Grundlagen Rechnernetze und Verteilte Systeme

SoSe 2014

Kapitel 6: Netzsicherheit

Prof. Dr.-Ing. Georg Carle

Nadine Herold, M. Sc.
Dipl.-Inf. Stephan Posselt
Johannes Naab, M. Sc.
Marcel von Maltitz, M. Sc.
Stephan Günther, M. Sc.

Fakultät für Informatik
Lehrstuhl für Netzarchitekturen und Netzdienste
Technische Universität München

Worum geht es in diesem Kapitel?

1 Motivation

2 Grundlegende Begriffe

3 Kryptografie

4 Public Key Infrastrukturen (PKI)

5 Sichere Protokolle und Protokollvarianten

6 Netzwerkmonitoring und -schutz

Warum ist Sicherheit in Netzen von Bedeutung?

- Das ISO/OSI-Modell stellt eine Netzarchitektur zur Verfügung, welche Systemen erlaubt, (zuverlässig) miteinander zu kommunizieren.
- Typischerweise findet diese Kommunikation über viele verschiedene Zwischensysteme (Router, Proxies, Mailserver) statt.
- Da der Austausch im Standardfall im Klartext durchgeführt wird, sind alle zwischengeschalteten Systeme in der Lage, die Daten mitzulesen und zu speichern.
- Bisher eingesetzten Mechanismen zur Integritätswahrung dienen lediglich eine Absicherung gegen (zufällige) Übertragungsfehler.
- Eine „sichere“ Kommunikation ist so nicht möglich.
- Zudem sind mittlerweile Netze aller Art Ziele von Angriffen, deren Techniken immer ausgefeilter werden.

Das Problemfeld vertraulicher und geschützter Kommunikation wird also im ISO/OSI-Modell nicht behandelt. Aspekte, welche die Sicherheit der Kommunikation betreffen, sind in diesem Modell nicht verankert.

$\rightarrow$ weitere Strategien zur Erreichung dieser Ziele sind notwendig

Übersicht

Grundbegriffe

- Der Begriff Sicherheit umfasst eine Sammlung verschiedener Sicherheitsbegriffe, welche sich auf unterschiedliche zu schützende Werte (assets) beziehen.

Definition (Sicherheit)

Unter dem Begriff Betriebssicherheit (safety) versteht man hauptsächlich die technische Sicherheit, d. h. das System funktioniert unter normalen Umständen wie es soll. Insbesondere darf es keine Gefahrenquelle für Benutzer darstellen.
Informationssicherheit (security) hingegen besteht, wenn Informationen weder unautorisiert gewonnen noch verändert werden können.

- Zudem ist der Begriff Datensicherheit (protection) zentral. Dieser behandelt den Schutz vor unautorisierter Manipulation von **Systemressourcen** sowie Maßnahmen zum Schutz vor Datenverlusten (z. B. durch Backups).
- Der Begriff Datenschutz (privacy) ist im deutschen Bundesdatenschutzgesetz (BDSG) festgelegt und beschreibt die Selbstbestimmung natürlicher Personen über die Weitergabe der eigenen personenbezogenen Daten.

Definition (Verlässlichkeit)

Ein System erfüllt die Eigenschaft der Verlässlichkeit (dependability), wenn es betriebssicher ist und die Funktionalitäten zuverlässig (Reliability) erfüllt.

Schutzziele

- Sicherheit in IT-Systemen im Allgemeinen fasst sowohl Informationen als auch die sie repräsentierenden Daten als zu schützende Werte auf.
- Um diese Werte schützen zu können, werden eine Reihe verschiedener Schutzziele definiert.

Definition (Authentizität)

Unter dem Begriff der Authentizität (authenticity) wird die Echtheit eines Objekts bzw. Subjekts verstanden. Die Echtheit kann mittels einer eindeutigen Identität oder charakteristischen Eigenschaft überprüfbar sein.

Ablauf eines Authentizitätsnachweises:

- Soll die Echtheit von A gegenüber B nachgewiesen werden, muss zuerst A einen vermeintlichen Nachweis der eigenen Authentizität erbringen (Authentisierung).
- Danach verifiziert B den dargebotenen Nachweis (Authentifikation) und schließt daraus auf die Authentizität von A.

Welche Verfahren sind hier möglich?

- Authentisierung durch Wissen, z. B. Passwörter, PIN
- Authentisierung durch Besitz, z. B. Smartcards
- Authentisierung durch biometrische Merkmale, z. B. Fingerabdruck, Iris

Eine Kombination der genannten Verfahren ist möglich (multi-factor authentication)

Schutzziele

Definition (Datenintegrität)

Datenintegrität (integrity) beschreibt die Unveränderbarkeit der zu schützenden Daten, d. h. die Daten können nicht unautorisiert und unbemerkt manipuliert werden.

A-Priori Mechanismen

- Mit Hilfe eines entsprechenden Rechtesystems können einzelne Befugnisse (z. B. Leserechte, Schreibrechte, ...) bestimmte Daten betreffend an Identitäten oder Gruppen gebunden werden. Man spricht hierbei von Zugriffskontrolle.
- Um die Weiterverarbeitung der gewonnenen Informationen zu beschreiben und zu kontrollieren, werden Mechanismen der Informationsflusskontrolle verwendet.

A-Posteriori Mechanismen

- Teilweise können Manipulationen nicht verhindert werden. Daher ist es notwendig, diese zu erkennen, um der Definition von Integrität gerecht zu werden.
- Hierfür können beispielsweise kryptografisch sichere Hashfunktionen verwendet werden (siehe S. 29ff).

Schutzziele

Definition (Informationsvertraulichkeit)

Informationsvertraulichkeit (confidentiality) ist gegeben, wenn keine unautorisierte Gewinnung von Informationen möglich ist.

- Dies ist eng mit der Datenintegrität verbunden und setzt ebenfalls ein Rechtesystem voraus.
- Maßnahmen zur Bestimmung, Festlegung und Kontrolle zulässiger Datenflüsse sollen ausschließen, dass Informationen an unautorisierte Teilnehmer gelangen (Confinement Problem).
- Das sog. Interferenz Problem ist eng mit dem Confinement Problem verknüpft und beschreibt die Möglichkeit aus Einzelinformationen weitere Informationen ableiten zu können.

Diese drei Schutzziele spielen eine zentrale Rolle und werden meist unter dem Akronym CIA (Confidentiality – Integrity – Authenticity) zusammengefasst.

Neben diesen wichtigen Schutzzielen gibt es noch eine Reihe weiterer Schutzziele, welche ebenfalls betrachtet werden müssen.

Schutzziele

Definition (Verfügbarkeit)

Unter Verfügbarkeit (availability) versteht man, dass ein System seinen Dienst erbringt und seine Nutzer ihre Berechtigungen wahrnehmen können, ohne unautorisiert beeinträchtigt zu werden.

Definition (Verbindlichkeit)

Unter Verbindlichkeit (non-repudiation) versteht man die Fähigkeit eines Systems, eine Zuordnung zwischen Aktionen und Benutzern herzustellen, sodass Benutzer ihre durchgeführten Aktionen nicht abstreiten können.

Im Konflikt mit der Forderung der Verbindlichkeit stehen die Forderungen nach Privatsphäre. Es werden zunehemend Mechanismen wichtig, welche sich dem Aspekt der Privatsphäre widmen.

Definition (Anonymisierung, Pseudonymisierung)

Anonymisierung beschreibt das Verändern personenbezogener Daten, sodass ein Rückschluss auf die Identität nur mit unverhältnismäßig großem Aufwand möglich ist.
Eine schwächere Form der Anonymisierung wird durch die Pseudonymisierung erreicht, bei der die personenbezogenen Daten mit Hilfe einer Zuordnungsvorschrift von realen Personen getrennt werden.

Angriffe, Bedrohungen und Risiko

- Eine Schwachstelle (weakness) bezeichnet eine Schwäche im System, z. B. unsichere Kommunikationskanäle.
- Eine Verwundbarkeit (vulnerability) ist eine Schwachstelle, über welche ein Angreifer die Sicherheitsmechanismen des Systems umgehen kann, z. B. Heartbleed.
- Eine Bedrohung (threat) ergibt sich dann, wenn Schwachstellen oder Verwundbarkeiten so ausgenutzt werden können, dass die Schutzziele gefährdet sind.
- Das Risiko einer Bedrohung ist die Eintrittswahrscheinlichkeit des Ereignisses, welches den Schaden verursacht, kombiniert mit der potentiellen Höhe des entstehenden Schadens.

Als Angriff (attack) versteht man nun den (versuchten) unautorisierten Zugriff auf ein System. Man kann zwischen aktiven und passiven Angriffen unterscheiden.

Passiv	**Aktiv**
unautorisierte Informationsgewinnung	unautorisierte Modifikation
$\rightarrow$ Verlust der Vertraulichkeit	$\rightarrow$ Verlust der Integrität oder Verfügbarkeit

Beispiele für passive Angriffe sind etwa das Abhören der Kommunikationsleitung (eavesdropping) oder von Passwörtern (sniffing).

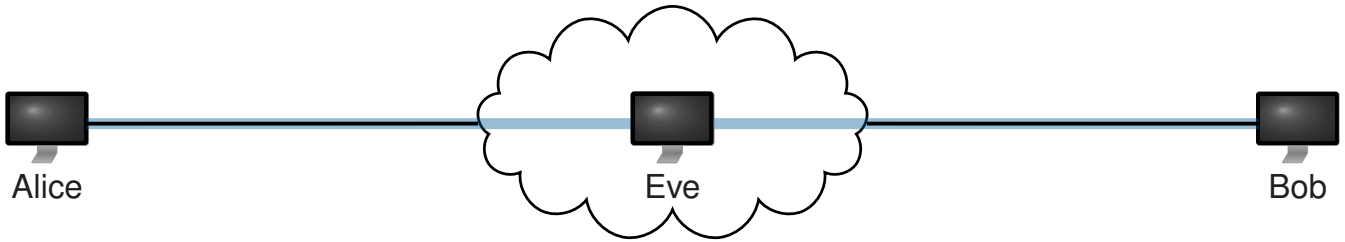
Aktive Angriffe im Netzbereich sind vor allem das Verändern, Entfernen, Wiedereinfügen (Replay) von Nachrichten.
Zwei spezielle Klassen von aktiven Angriffen sind zudem die sog. Maskierung (Spoofing) und Denial-of-Service-Angriffe.
Ersteres richtet sich gegen die Verbindlichkeit. DoS-Angriffe richten sich gegen die Verfügbarkeit.

Übersicht

Kryptografische Grundlagen

Problemstellung:

- Alice und Bob wollen miteinander kommunizieren.



- Alice hat keinen Einfluss darauf, wie Datenpakete durch das Internet geleitet werden, wer sie mitlesen oder sogar modifizieren kann.
- Alice kann sich (trotz korrekter Adressierung von Bob) nicht einmal sicher sein, überhaupt mit Bob zu kommunizieren:
  - Der Angreifer Eve könnte sich sowohl Alice als auch Bob gegenüber als der jeweils andere Kommunikationspartner ausgeben.
  - Einzige Voraussetzung dafür ist, dass Eve die Pakete von Alice und Bob „abfangen“ kann.
  - Befindet sich Eve im lokalen Netz von Alice oder Bob, ist das sehr einfach: ARP-Spoofing.

Idee: Alice und Bob authentifizieren sich gegenseitig und verschlüsseln ihre Daten, sodass

- sichergestellt ist, dass Alice und Bob auch wirklich diejenigen sind, die sie vorgeben zu sein, und
- nur der jeweils andere Kommunikationspartner in der Lage ist, die Daten wieder zu entschlüsseln.

Ziele kryptografischer Verfahren

Mittels kryptographischer Verfahren werden im Speziellen folgende bereits beschriebenen Schutzziele verfolgt:

- Integrität
  Bob will sich sicher sein, dass die Daten von Alice auf dem Weg nicht verändert wurden.
- Authentizität
  Bob will sich sicher sein, dass die Daten auch wirklich von Alice stammen und nicht von jemandem, der sich für Alice ausgibt.
- Vertraulichkeit
  Es soll verhindert werden, dass unbefugte Dritte Nachrichten zwischen Alice und Bob mitlesen können.
- Verbindlichkeit / Nichtabstreitbarkeit
  Dem Sender einer Nachricht soll es nicht möglich sein, zu bestreiten, dass er der Urheber einer bestimmten Nachricht ist.

Diese Ziele werden i. d. R. nur durch das (komplizierte) Zusammenspiel aus

- Sitzungsprotokollen
- Schlüsselaustauschverfahren bzw. -protokollen,
- Verschlüsselungsalgorithmen und
- Hashverfahren

erreicht.

Grundlegende Terminologie

Definition (Kryptographie, Kryptoanalyse, Kryptologie)

Unter dem Begriff Kryptographie versteht man Methoden zur Absicherung von Nachrichten mit Hilfe von Verschlüsselung. Mit dem Begriff Kryptoanalyse wird das Vorgehen beschrieben, diese Verschlüsselung zu brechen und die Klartext zu reproduzieren. Kryptologie verbindet Kryptographie und Kryptoanalyse.

Ein **Kryptosystem KS** kann dargestellt werden als ein Sechstupel

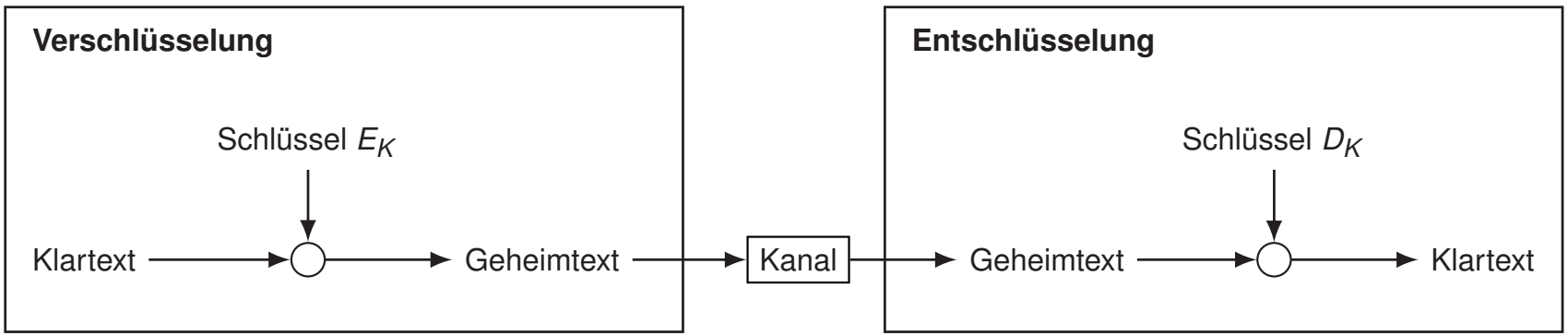
$$\boldsymbol{KS} = (\mathcal{M}, \mathcal{C}, \boldsymbol{EK}, \boldsymbol{DK}, \boldsymbol{E}, \boldsymbol{D})$$

mit

- $\mathcal{M}$: einer nicht-leeren Menge von **Klartexten** $\mathcal{M} \subseteq A_1^*$, wobei A_1^* die Menge aller Worte über dem Alphabet A_1 beschreibt
- $\mathcal{C}$: einer nicht-leeren Menge von **Chiffretexten** $\mathcal{C} \subseteq A_2^*$
- EK: einer nicht-leeren Menge von **Verschlüsselungschlüsseln**
- DK: einer nicht-leeren Menge von **Entschlüsselungsschlüsseln** sowie einer Bijektion $f : EK \rightarrow DK$, sodass $\exists K_E \in EK : f(K_E) = K_D$
- E: einem linkstotalen und injektiven **Verschlüsselungsverfahren**
- D: einem korrespondierenden **Entschlüsselungsverfahren**

Verschlüsselungsverfahren

- **Verschlüsselung (Encryption)** beschreibt die Abbildung von $M \in \mathcal{M}$ auf $C \in \mathcal{C}$: $E(M) = C$.
- **Entschlüsselung (Decryption)** beschreibt die Umkehrfunktion: $D(C) = M$
- Daher gilt: $D(E(M)) = M$



Für Verschlüsselungsverfahren soll im Allgemeinen gelten:

Kerkhoffs-Prinzip

Die Sicherheit eines Verschlüsselungsverfahrens darf nur von der Geheimhaltung des Schlüssels abhängen, also auch nicht von der Geheimhaltung des Verschlüsselungsalgorithmus.

Klassifizierung von Verschlüsselungsverfahren I

Symmetrische Verfahren

- Sender und Empfänger einigen sich auf ein gemeinsames Geheimnis (engl. Shared Secret, ugs. „Passwort“ oder „Schlüssel“).
- Mittels dieses Schlüssels können Daten verschlüsselt und auch wieder entschlüsselt werden (daher „symmetrische“ Verfahren).
- Das bedeutet: $E_K(M) = C$ und $D_K(C) = M$, woraus folgt: $D_K(E_K(M)) = M$
- Man beachte, dass für die Ent- sowie Verschlüsselung dasselbe K verwendet wird.

Asymmetrische Verfahren

- Alice und Bob besitzen jeweils zwei Schlüssel: Einen Public Key und einen Private Key.
- Der öffentliche Schlüssel ist prinzipiell jedem zugänglich, der private Schlüssel wird geheim gehalten.
- Alice verschlüsselt eine Nachricht mit dem öffentlichen Schlüssel von Bob.
- Zur Entschlüsselung ist der passende private Schlüssel notwendig, den nur Bob besitzt.
- Das bedeutet: $E_{K_{pub}}(M) = C$ und $D_{K_{priv}}(C) = M$, woraus folgt: $D_{K_{priv}}(E_{K_{pub}}(M)) = M$
- Man beachte, dass hier K_{priv} der private und K_{pub} der öffentliche, d.h. zwei verschiedene Schlüssel sind.
- *Praktischer Einsatz:* Dieses Prinzip steht hinter den RSA-Schlüsseln, die Sie zum Zugriff auf Ihre GRNVS-VMs benötigen.

Klassifizierung von Verschlüsselungsverfahren II

Stromchiffren

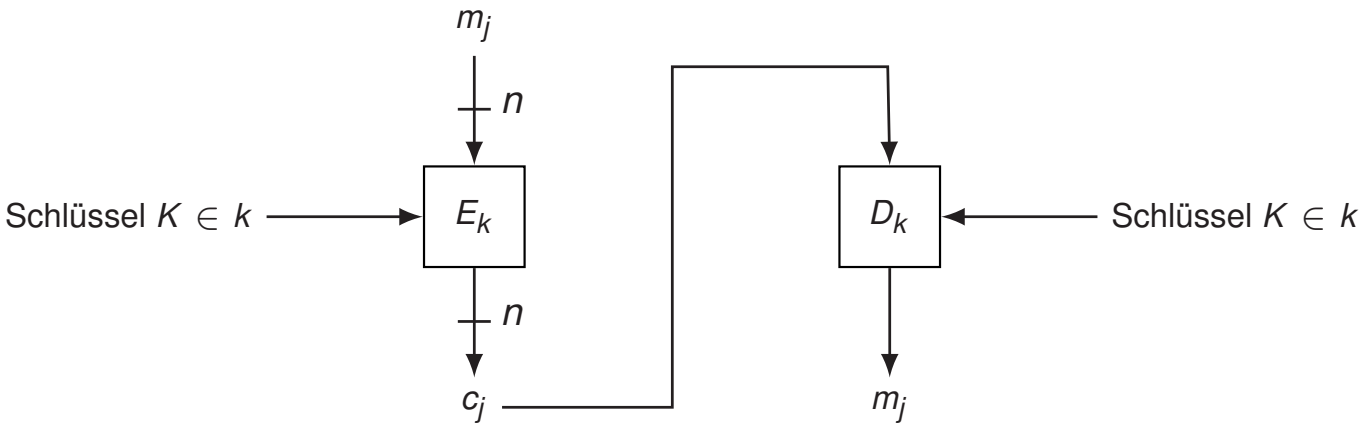
- Bei Stromchiffren wird ein Zeichen nach dem anderen verschlüsselt.
- Im Folgenden wird der RC4 (Rivest Cipher 4) Algorithmus als Vertreter der Stromchiffren sowie der symmetrischen Verschlüsselungsverfahren betrachtet.

