwerte = Gemeinsamer Geburtstag zweier Personen

$\Rightarrow$ Man muss dafür sorgen, dass man einen langen Hashwert hat.

11.6.2 Aufgabe 2

Bestimmen Sie mittels eines Java-Programms die Hashwerte des Strings HELLO_WORLD! unter den Hashfunktionen md5 bzw. SHA-1, jeweils byte-Array.

Lösung

Implementationen von Hashfunktionen befinden sich im package `java.security`.

Listing 11.3: Code für Message Digest (MD5 und SHA-1)

```
1 import java.security.*;
2 import java.io.*;
3
4 public class MessageDigestTest {
5
6     public static String eingabe() throws Exception {
7
8         System.out.println("Please type in your Cleartext.");
9         BufferedReader reader = new BufferedReader(new InputStreamReader(
10             System.in));
11         String cleartext = reader.readLine();
12         return cleartext;
```

```

13 }
14
15 public static byte[] md5Hash(String s) throws NoSuchAlgorithmException {
16     byte[] hashvalue;
17     MessageDigest md5;
18     md5 = MessageDigest.getInstance("MD5");
19     hashvalue = md5.digest(s.getBytes());
20     return hashvalue;
21 }
22
23 public static byte[] sha1Hash(String s) throws NoSuchAlgorithmException {
24     byte[] hashvalue;
25     MessageDigest sha1;
26     sha1 = MessageDigest.getInstance("SHA1");
27     hashvalue = sha1.digest(s.getBytes());
28     return hashvalue;
29 }
30
31 public static void main(String[] args) throws Exception {
32
33     String s = eingabe();
34     byte[] hash = md5Hash(s);
35     System.out.println("MD5-Digest Value of your Input string:");
36     for (int i = 1; i < hash.length; i++) {
37         System.out.print(hash[i] + " ");
38     }
39     System.out.println();
40     System.out.println(hash.length + " Bytes");
41
42     System.out.println();
43     hash = sha1Hash(s);
44     System.out.println("SHA-1-Digest Value of your Input string:");
45     for (int i = 1; i < hash.length; i++) {
46         System.out.print(hash[i] + " ");
47     }
48     System.out.println();
49     System.out.println(hash.length + " Bytes");
50
51 }
52 }

```

11.6.3 Aufgabe 3

Schreiben Sie ein Java-Programm, das für einen beliebig langen String m einem Message Authentication Code in folgender Form liefert:

$$MAC(m) = DES_K(SHA-1(m)),$$

wobei ein K ein DES -Key ist.

Lösung

Listing 11.4: Code für SHA-1 Hash und MAC

```

1 import java.security.*;
2 import javax.crypto.*;
3 import sun.misc.*;
4
5 class DesSha1Mac {
6
7     //computes a DES-SHA.1 based MAC over a given message

```

```

8  public static byte[] computeHash(String message) throws Exception {
9
10     byte[] hashvalue;
11     MessageDigest sha1;
12     sha1 = MessageDigest.getInstance("SHA1");
13     hashvalue = sha1.digest(message.getBytes());
14     return hashvalue;
15 }
16
17 public static String computeMAC(byte[] hash) throws Exception {
18
19     //generate a DES key:
20     KeyGenerator generator = KeyGenerator.getInstance("DES");
21     Key key;
22     generator.init(new SecureRandom());
23     key = generator.generateKey();
24
25     //obtain a cipher:
26     Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
27     cipher.init(Cipher.ENCRYPT_MODE, key);
28
29     //Encrypt Hashvalue:
30     byte[] macbytes = cipher.doFinal(hash);
31     System.out.println("MAC has " + macbytes.length + " bytes");
32
33     //Encode macbytes in Base64:
34     BASE64Encoder encoder = new BASE64Encoder();
35     String macstring = encoder.encode(macbytes);
36
37     return macstring;
38 }
39
40 public static void main(String[] args) throws Exception {
41
42     byte[] hashvalue = computeHash(args[0]);
43     String mac = computeMAC(hashvalue);
44     System.out.println("This is the DES-SHA-1 MAC of your cleartext:");
45     System.out.println(mac);
46 }
47 }

```

11.7 Übungsblatt 7

11.7.1 Aufgabe 1

Gegeben seien die Primzahlen $p = 23$ und $q = 37$. Berechnen Sie aus p und q ein RSA-Schlüsselpaar und verschlüsseln Sie die Nachricht $m = 4$.

Lösung

$$p = 23, q = 37$$

$$(i) \ n = p \cdot q = 851, \ \phi(n) = (p - 1)(q - 1) = 792$$

(ii) Wähle zu 792 teilerfremdes e :

$$e = 5$$

(iii) Invertiere $e = 5 \bmod 792$

$$\begin{aligned} 792 &= 158 \cdot 5 + 2 \Rightarrow 2 = 792 - 5 \cdot 158 \\ 5 &= 2 \cdot 2 + 1 \Rightarrow 1 = 5 - 2 \cdot (792 - 5 \cdot 158) \\ &= 317 \cdot 5 - 2 \cdot 792 \\ &= e^{-1} \bmod 792 = d \end{aligned}$$

(iv) $PK = (5, 851)$
 $SK = 317$

Verschlüssele $m = 4$:

$$c = 4^5 \bmod 851 = 1024 \bmod 851 = 173$$

11.7.2 Aufgabe 2

Sie bekommen von Ihrem Arbeitgeber den RSA-Public Key $(e, n) = (3, 14803)$ und den Secret Key $d = 9707$ zugewiesen. Faktorisieren Sie mit Hilfe dieser Werte den Modul n .

Lösung

$$\underbrace{(e, n) = (3, 14803)}_{PK}, \underbrace{d = 9707}_{SK}$$

$$\text{Schätze } \phi(n) = (p-1)(q-1) = pq - p - q + 1 \approx n - 2\sqrt{n}$$

(Man nutzt aus, dass die Primfaktoren p und q etwa in der Grössenordnung $\sqrt{n}$ sind)

$$\text{Hier: } \phi(n) \approx 14563$$

$$\begin{aligned} \text{Wir wissen: } e - d \bmod \phi(n) &\Leftrightarrow ed - k \cdot \phi(n) = 1 \quad (*) \\ &\Leftrightarrow ed \approx k \cdot \phi(n) \end{aligned}$$

$$\text{Hier: } ed = 29121 \approx k \cdot 14563 \Rightarrow k = 2$$

$$\text{Einsetzen in } (*): 29121 = 2 \cdot \phi(n) + 1 \Rightarrow \phi(n) = 14560$$

Bestimme dann p, q . Siehe Abschnitt [11.7.3](#).

11.7.3 Aufgabe 3

Die Zahl $n = 14803$ besitzt zwei Primfaktoren p und q . Es ist $\phi(n) = 14560$. Bestimmen Sie p und q !

Lösung

$$\begin{aligned}\phi(n) &= 14560 = \overbrace{p \cdot q}^{\phi(n)} - p - q + 1 \\ 14560 &= 14560 - p - q + 1\end{aligned}$$

$$\Rightarrow \begin{array}{l} i) \quad p + q = 244 \\ ii) \quad p \cdot q = 14560 \end{array} \Rightarrow q = 244 - p \quad (\text{Setze } q \text{ in } ii)$$

$$\Rightarrow p \cdot (244 - p) = 14803$$

$$\Rightarrow p^2 - 244p + 14803 = 0$$

$$\Rightarrow p_{1,2} = 122 \pm \sqrt{122^2 - 14803} = \underbrace{131}_p - \underbrace{113}_q$$

$\Rightarrow \phi(n)$ muss also genauso geheim gehalten werden wie $\phi(n)$ (TODO mitschriebfehler überprüfen!).