Blockchiffren

- Bei Blockchiffren wird die zu verschlüsselnde Nachricht in einzelne Blöcke (z. B. 64 Bit) zerlegt.
- Anschließend wird ein Block nach dem anderen verschlüsselt.
- Zwei wichtige Vertreter der Blockchiffre sind der DES[1] - und der AES-Algorithmus.
- Blockchiffren können in verschiedenen Modi verwendet werden.
- Im Folgenden betrachten wir zwei einfache Vertreter zur Veranschaulichung: ECB (Electronic Codebook Mode) und CBC (Cipherblock Chaining Mode).

[1] DES gilt heute als unsicher und sollte daher nicht mehr verwendet werden.

Blockchiffren – ECB Mode

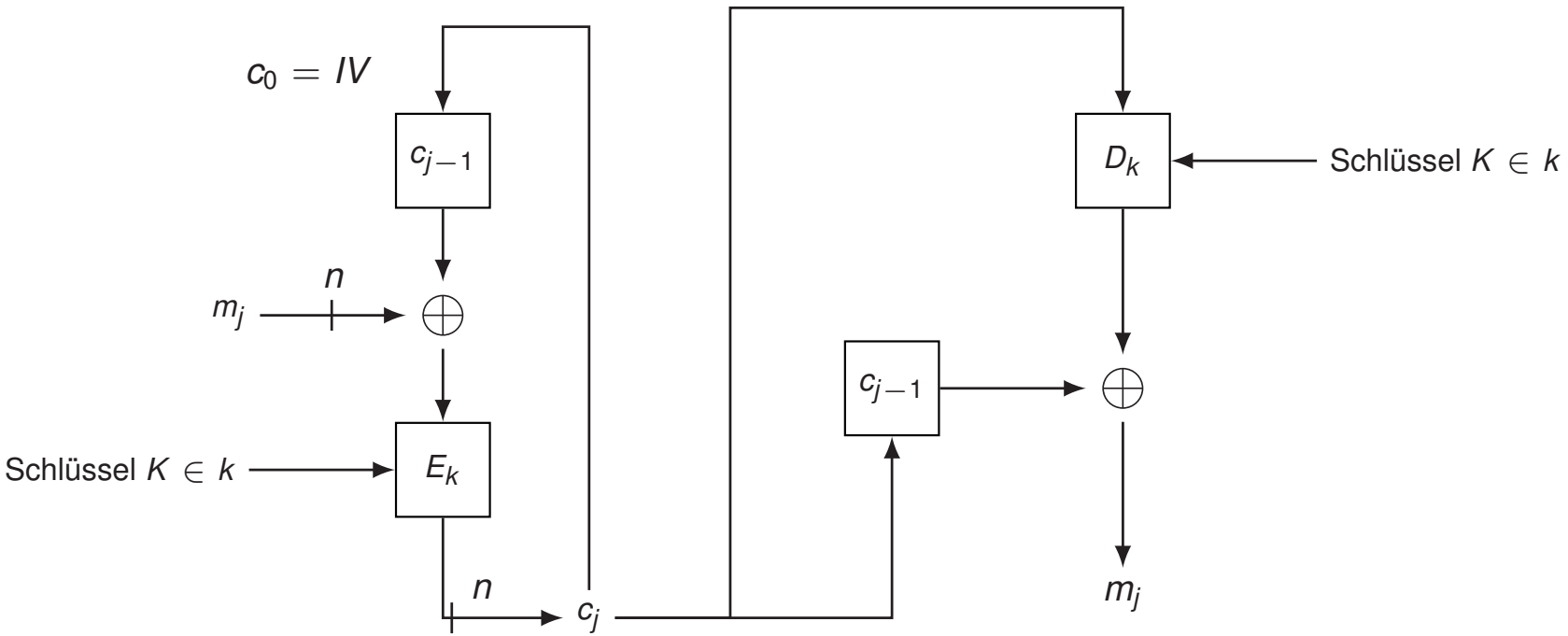


- Beliebig lange Klartexte werden in Blöcke der Länge n aufgeteilt. Geht die Zerlegung nicht auf, muss der Klartext gepaddet werden.
- Jeder Block wird eigenständig verschlüsselt und wieder entschlüsselt.
- Die Konkatenation der einzelnen entschlüsselten Blöcke ergibt die ursprüngliche Nachricht.
- Dieses Verfahren ist für symmetrische wie auch asymmetrische Verschlüsselung möglich.

Beispiel: $m = 110100111001$

- Es ergeben sich mit $n = 4$ folgende Blöcke: $m_1 = 1101$, $m_2 = 0011$ und $m_3 = 1001$
- Mittels Verschlüsselung durch Permutation $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$
- ergibt sich der folgende Chiffretext: $c = 1011\ 1100\ 1001$

Blockchiffren – CBC Mode



- Es wird ein fester Initialisierungsvektor (IV) der Blocklänge n und als Verknüpfung *XOR* verwendet .
- Der Klartext wird auch hier in feste Blöcke der Länge n aufgeteilt.

Blockchiffren – CBC Mode: Anwendung und Beispiel

Für die Verschlüsselung der Klartextblöcke $m_1, \ldots, m_x$ der Länge n mit Schlüssel k wird wie folgt vorgegangen:

- $c_0 = IV$ und $c_j = E_k(c_{j-1} \oplus m_j)$ für alle $1 \leq j \leq x$
- Ergebnis: Chiffretextblöcke $c_1, \ldots, c_x$

Für die Entschlüsselung der Chiffretextblöcke $c_1, \ldots, c_x$ der Länge n mit Schlüssel k wird wie folgt vorgegangen:

- $c_0 = IV$ und $m_j = c_{j-1} \oplus D_k(c_j)$ für alle $1 \leq j \leq x$
- Ergebnis: Klartextblöcke $m_1, \ldots, m_x$
- Die Schlüssel für Ver- und Entschlüsselung müssen nicht notwendigerweise gleich, aber korrespondierend sein.

Im Folgenden betrachten wir erneut das bekannte Beispiel: $m = 110100111001$

- Es ergeben sich mit $n = 4$ folgende Blöcke: $m_1 = 1101$, $m_2 = 0011$ und $m_3 = 1001$
- Mittels Verschlüsselung durch Permutation $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$
- IV sei gegeben mit 1011
- ergibt sich der folgende Chiffretext: $c = 0110\ 1010\ 1100$

Bewertung der beiden Modi

Nachteile und Probleme im ECB-Mode

- Gleiche Klartextblöcke werden in gleiche Chiffretextblöcke umgewandelt, daher übertragen sich Regelmäßigkeiten und Muster des Klartextes in den Chiffretext.
- Diese Informationen können die Kryptoanalyse unterstützen.
- Der Angreifer kann zudem unbemerkt Chiffretext einfügen, der mit demselben Schlüssel verschlüsselt wurde.
- Eine Änderung der Blockreihenfolge bleibt zudem ebenfalls unbemerkt.

Der ECB-Mode sollte daher nicht für längere Nachrichten benutzt werden.

Verbesserungen durch CBC-Mode gegenüber ECB-Mode

Verbesserungen durch CBC-Mode gegenüber ECB-Mode

- Gleiche Texte werden bei geändertem Initialisierungsvektor unterschiedlich verschlüsselt.
- Gleiche Klartextblöcke werden unterschiedlich (kontextabhängig) verschlüsselt.
- Änderungen des Chiffretextes oder seiner Reihenfolge werden erkannt.

Nachteile und Probleme im CBC-Mode

- Übertragungsfehler in einem Block werden in den nächsten Block propagiert (jedoch nicht weiter).
- Benutzen Sender und Empfänger nicht denselben Initialisierungsvektor, kann der erste Block nicht entschlüsselt werden.

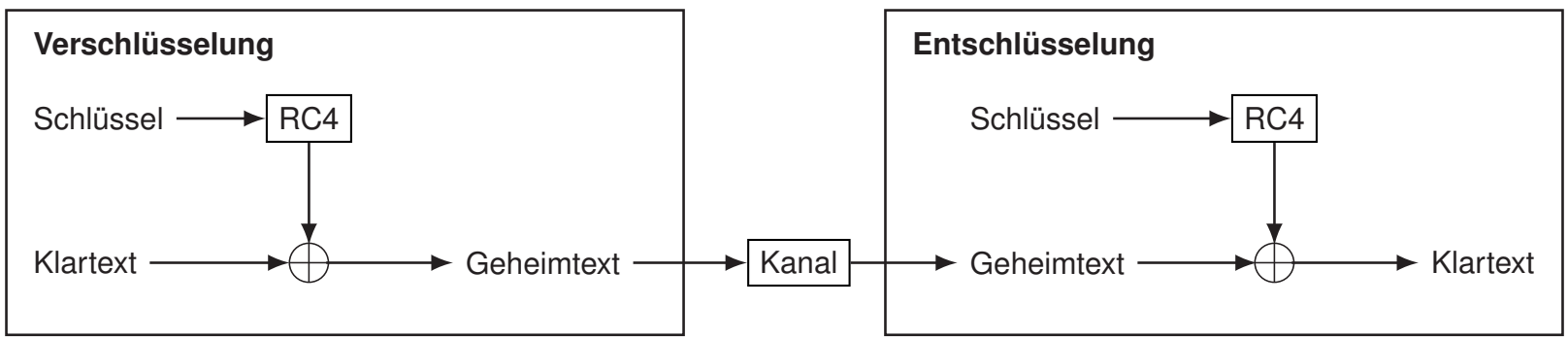
Stromchiffren: RC4

Verschlüsselung:

- Der Sender erzeugt mit Hilfe eines gemeinsamen Geheimnisses einen Schlüsselstrom (engl. cipher stream).
- Da der Bitstrom pseudozufällig ist, kann er mit Hilfe des gemeinsamen Geheimnisses von beiden Kommunikationspartnern reproduziert werden.
- Der Schlüsselstrom wird mit der zu verschlüsselnden Nachricht (Klartext, engl. plain text) bitweise XOR verknüpft.
- Der resultierende Datenstrom wird als Geheimtext (engl. cipher text) bezeichnet.

Entschlüsselung:

- Der Empfänger kann mit Hilfe des gemeinsamen Geheimnisses denselben Schlüsselstrom erzeugen.
- XOR-Verknüpfung aus Schlüsselstrom und Geheimtext ergibt wieder den Klartext.



Erzeugung des Schlüsselstroms (KSA, key-scheduling algorithm):

- RC4 initialisiert in Abhängigkeit des Schlüssels eine 256 Byte lange sog. Substitutions-Box (S-Box) mit den Zahlen 0 bis 255.
- Im Anschluss werden in jedem Schritt zwei Zahlen aus der S-Box permutiert und die Summe der beiden permutierten Zahlen modulo 256 als Index zu einer neuen Zahl in der S-Box verwendet, welche als „Zufallszahl“ zurückgegeben wird.[2]

Bei RC4 ist zu beachten:

- Es dürfen niemals zwei Nachrichten mit demselben Schlüsselstrom verschlüsselt werden, da andernfalls ein Angreifer, dem eine Nachricht im Klartext bekannt ist, auch die andere Nachricht im Klartext berechnen kann.
- Aufgrund der einfachen Funktionsweise lassen die ersten Bytes des Schlüsselstroms Rückschlüsse auf den Schlüssel selbst zu und sollten daher verworfen werden.
- Wie jeder Generator für pseudozufällige Zahlen hat RC4 eine begrenzte Periodendauer, d. h. irgendwann wiederholt sich der Schlüsselstrom.
- Bei RC4 wurden bestimmte Korrelationen zwischen Schlüssel und Schlüsselstrom gefunden, so dass RC4 heute als nicht mehr sicher angesehen wird.

Verschlüsselungsprotokolle (Cipher Suites), die RC4 nutzen:

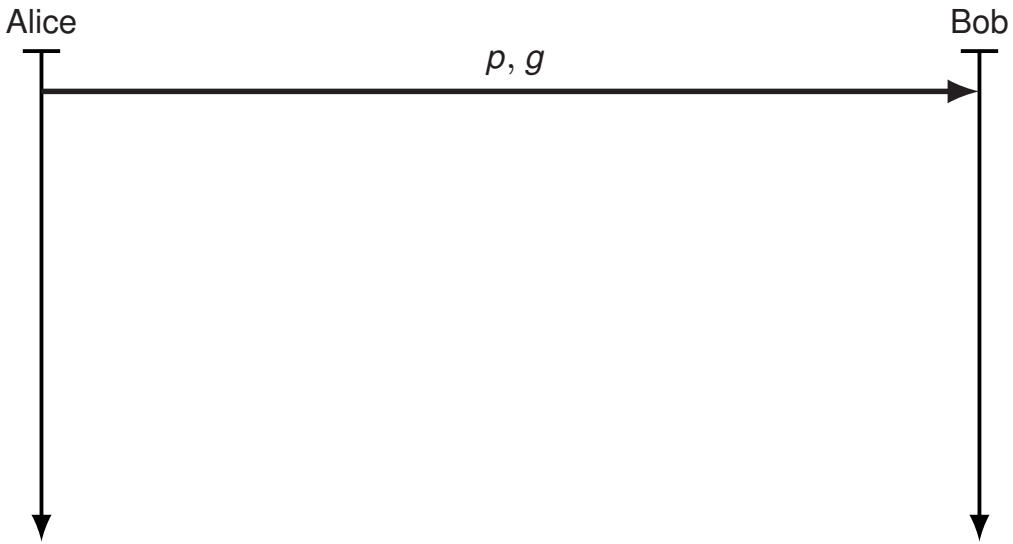
- WEP[3] (Wired Equivalent Privacy)
- WPA-TKIP (WiFi Protected Access Temporal Key Integrity Protocol)
- Als optionaler Algorithmus in IPSec, SSL/TLS, SSH, Kerberos

[2] Diese Darstellungsweise ist etwas vereinfachend.

[3] Die Schwäche liegt hier nicht bei RC4 selbst sondern bei der Art, wie RC4 genutzt wird.

Wie tauschen Sender und Empfänger die gemeinsamen Schlüssel aus?

Eine Möglichkeit besteht in der Nutzung des Diffie-Hellman-Verfahrens:

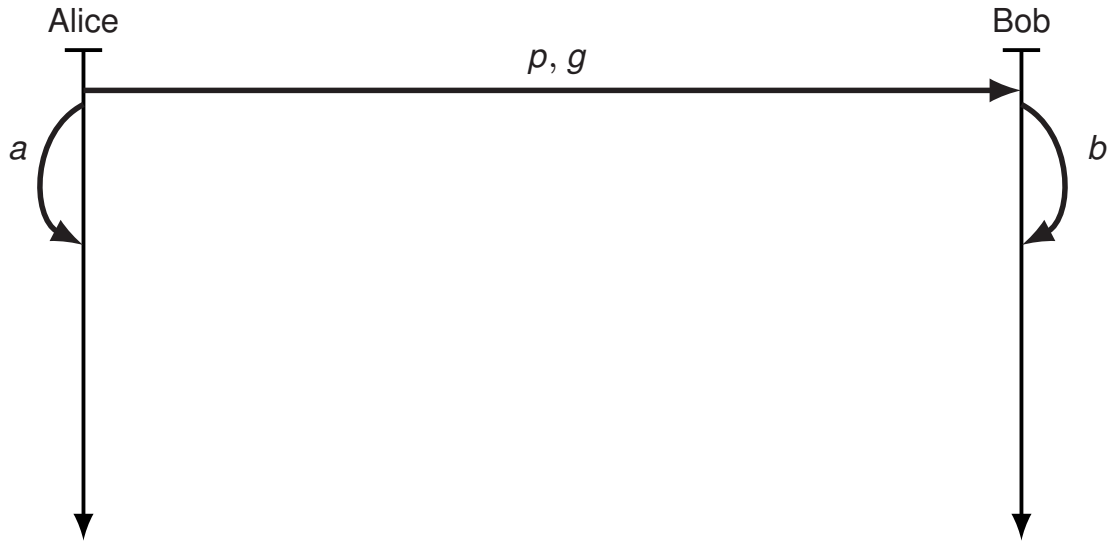


- Alice und Bob einigen sich zunächst auf eine Primzahl p und eine sog. primitive Kongruenzwurzel[4] g mod p.

[4] g ist eine primitive Kongruenzwurzel zur Primzahl p, wenn g^i mod p für $i = \{1, 2, \ldots, p-1\}$ alle Zahlen zwischen 0 und einschließlich $p-1$ erzeugt. Werte für p und g finden sich u.a. in RFC5114.

Wie tauschen Sender und Empfänger die gemeinsamen Schlüssel aus?

Eine Möglichkeit besteht in der Nutzung des Diffie-Hellman-Verfahrens:

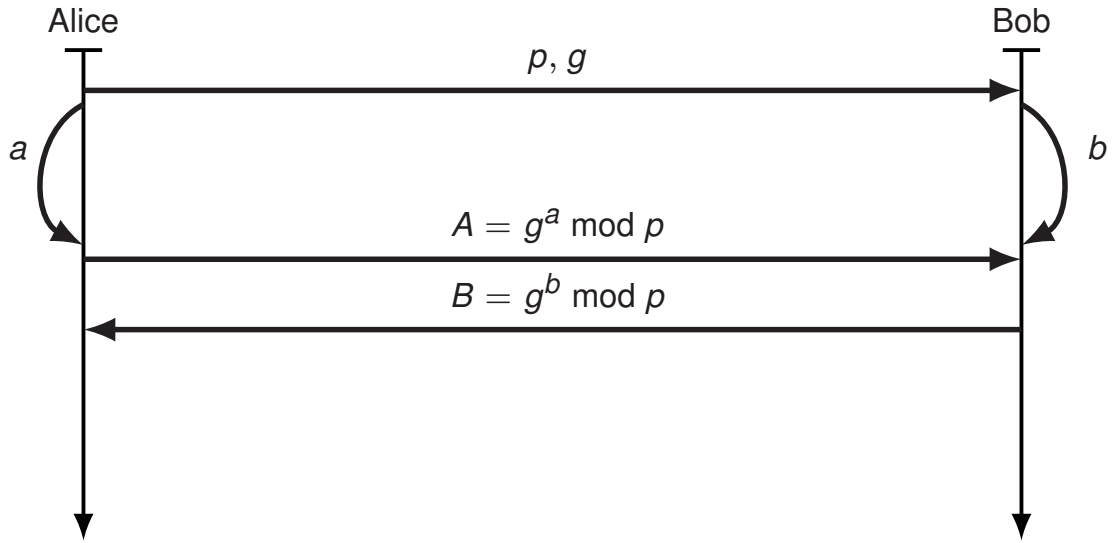


- Alice und Bob einigen sich zunächst auf eine Primzahl p und eine sog. primitive Kongruenzwurzel[4] g mod p.
- Beide erzeugen unabhängig voneinander jeweils eine Zufallszahl a und b.

[4] g ist eine primitive Kongruenzwurzel zur Primzahl p, wenn g^i mod p für $i = \{1, 2, \ldots, p-1\}$ alle Zahlen zwischen 0 und einschließlich $p-1$ erzeugt. Werte für p und g finden sich u.a. in RFC5114.

Wie tauschen Sender und Empfänger die gemeinsamen Schlüssel aus?

Eine Möglichkeit besteht in der Nutzung des Diffie-Hellman-Verfahrens:

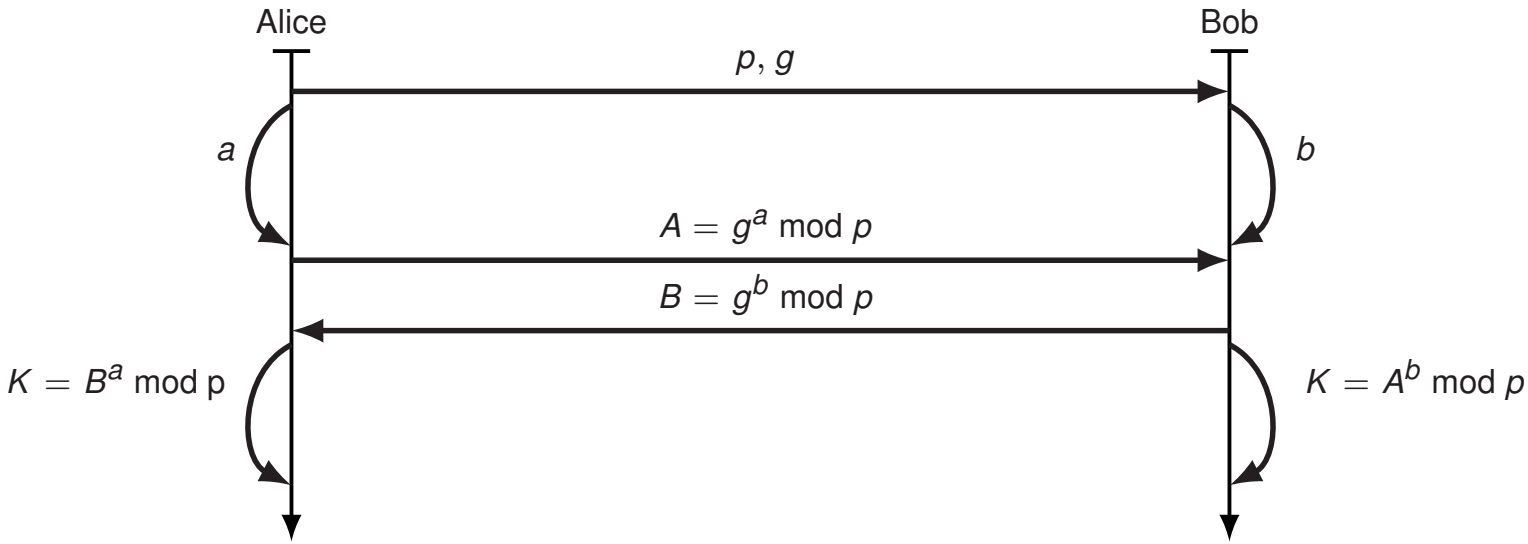


- Alice und Bob einigen sich zunächst auf eine Primzahl p und eine sog. primitive Kongruenzwurzel[4] g mod p.
- Beide erzeugen unabhängig voneinander jeweils eine Zufallszahl a und b.
- Nun werden die Zahlenwerte $B = g^b \bmod p$ bzw. $A = g^a \bmod p$ ausgetauscht. Die Zufallszahlen a und b bleiben aber geheim.