11.7.4 Aufgabe 4

Erstellen Sie eine Klasse `RSAPublicKey` in Java. Schreiben Sie insbesondere einen geeigneten Konstruktor für die Klasse, der aus zwei gegebenen Primzahlen ein RSA-Schlüsselpaar erzeugt.

Hinweis: Benutzen Sie die Klasse `BigInteger` aus `java.math`. Diese Klasse enthält u.a. auch einen Konstruktor `BigInteger(int bitLength, int certainty, Random rnd)`, der zufällig eine `BigInteger`-Zahl der Länge `bitLength` erzeugt, die mit der Wahrscheinlichkeit $1 - 2^{-\text{certainty}}$ prim ist. Sehr nützlich sind auch die Methoden `modInverse` für modulare Inversenbildung und `modPow` für modulares Potenzieren.

Lösung

Listing 11.5: `RSAPublicKey`

```
1 import java.math.*;
2 import java.util.*;
3
4 public class RSAPublicKey {
5
6     public BigInteger e, n, d, p, q;
7
8     public RSAPublicKey(BigInteger p1, BigInteger q1) {
9         p = p1;
10        q = q1;
11        Random rnd = new Random();
12        n = p1.multiply(q1); // Public module
13        BigInteger phi = (p1.subtract(BigInteger.ONE)).multiply(q1
14            .subtract(BigInteger.ONE));
15        // compute Phi-Function
16        e = new BigInteger(3, rnd); // Public Exponent
17        while ((e.equals(BigInteger.ONE))
18            || !((e.gcd(phi)).equals(BigInteger.ONE))) {
19            e = new BigInteger(3, rnd);
20        }
```

```

21     d = e.modInverse(phi); //Private Key
22 }
23 }

```

11.7.5 Aufgabe 5

Schreiben Sie ein Java-Programm, welches eine als Zahl gegebene Nachricht mit dem RSA-Algorithmus verschlüsselt. Erzeugen Sie sich Ihr Schlüsselpaar selbst. Benutzen Sie dazu die Klasse `RSAPublicKey` aus Abschnitt 11.7.4.

Lösung

Listing 11.6: RSACrypt

```

1  import java.math.*;
2  import java.util.*;
3
4  public class RSACrypt {
5
6      static Random rnd = new Random();
7      static BigInteger p = new BigInteger(50, 50, rnd);
8      static BigInteger q = new BigInteger(50, 50, rnd);
9
10     static sign.RSAPublicKey rsaKeys = new sign.RSAPublicKey(p, q);
11
12     public static BigInteger rsaEncrypt(BigInteger message) {
13         BigInteger c = message.modPow(rsaKeys.e, rsaKeys.n);
14         return c;
15     }
16
17     public static String rsaDecrypt(BigInteger c) {
18         BigInteger m = c.modPow(rsaKeys.d, rsaKeys.n);
19         String message = m.toString();
20         return message;
21     }
22
23     public static void main(String[] args) throws Exception {
24
25         System.out.println("Bitte geben Sie die zu verschlüsselnde Zahl ein.");
26         BigInteger cleartext = new BigInteger(sign.Input.stringEingabe());
27         System.out.println();
28         System.out.println("Klartext: " + cleartext);
29         BigInteger cipher = rsaEncrypt(cleartext);
30         String ciphertext = cipher.toString();
31         System.out.println("Ciphertext: " + ciphertext);
32
33         String clear = rsaDecrypt(cipher);
34         System.out.println("Deciphered Cleartext: " + clear);
35         System.out.println("Keypair used: Public Key (e,n) = (" + rsaKeys.e
36             + ", " + rsaKeys.n + ")");
37         System.out.println("Private Key d = (" + rsaKeys.d + ")");
38     }
39 }

```

11.8 Übungsblatt 8

11.8.1 Aufgabe 1

Berechnen Sie folgende diskrete Logarithmen:

- a) Logarithmus von 6 zur Basis 3 (modulo 7)
- b) Logarithmus von 1 zur Basis 2 (modulo 11)

Lösung

→ a) Suche x mit $3^x \bmod 7 = 6$ und $x \in \{0, 1, \dots, 6\}$

$$\begin{aligned}x = 0: & 3^0 \bmod 7 = 1 \\x = 1: & 3^1 \bmod 7 = 3 \\x = 2: & 3^2 \bmod 7 = 2 \\x = 3: & 3^3 \bmod 7 = 6 \Rightarrow x = 3\end{aligned}$$

→ b) Suche x mit $2^x \bmod 11 = 1$ und $x \in \{0, 1, \dots, 10\}$

$$\begin{aligned}x = 0: & 2^0 \bmod 11 = 1 \Rightarrow x = 0 \quad \text{triviale Lösung. Weitere Lösungen?} \\x = 1: & 2^1 \bmod 11 = 2 \\x = 2: & 2^2 \bmod 11 = 4 \\x = 3: & 2^3 \bmod 11 = 8 \\x = 4: & 2^4 \bmod 11 = 5 \\x = 5: & 2^5 \bmod 11 = 10 \\x = 6: & 2^6 \bmod 11 = 9 \\x = 7: & 2^7 \bmod 11 = 7 \\x = 8: & 2^8 \bmod 11 = 3 \\x = 9: & 2^9 \bmod 11 = 6 \\x = 10: & 2^{10} \bmod 11 = 1 \Rightarrow x = 10\end{aligned}$$

Satz



p Primzahl, $m < p \Rightarrow m^{p-1} \bmod p = 1$

11.8.2 Aufgabe 2

Schreiben Sie ein Java-Programm `RSASign`, mit dem Sie beliebige Strings signieren und als Byte-Array gegebene Signaturen verifizieren können.

Ihr Programm sollte also folgende Funktionen enthalten:

- `public static byte[] sign(String cleartext, RSAKeyPair signatureKey)`
- `public static boolean verify(String cleartext, byte[] signaturebytes, BigInteger e, BigInteger n)`

Verwenden Sie MD5 oder SHA-1 aus `java.security` als Hash-Funktionen.

Lösung

- `sign()`:
 1. Hashe Input-String
 2. Übersetze Hashwert in Array von `BigInteger`-Zahlen `s[]`
(Man braucht ein Array, da ansonsten die Zahl möglicherweise grösser als das Modul wird.)
 3. Jedes `s[i] < Modul n`
 4. Signiere: Bilde für alle `i`: $s[i] = s[i]^d \bmod n$
 5. Übersetze jedes `s[i]` in Byte-Array `signaturebytes`
 6. Ergebnis: 2-dimensionaler Byte-Array
- `verify()`:
 1. Teile `signaturebytes` auf in 4 Byte-Blöcke
 2. Erzeuge `BigInteger`-Zahl `signaturevalue` auf jedem Block
 3. Für jeden `signaturevalue` bilde: $\text{verifyvalue} = \text{signaturevalue}^e \bmod n$
 4. Erzeuge wie beim Signieren `BigInteger`-Zahlen aus Hashwert des Klartexts
 $\Rightarrow \text{BigInteger}[] \text{ test};$
 5. Vergleiche `test` mit `verifyvalue`