[4] g ist eine primitive Kongruenzwurzel zur Primzahl p, wenn $g^i \bmod p$ für $i = \{1, 2, \ldots, p-1\}$ alle Zahlen zwischen 0 und einschließlich $p-1$ erzeugt. Werte für p und g finden sich u.a. in RFC5114.

Wie tauschen Sender und Empfänger die gemeinsamen Schlüssel aus?

Eine Möglichkeit besteht in der Nutzung des Diffie-Hellman-Verfahrens:



- Alice und Bob einigen sich zunächst auf eine Primzahl p und eine sog. primitive Kongruenzwurzel[4] g mod p.
- Beide erzeugen unabhängig voneinander jeweils eine Zufallszahl a und b.
- Nun werden die Zahlenwerte $B = g^b \bmod p$ bzw. $A = g^a \bmod p$ ausgetauscht. Die Zufallszahlen a und b bleiben aber geheim.
- Alice und Bob können nun jeweils $A^b \bmod p = B^a \bmod p = K$ berechnen.

[4] g ist eine primitive Kongruenzwurzel zur Primzahl p, wenn $g^i \bmod p$ für $i = \{1, 2, \ldots, p-1\}$ alle Zahlen zwischen 0 und einschließlich $p-1$ erzeugt. Werte für p und g finden sich u.a. in RFC5114.

Funktioniert das denn?

Zu zeigen: $K \stackrel{!}{=} A^b \bmod p \stackrel{!}{=} B^a \bmod p$

$$\begin{aligned} A^b \bmod p &= \left(g^a\right)^b \bmod p \\ &= \left(g^b\right)^a \bmod p \\ &= B^a \bmod p \end{aligned}$$

Ist das Verfahren sicher?

Gegenfrage: Was ist sicher?

- RC4 und das Diffie-Hellman-Verfahren wie hier vorgestellt sind anfällig gegenüber Man-in-the-Middle-Angriffe:
  - Schafft es Eve, während des Verbindungsaufbaus sowohl die Nachrichten von Alice als auch von Bob abzufangen, kann sie sich für den jeweils anderen Kommunikationspartner ausgeben und unterschiedliche Schlüssel mit Alice und Bob aushandeln.
  - Nach dem Schlüsselaustausch muss Eve weiterhin in der Lage sein, alle Nachrichten von Alice bzw. Bob abzufangen.
  - Diese können dann entschlüsselt, gelesen und modifiziert und anschließend neu verschlüsselt und weitergeleitet werden.
- Aus diesem Grund ist keines der Kriterien Integrität, Authentizität, Vertraulichkeit und Verbindlichkeit wirklich erfüllt.

$\Rightarrow$ RC4 und das Diffie-Hellman-Verfahren alleine sind ein Anfang. Für eine sichere Kommunikation wird aber ein komplettes Protokoll benötigt.

Hashfunktionen

Mittels kryptografisch sicheren Hashfunktionen soll die Integrität der Nachrichten sichergestellt werden. Man kann zwischen starken und schwachen Hashfunktionen unterscheiden.

Eine injektive Funktion $H : A^* \rightarrow B^k$ über die Alphabete A und B ist eine schwache Hashfunktion, wenn gilt:

- H ist eine Einweg-Funktion, d.h. die Umkehrfunktion ist nicht effizient berechenbar
- Der Hashwert $H(M) = h$ mit $|h| = k$ ist für gegebene M effizient berechenbar
- Es ist praktisch unmöglich, ein M' zu bestimmen, für das bei gegebenem $h = H(M)$ gilt: $h = H(M')$ mit $M \neq M'$

Eine starke Hashfunktion ist eine schwache Hasfunktion, für welche zusätzlich gilt:

- Es ist praktisch unmöglich, ein Paar von Nachrichten M und M' mit $M \neq M'$ zu bestimmen, für das gilt: $H(M) = H(M')$

Vorgehen bei der Nachrichtenüberprüfung mit Hinblick auf Integrität:

- Der Sender bestimmt zu Nachricht M den Hashwert $h = H(M)$ und sendet das Tupel (M, h)
- Der Empfänger bestimmt für die erhaltene Nachricht M' den zugehörigen Hashwert $H' = H(M')$
- Die Nachricht ist integer, wenn gilt: $h = h'$

Digitale Signaturen

- Bisher konnte Integritätsschutz der Nachricht sicher gestellt werden.
- Es ist aber weiterhin möglich, die Nachricht zu manipulieren und den Hash erneut zu berechnen.
- Im nächsten Schritt soll dies nun verhindert werden, indem auch der Sender der Nachricht mit dem Inhalt verbunden wird.

Definition (Digitale Signatur)

Um ein elektronisches Dokument zu unterschreiben, werden digitale Signaturen verwendet. Diese stellen die Verbindlichkeit bezüglich der signierenden Partei sicher und ermöglichen die Verifikation, dass ein Dokument auch wirklich von der signierenden Partei stammt.

Eigenschaften digitaler Signaturen:

- authentisch, d.h. die signierende Partei hat willentlich unterschrieben
- fälschungssicher, d.h. die signierende Partei hat unterschrieben und niemand anders
- nicht wiederverwendbar, d.h. die Signatur kann nicht auf ein anderes Dokument angewandt werden
- unveränderbar, d.h. nach dem Signieren kann die Unterschrift nicht unbemerkt geändert werden
- bindend, d.h. der Unterzeichner kann das Unterschreiben nicht abstreiten

Verschiedene Umsetzungsmöglichkeiten

Message Authentication Code

- Die Kommunikationspartner einigen sich auf einen geheimen Schlüssel K.
- Dieser wird zusätzlich in die Hashfunktion eingebracht.
- Das weitere Vorgehen ist dann analog zur Bestimmung der Integrität mittels Hashfunktionen:
  $\text{MAC}(M', K) = mac' \stackrel{?}{=} mac = \text{MAC}(M, K)$

Bei asymmetrischen Verfahren (z.B. RSA):

- Eine Nachricht m soll mittels einer digitalen Signatur signiert werden.
- Hierfür wird der private (geheime) Schlüssel K_{priv} verwendet: $E(K_{priv}, m) = sig_m$
- das signierte Dokument sig_m kann nun verschickt werden.
- Diese Signatur kann mit dem öffentlichen Schlüssel K_{pub} verifiziert werden:
  $m = D(sig_m, K_{pub})$
- Dieser öffentliche Schlüssel befindet sich öffentlich zugänglich, z.B. auf einem Key-Server.

Übersicht

Motivation

Bisher gelöste Probleme:

- Die Nachricht kann nicht von Unbefugten gelesen werden (Verschlüsselung)
- Die Integrität der Nachricht ist gesichert (Hashfunktionen)
- Der Sender der Nachricht kann verifiziert werden (Digitale Signatur mittels MAC)

Motivation

Bisher gelöste Probleme:

- Die Nachricht kann nicht von Unbefugten gelesen werden (Verschlüsselung)
- Die Integrität der Nachricht ist gesichert (Hashfunktionen)
- Der Sender der Nachricht kann verifiziert werden (Digitale Signatur mittels MAC)

Kann man wirklich sicher sein, dass Alice auch Alice ist?

Motivation

Bisher gelöste Probleme:

- Die Nachricht kann nicht von Unbefugten gelesen werden (Verschlüsselung)
- Die Integrität der Nachricht ist gesichert (Hashfunktionen)
- Der Sender der Nachricht kann verifiziert werden (Digitale Signatur mittels MAC)

Kann man wirklich sicher sein, dass Alice auch Alice ist?

Definition (Man-in-the-Middle (MITM) Attack)

Unter dem Begriff Man-in-the-Middle (MITM) Attack versteht man einen Angriff auf Rechnernetze bei denen sich ein Angreifer logisch oder physikalisch zwischen die beiden Kommunikationsendpunkte bringt und somit den gesamten Netzverkehr abhören und manipulieren kann.

Motivation

Bisher gelöste Probleme:

- Die Nachricht kann nicht von Unbefugten gelesen werden (Verschlüsselung)
- Die Integrität der Nachricht ist gesichert (Hashfunktionen)
- Der Sender der Nachricht kann verifiziert werden (Digitale Signatur mittels MAC)

Kann man wirklich sicher sein, dass Alice auch Alice ist?

Definition (Man-in-the-Middle (MITM) Attack)

Unter dem Begriff Man-in-the-Middle (MITM) Attack versteht man einen Angriff auf Rechnernetze bei denen sich ein Angreifer logisch oder physikalisch zwischen die beiden Kommunikationsendpunkte bringt und somit den gesamten Netzverkehr abhören und manipulieren kann.

- Dieser Angriff untergräbt die bisherigen Sicherheitsmechanismen, da nicht sicher gestellt ist, dass die Absicherungen mit dem entsprechenden beabsichtigten Kommunikationspartner erfolgen.
- Im Folgenden soll der MITM Angriff kurz illustriert werden und entsprechende Absicherungen aufgezeigt werden.

MITM - Angriff

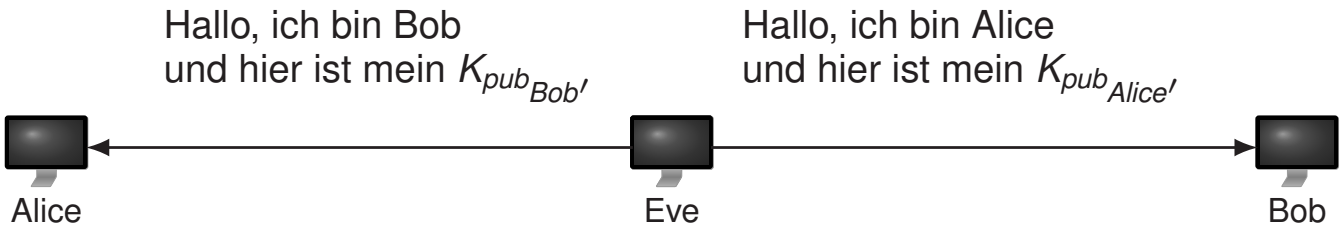

Alice


Eve


Bob

- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.

MITM - Angriff

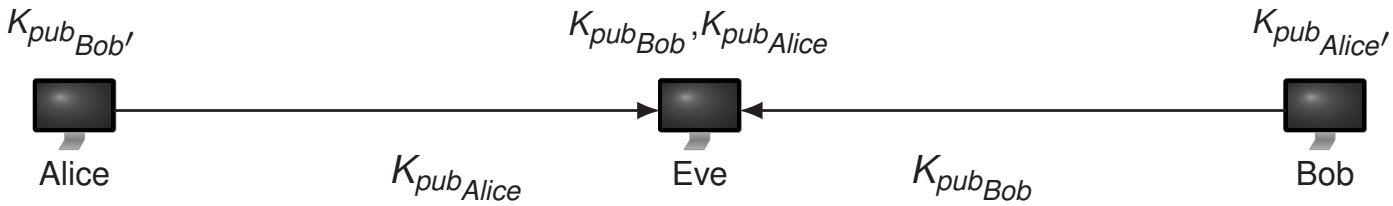


- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.
- Eve sendet ihren öffentlichen Schlüssel an Alice und Bob.

MITM - Angriff

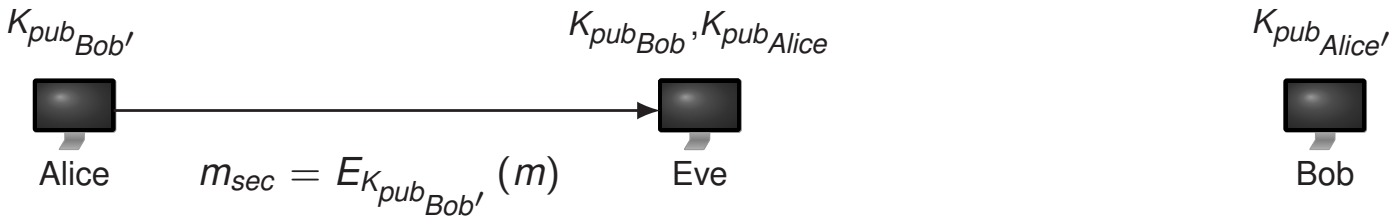
- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.
- Eve sendet ihren öffentlichen Schlüssel an Alice und Bob.
- Sowohl Alice als auch Bob besitzen nun die vermeintlichen Schlüssel des jeweils anderen.

MITM - Angriff



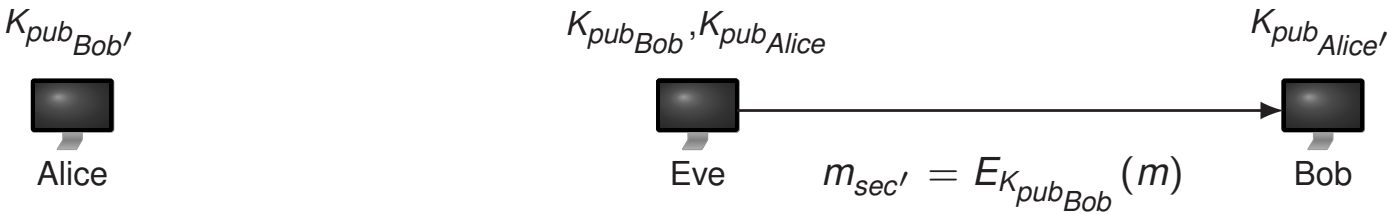
- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.
- Eve sendet ihren öffentlichen Schlüssel an Alice und Bob.
- Sowohl Alice als auch Bob besitzen nun die vermeintlichen Schlüssel des jeweils anderen.
- Alice und Bob können nun auch ihre öffentlichen Schlüssel austauschen, sodass auch Eve die beiden entsprechenden Schlüssel hat.

MITM - Angriff



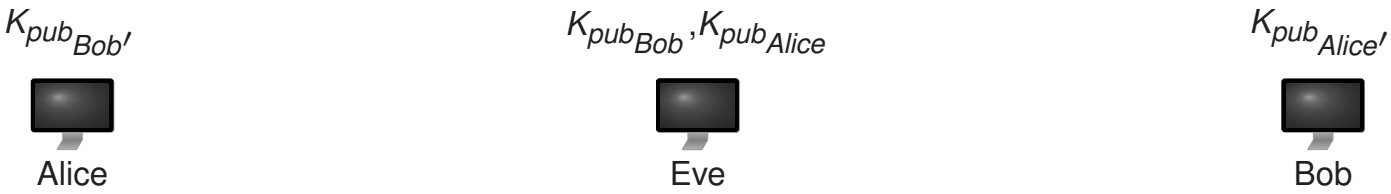
- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.
- Eve sendet ihren öffentlichen Schlüssel an Alice und Bob.
- Sowohl Alice als auch Bob besitzen nun die vermeintlichen Schlüssel des jeweils anderen.
- Alice und Bob können nun auch ihre öffentlichen Schlüssel austauschen, sodass auch Eve die beiden entsprechenden Schlüssel hat.
- Alice sendet nun eine geheime Nachricht m unter Zuhilfenahme von $K_{pub_{Bob'}}$ an Bob.

MITM - Angriff



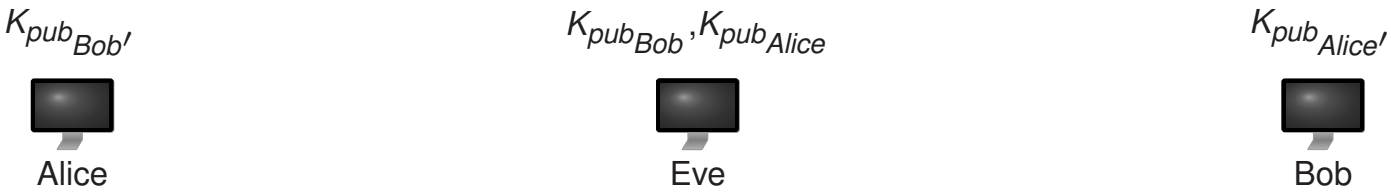
- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.
- Eve sendet ihren öffentlichen Schlüssel an Alice und Bob.
- Sowohl Alice als auch Bob besitzen nun die vermeintlichen Schlüssel des jeweils anderen.
- Alice und Bob können nun auch ihre öffentlichen Schlüssel austauschen, sodass auch Eve die beiden entsprechenden Schlüssel hat.
- Alice sendet nun eine geheime Nachricht m unter Zuhilfenahme von $K_{pub_{Bob'}}$ an Bob.
- Eve kann m_{sec} mit Hilfe ihre privaten Schlüssels entschlüsseln und durch Zuhilfenahme von Bobs öffentlichen Schlüssel unbemerkt weiterleiten.

MITM - Angriff



- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.
- Eve sendet ihren öffentlichen Schlüssel an Alice und Bob.
- Sowohl Alice als auch Bob besitzen nun die vermeintlichen Schlüssel des jeweils anderen.
- Alice und Bob können nun auch ihre öffentlichen Schlüssel austauschen, sodass auch Eve die beiden entsprechenden Schlüssel hat.
- Alice sendet nun eine geheime Nachricht m unter Zuhilfenahme von $K_{pub_{Bob'}}$ an Bob.
- Eve kann m_{sec} mit Hilfe ihre privaten Schlüssels entschlüsseln und durch Zuhilfenahme von Bobs öffentlichen Schlüssel unbemerkt weiterleiten.
- Bob kann die Nachricht mit Hilfe seines privaten Schlüssels entschlüsseln und alle Beteiligten (auch Eve) sind nun in Besitz der geheimen Nachricht.

MITM - Angriff



- Alice möchte mit Bob kommunizieren und beide besitzen ein Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel.
- Eve sendet ihren öffentlichen Schlüssel an Alice und Bob.
- Sowohl Alice als auch Bob besitzen nun die vermeintlichen Schlüssel des jeweils anderen.
- Alice und Bob können nun auch ihre öffentlichen Schlüssel austauschen, sodass auch Eve die beiden entsprechenden Schlüssel hat.
- Alice sendet nun eine geheime Nachricht m unter Zuhilfenahme von $K_{pub_{Bob'}}$ an Bob.
- Eve kann m_{sec} mit Hilfe ihre privaten Schlüssels entschlüsseln und durch Zuhilfenahme von Bobs öffentlichen Schlüssel unbemerkt weiterleiten.
- Bob kann die Nachricht mit Hilfe seines privaten Schlüssels entschlüsseln und alle Beteiligten (auch Eve) sind nun in Besitz der geheimen Nachricht.

Die Nutzung von Certification Authorities oder dem Web of Trust können hier helfen.

Zertifikate

Definition (Zertifikate)

Ein Zertifikat bindet einen öffentlichen Schlüssel an eine natürliche oder juristische Person.

- Die Struktur eines Zertifikats ist im *Standard X.509* festgelegt.
- Dieser Standard ist Teil eines größeren Frameworks, von allerdings nur noch dieser Teil von praktischer Relevanz ist.