Listing 11.7: RSASign

```

1 import java.math.*;
2 import java.security.*;
3 import java.util.*;
4
5 public class RSASign {
6
7     public static RSAKeyPair generateKey() {
8         Random rnd = new Random();
9         BigInteger p = new BigInteger(50, 50, rnd);
10        BigInteger q = new BigInteger(50, 50, rnd);
11        RSAKeyPair signingkey = new RSAKeyPair(p, q);
12        return signingkey;
13    }
14
15    public static byte[][] sign(String cleartext, RSAKeyPair signatureKey)
16        throws NoSuchAlgorithmException {
17        byte[] hashvalue = Hashing.md5Hash(cleartext); // hashvalue consists of
18                                                    // 20 bytes
19        byte[] b = new byte[4]; // array b represents one block
20        BigInteger[] s = new BigInteger[hashvalue.length / b.length];
21        byte[][] signaturebytes = new byte[(hashvalue.length / b.length)][20];
22
23        for (int i = 0; i < (hashvalue.length / b.length); i++) {
24            for (int j = 0; j < b.length; j++) {
25                b[j] = hashvalue[i * b.length + j];
26            }
27        }
28    }
29
30 }
```

```

28         BigInteger m = new BigInteger(b); // create one BigInteger out of
29                                           // one Block
30         if (m.compareTo(BigInteger.ZERO) < 0) {
31             m = m.add(signatureKey.n);
32         }
33         s[i] = m.modPow(signatureKey.d, signatureKey.n); // signature of
34                                                         // one Block
35         signaturebytes[i] = s[i].toByteArray();
36     }
37
38     return signaturebytes;
39 }
40
41 /**
42  * implements fast RSA signing algorithm used in PGP cf. Ertel, p.82
43  */
44 public static byte[][] fastSign(String cleartext, RSAKeyPair signatureKey)
45     throws NoSuchAlgorithmException {
46     byte[] hashvalue = Hashing.md5Hash(cleartext); // hashvalue consists of
47                                                    // 20 bytes
48     byte[] b = new byte[4];
49     BigInteger[] s = new BigInteger[hashvalue.length / b.length];
50     byte[][] signaturebytes = new byte[(hashvalue.length / b.length)][20];
51
52     BigInteger s1, s2, h;
53     BigInteger pstar = signatureKey.p.modInverse(signatureKey.q);
54
55     for (int i = 0; i < (hashvalue.length / b.length); i++) {
56         for (int j = 0; j < b.length; j++) {
57             b[j] = hashvalue[i * b.length + j];
58         } // array b represents one block
59
60         BigInteger m = new BigInteger(b); // create one BigInteger out of
61                                           // one Block
62         if (m.compareTo(BigInteger.ZERO) < 0) {
63             m = m.add(signatureKey.n);
64         }
65
66         s1 = m.modPow(signatureKey.d, signatureKey.p);
67         s2 = m.modPow(signatureKey.d, signatureKey.q);
68         h = (pstar.multiply(s2.subtract(s1))).mod(signatureKey.q);
69         // signature of one Block
70         s[i] = (s1.add(signatureKey.p.multiply(h))).mod(signatureKey.n);
71         signaturebytes[i] = s[i].toByteArray();
72     }
73
74     return signaturebytes;
75 }
76
77 public static boolean verify(String cleartext, byte[][] signaturebytes,
78     BigInteger e, BigInteger n) throws NoSuchAlgorithmException {
79     boolean verified = false;
80     byte[] hashvalue1 = Hashing.md5Hash(cleartext);
81     byte[] b = new byte[4];
82     BigInteger[] test = new BigInteger[(hashvalue1.length / b.length)];
83
84     // create array of BigIntegers out of hash of cleartext
85     for (int i = 0; i < (hashvalue1.length / b.length); i++) {
86         for (int j = 0; j < b.length; j++) {
87             b[j] = hashvalue1[i * b.length + j];
88         } // array b represents one block
89
90         test[i] = new BigInteger(b);
91         if (test[i].compareTo(BigInteger.ZERO) < 0)
92             // Testvalues must not be negative,
93             // therefore add the public module n
94         {

```

```

95         test[i] = test[i].add(n);
96     }
97 }
98
99 BigInteger[] signaturevalue = new BigInteger[signaturebytes.length];
100 BigInteger[] verifyvalue = new BigInteger[signaturebytes.length];
101
102 for (int i = 0; i < signaturebytes.length; i++) {
103     signaturevalue[i] = new BigInteger(signaturebytes[i]);
104 }
105
106 for (int i = 0; i < verifyvalue.length; i++) {
107     verifyvalue[i] = signaturevalue[i].modPow(e, n);
108 }
109
110 // compare verifyvalue with test
111 for (int i = 0; i < test.length; i++) {
112     verified = (test[i].equals(verifyvalue[i]));
113     if (!verified) {
114         break;
115     }
116 }
117
118 return verified;
119 }
120
121 public static void signatureOutput(byte[][] sig) {
122     for (int i = 0; i < sig.length; i++) {
123         for (int j = 0; j < sig[0].length; j++) {
124             System.out.print(sig[i][j] + "\t");
125         }
126         System.out.println();
127     }
128 }
129
130 public static void main(String[] args) throws Exception {
131     RSAKeyPair signatureKey = generateKey();
132     System.out.println("Please type in your cleartext.");
133     String cleartext = Input.stringEingabe();
134     byte[][] signature = sign(cleartext, signatureKey);
135     System.out.println("This is the signature of your cleartext.");
136     signatureOutput(signature);
137     if (verify(cleartext, signature, signatureKey.e, signatureKey.n)) {
138         System.out.println("Signature has been succesfully verified.");
139     } else {
140         System.out.println("Signature could not be verified");
141     }
142
143     System.out.println();
144
145     signature = fastSign(cleartext, signatureKey);
146     System.out.println("This is the FAST signature of your cleartext.");
147     signatureOutput(signature);
148     if (verify(cleartext, signature, signatureKey.e, signatureKey.n)) {
149         System.out.println("Signature has been succesfully verified.");
150     } else {
151         System.out.println("Signature could not be verified");
152     }
153 }
154 }

```

11.8.3 Aufgabe 3

Die Firma TC Trustcenter (<http://www.trustcenter.de>) bietet verschiedene Zertifikatsklassen für Firmen und Privatleute an. Finden Sie anhand der "Zertifizierungsrichtlinien" von TC Trustcenter heraus, in welcher Hinsicht sich diese Zertifikatsklassen unterscheiden.

Lösung

Je höher die Zertifikatsklasse, desto umfangreicher ist die Überprüfung der Identität des Beantragenden.

Class 0-Zertifikate:	Keine Prüfung
Class 1-Zertifikate:	Überprüfen der Email-Adresse
Class 2-Zertifikate:	Kopie eines Ausweises muss eingereicht werden, oder Zusicherung durch von TC TrustCenter zugelassene Dritte
Class 3-Zertifikate:	Persönliches Erscheinen der Person mit Ausweis oder beglaubigte Kopien des Handelsregisters

11.8.4 Aufgabe 4

Welche der folgenden E-Mail Anbieter unterstützen Digitale Signaturen? Auf welche Weise wird die Identität der Zertifikatsinhaber geprüft?

- a) web.de
- b) gmx
- c) T-Online

Lösung

- a) web.de: siehe <http://trust.web.de>. Es wird die postalische Adresse überprüft.
- b) gmx: scheint keine Zertifikate zu unterstützen
- c) T-Online: Zertifikate stehen nur noch Benutzern zur Verfügung, die bereits ein Zertifikat beantragt haben, wird aber nicht mehr angeboten. Es hiess damals secureMail und war kostenpflichtig.