Bezeichnung	Beschreibung
Versionsnummer	Angabe des verwendeten Zertifikatformats
Seriennummer	eindeutige ID
Signatur	benutzte Parameter und Algorithmen
Zertifikataussteller	Name der Instanz, welche das Zertifikat ausstellt
Gültigkeitsdauer	Zeitintervall
Benutzername	eindeutiger Name des Benutzers
Schlüsselinformationen	verwendeter Schlüssel des Benutzers inkl. Algorithmen

- Der Aussteller ist verantwortlich, dass die Seriennummer eindeutig bleibt.
- Um ein Zertifikat verifizieren zu können, müssen die verwendeten Hash- und Signierverfahren, sowie der öffentliche Schlüssel des Inhabers im Zertifikat enthalten sein.

Beispiel - Root Certificate der Telekom I

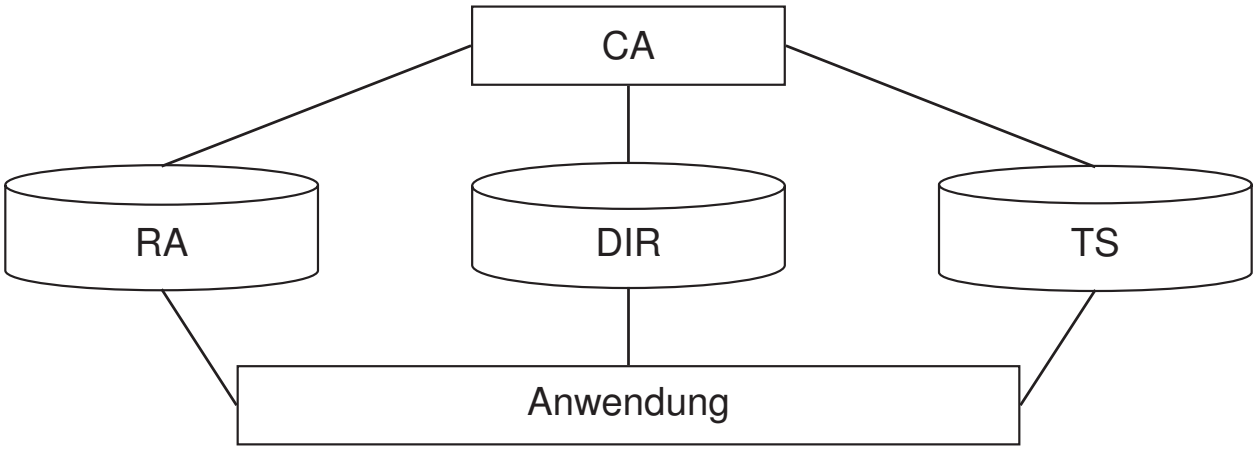
```
Certificate:

  Data:
    Version: 3 (0x2)
    Serial Number: 38 (0x26)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=DE, O=Deutsche Telekom AG, OU=T-TeleSec Trust Center,
      CN=Deutsche Telekom Root CA 2
    Validity
      Not Before: Jul  9 12:11:00 1999 GMT
      Not After : Jul  9 23:59:00 2019 GMT
    Subject: C=DE, O=Deutsche Telekom AG, OU=T-TeleSec Trust Center,
      CN=Deutsche Telekom Root CA 2
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
        Modulus (2048 bit):
          00:ab:0b:a3:35:e0:8b:29:14:b1:14:85:af:3c:10:
          e4:39:6f:35:5d:4a:ae:dd:ea:61:8d:95:49:f4:6f:
          ...
          bc:da:03:34:d5:8e:5b:01:f5:6a:07:b7:16:b6:6e:
          4a:7f
        Exponent: 65537 (0x10001)
```

Beispiel - Root Certificate der Telekom II

```
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        31:C3:79:1B:BA:F5:53:D7:17:E0:89:7A:2D:17:6C:0A:B3:2B:9D:33
      X509v3 Basic Constraints:
        CA:TRUE, pathlen:5
      X509v3 Key Usage: critical
         Certificate Sign, CRL Sign

  Signature Algorithm: sha1WithRSAEncryption
    94:64:59:ad:39:64:e7:29:eb:13:fe:5a:c3:8b:13:57:c8:04:
    24:f0:74:77:c0:60:e3:67:fb:e9:89:a6:83:bf:96:82:7c:6e:
    ...
    57:99:94:0a:6d:ba:39:63:28:86:92:f3:18:84:d8:fb:d1:cf:
    05:56:64:57

-----BEGIN CERTIFICATE-----
MIIDnzCCAoegAwIBAgIBJjANBgkqhkiG9w0BAQUFADBxMQswCQYDVQQGEwJERTEc
MBoGA1UEChMTRGV1dHNjaGUgVGVsZWtvbSBBRzEfMB0GA1UECxMWVC1UZWxlU2Vj
IFRydXN0IEN1bnRlcjEjMCEGA1UEAxMaRGV1dHNjaGUgVGVsZWtvbSBSb290IENB
...
6iFhkOQxIY4OsfcvNUqFENrnijchvllj4PKFiDFT1FQUhXB59C4Gdyd1Lx+4ivn+
xbrYNuSD7Odlt79jWvNGr4GUN9RBjNYj1h7P9WgbRGOiWrqnNVmh5XAFmw4jV5mU
Cm26OWMohpLzGITY+9HPBVZkVw==
-----END CERTIFICATE-----
```

Zertifizierungsstellen

- Zertifizierungsinstanz (Certification Authority (CA)) bieten Dienste zur Ausstellung von Zertifikaten an.
- Das ausgestellte Zertifikat wird mit dem privaten Schlüssel der CA signiert.
- Die Gesamtheit der benötigten Komponenten wird in diesem Zusammenhang meist als Public-Key Infrastructure (PKI) bezeichnet.



- Die Registration Authority (RA) übernimmt die Verknüpfung zwischen öffentlichem Schlüssel und Indentitäten.
- Der Verzeichnisdienst (DIR), z. B. in Form eines LDAP-Verzeichnisses gibt Auskunft über die eigenen Zertifikate.
- Ein Zeitstempeldienst (TS) kann Daten vertrauenswürdig mit Zeitpunkten verbinden.

Hierarchisches Vertrauensmodell

- Bei den beschriebenen Zertifizierungsstellen wird von einem hierarchischen Vertrauensmodell ausgegangen.
- Basis für ein solches System ist eine Wurzelzertifizierungsinstanz (Root-CA) unterhalb derer sich mehrere CAs befinden, welche weiterhin hierarchische untergliedert sein können und von der Root-CA zertifiziert sind.
- Die signierten Benutzerschlüssel können über sie entsprechende Zertifikatskette bis zum Wurzelzertifikat geprüft werden.

Wollen zwei Benutzer das Zertifikat des jeweils anderen prüfen, dann können beide:

- bei der gleichen CA sein und sind somit im Besitz des öffentlichen Schlüssels der Zertifizierungsstelle.
- bei der gleichen Root-CA sein und somit über den Zertifizierungspfad die Prüfung durchführen.
- bei unterschiedlichen Root-CAs, d.h. unterschiedlichen Vertrauensbereichen sein, sodass ein Cross-Zertifikat notwendig ist.

Definition (Cross-Zertifikate)

Ein Cross-Zertifikat baut Zertifizierungspfade über verschiedene CAs unabhängig von der Hierarchie hinweg auf um die Bildung von Zertifizierungspfaden zu ermöglichen. Eine CA_A zertifiziert dabei das Zertifikat einer CA_B.

Statt einer Vertrauenshierarchie, kann auch eine netzartige Vertrauensstruktur etabliert werden: Web of Trust.

Web of Trust

Definition (Web of Trust)

Dem Web of Trust liegt ein Vertrauensnetz zu Grunde, welches auf transitiven Vertrauensbeziehungen beruht.

- Die Funktion die Echtheit eines öffentlichen Schlüssels zu bestätigen, übernehmen hier die Benutzer anstelle einer zentralen Zertifizierungsinstanz.
- Zu jedem öffentlichen Schlüssel anderer Benutzer wird der sog. owner trust festgelegt, welcher beschreibt, wie sehr man den Signaturen des jeweiligen Benutzer vertraut.
- Zusätzlich werden den Schlüsseln eine Menge von Signaturen zugeordnet, welche den öffentlichen Schlüssel zertifizieren (inkl. des Vertrauensgrades (signatory trust)).
- Die Schlüssel können beispielsweise über sog. Key-Server oder als e-Mail Anhänge zugänglich gemacht werden.

Unterschiedliche Vertrauensstufen:

- *ultimate:* eigener Schlüssel
- *complete:* dem Besitzer wird immer vertraut
- *marginal:* dem Besitzer wird normalerweise vertraut
- *untrusted:* dem Besitzer wird nicht vertraut
- *unknown*

Dieses System kommt Beispielsweise bei PGP (Pretty Good Privacy) zum Einsatz.

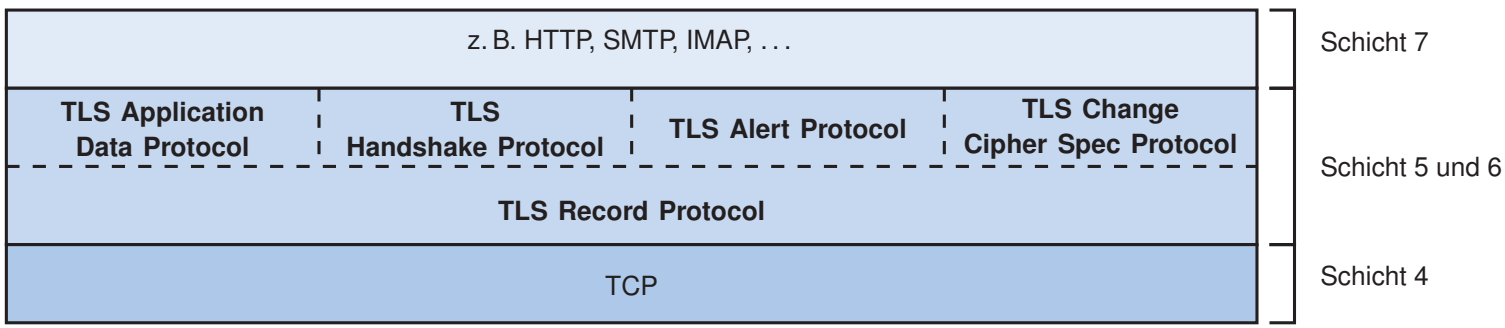
Übersicht

TLS

Das Transport Layer Security (TLS) Protokoll ist ein von der IETF standardisiertes Protokoll. In seiner neuesten Version (Version 1.2) ist es durch den RFC 5246 spezifiziert. Ergänzungen dazu finden sich in den RFCs 5746, 5878 und 6176.

TLS setzt auf ein verbindungsorientiertes Transportprotokoll (TCP) auf. Die zu TLS gehörenden Teilprotokolle unterstützen zahlreiche Funktionen. Im Rahmen dieser Vorlesung wird nur ein Teil der Funktionen angesprochen.

- **Handshake Protocol**
  - Aushandlung der Eigenschaften einer Sitzung (Verschlüsselungsalgorithmen, Kompressionsverfahren)
  - Authentifizierung von Instanzen
- **Record Protocol**
  - Fragmentierung der Anwendungsdaten
  - Ver- und Entschlüsselung
  - Kompression und Dekompression



Dienste von TLS

Die in TLS enthaltenen Funktionen dienen zur Realisierung folgender Dienste

- Authentisierung der Instanzen
- Vertraulichkeit der Nachrichten (bzw. der Nutzerdaten)
- Authentisierung und Integritätswahrung der Nachrichten

Diese Dienste werden auf folgende Weise realisiert

- Authentisierung der Instanzen: Bei TLS wird hierzu ein Dialog im Rahmen des TLS Handshake Protocols durchgeführt. Entweder authentisiert sich nur der Server gegenüber dem Client mit Hilfe von Zertifikaten, oder auch (zusätzlich) der Client gegenüber dem Server.
- Vertraulichkeit der Nachrichten: Die ausgetauschten Nutzerdaten werden hierzu mit einem symmetrischen Verschlüsselungsverfahren wie RC4 oder AES verschlüsselt.
- Authentisierung der Nachrichten und Integrität der Nachrichten: Hierzu wird für jede ausgetauschte Nachricht eine kryptografische Hash-Funktion berechnet. In die Berechnung des Hash-Werts fließt ein beiden Seiten bekannter Schlüssel ein.

Dienste von TLS

Das Protokoll TLS kann zum Teil der Sitzungsschicht zugeordnet werden, da es den Kommunikationspartnern erlaubt, einen Sitzungszustand mit gemeinsam bekannten Informationen aufzubauen.

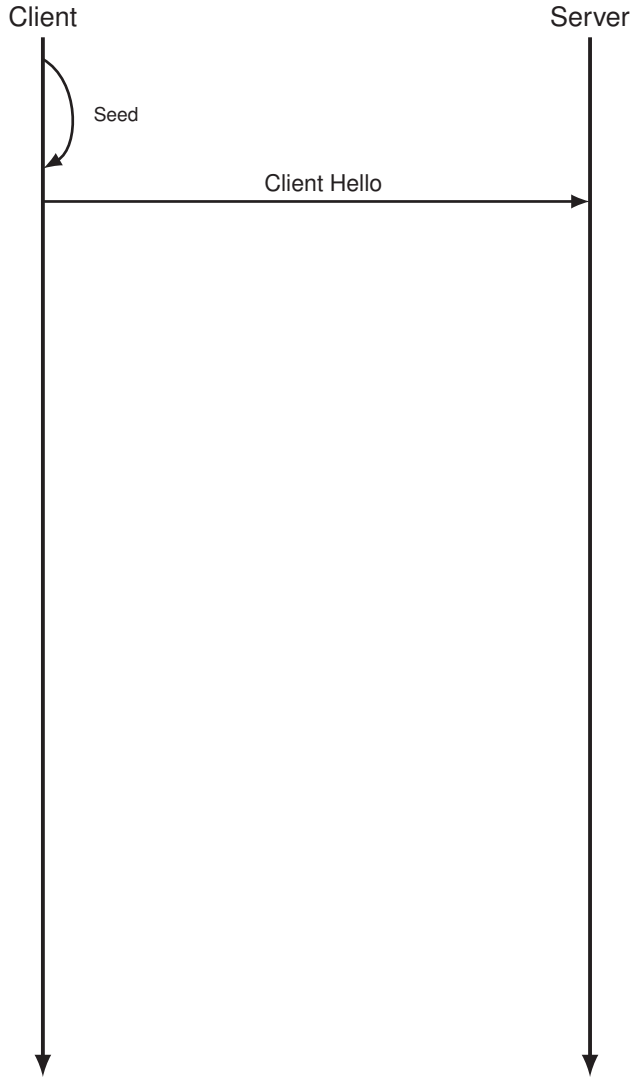
Als Sitzungszustand wird eine Reihe von Informationen (sog. Attribute) verwaltet. Dazu gehören:

- Sitzungsidentifikator (Session ID)
- Zertifikat der Instanz des Kommunikationspartners (Peer Certificate)
- Kryptografische Algorithmen und ihre Parameter (Cypher Suite)
- Funktion zur Kompression (Compression Method)
- Jeweils eine von jeder Instanz generierte Zufallszahl (Seed: ClientHello.random und ServerHello.random)
- Mehrere Schlüssel zur Verschlüsselung und zur Nachrichtenauthentisierung. Zunächst wird ein Premaster-Secret ausgehandelt. Daraus werden dann weitere Schlüssel abgeleitet.

Hinweis: TLS verwendet abhängig von den zur Verfügung stehenden Authentifizierungsmethoden unterschiedliche Handshakes. Im folgenden betrachten wir den TLS Simple Handshake mit serverseitigem Zertifikat.

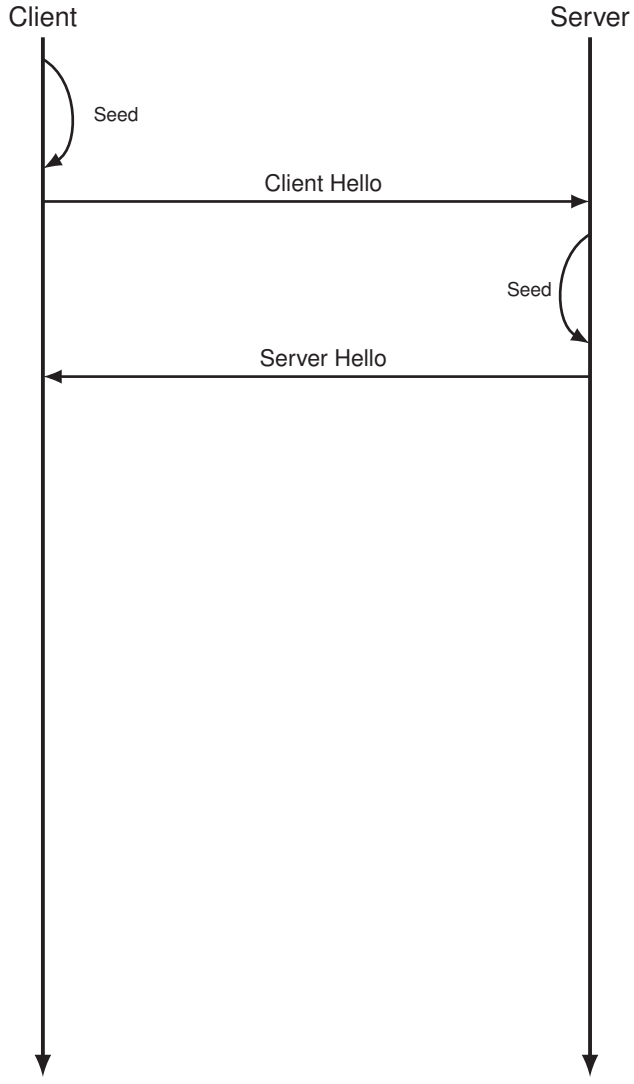
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)



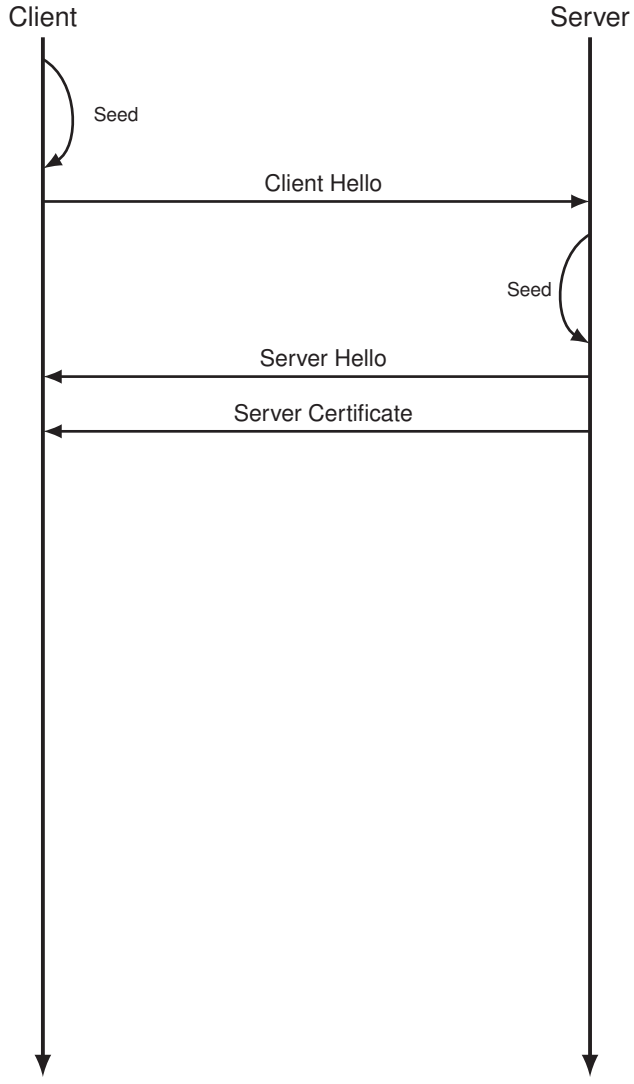
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl



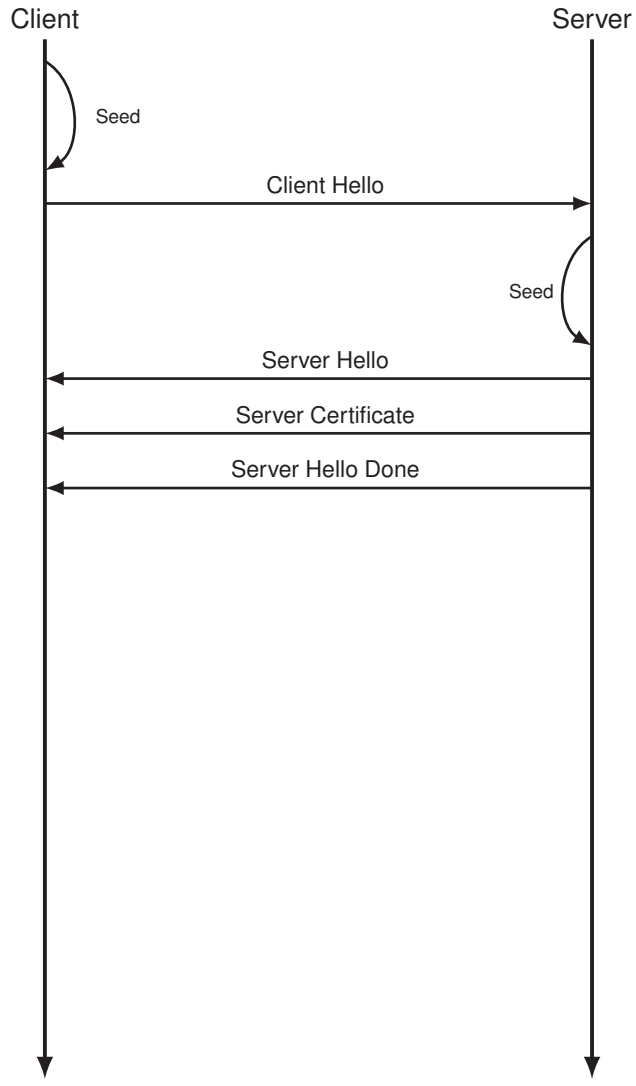
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl
- **Server Certificate**
  Enthält den öffentlichen Schlüssel des Servers



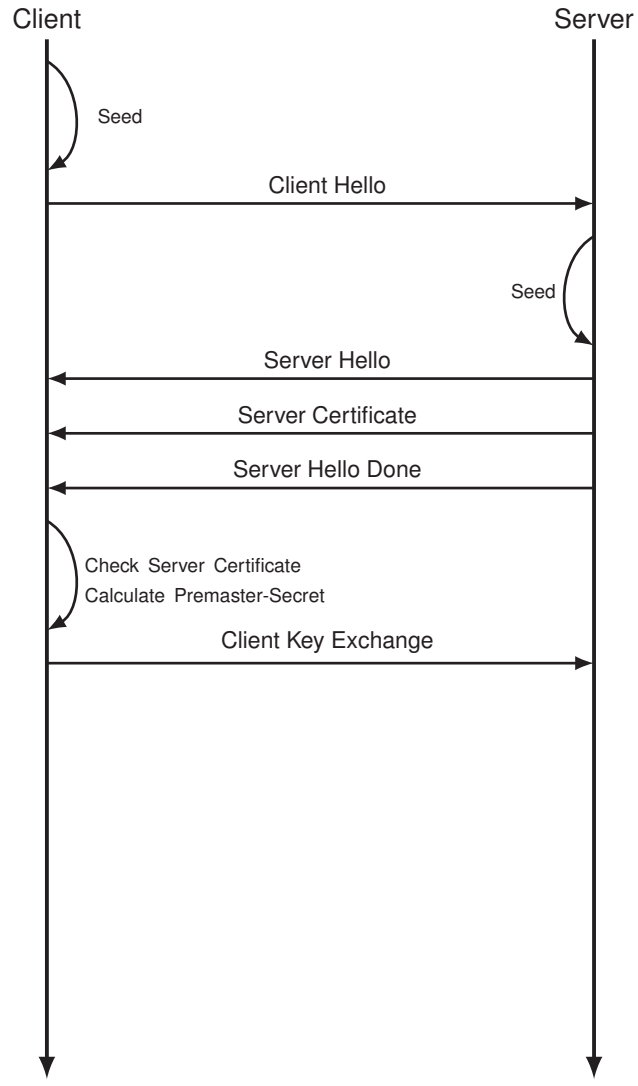
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl
- **Server Certificate**
  Enthält den öffentlichen Schlüssel des Servers
- **Server Hello Done**
  Ende des serverseitigen Handshakes



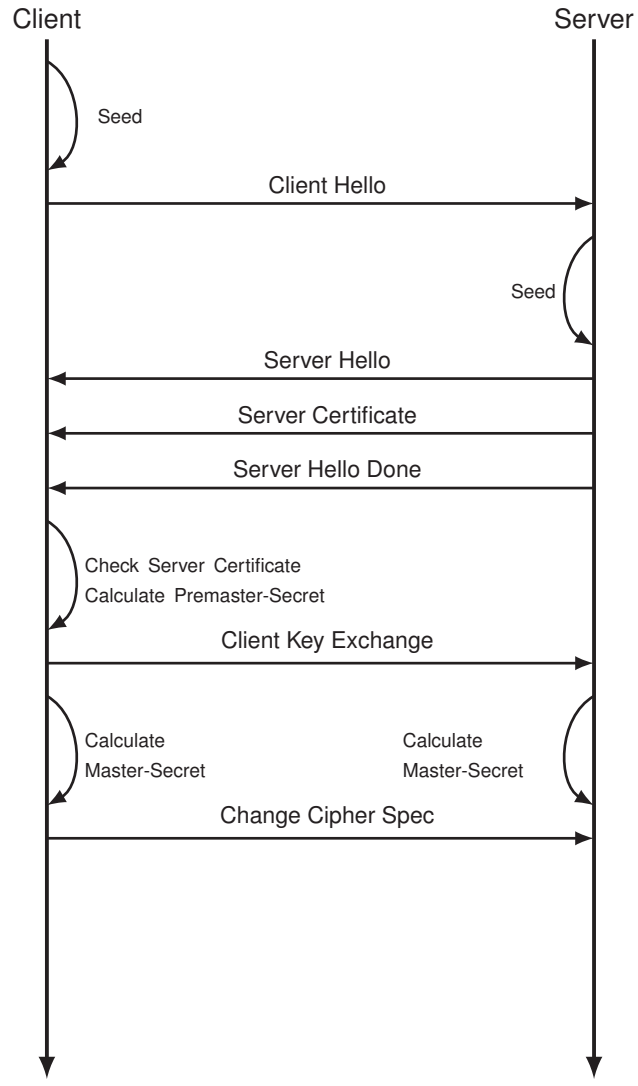
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl
- **Server Certificate**
  Enthält den öffentlichen Schlüssel des Servers
- **Server Hello Done**
  Ende des serverseitigen Handshakes
- **Client Key Exchange**
  Mit RSA asymmetrisch verschlüsseltes Premaster-Secret, aus dem zusammen mit den Seeds ein gemeinsames Master-Secret für die symmetrische Verschlüsselung abgeleitet wird



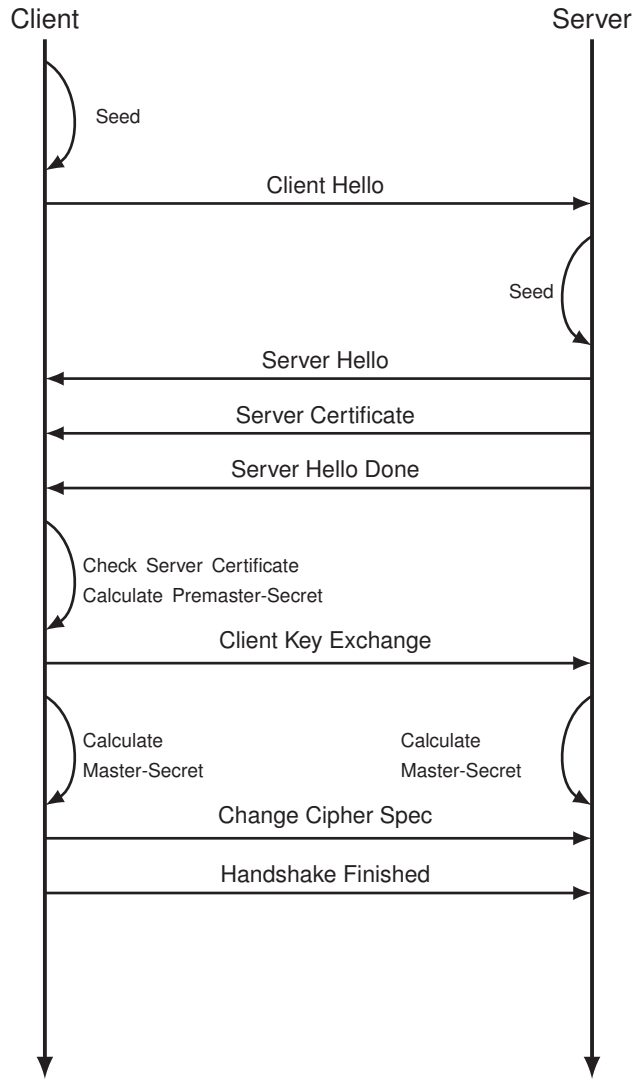
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl
- **Server Certificate**
  Enthält den öffentlichen Schlüssel des Servers
- **Server Hello Done**
  Ende des serverseitigen Handshakes
- **Client Key Exchange**
  Mit RSA asymmetrisch verschlüsseltes Premaster-Secret, aus dem zusammen mit den Seeds ein gemeinsames Master-Secret für die symmetrische Verschlüsselung abgeleitet wird
- **Change Cipher Spec**
  Letzte unverschlüsselte Nachricht des Clients



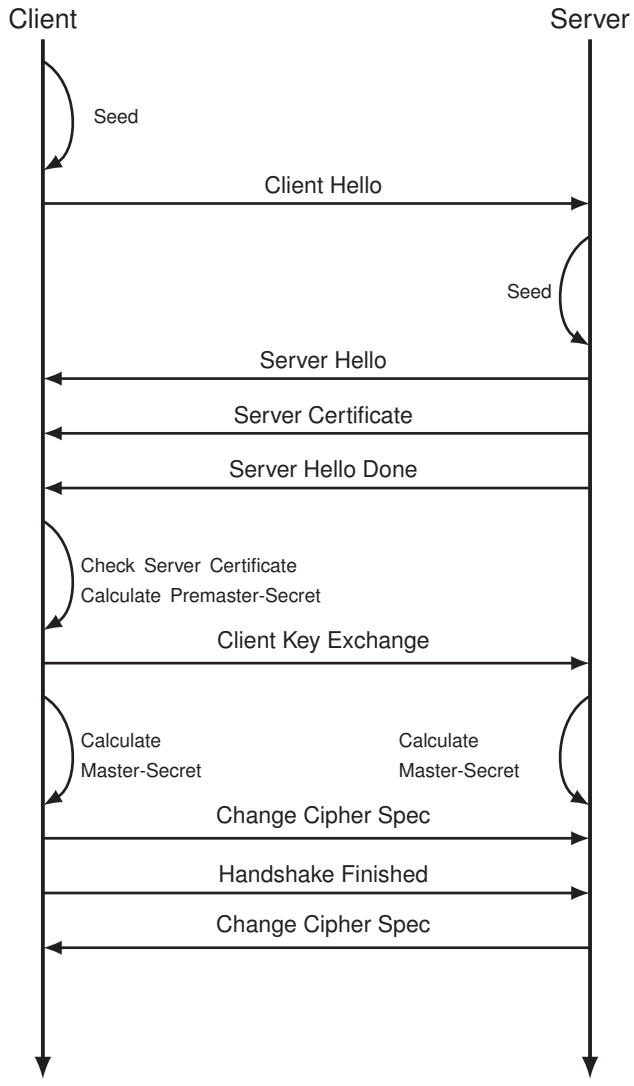
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl
- **Server Certificate**
  Enthält den öffentlichen Schlüssel des Servers
- **Server Hello Done**
  Ende des serverseitigen Handshakes
- **Client Key Exchange**
  Mit RSA asymmetrisch verschlüsseltes Premaster-Secret, aus dem zusammen mit den Seeds ein gemeinsames Master-Secret für die symmetrische Verschlüsselung abgeleitet wird
- **Change Cipher Spec**
  Letzte unverschlüsselte Nachricht des Clients
- **Handshake Finished**
  Erste verschlüsselte Nachricht des Clients



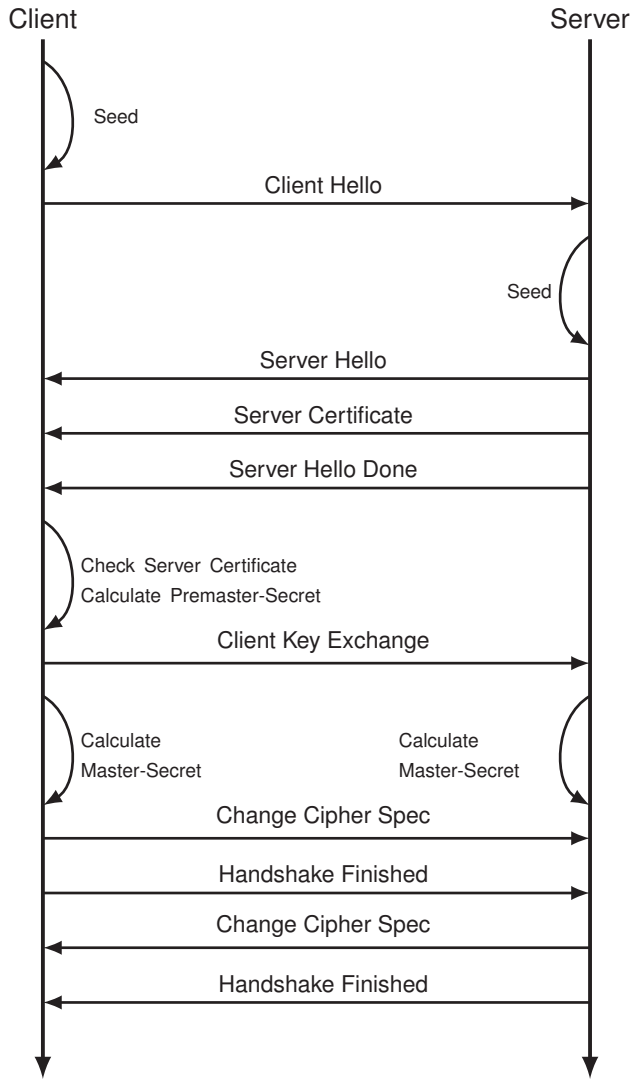
TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl
- **Server Certificate**
  Enthält den öffentlichen Schlüssel des Servers
- **Server Hello Done**
  Ende des serverseitigen Handshakes
- **Client Key Exchange**
  Mit RSA asymmetrisch verschlüsseltes Premaster-Secret, aus dem zusammen mit den Seeds ein gemeinsames Master-Secret für die symmetrische Verschlüsselung abgeleitet wird
- **Change Cipher Spec**
  Letzte unverschlüsselte Nachricht des Clients
- **Handshake Finished**
  Erste verschlüsselte Nachricht des Clients
- **Change Cipher Spec**
  Letzte unverschlüsselte Nachricht des Servers



TLS Handshake Protocol

- **Client Hello**
  Unterstützte TLS-Versionen, unterstützte Verschlüsselungsprotokolle und Kompressionsmethoden, Zufallszahl (Seed)
- **Server Hello**
  Gewählte TLS-Version, gewählte Verschlüsselungs- und Kompressionsmethoden, Zufallszahl
- **Server Certificate**
  Enthält den öffentlichen Schlüssel des Servers
- **Server Hello Done**
  Ende des serverseitigen Handshakes
- **Client Key Exchange**
  Mit RSA asymmetrisch verschlüsseltes Premaster-Secret, aus dem zusammen mit den Seeds ein gemeinsames Master-Secret für die symmetrische Verschlüsselung abgeleitet wird
- **Change Cipher Spec**
  Letzte unverschlüsselte Nachricht des Clients
- **Handshake Finished**
  Erste verschlüsselte Nachricht des Clients
- **Change Cipher Spec**
  Letzte unverschlüsselte Nachricht des Servers
- **Handshake Finished**
  Erste verschlüsselte Nachricht des Servers



Anmerkungen zum TLS-Handshake

- Der hier gezeigte TLS-Handshake ist der sog. Simple TLS Handshake. Verfügt auch der Client über ein Zertifikat und wird dies vom Server angefordert, so gibt es einen modifzierten Client-Authenticated TLS Handshake.
- Bei Wiederaufnahme einer Session enthalten Client- und Server-Hello eine Session-ID, mit der die zuvor unterbrochene Session identifiziert werden kann. Hierfür gibt es einen abgekürzten Resumed TLS Handshake.
- TLS arbeitet mit unterschiedlichen Anwendungsprotokollen zusammen. Diese werden dann häufig durch ein angehängtes „S“ gekennzeichnet, z. B. HTTPS.

Die Themen

- symmetrische Verschlüsselung,
- asymmetrische Verschlüsselung und
- Public Key Infrastructures

werden detaillierter in der Vorlesung Netzsicherheit (Wintersemester) behandelt.

TLS Record

- Das TLS Record Protocol kapselt die Nutzdaten höherer Ebenen als TLS Record.

- Die Nutzdaten sind durch die kryptografische Hashsumme integritätsgeschützt.
- Data + Hash + pad + pad length sind verschlüsselt.

HTTPS

Kommunikationen über HTTP können transparent durch TLS gesichert werden.

- Zur Unterscheidung von HTTP findet HTTPS über Port 443 statt.
- Browser nutzen hierfür das Schema https://

Typischerweise werden die Authentizität des kontaktierten Servers sichergestellt sowie die kommunizierten Daten vor dem Austausch verschlüsselt und vor Manipulationen geschützt.

- Aufgrund der Kapselung werden insbesondere auch die HTTP-Header verschlüsselt übertragen. Es ist somit lediglich ersichtlich, mit welcher IP-Adresse über welchen Port Daten ausgetauscht werden.
- Zur Authentifikation des Servers muss die Zertifikatskette überprüft werden. In den meisten Browsern ist hierfür standardmäßig eine Vielzahl an gängigen Root-Zertifikaten hinterlegt.

CVE-2014-0160 (Heartbleed)

Heartbeat: Keep-Alive-Extension für TLS (RFC 6520; Februar 2012)

- **Request**: Client sendet beliebigen Text + Längenangabe desselben
- **Response**: Server soll mit demselben Text antworten

Exploit:

- Client sendet Text + zu große Längenangabe
- Server antwortet mit Text plus den im Speicher dahinter liegenden Daten (Read overrun)
- Das Längenfeld besitzt 2 Byte $\Rightarrow$ pro Request sind maximal 2^{16} Byte = 64KB auslesbar.

Fehler:

- OpenSSL-Implementation vertraute auf Längenangabe des Clients
- Bug wurde beim Code-Review nicht entdeckt
- Längenfeld u. U. länger als notwendig
- OpenBSD Speicherschutzmechanismen waren bei OpenSSL deaktiviert

Originalcommit bei GitHub:

https://github.com/openssl/openssl/commit/4817504d069b4c5082161b02a22116ad75f822b1

Sichere E-Mail

Ohne weiteres Zutun werden E-Mails im Klartext versandt.

- Sie können während des Transportes und der (Zwischen-)Speicherung eingesehen und
- manipuliert werden

$\Rightarrow$ Zur Wahrung von Vertraulichkeit, Authentizität und Integrität sind auch hier weitere Zusatzmaßnahmen notwendig.

Einschub: E-Mail made in Germany

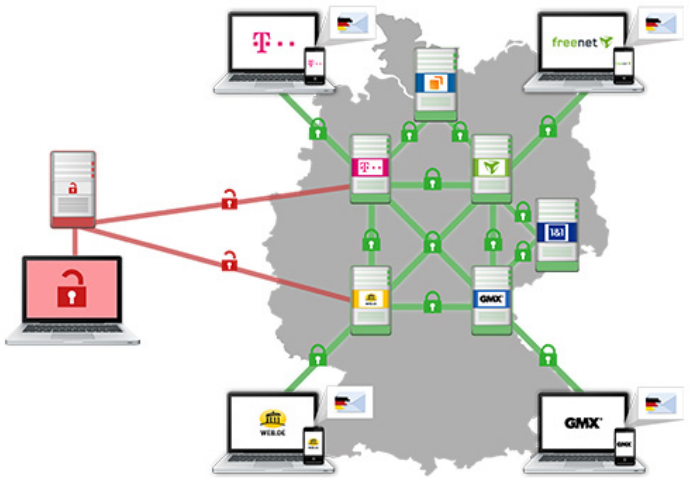
- Initiative von 1&1 und Telekom
- Freenet und Strato sind weitere Teilnehmer
- Umgesetzt in 2013

Auszüge aus der Beschreibung der Initiative:
. . . Ihre Daten werden automatisch verschlüsselt. Um das Mitlesen Ihres E-Mail-Verkehrs im Internet zu verhindern. . .