11.8.5 Aufgabe 5

Surfen Sie zur Regulierungsbehörde für Telekommunikation und Post (<http://www.regtp.de>) und besorgen Sie sich folgende Informationen:

- Wie lang ist der Modul n des am 16.11.2001 im Bundesanzeiger veröffentlichten Public Keys der RegTP? Wie lautet der Encryption Exponent e in dezimaler Schreibweise?
- Welche Zertifizierungsdiensteanbieter (ZDAs) sind zur Zeit bei der RegTP akkreditiert?
- Welche anerkannten Prüfstellen können die Konformität eines ZDA mit dem Signaturgesetz/ der Signaturverordnung bestätigen?
- Welche Algorithmen und welche Schlüssellängen werden für die Erfüllung der Anforderungen des Signaturgesetzes als geeignet angesehen?
- Was versteht man unter "einfachen", "fortgeschrittenen", "qualifizierten" und "akkreditierten" Signaturen? Wie ist jeweils ihr rechtlicher Status?

Lösung

→ a) TODO aufgabe noch fertig machen

11.9 Übungsblatt 9

11.9.1 Aufgabe 1

Besorgen Sie sich die neueste PGP-Freeware Version bei <http://www.pgpi.org>. Überlegen Sie, wie Sie sicherstellen könnten, dass Sie eine **authentische** Version von PGP bei sich installieren.

Lösung

PGP-Download ist mit PGP signiert!

Um Download zu authentifizieren, ist die Version integral! CHECKME

- md5 Vergleich
- guter Freund der PGP hat
- Sourcecode-Commandozeile → Build → dann xxxx CHECKME Version
- einfache Bestellung

11.9.2 Aufgabe 2

Im Jahr 2000 wurde von Ralf Senderek ein Angriff auf das ADK Feature veröffentlicht. (s. <http://senderek.de/security/key-experiments.html>) Versuchen Sie den Angriff mit eigenen Worten zu beschreiben.

Lösung

ADK = 2.ter Public Key (mit 2tem Empfänger)

1. PGP $\rightarrow$ SessionKey KG
2. Bilde $E_{PK_{Bob}}(KG)$ und zusätzlich $E_{ADK}(KG)$
 - Empfängerliste (z.B. 20 Businesspartner)
 - ADK wird bei Mails hinzugebunden (ADK wird mit Public Key signiert!)

PK $\begin{cases} \text{signierte Elemente} \\ \text{unsignede Elemente (z.B. Bild)} \end{cases}$

Unsignede Elemente können verändert werden!

2 mal verschlüsselte Mails:

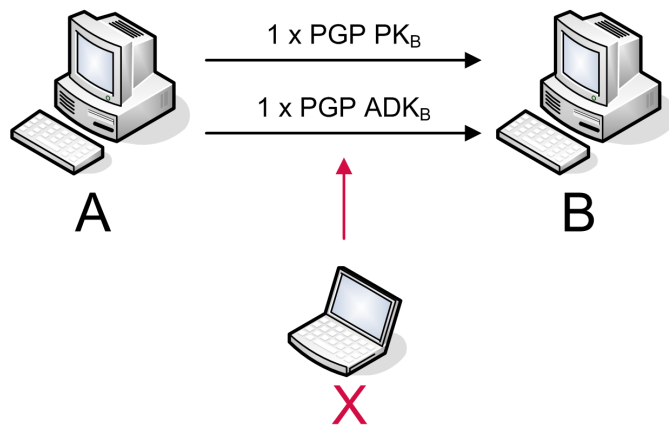


Abbildung 11.1: Einschleusen des ADK in 2 mal verschlüsselte Mail

In Abbildung 11.1 hat **X** den ADK eingeschleust und kann gleiche Mail entschlüsseln!

(nichts mit signieren zu tun!)

<http://www.pgpdump.net>

$\rightarrow$ key einfügen $\rightarrow$ Output

11.9.3 Aufgabe 3

Sie möchten Ihren PGP Private Key mit einer Passphrase schützen, die dieselbe Sicherheit bietet wie ein zufälliger 128 Bit Key.

- a) Aus wieviel **Buchstaben** der Menge a, b, c, ..., z (insges. 26 Buchstaben) müsste die Passphrase bestehen, wenn die Buchstaben **zufällig** gewählt werden?
- b) Wieviele Zeichen braucht man, wenn man Groß- und Kleinbuchstaben (52 Zeichen) zulässt? (mit Sonderzeichen: 70)
- c) Aus wievielen **Wörtern** eines Wörterbuch mit 60000 Einträgen müsste die Passphrase bestehen?

Lösung

$$\rightarrow \text{a) } 2^{128} = 26^x$$

Passphrase $\rightarrow$ Hash $\rightarrow$ 128 Bit $\Rightarrow$ symmetrischer Schlüssel K zum Schutz des Private Key

$$x \cdot \ln(26) = 128 \cdot \ln(2) \Rightarrow x = \frac{128 \cdot \ln(2)}{\ln(26)} = 27,2$$

$$\rightarrow \text{b) } 52^x \stackrel{!}{=} 2^{128} \Rightarrow x = 22,45$$

Bei allen Sonderzeichen $\Rightarrow 12$

$$\rightarrow \text{c) } 60000^x \stackrel{!}{=} 2^{128} \mid \ln() \Rightarrow x = 8,06$$

11.9.4 Aufgabe 4

Welches Sicherheitsniveau (in Bit) bietet ein 5-Buchstabiges Passwort aus Kleinbuchstaben?

Lösung

$$\begin{aligned} 26^5 &= 2^x \quad \mid \ln() \\ 5 \cdot \ln(26) &= x \cdot \ln(2) \\ x &= \frac{5 \cdot \ln(26)}{\ln(2)} \\ &\approx 23,5 \end{aligned}$$

11.10 Übungsblatt 10

11.10.1 Aufgabe 1

Die folgenden Rechenzeiten für kryptografische Operationen wurden auf einem 500 MHz Pentium III Rechner ermittelt:

- MD5 Hashing: 72 500 Kbyte/sec
- SHA-1 Hashing: 41 000 Kbyte/sec
- RSA Public Key Anwendung bei 1024 Bit Modul: 8,5 Kbyte/sec
- RSA Private Key Anwendung bei 1024 Bit Modul: 5,8 Kbyte/sec

Berechnen Sie, ausgehend von diesen Daten, die ungefähre Zeit, die für einen SSL-Handshake benötigt wird. Gehen Sie von folgenden Voraussetzungen aus:

- Größe des Server- Zertifikats: 500 Byte
- Einstufige Zertifikathierarchie (d.h. der Server präsentiert ein Zertifikat von einer CA, der der Client vertraut)
- Keine Client-Authentisierung (d.h. es wird kein Client-Zertifikat präsentiert)
- Gewählte CipherSuite: FixedRSA
- Gewählter Signaturalgorithmus: RSA mit SHA-1.