. . . Ihre Daten werden immer SSL-verschlüsselt übermittelt und so vor dem Zugriff von Dritten geschützt. . .

. . . Um eine sichere Übertragung auf allen Übertragungswegen zu gewährleisten, werden auch zwischen den E-Mail-Servern von freenet, GMX, Telekom und WEB.DE alle Daten ausschließlich verschlüsselt übertragen. . .

. . . De-Mail geht noch weiter: Auf Grundlage der De-Mail Gesetze entwickelt, gewährleistet De-Mail . . . zusätzlich die einwandfreie Identität von Sender und Empfänger. . .



Quelle: http://www.e-mail-made-in-germany.de/Verschluesselung.html

Einschub: De-Mail

De-Mail-Gesetz (2011 in Kraft getreten): De-Mail-Dienste, “die einen sicheren, vertraulichen und nachweisbaren Geschäftsverkehr für jedermann im Internet sicherstellen sollen” (§1, De-Mail-G)

Vorgehensweise:

- Kunde liefert einmalig Identitätsnachweis gegenüber Mailanbieter
- Login via Two-Factor-Auth (“sicher”) oder Username/Passwort (“unsicher”)
- Verfassen von Nachrichten via Webinterface
- Verschlüsselung und Signatur NACH Erhalt durch Mailanbieter
- Versand via SMTPS (RFC 3207 von 2002)

Probleme:

- Keine Ende-zu-Ende-Verschlüsselung
- Elektronische Signatur, jedoch nicht beim Absender, sondern durch Mailanbieter
- Serverseitige Entschlüsselung der e-Mails (Argument: Virenscan)

$\Rightarrow$ weiterhin Vertrauen in Mailanbieter notwendig $\Rightarrow$ Aussagekraft der Signatur massiv gemindert

Sichere E-Mail - Fortsetzung

Ohne weiteres Zutun werden E-Mails im Klartext versandt.

- Sie können während des Transportes und der (Zwischen-)Speicherung eingesehen und
- manipuliert werden

$\Rightarrow$ Zur Wahrung von Vertraulichkeit, Authentizität und Integrität sind auch hier weitere Zusatzmaßnahmen notwendig.

Reicht TLS?

- TLS schützt die Nachricht nur auf dem Transportweg vor Dritten
- Auf Mail-Relay-Servern kann die Nachricht weiterhin eingesehen und manipuliert werden
- Zusätzlich wird die Authentizität des Absenders nie verifiziert

Zur tatsächlichen Wahrung dieser Schutzziele müssen die Verschlüsselung und Mechanismen zur Authentifikation **bereits beim Client** durchgeführt werden.

Zwei verbreitete Technologien hierfür sind

- S/MIME
- OpenPGP

S/MIME & OpenPGP

Das S/MIME (RFC 5750 und 5751) und OpenPGP (RFC 4880) sind Standards, den den Versand signierter und verschlüsselter Nachrichten ermöglichen.

Es handelt sich jeweils um ein **hybrides Public-Key-Verfahren**:

- Nachricht wird symmetrisch mit Einmalschlüssel verschlüsselt
- Einmalschlüssel wird asymmetrisch mit Public Key des Empfängers verschlüsselt
- Beide Informationen werden zusammen verschickt

Frage: Warum wird die Nachricht nicht direkt mit dem Public Key asymmetrisch verschlüsselt?

Unterschied zwischen beiden liegt im Trust Model:

- S/MIME nutzt hierarchische X.509 Zertifizierungsinfrastruktur
- OpenPGP verwendet ein Web of Trust-Modell

Die Einbettung der verschlüsselten beziehungsweise signierten Daten in e-Mails wird MIME verwendet. Es existieren dafür entsprechende Content-Types:

- multipart/signed
- multipart/encrypted
- multipart/pkcs7-mime

Mögliche Verschlüsselungs- und Signaturalgorithmen sind im RFC spezifiziert. RSA beziehungsweise RSA mit SHA-256 werden bei S/MIME als obligatorischer Standard vorgegeben.

Auch wenn die Ziele und teilweise sogar die verwendeten Algorithmen identisch sind, unterscheidet sich dennoch das Datenformat. Daher sind S/MIME und OpenPGP zueinander nicht kompatibel.

Format einer mit S/MIME signierten Nachricht

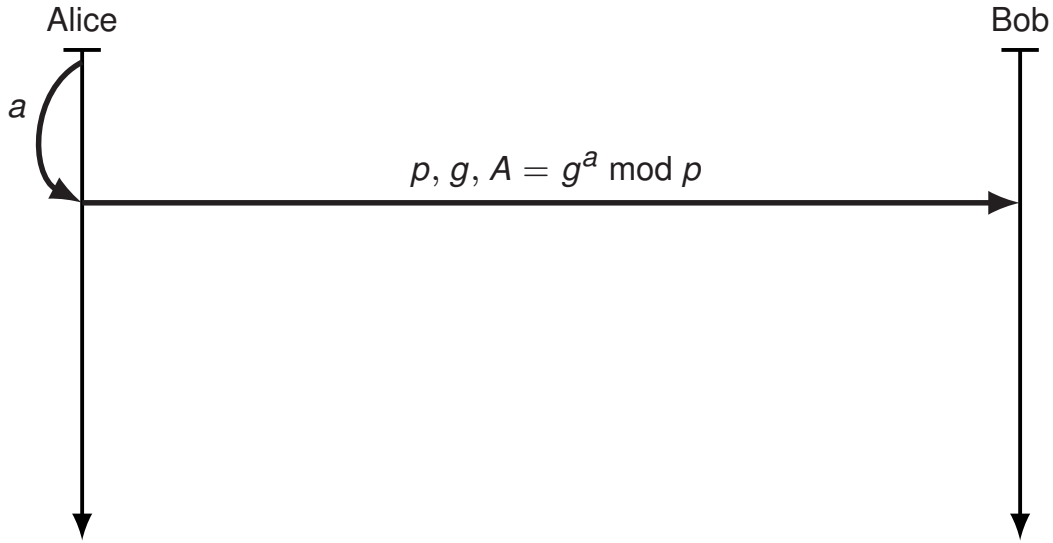
```
...
Subject: ...
Content-Type: multipart/signed; protocol="application/pkcs7-signature";
micalg=sha1; boundary="------------ms020508090406050008090703"

This is a cryptographically signed message in MIME format.

--------------ms020508090406050008090703
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

Hello, ...

--------------ms020508090406050008090703
Content-Type: application/pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAM...
--------------ms020508090406050008090703--
```

Format einer mit GPG verschlüsselten Nachricht

```
...
Subject: ...
X-Enigmail-Version: 1.6
Content-Type: multipart/encrypted;
 protocol="application/pgp-encrypted";
 boundary="sReSwidkN7wigApS7WvmWHOwTRXFRXCpI"

This is an OpenPGP/MIME encrypted message (RFC 4880 and 3156)
--sReSwidkN7wigApS7WvmWHOwTRXFRXCpI
Content-Type: application/pgp-encrypted
Content-Description: PGP/MIME version identification

Version: 1

--sReSwidkN7wigApS7WvmWHOwTRXFRXCpI
Content-Type: application/octet-stream; name="encrypted.asc"
Content-Description: OpenPGP encrypted message
Content-Disposition: inline; filename="encrypted.asc"

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.14 (GNU/Linux)
Comment: Using GnuPG with Thunderbird - http://www.enigmail.net/

hQEMAOtQHerC/rXTAQf...
-----END PGP MESSAGE-----
```

ElGamal-Verschlüsselung

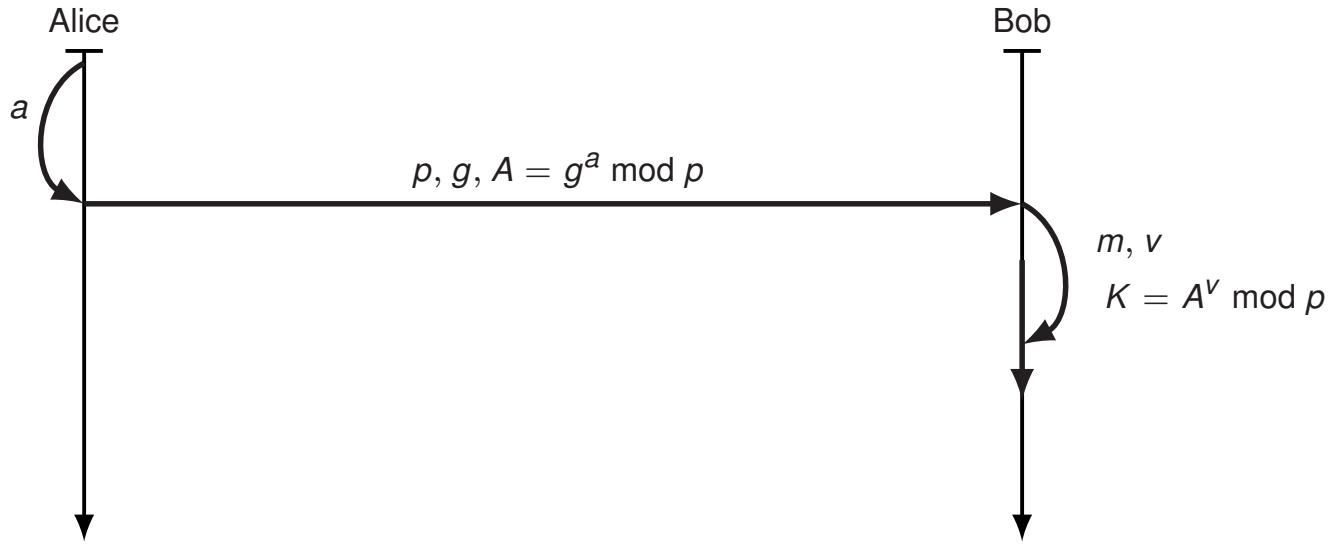
Das ElGamal-Verschlüsselungsverfahren basiert auf dem bereits vorgestellten Diffie-Hellman-Verfahren:



- Alice berechnet mithilfe der primitiven Kongruenzwurzel g zur Primzahl p ihren öffentlichen Schlüssel A und veröffentlicht ihn zusammen mit der Werten für g und p.

ElGamal-Verschlüsselung

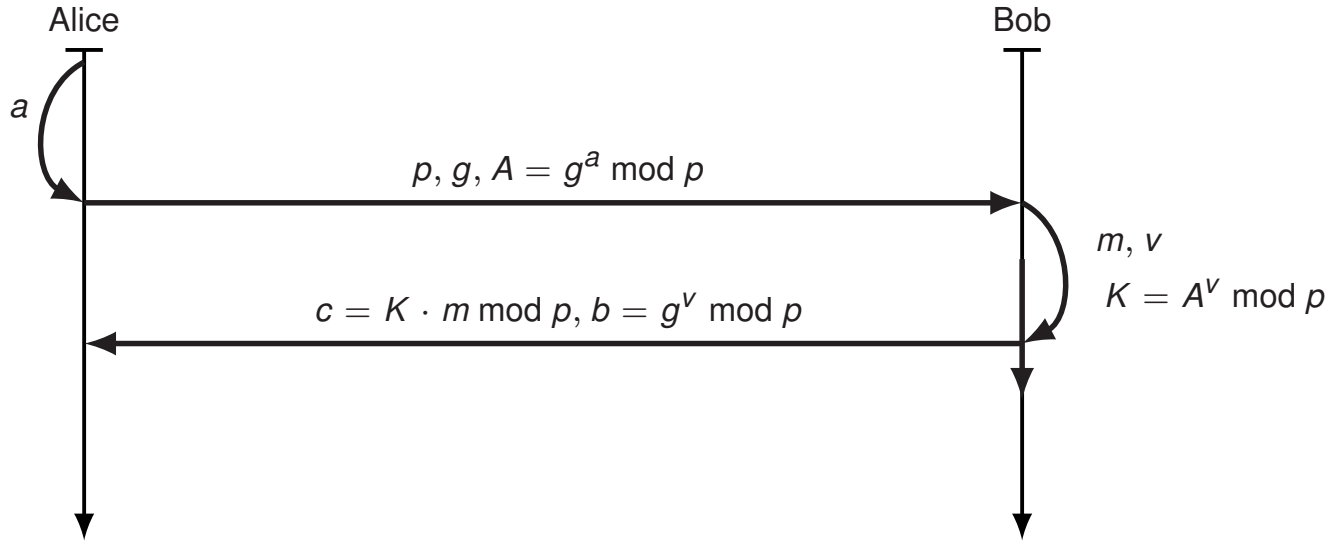
Das ElGamal-Verschlüsselungsverfahren basiert auf dem bereits vorgestellten Diffie-Hellman-Verfahren:



- Alice berechnet mithilfe der primitiven Kongruenzwurzel g zur Primzahl p ihren öffentlichen Schlüssel A und veröffentlicht ihn zusammen mit der Werten für g und p.
- Bob möchte nun Nachricht m an Alice schicken. Hierzu erzeugt er eine Zufallszahl v.

ElGamal-Verschlüsselung

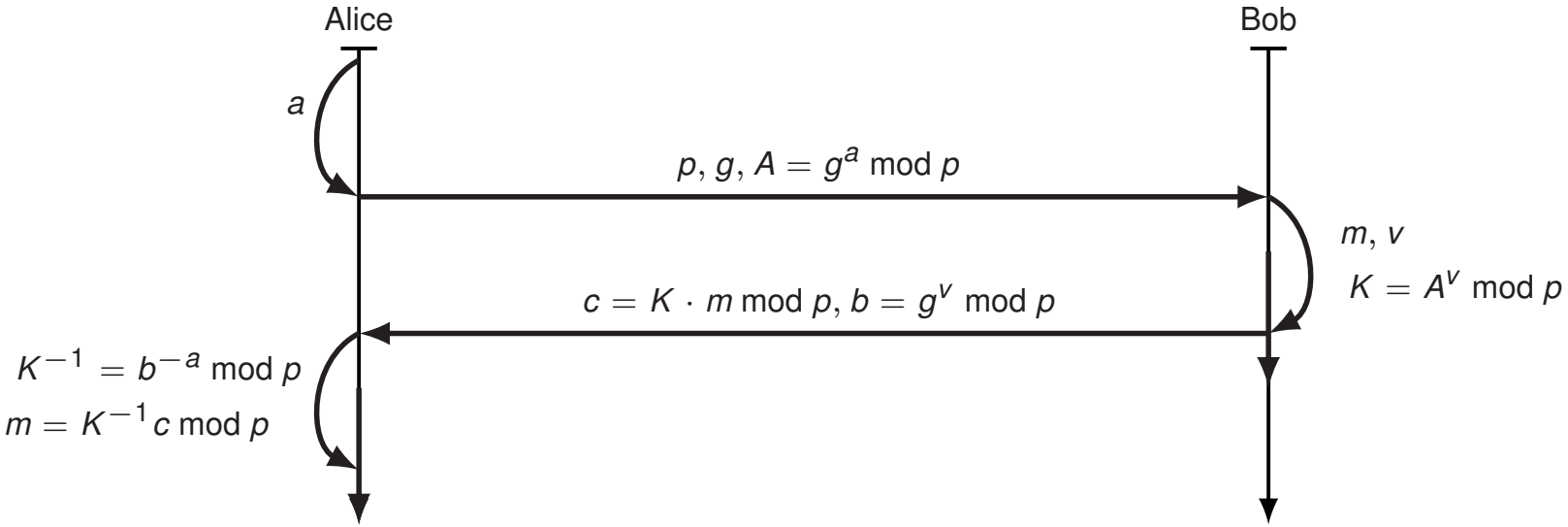
Das ElGamal-Verschlüsselungsverfahren basiert auf dem bereits vorgestellten Diffie-Hellman-Verfahren:



- Alice berechnet mithilfe der primitiven Kongruenzwurzel g zur Primzahl p ihren öffentlichen Schlüssel A und veröffentlicht ihn zusammen mit der Werten für g und p.
- Bob möchte nun Nachricht m an Alice schicken. Hierzu erzeugt er eine Zufallszahl v.
- Bob berechnet nun den Chiffretext $c = A^v \cdot m$ mod p und verschickt ihn zusammen mit $b = g^v$mod p

ElGamal-Verschlüsselung

Das ElGamal-Verschlüsselungsverfahren basiert auf dem bereits vorgestellten Diffie-Hellman-Verfahren:



- Alice berechnet mithilfe der primitiven Kongruenzwurzel g zur Primzahl p ihren öffentlichen Schlüssel A und veröffentlicht ihn zusammen mit der Werten für g und p.
- Bob möchte nun Nachricht m an Alice schicken. Hierzu erzeugt er eine Zufallszahl v.
- Bob berechnet nun den Chiffretext $c = A^v \cdot m$ mod p und verschickt ihn zusammen mit $b = g^v$mod p
- Alice kann nun die Inverse des Schlüssels berechnen und damit aus c wieder die Nachricht m gewinnen.

DNSSEC

DNSSEC ist eine Erweiterung des DNS-Protokolls um Authentizitäts- und Integritätsfunktionen. Die aktuelle Version dieser Erweiterung wurde 2005 in RFC 4033, 4034 und 4035 spezifiziert.

Schutzziele in DNSSEC für DNS-Daten

- Authentizität
- Integrität
- Nicht: Vertraulichkeit

Ein Angreifer kann DNS-Daten nicht mehr manipulieren (also z. B. die Manipulation der Namensauflösung für die Vorbereitung eines MitM-Angriffs) aber immer noch sehen, welche Namen angefragt werden.

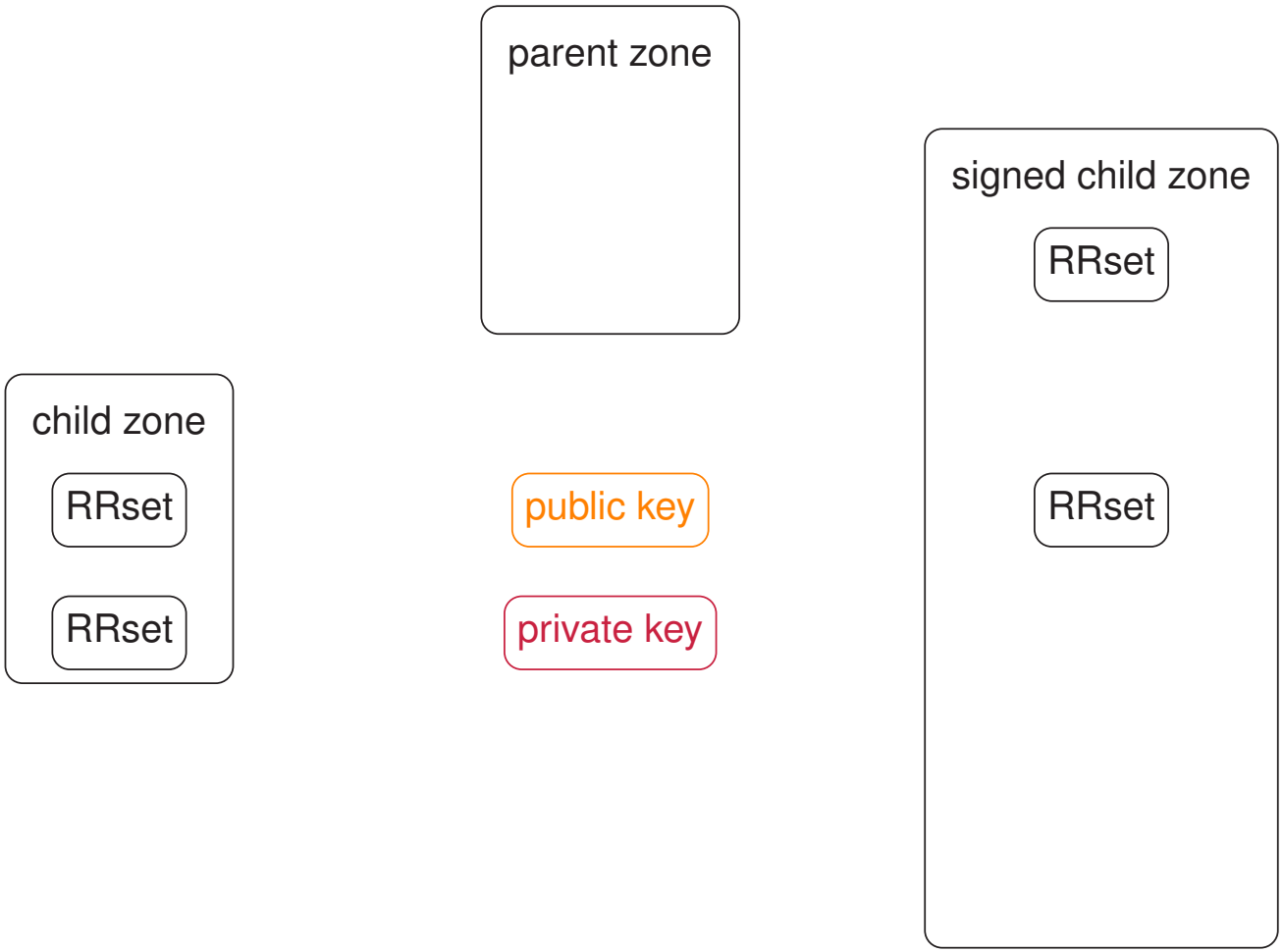
Integration

- DNSSEC erweitert das DNS-Protokoll direkt, es gibt keinen zusätzlichen Security Layer.
- Abwärtskompatibel: bestehende Systeme funktionieren weiterhin, wenn auch ohne die zusätzliche Sicherheit von DNSSEC.
- DNSSEC sichert zunächst nur die Namensauflösung, nicht jedoch die darauf aufbauenden Verbindungen.