Lösung

(1) **ClientHello**

(2) **ServerHello**

(3) **Certificate**: 500 Byte Klartext

Client verifiziert Signature der CA unter Server-Zertifikat:

- Hashe 500 Byte Klartext: $t = \frac{0,5 \text{ KB}}{41000 \text{ KB/s}} = 0,0122 \text{ ms}$
- Wende PK_{CA} auf Signature in Certificate an: $t = \frac{20 \text{ B}}{8,5 \text{ KB/s}} = 2,36 \text{ ms}$

(4) *entfällt* (wegen fixedRSA)

(5) *entfällt* (weil wir kein Zertifikat anfordern)

(6) **ServerDone**

(7) *entfällt*

(8) **ClientKeyExchange**: Verschlüssele 48 Byte mit PK_{Server} : $t = \frac{48 \text{ B}}{8,5 \text{ KB/s}} = 5,65 \text{ ms}$

(9) *entfällt*

(10) Server entschlüsselt PMS : $t = \frac{48 \text{ B}}{5,8 \text{ KB/s}} = 8,28 \text{ ms}$

Ableiten des MS aus PMS :

$$MS = \overbrace{MD5(\overbrace{PMS}^{48 \text{ B}} \parallel \overbrace{SHA('A' \parallel PMS \parallel RAND_{Client} \parallel RAND_{Server})}^{112 \text{ B}})}^{68 \text{ B}} \parallel \overbrace{MD5(PMS \parallel \overbrace{SHA('BB' \parallel PMS \parallel RAND_{Client} \parallel RAND_{Server})}^{112 \text{ B}})}^{20 \text{ B}} \parallel MD5(PMS \parallel \overbrace{SHA('CCC' \parallel PMS \parallel RAND_{Client} \parallel RAND_{Server})}^{112 \text{ B}})$$

$$t = 3 \cdot \left(\frac{112 \text{ B}}{4100 \text{ KB/s}} + \frac{68 \text{ B}}{72500 \text{ KB/s}} \right) = 0,011 \text{ ms}$$

(11) **Finished**

$$MD5(\text{ca. } 120 \text{ B}) + MD5(\text{ca. } 800 \text{ B}) + SHA(\text{ca. } 120 \text{ B}) + SHA(\text{ca. } 800 \text{ B}) \\ \Rightarrow t = 0,035 \text{ ms (rechne 2 mal)}$$

Alles zusammen ergibt 16,4 ms

11.10.2 Aufgabe 2

Überlegen Sie, welche Probleme auftreten können, falls mehrere SSL-Server zusammen mit einem Load-Balancing Schema betrieben werden.

Lösung

- Private Keys müssen auf allen Rechner sein.
- Bei Session Resumption muss eine zentrale Datenbank existieren, welche die Sessions hält

11.10.3 Aufgabe 3

Vergleichen Sie SSH und SSL miteinander: Wo liegen Gemeinsamkeiten, wo Unterschiede?

Lösung

- Gemeinsamkeiten:
 - Sicherer Kanal auf der Transportschicht
 - Gleiche Sicherheitsdienste
- Unterschiede:
 - SSL im Browser integriert, SSH braucht eigenen Client
 - SSL mehr Varianten zur Schlüsselvereinbarung, SSH hat festgeschriebene Methode

- SSL braucht X.509 Zertifikat
- SSH unterscheidet zwischen Maschinen-Authentifizierung und User-Authentifizierung

Kapitel 12

Testfragen

1. Versuchen Sie mit eigenen Worten und ohne Formeln zu erklären, was ein perfektes Kryptosystem ist.

Lösung: Perfekte Sicherheit eines Kryptosystems bedeutet, daß ein Angreifer selbst mit unbegrenzter Rechenkapazität keine größere Erfolgswahrscheinlichkeit erhält beim Raten des Klartexts mit Kenntnis des Chiffretexts als beim Raten des Klartexts ohne Kenntnis des Chiffretexts.

Anders formuliert: Der Chiffretext gibt dem Angreifer Null Information über den Klartext.

2. Was sind hybride Verschlüsselungsverfahren? Nennen Sie drei konkrete Systeme, in denen hybride Verfahren eingesetzt werden.

Lösung: Bei hybriden Verfahren werden asymmetrische und symmetrische Verfahren miteinander gekoppelt. Die symmetrischen Verfahren übernehmen die eigentliche Verschlüsselung der zu transportierenden Daten, die asymmetrischen dienen zum Schlüsseltransport. Systeme, die hybride Verfahren nutzen, sind z.B. S/MIME, PGP und IPsec.

3. Was sind schwach kollisionsresistente Hashfunktionen? Nennen Sie zwei Einsatzgebiete.

Lösung: Schwach kollisionsresistente Hashfunktionen sind solche Hashfunktionen, bei denen es schwer ist, zu einem gegebenen Paar $(x, \text{hash}(x))$ ein y zu bestimmen mit $\text{hash}(y) = \text{hash}(x)$. Einsatzgebiete sind digitale Signaturen und MACs, die durch Verschlüsselung des Hashwertes gebildet werden.

4. Was versteht man unter dem ECB-Betriebsmodus einer Blockchiffre? Welche Nachteile hat dieser Modus?

Lösung: Beim ECB-Mode wird jeder Block einer langen Nachricht in der gleichen Weise verschlüsselt, d.h. ohne Interaktion mit den anderen Blöcken. Das führt dazu, daß sich wiederholende Klartextblöcke auch im Chiffretext als sich wiederholende Blöcke ersichtlich werden. Außerdem ist einem Angreifer evtl. möglich, Chiffreblöcke auszutauschen oder zu unterdrücken, ohne daß der Empfänger dies bemerkt.

5. Auf welchen beiden mathematischen Problemen lassen sich asymmetrische Verschlüsselungsverfahren aufbauen?

Lösung: Auf der Schwierigkeit, große Zahlen in ihre Primfaktoren zu zerlegen und auf dem diskreten Logarithmusproblem, d.h. bei gegebenem A, g, p in der Gleichung $A = g^x \bmod p$ das x zu bestimmen.

6. Wie lang (in Bit) sollten heutzutage die Schlüssel asymmetrischer bzw. symmetrischer Verschlüsselungsverfahren sein?

Lösung: Symmetrische Schlüssel sollten nicht kürzer als 128 Bit sein. Der Public Module n beim RSA Verfahren bzw. die Primzahl p beim ElGamal Verfahren sollten nicht kürzer als 1024 Bit sein.

7. Was versteht man in asymmetrischen Kryptosystemen unter der *Authentizität* der Public Keys? Warum müssen die Public Keys authentisch sein?

Lösung: Authentizität bedeutet die Gewissheit, dass der Public Key wirklich der Person gehört, für die die mit diesem Key verschlüsselte Nachricht gedacht ist. Fehlt diese Gewissheit, könnte ein Angreifer den Public Key des Empfängers einer Nachricht mit seinem eigenen vertauscht haben und somit die für den Empfänger bestimmte Nachricht lesen.

8. Wozu dient in SET die "Dual Signature"? Warum benutzt man keine "normale" digitale Signatur?

Lösung: Die Dual Signature dient dazu, den Kunden bei einer SET-Transaktion zu authentifizieren. Die Authentifikation ist nicht abstreitbar, d.h. der Kunde kann seine Bestellung im Nachhinein nicht mehr abstreiten. Zusätzlich werden durch die Signatur Bestellung und Kontoinformationen des Kunden aneinander gebunden. Diese Ziele könnten auch durch eine normale Signatur erreicht werden, aber dann wären die Kontodaten des Kunden für den Händler und die Bestelldaten für die Bank sichtbar, was beides nicht erwünscht ist.

9. Erläutern Sie das Konzept der Security Association in IPSec.

Lösung: Eine Security Association ist eine Verbindung in einer Richtung zwischen zwei IPSec Hosts, die die zu verwendenden Sicherheitsprotokolle, Schlüssel und Algorithmen beinhaltet. Durch die Zusammenfassung aller relevanten Daten in einem Datensatz "Security Association" wird es möglich, diese sehr einfach über eine Nummer (den Security Parameter Index) zu referenzieren.

Kapitel 13

Fragen zu diesem Dokument

13.1 Fragen zu Kapitel 1 (Seite 11)

- Welche Bedrohungen gibt es im Internet?
- Was ist eine "Security Policy"?
- Was ist ein "Security Breach"?
- Was versteht man unter "Sicherheitsdiensten"?
- Welche Sicherheitsdienste gibt es?
- Was versteht man unter "Sicherheitsmechanismen"?
- Welche Bedrohungen gibt es im Internet?
- Was ist eine "Security Policy"?
- Was ist ein "Security Breach"?
- Was versteht man unter "Sicherheitsdiensten"?
- Welche Sicherheitsdienste gibt es?
- Was versteht man unter "Sicherheitsmechanismen"?

13.2 Fragen zu Kapitel 2 (Seite 15)

- Mobiler Code:
 - Wo findet man mobilen Code?
 - Welche Bedrohungen gibt es für/durch mobilen Code?
 - Welchen Schutz gibt es für den mobilen Code/für den Gastrechner?
- Wieso und wodurch ist Java sicher?
- Welche Operationen darf ein nicht-signiertes Java-Applet *nicht* durchführen?
- Wie lässt sich ein Applet signieren?
- Wieso sind ActiveX Controls generell gefährlicher als Java-Applets?

13.3 Fragen zu Kapitel 3 (Seite 19)

- Viren:
 - Was ist ein Virus?
 - Welche Typen von Viren gibt es?
 - Wie verhält sich ein Virus?
 - Wie arbeitet ein Virenschanner?
 - Wie kann man sich gegen Viren schützen?
- Was ist ein trojanisches Pferd?
- Würmer:
 - Was ist ein Wurm?

- Wie arbeiten Würmer meistens?
- Was ist eine Trap door?
- Was ist eine Logic Bomb?
- Was ist ein Bakterium?
- Was ist ein Hoax?
- Spam:
 - Was ist Spam?
 - Wie kommt man an die etlichen Email-Adresse ran?
 - Welche bedrohungen hat man durch Spam?
 - Wie kann man sich gegen Spam schützen?
- Was ist Phishing?

13.4 Fragen zu Kapitel 4 (Seite 27)

- TCP:
 - Wozu ist TCP zuständig?
 - Wie funktioniert der 3-Way handshake?
 - Welche sind die Hauptprobleme des TCP?
 - Wie funktioniert sniffing?
 - Wie funktioniert portscanning?
 - Erläutere zwei Portscan-Methoden.
 - Wie funktioniert TCP-SYN-Flooding?
 - Wie funktioniert Session Hijacking?
 - Wie funktioniert ein RST-Angriff?
 - Wie wird die Sequenznummer unter TCP erzeugt?
 - Was funktioniert DNS-Spoofing/Poisoning?
 - Wie kommt es dass diese Angriffe alle möglich sind?
- UDP:
 - Welche Gefahr bringt UDP mit sich?
 - Wie funktioniert der UDP-Flood?
- ICMP:
 - Welche Gefahr bringt ICMP mit sich?
- Was sind Cookies?
- Welche Probleme gibt es bei Cookies und wie kann man sich dagegen schützen?

13.5 Fragen zu Kapitel 5 (Seite 35)

- Was ist ein Paketfilter und welche Policies (Regeln) gibt es?
- Wie kann man ein Paketfilter angreifen?
- Welche Vor- und Nachteile haben Paketfilter?
- Was ist ein Application-level Gateway?
- Was ist ein Circuit-level Gateway?
- Wie funktionieren folgende Firewall-Konfigurationen:
 - Single-Homed Bastion Host
 - Dual-Homed Bastion Host
 - Screened-subnet Firewall System

13.6 Fragen zu Kapitel 6 (Seite 43)

- Erkläre die Begriffe "Symmetrisch", "Plain- und Chiffretext" anhand der Caesar-Chiffre.
- Was besagt das Kerkhoffsches Prinzip?
- Wann ist ein Algorithmus "stark"?
- Wie funktionieren folgende Angriffe:
 - Brute-Force-Attack
 - Known-Plaintext-Attack
 - Cyphertext-Only-Attack
- Was sind monoalphabetische und polyalphabetische Substitutionschiffren?
- Haben polyalphabetische Substitutionschiffren Vorteile gegenüber monoalphabetischen?
- Wie funktioniert die Vigenère-Chiffre? Wie kann man diese Chiffre brechen?
- Wie funktioniert die ENIGMA-Chiffre?
- Wahrscheinlichkeitstheorie:
 - Definiere die Begriffe Ereignismenge, und Ereignisse.
 - Definiere den Begriff Wahrscheinlichkeitsverteilung.
 - Definiere den Begriff bedingte Wahrscheinlichkeit.
 - Schaue Beispiele zu Definitionen an.
 - Erläutere den Zusammenhang Wahrscheinlichkeitstheorie / perfektes Kryptosystem

Abbildungsverzeichnis

1.1	Internet-Security liegt im Zentrum der Begriffe	11
3.1	Übersicht bössartiger Software	19
4.1	TCP Header	29
4.2	3-Way handshake	29
4.3	Session Hacking	32
4.4	DNS-Spoofing/Poisoning	33
4.5	UDP Header	33
4.6	UDP-Flood (Echo)	34
5.1	Paketfiltering Router	35
5.2	Angriff auf Paketfilter	37
5.3	Application-level Gateway	38
5.4	Circuit-level Gateway	39
5.5	Single-Homed Bastion Host	40
5.6	Dual Homed Bastion Host	40
5.7	Screened-subnet Firewall System	41
6.1	Häufigste Buchstaben (ENIRSAT)	45
6.2	Skytale	46
6.3	Stromchiffren	52
6.4	Linear Shift Register	53
6.5	A5 Stream Generation	54
6.6	Designpattern zur Verschlüsselung bei Feistel-Chiffren	55
6.7	Designpattern zur Entschlüsselung bei Feistel-Chiffren	55
6.8	Ver- und Entschlüsselung bei CBC	60
6.9	Ver- und Entschlüsselung bei CFB	61
6.10	PKCS5 Padding	61
6.11	Java Cryptographic Architecture	62
6.12	Manipulation einer Nachricht	62
6.13	MACs: Allgemeiner Ablauf	63
7.1	MACs: Public Key Kryptosystem: Allgemeiner Ablauf	68
7.2	Adi Shamir, Ron Rivest, Len Adleman	68
7.3	Diffie-Hellman Schlüsseltausch-Protokoll	74
7.4	Man-in-the Middle Angriff auf Diffie-Hellman	75
7.5	Digitale Signaturen: Allgemeiner Ablauf	77
7.6	Basic Challenge-Response	81
7.7	HTTP Digest Authentication	81
7.8	Challenge-Response Scheme based on PK Decryption	81