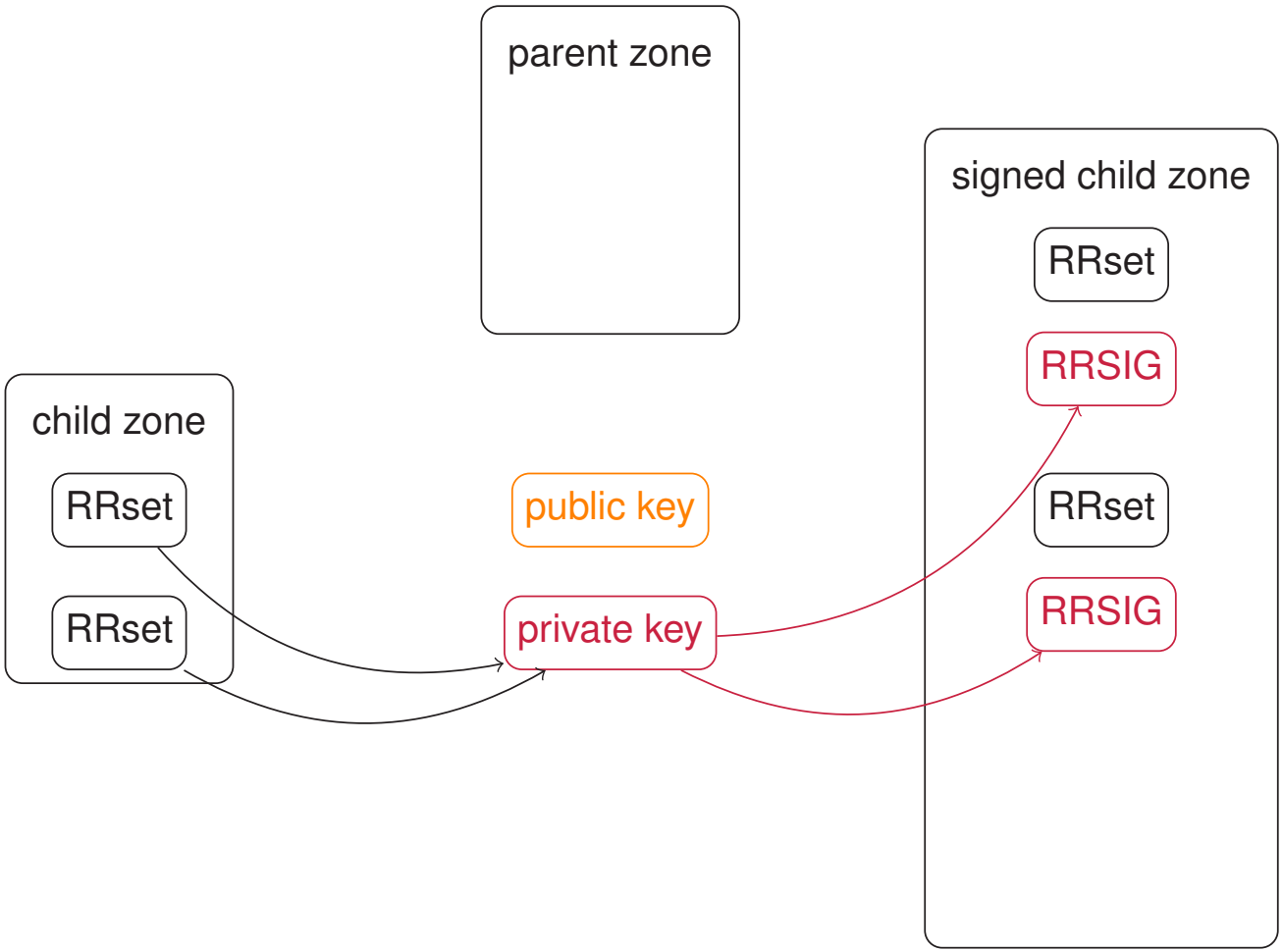
Funktionsweise DNSSEC

- **Zusätzliche DNS Resource Records für DNSSEC**
  - DNSKEY – öffentlicher Schlüssel
    Enthält den öffentlichen Schlüssel für die Signaturüberprüfung. Der private Schlüssel wird natürlich nicht veröffentlicht.
  - RRSIG – Signatur
    Enthält die Signatur über die Daten eines Resource Record Set (RRSet), z.B. IPv4-Addressen.
  - DS – Hash über ein DNSKEY
    Der Hash (zusammen mit der zugehörigen RRSIG) stellt die Signaturkette zu einem DNSKEY in einer untergeordneten Zone sicher.
  - NSEC – Nachweis der Nicht-Existenz von Namen
    Ein NSEC Record (mit entspechender RRSIG) wird verwendet um nachzuweisen, dass ein Domain Name nicht existiert. Der NSEC Record enthält den lexikographisch vorhergehenden und nachfolgenden Namen und kann somit beweisen, dass zwischen diesen beiden Namen keine weiteren Namen existieren. NSEC Records geben die Löcher zwischen den Domain Names an.
- Die DNS Zone signiert die Daten der DNS Resource Records (z.B. die IPv4-Addressen für den Record) mit dem privaten Schlüssel.
- Diese Signatur (RRSIG) wird zusammen mit den jeweiligen DNS Resource Records zurückgeliefert.
- Ein Resolver verifiziert die Signatur mit dem öffentlichen Schlüssel der Zone (DNSKEY).
- Der öffentliche Schlüssel wird von der darüberliegenden Zone durch den DS Record validiert.
- Der Hash (DS) des öffentliche Schlüssel der Root Zone ist bekannt.
- Damit eine Zone DNSSEC signiert werden kann, müssen alle darüberliegenden Zonen bis zur Root Zone signiert sein.
- Jede Zone kann nur die eigenen Records signieren.

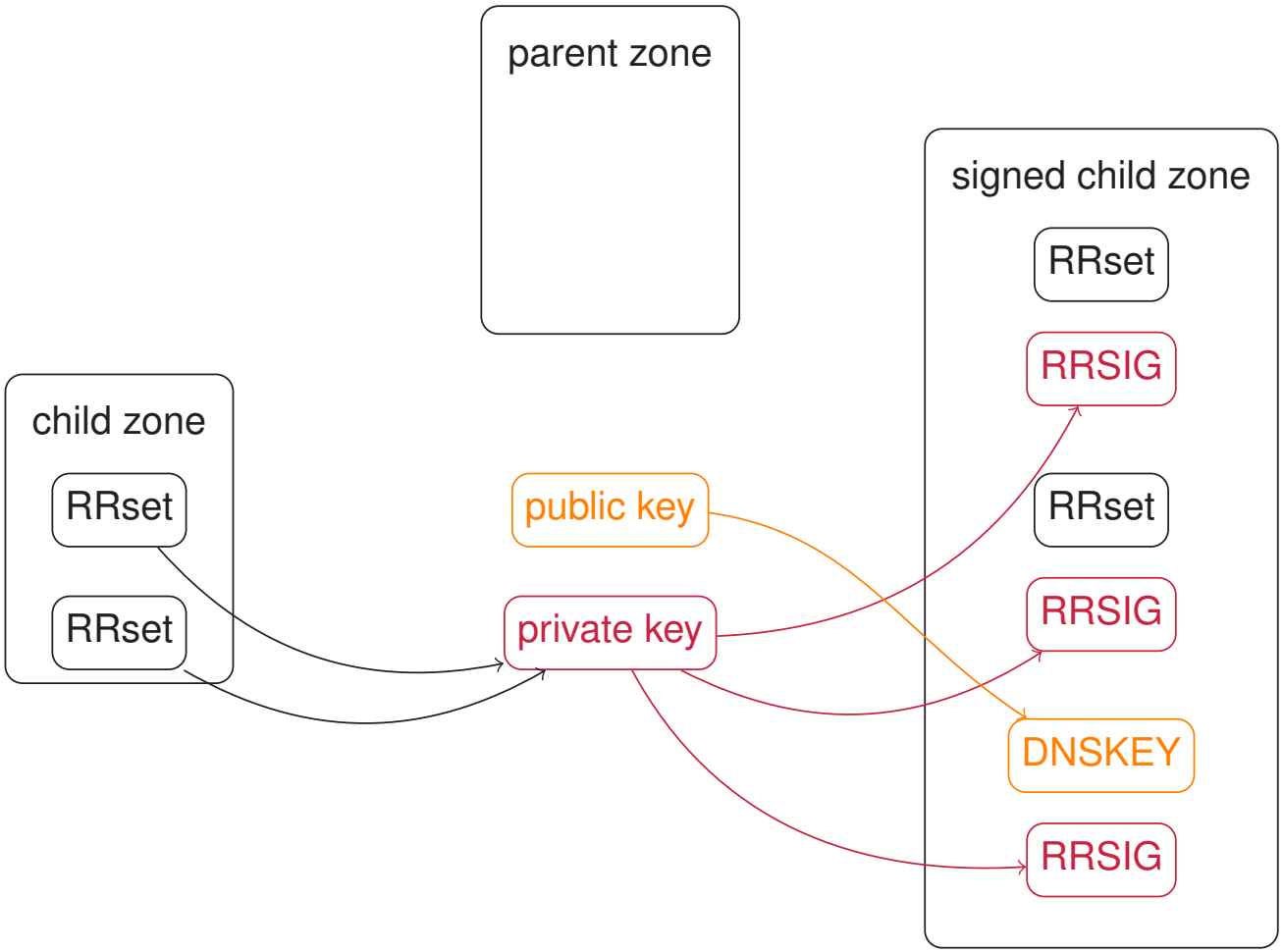
DNSSEC Zone Signing



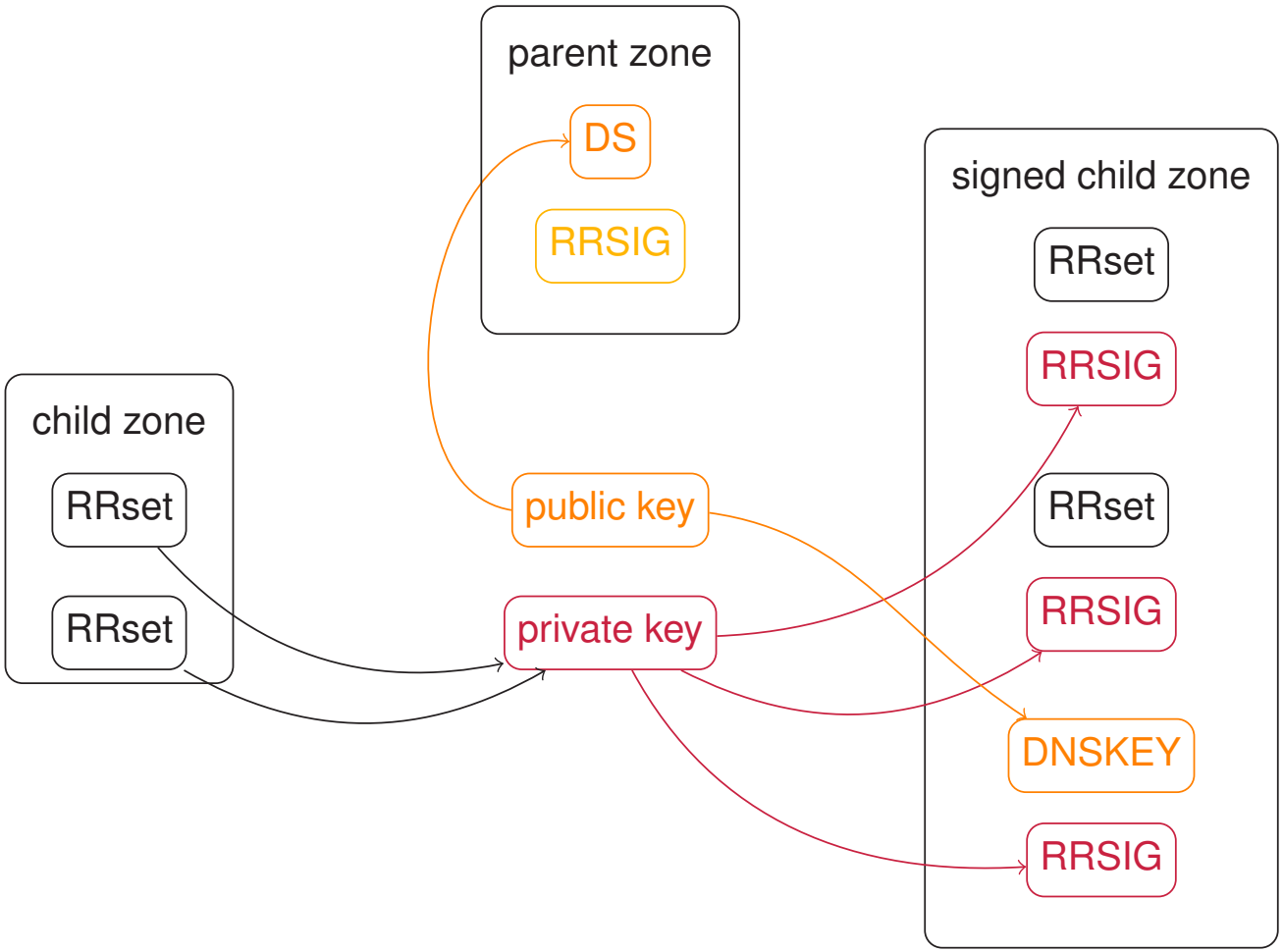
DNSSEC Zone Signing



DNSSEC Zone Signing



DNSSEC Zone Signing



Implementierung DNSSEC

- Die Root Zone ist seit Juli 2010 signiert.
- Die `de` Zone ist seit Mai 2011 signiert.
- Der DNSKEY wird oft in ein Key Signing Key (KSK) und ein Zone Signing Key (ZSK) aufgesplittet.
  - Der KSK ist länger (2048 bit RSA) und länger gültig (2–4 Jahre).
  - Der DS dieses Schlüssels wird in der übergeordneten Zone eingetragen.
  - Der ZSK ist kürzer (512 bit RSA) und nur 1–2 Monate gültig.
  - Der ZSK wird mit dem KSK signiert.
  - Die RRSIGs werden mit dem ZSK erstellt.
  - Dies wird gemacht, damit die Signaturen nicht zu lang werden und auch schneller zu verifizieren sind.
- NSEC3 verwendet für den Nachweis der Nicht-Existenz von Namen deren Hashs und kann somit verhindern, dass alle Namen in der Zone sichtbar werden.
- Der Client vertraut seinem DNS Resolver. Der (rekursive) Resolver führt die eigentliche DNSSEC-Überprüfung durch. Problem: Wie bekommt der Client die Antwort vom Resolver?
- DNSSEC sichert nicht die Übertragung zwischen zwei Endsystemen.
- DNSSEC kann aber verwendet werden um Schlüsselmaterial für eine solche Verbindung sicher zu verteilen:
  - TLSA Resource Records (RFC 6698) speichern den Hash eines X.509 Zertifikats, sodass DNSSEC als Trust Anchor für TLS verwendet werden kann.
  - SSHFP Resource Records (RFC 4255) speichern den SSH Server Fingerprint im DNS.

Übersicht

Reichen bisherige Maßnahmen aus?

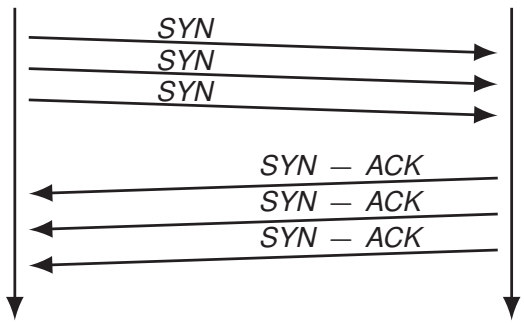
- Nein, leider nicht...
- Fehlerhafte Implementierungen, Fehlkonfigurationen oder falsche Benutzung können zu Schwachstellen führen, welche für Angriffe ausgenutzt werden können.
- Zudem zielen einige Angriffe auf bisher nicht betrachtete Schutzziele ab (z.B. Availability).

Beispiel: Denial-of-Service durch SYN-Flooding

Definition (Denial-of-Service)

Unter dem Begriff Denial-of-Service werden Angriffe zusammengefasst, welche durch gezielt erzeugte Überlastsituationen Netzkomponenten in deren Erreichbarkeit komplett oder wenigstens teilweise einschränken.

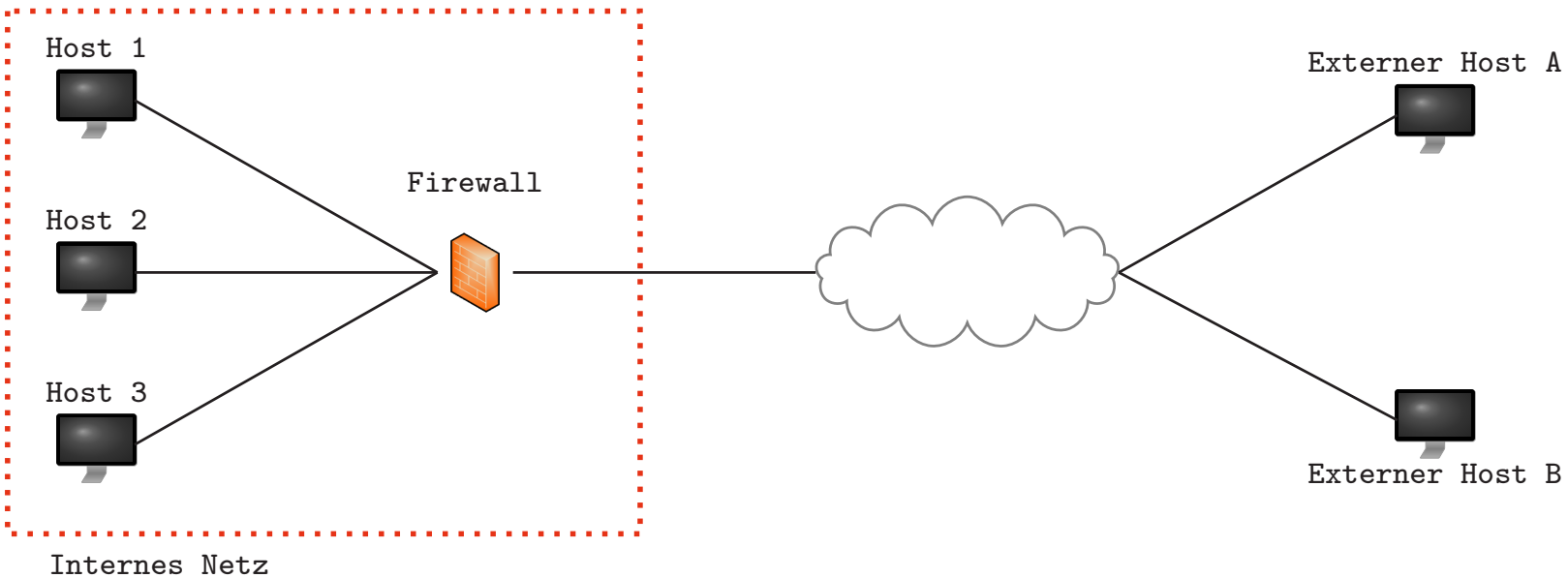
- Der Angreifer sendet TCP-SYN Paketen an ein Opfer.
- Das Opfer antwortet mit einem SYN-ACK und hält die Verbindung einseitig offen, da kein weiteres ACK eintrifft.
- Um den Zustand zu halten müssen Systemressourcen eingesetzt werden.
- Reguläre Anfragen können im schlimmsten Fall nicht mehr angenommen werden, und das Opfer ist nicht mehr erreichbar.