7.9	X.509 Challenge-Response based on Dig. Sig.: 1-Way with Timestamp . . .	82
7.10	X.509 Challenge-Response based on Dig. Sig.: 2-Way with Rand. Challenge	82
7.11	X.509 Challenge-Response based on Dig. Sig.: 3-Way Mutual Auth.	82
7.12	Basic Kerberos	83
7.13	Kerberos in a network	84
8.1	PGPTools Screenshot	89
8.2	Format of PGP Message (From User A to User B, signed by A)	90
8.3	PGP Message Generation (User A to User B)	90
8.4	PGP Message Reception (From User A to User B)	91
8.5	PGP "Web of Trust"	91
8.6	SET Beteiligte	92
8.7	SET Transaction	93
8.8	SET Dual Signature	95
8.9	SET: Building the Purchase Request	95
8.10	SET: Verifying a Dual Signature: Merchant's Site	96
9.1	SSL/TLS, SSH: Lage von im Protokollstapel	97
9.2	SSL Protokollstapel	98
9.3	SSL Record Format	98
9.4	SSL Handshake	99
9.5	SSH-2 Protokollstapel	101
9.6	SSH-2 Protokollablauf	102
10.1	Beispiel für eine IPSec-Übertragung im ESP-Transport Mode	107
10.2	ESP Header	108
10.3	ESP Encryption and Authentication (Transport Mode)	108
10.4	ESP Encryption and Authentication (Tunnel Mode)	109
10.5	AH Header	109
10.6	IP Headers before applying AH	110
10.7	AH Authentication (Transport Mode)	110
10.8	AH Authentication (Tunnel Mode)	110
10.9	SA Combinations - case 1	112
10.10	SA Combinations - case 2	112
10.11	SA Combinations - case 3	113
10.12	SA Combinations - case 4	113
10.13	STS Protokollablauf	114
10.14	ISAKMP Header	115
10.15	Generic Payload Header	115
10.16	IPv4 Header	116
10.17	IPv6 Header	116
10.18	NAT-Gateway	117
10.19	ESP und AH im Protokollstapel	118
10.20	IKE im Protokollstapel	118
11.1	Einschleusen des ADK in 2 mal verschlüsselte Mail	146

Tabellenverzeichnis

1.1	Sicherheitsdienste - Vergleich mit dem Alltag	12
5.1	Paketfilter default policy: Discard	36
5.2	Paketfilter default policy: Forward	36
5.3	Paketfilter example Filtering Rule 1	36
5.4	Paketfilter example Filtering Rule 2	36
6.1	DES-AES-Vergleich	59
11.1	Paketfilter Übungsaufgabe Filtering Rule	124

Quellcodeverzeichnis

3.1	Die Fortran-Zeile	19
3.2	Dateivirus Pseudocode	20
6.1	RC4 Key Schedule Pseudocode	52
6.2	RC4 Stream Generation Pseudocode	53
6.3	Getting a Cipher Instance	61
7.1	Euklid'scher Algorithmus Pseudocode	70
11.1	Caesar-Chiffre En-/Decoder	126
11.2	Umgang mit DES	132
11.3	Code für Message Digest (MD5 und SHA-1)	134
11.4	Code für SHA-1 Hash und MAC	135
11.5	RSAPublicKey	138
11.6	RSACrypt	139
11.7	RSASign	141

Abkürzungsverzeichnis

3DES	Tripe DES
ACL	A ccess C ontrol L ist
AES	A dvanced E ncryption S tandard
AH	A uthentication H header
API	A pplication P rogramming I nterface
ARPA	A dvanced R esearch P roject A gency
ASCII	A merican S tandard C ode for I nformation I nterchange
ASIC	A pplication S pecific I ntegrated C ircuit
BSI	B undesamt für S icherheit in der I nformationstechnik
CA	C ertificate A uthority
CAST	Blockchiffre, benannt nach C arlisle, A dams, S tafford und T averes
CBC	C ipher B lock C haining M ode
CERT	C omputer E mergency R esponse T eam
cert	C ertificate
CFB	C ipher F eedback M ode
CoD	C ode o n D emand
CPS	C ertificate P ractice S tatement
CPU	C entral P rocessing U nit
CRL	C ertificate R evocation L ist
CTR	C ounter M ode
DES	D ata E ncryption S tandard
DH	D iffie H ellman
DNAT	D ynamic N AT
DNS	D omain N ame S ervice
DoI	D omain o f I nterpretation
DoS	D enial o f S ervice
DSA	D igital S ignature A lgorithm
DSS	D igital S ignature S tandard
ECB	E lectronic C odebook M ode
ESP	E ncapsulating S ecurity P ayload
FPGA	F ield P rogrammable G ate A rray
FTP	F ile T ransmission P rotocol

ggT	g rösster g emeinsamer T eiler
GSM	G lobal S ystem for M obile Communications
HMAC	M AC based on H ash function
HTTP	H yper T ext T ransfer P rotocol
HW	H ard w are
ICMP	I nternet C ontrol M essage P rotocol
IDEA	I nternational D ata E ncryption A lgorithm
IDS	I ntrusion D etection S ystem
IEEE	I nstitute of E letrical and E lectronic E ngineers
IETF	I nternet E ngineering T ask F orce
IIS	I nternet I nformation S erver
IKE	I nternet K ey E xchange
IMAP	I nternet M essage A ccess P rotocol
IP	I nternet P rotocol
IPSec	I P S ecurity
ISAKMP	I nternet S ecurity A ssociation K ey M anagement P rotocol
ISN	I nitial S equence N umber
ISO	I nternational S tandards O rganisation
ITU	I nternational T elecommunications U nit
JCA	J ava C ryptographic A rchitecture
JCE	J ava C ryptography E xtension
JDK	J ava D evelopment K it
JPEG	J oint P hotographic E xperts G roup
LAN	L ocal A rea N etwork
LFSR	L inear F eedback S hift R egister
MAC	M essage A uthentication C ode
MD5	M essage D igest A lgorithm 5
MIME	M ultipurpose I nternet M ail E xtension
mod	m odulo
MPEG	M oving P icture E xperts G roup
MSP	M anaged S ecurity P rovider
NAPT	N etwork A ddress P ort T ranslation
NAT	N etwork A ddress T ranslation
NIST	N ational I nstitute of S tandards and T echnology
OFB	O utput F eed b ack M ode
OTP	O ne- T ime- P ad
PGP	P retty G ood P rivacy

PIN	P ersonal I dentification N umber
PK	P ublic K ey
PKCS	P ublic K ey C ryptographic S tandards
POP	P ost O ffice P rotocol
RAND	R ANDOM
RC	R elease C andidate
REV	R emote E valuation
RFC	R equest F or C omment
RIPEMD	R ACE Integrity P rimitives E valuation M essage D igest
RSA	Kryptosystem, benannt nach R ivest, S hamir und A dleman
S/MIME	S ecure/ M IME
SA	S ecurity A ssociation
SAD	S ecurity A ssociation D atabase
SET	S ecure E letronic T ransaction
SETI	S earch for E xtraterrestrial I ntelligence
SHA	S ecure H ash A lgorithm
sig	s ignature
SK	S ecret K ey (Private Key)
SMTP	S imple M ail T ransfer P rotocol
SPD	S ecurity P olicy D atabase
SPI	S ecurity P arameter I ndex
SPI	S ervice P rovider I nterface
SSL	S ecure S ocket L ayer
SW	S oftware
TCP	T ransmission C ontrol P rotocol
TGS	T icket G ranting S erver
TGT	T icket G ranting T icket
TLS	T ransport L ayer S ecurity
UDP	U ser D atagram P rotocol
URL	U niform R essource L ocator
WLAN	W ireless L AN
XOR	exclusive O R

Index

- 3-Way handshake, 29–30
- A5, 53
- Access Control, *siehe* Zugangskontrolle
- ActiveX Control, 18
- Advanced Encryption Standard (AES), 58–60
- Anwendungsschicht, 28
- Applet, 17–18
- Asymmetrische Kryptographie, 67
- Authentication, *siehe* Authentifikation
- Authentication Header, 105
- Authentication Header (AH), 109
- Authenticator, 83
- Authentifikation, 12
- Authentifikationsprotokolle, 79
- Authorisierung, 12
- Authorization, *siehe* Authorisierung
- Availability, *siehe* Verfügbarkeit
- Bakterien, 23
- Bastion Hosts, 39
 - Dual-Homed Bastion Host, 40
 - Single-Homed Bastion Host, 39
- Bedingte Wahrscheinlichkeit, 48–49
- Betriebsmodi für Blockchiffren, 60–61
- Blockchiffren, 54
- Blowfish, 57
- Brute-Force-Attack, 44
- Buffer Overflow Attack, 23
- Caesar-chiffre, 43
- CAST, 57
- Certificate Authority (CA), 79
- Certificate Policy, 79
- Certification Practice Statement, 79
- Certification Practice Statement (CPS), 79
- Challenge-Response, 80
- Chiffren, 43
- Chiffretext, 43
- Cipher Block Chaining Mode (CBC), 60
- Cipher Feedback Mode (CFB), 60
- Ciphertext, *siehe* Chiffretext
- Ciphertext-Only-Attack, 45
- Code Red Virus, 23
- Code-on-Demand (COD), 15
- Computervirus, 19
 - Bootvirus, 20
 - Dateivirus, 20
 - Makrovirus, 20
- Confidentiality, *siehe* Vertraulichkeit
- Cookies, 122
- Counter Mode (CTR), 61
- cryptographic protocol, 79
- Darstellungsschicht, 28
- Data Encryption Standard (DES), 56–57
 - Triple-DES (3DES), 56
- Dialerprogramme, 15
- Diffie-Hellman, 74
- Digitale Signaturen, 76–77
- Diskrete Logarithmen, 73
- Diskretes Logarithmusproblem, 73
- DNS-Poisoning, 33
- DNS-Spoofing, *siehe* DNS-Poisoning
- Dual Signature, 94
- Electronic Codebook Mode (ECB), 60
- Elementarereignisse, 47
- ElGamal-Verfahren, 76
- Encapsulating Security Payload (ESP), 107
- Encryption Exponent, 69
- ENIGMA, 47
- Ereignismenge, 47–48
- Ereignisse, 47
- Erweiterter Euklid'scher Algorithmus, 70–71
- Euklid'scher Algorithmus, 70
- Feistel Chiffren, 55
- Firewalls, 35
 - Personal Firewalls, 41–42

- Typen, 35
- Gateways
 - Application-Level Gateway, 38
 - Circuit-Level Gateway, 39
- ggT, 69
- Half-Open Scan, 31
- Hashfunktion, 63–64
- Hoax, 23
- Identifikation, *siehe* Authentifikation
- Integritätsschutz, 12
- Integrity Protection, *siehe* Integritätsschutz
- International Data Encryption Algorithm (IDEA), 58
- Internet Control Message Protocol (ICMP), 34
- Internet Key Exchange, 105
- Internet Protokoll (IP), 28
- Intrusion Detection Systems (IDS), 42
- IPSec, 105
- ISAKMP, 115
- Java Cryptographic Architecture (JCA), 61–62
- Kasiski, 47
- Kerberos, 83
- Kerkhoffsches Prinzip, 44
- Klartext, 43
- Known-Plaintext-Attack, 45
- Kollision, 64
- Kollisionsresistent
 - schwach Kollisionsresistent, 64
- Kollisionsresistenz
 - schwach Kollisionsresistent, 64
 - stark Kollisionsresistent, 64
- Kryptographie, 43
- Layers
 - Application, *siehe* Anwendungsschicht
 - Data Link, *siehe* Sicherungsschicht
 - Network, *siehe* Vermittlungsschicht
 - Physical, *siehe* Physikalische Schicht
 - Presentation, *siehe* Darstellungsschicht
 - Session, *siehe* Sitzungsschicht
 - Transport, *siehe* Transportschicht
- Linear Shift Registers, *siehe* Shift Registers
- Logic Bombs, 23
- MAC-Bildung, 64–65
- Managed Security Provider (MSP), 42
- MD5, 65
- Message Authentication Codes (MACS), 62–63
- Mobiler Code, 15
- Moblile Agenten, 15
- Morris Wurm, 123
- Multipurpose Internet Mail Extension (MIME), 86
- Nachrichtenauthentifikation, *siehe* Authentifikation
- Network Address Translation (NAT), 116
- Nichtabstreitbarkeit, 12
- Nonrepudiation, *siehe* Nichtabstreitbarkeit
- Oakley, 114
- One-Time-Pad (OTP), 51
- One-Way-Eigenschaft, 65
- OSI Schichtenmodell, 27–28
- Output Feedback Mode (OFB), 60–61
- Packet Filter, 35–36
- Perfekte Sicherheit, 49
- Perfektion von symmetrisches Kryptosystem, 51
- Phishing, 25
- Physikalische Schicht, 27
- PKCS5 Padding, 61
- Plaintext, *siehe* Klartext
- Portscanning, 31
- Potenzmenge, 47
- Pretty Good Privacy (PGP), 88
- primitives Element, 73
- Primzahlen, 69
- public module, 69
- Public-Key-Kryptosystem, 67
- RC4, 52–53
- Remote Evaluation (Rev), 15
- RIPEMD-160, 65
- RSA, 67–69
- RSA-Problem, 72
- RST-Angriff, 32

- Safety, 9
- Schiebe-Registers, *siehe* Shift Registers
- Secure Electronic Transactions (SET), 92
- Secure Multipurpose Internet Mail Extension (S/MIME), 85
- Secure Shell (SSH), 101
- Secure Socket Layer (SSL), 97
- Security, 9
 - Computer Security, 11
 - Information Security, 11
 - Internet Security, 11
 - Network Security, 11
- Security Association Database, 106
- Security Association Database (SAD), 106
- Security Associations (SA), 106
- Security Breach, 12
- Security Parameter Index (SPI), 106
- Security Policy, 11
- Security Policy Database (SPD), 106
- Security Protocol Identifier, 106
- Session Hijacking, 32
- SHA-1, 65
- Shift Registers, 53
- Sicheres System, 11
- Sicherheitsdienste, 12
- Sicherheitsmechanismen, 13
- Sicherungsschicht, 27
- Simple Mail Transfer Protocol (SMTP), 85
- Single-Sign-On, 83
- Sitzungsschicht, 27
- Skytale, 46
- Sniffing, 30
- Spam, 24
- Starker Algorithmen, 44
- Stromchiffren, 51
- Substitutionschiffren, 45
 - Monoalphabetische, 45
 - Polyalphabetische, 45
- Symmetrisch, 43
- Symmetrisches Kryptosystem, 43
- SYN-ACK Scan, 31
- TCP-SYN-Flooding, 31
- Teiler, 69
- teilerfremd, 69
- Ticket Granting Ticket (TGT), 83
- Tiny Fragment Attack, 37
- Transmission Control Protocol (TCP), 29
- Transport Layer Security (TLS), 97
- Transportschicht, 27
- Trap doors, 23
- Trojanische Pferde, 22
- UDP-Flood, 34
- Unabhängige Ereignisse, 49
- User Datagram Protocol (UDP), 33
- Verfügbarkeit, 12
- Vermittlungsschicht, 27
- Vertraulichkeit, 12
- Vigenère-Chiffre, 46
- Virens Scanner, 21–22
- Virenschutzmaßnahmen, 22
- Virus, *siehe* Computervirus
- Würmer, 22
- Wahrscheinlichkeitsverteilungen, 48
- X.509, 78–79
- Zertifikate, 78–79
- Zufallsgeneratoren in Java, 52
- Zugangskontrolle, 12