Ein Schutz des Netzwerks im laufenden Betrieb durch spezielle Komponenten (z.B. Firewalls) als auch das stete Überwachen des Netzes (Netzwerkmonitoring) sind daher unerlässlich.

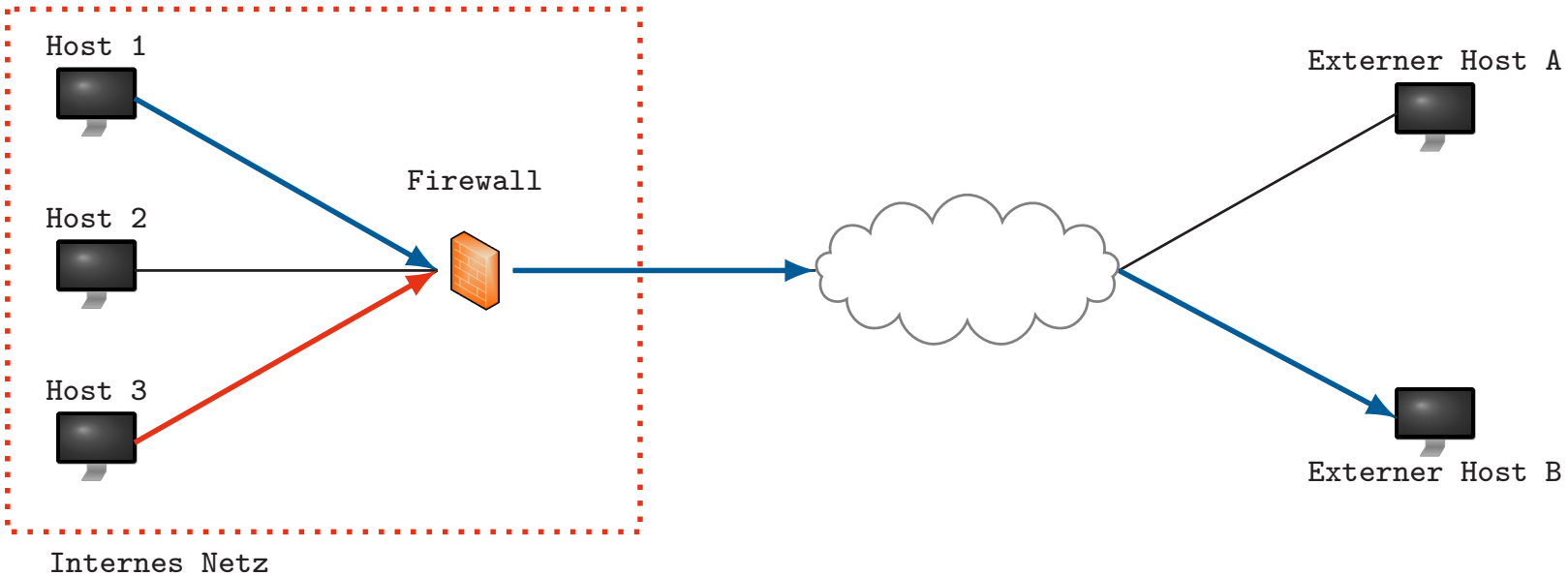
Firewall

- Firewalls schotten unterschiedliche Netze voneinander ab und kontrollieren die Übergänge in diese Netze, z.B. vom Internet in private Netze.
- Sie dienen als zentraler Kontrollpunkt, an welchem entschieden wird, welcher Verkehr von außen nach innen und umgekehrt weitergeleitet wird.
- Es kann sich dabei sowohl um Soft- als auch Hardwarekomponenten handeln.



Firewall

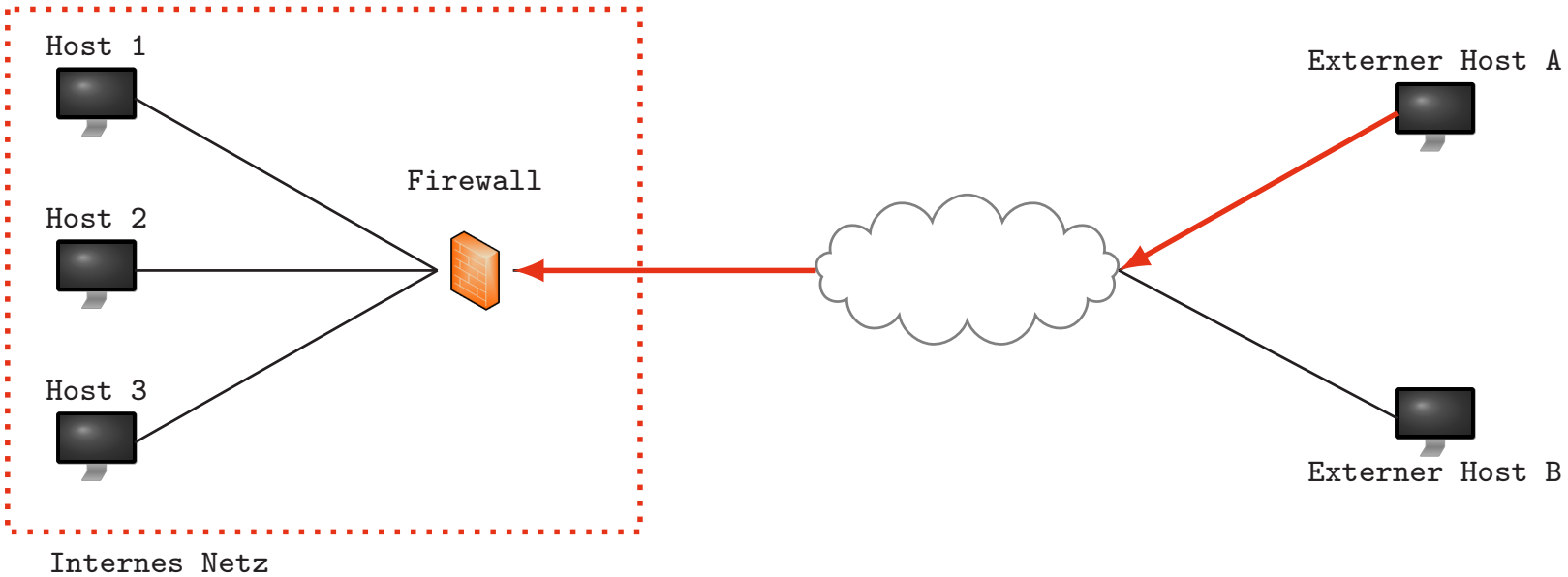
- Firewalls schotten unterschiedliche Netze voneinander ab und kontrollieren die Übergänge in diese Netze, z.B. vom Internet in private Netze.
- Sie dienen als zentraler Kontrollpunkt, an welchem entschieden wird, welcher Verkehr von außen nach innen und umgekehrt weitergeleitet wird.
- Es kann sich dabei sowohl um Soft- als auch Hardwarekomponenten handeln.



- Pakete aus dem internen Netz können geblockt oder weitergeleitet werden.

Firewall

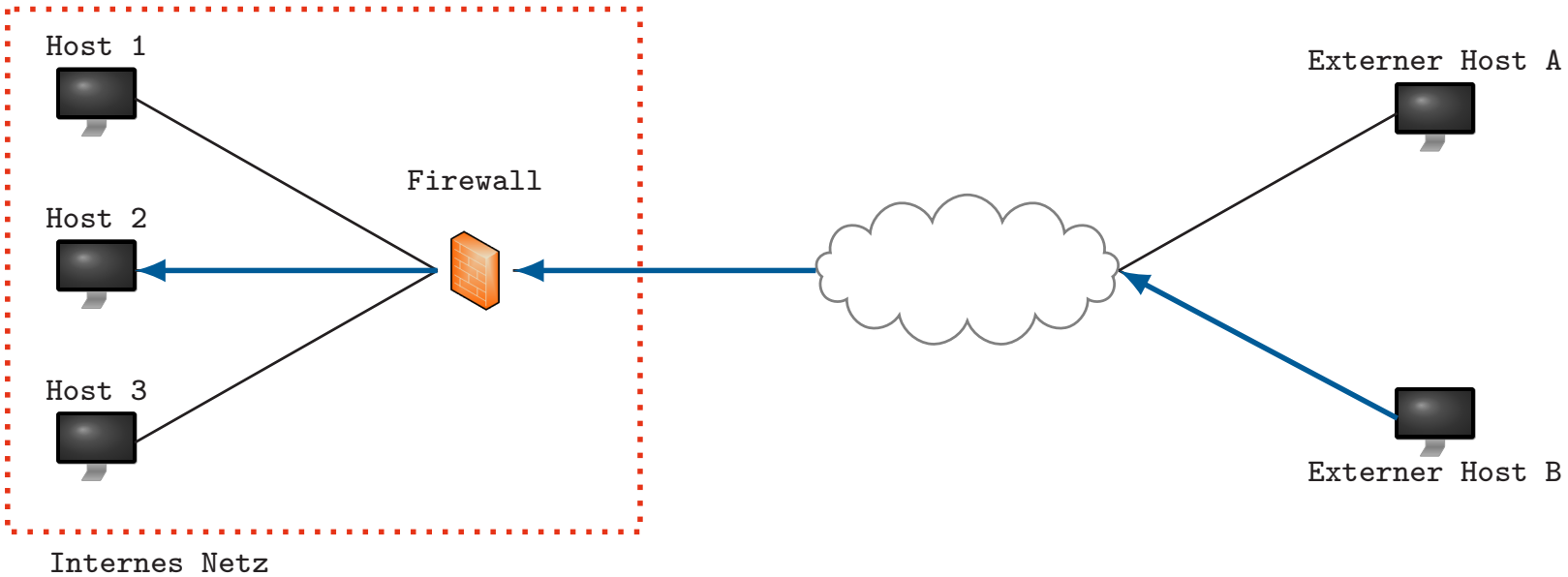
- Firewalls schotten unterschiedliche Netze voneinander ab und kontrollieren die Übergänge in diese Netze, z.B. vom Internet in private Netze.
- Sie dienen als zentraler Kontrollpunkt, an welchem entschieden wird, welcher Verkehr von außen nach innen und umgekehrt weitergeleitet wird.
- Es kann sich dabei sowohl um Soft- als auch Hardwarekomponenten handeln.



- Pakete aus dem internen Netz können geblockt oder weitergeleitet werden.
- Analoges geschieht bei Paketen aus dem externen in das interne Netz.

Firewall

- Firewalls schotten unterschiedliche Netze voneinander ab und kontrollieren die Übergänge in diese Netze, z.B. vom Internet in private Netze.
- Sie dienen als zentraler Kontrollpunkt, an welchem entschieden wird, welcher Verkehr von außen nach innen und umgekehrt weitergeleitet wird.
- Es kann sich dabei sowohl um Soft- als auch Hardwarekomponenten handeln.



- Pakete aus dem internen Netz können geblockt oder weitergeleitet werden.
- Analoges geschieht bei Paketen aus dem externen in das interne Netz.

Beispiel - ufw (Uncomplicated Firewall) für Ubuntu

```
### RULES ###
### tuple ### allow tcp 21 0.0.0.0/0 any 0.0.0.0/0 in
-A ufw-user-input -p tcp --dport 21 -j ACCEPT
### tuple ### allow any 123 0.0.0.0/0 any 0.0.0.0/0 in
-A ufw-user-input -p tcp --dport 123 -j ACCEPT
-A ufw-user-input -p udp --dport 123 -j ACCEPT
### tuple ### deny tcp 25 0.0.0.0/0 any 0.0.0.0/0 in
-A ufw-user-input -p tcp --dport 25 -j DROP
### tuple ### allow tcp any 0.0.0.0/0 25 131.87.63.9 in
-A ufw-user-input -p tcp -s 131.87.63.9 --sport 25 -j ACCEPT
### END RULES ###
```

Wie kann man das selbst mal ausprobieren?

- Bei Ubuntu in den Paketquellen (repositories), *iptables* ist nötig
- *ufw* bietet lediglich eine vereinfachte Bedienung zur Erstellung der *iptables*-Regeln
- Angelegte Regeln sind dann unter */var/lib/user.rules* oder */lib/ufw/user.rules* (siehe Ausschnitt oben)
- Eine Anleitung zum aktivieren der Firewall und zur Erstellung von Regeln ist unter: *wiki.ubuntuusers.de/ufw* zu finden

Nach welchen Kriterien kann gefiltert werden?

- Paketfilter können auf verschiedenen Schichten ansetzen
- Transportprotokoll (TCP, UDP)
- Portnummern
- IP-Adressen (sowohl Ziel- als auch Senderadressen)
- Services und Applikationen (ftp, http, smtp, ...)

Wie kann eine Firewall reagieren?

- ALLOW - Paket weiterleiten
- DENY - Paket verwerfen
- REDIRECT - Paket umleiten

Arten von Firewalls

- Zustandslos: jedes Paket wird einzeln betrachtet
- Zustandsbehaftet: Pakete können in Beziehung gesetzt werden (z.B. für TCP SYN/ACK)

Ist das Netz nun sicher?

Nach welchen Kriterien kann gefiltert werden?

- Paketfilter können auf verschiedenen Schichten ansetzen
- Transportprotokoll (TCP, UDP)
- Portnummern
- IP-Adressen (sowohl Ziel- als auch Senderadressen)
- Services und Applikationen (ftp, http, smtp, ...)

Wie kann eine Firewall reagieren?

- ALLOW - Paket weiterleiten
- DENY - Paket verwerfen
- REDIRECT - Paket umleiten

Arten von Firewalls

- Zustandslos: jedes Paket wird einzeln betrachtet
- Zustandsbehaftet: Pakete können in Beziehung gesetzt werden (z.B. für TCP SYN/ACK)

Ist das Netz nun sicher?

- Eine Firewall schützt das Netz nur vor Angriffen von außen.
- Bedrohungen können allerdings auch von innen kommen (empfehlenswert: Arte Netwars Dokumentation).

Intrusion Detection Systeme (IDS)

- Ist es nicht möglich, Angriffen vorzubeugen oder sie zu verhindern, dann ist es nötig, sie zu erkennen und im Anschluss entsprechende Maßnhamen zu ergreifen (z.B. Mitigation, Recovery, ...).

Definition (Intrusion und Intrusion Detection)

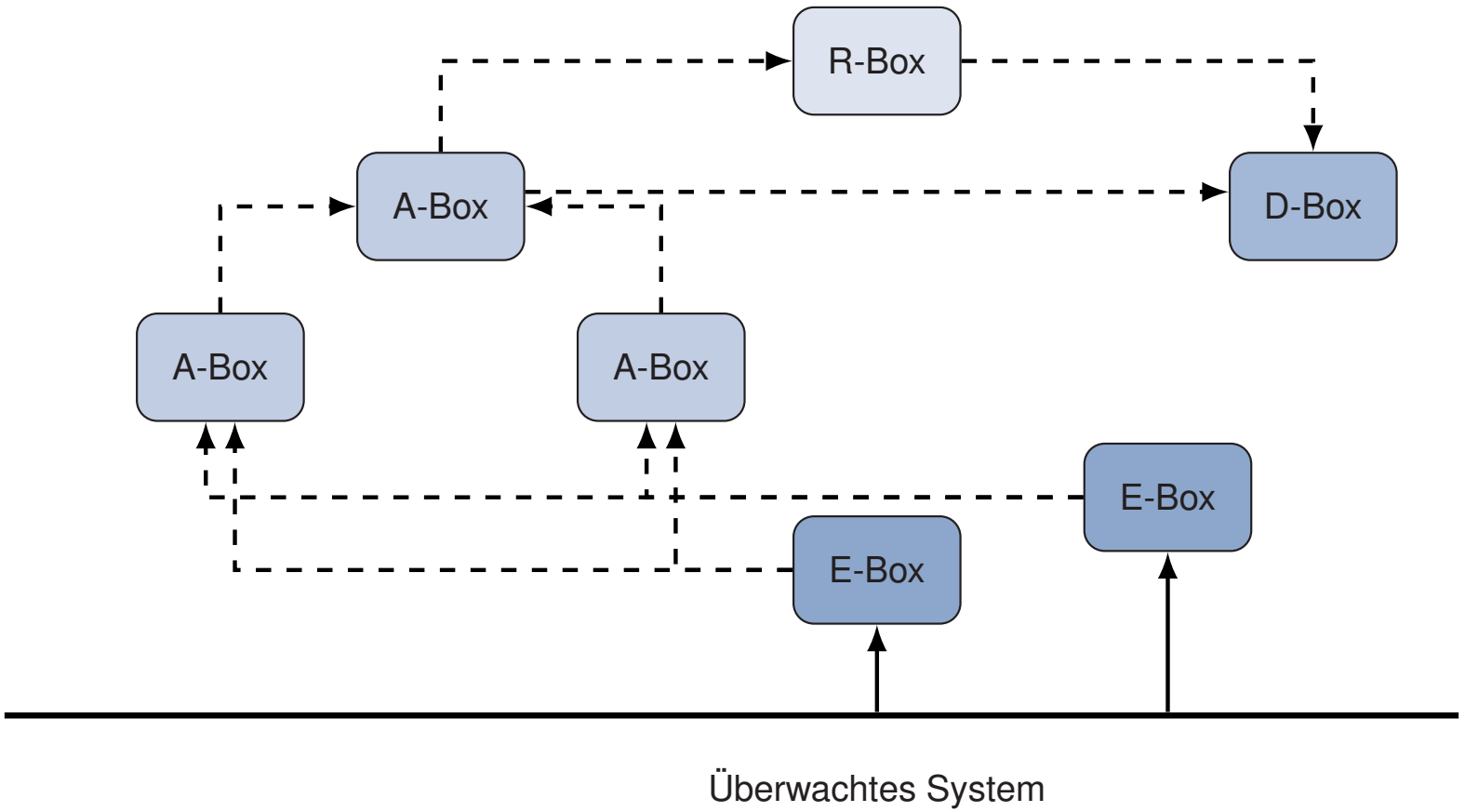
Als Intrusion bezeichnet man eine Abfolge zusammengehöriger Aktionen eines Angreifers mit der Absicht, ein Zielsystem zu kompromittieren.
Intrusion Detection bezeichnet den Prozess zur Identifizierung solcher schadhaften Aktionen.

Anforderungen an ein IDS:

- Genauigkeit, d.h. eine geringe False-Positive-Rate
- Performanz, um möglichst in Echtzeit arbeiten zu können
- Vollständigkeit, d.h. geringe False-Negative-Rate
- Fehlertolerant, d.h. resistent gegenüber Angriffen auf das IDS selbst
- Skalierbar, d.h. auch eine große Anzahl eingehender Events muss bearbeitet werden können ohne Informationen zu verlieren

CIDF Architektur

- Ein IDS kann mit Hilfe des Common Intrusion Detection Framework (CIDF) [5] umgesetzt werden.
- Es werden vier Komponenten und deren Rollen festgelegt.



[5] http://gost.isi.edu/cidf/drafts/architecture.txt

Erläuterungen zum CIDF

- **Event-Boxen (E-Box)** generieren Events, welche aus dem Monitoring gewonnen werden können.
- **Analyse-Boxen (A-Box)** analysieren die Events der E-Boxen. Das Resultat einer A-Box ist typischerweise ein Alarm, welcher von weiteren A-Boxen weiterverarbeitet werden kann. Diese arbeiten dann auf einem höheren Abstraktionslevel, beispielsweise Korrelation oder Aggregation von Alarmen.
- **Datenbank-Boxen (D-Box)** speichern Events und Alarme und dienen als Grundlage für Forensik und die persistente Haltung der Daten.
- **Response-Boxen (R-Box)** führen Aktionen gegen erkannte Angriffe aus. Eine mögliche Aktion ist die Umkonfiguration einer Firewall.

IDS Taxonomie

Erkennungsmethode

- *Anomalie-basierte Erkennung:* Das Normalverhalten des Systems wird dabei zu Grunde gelegt und mit dem aktuellen Verhalten abgeglichen. Bei Abweichungen spricht man von einer Anomalie. Dies kann allerdings sowohl ein Angriff, eine Fehlkonfiguration oder lediglich unerwartetes Verhalten sein. Letzteres zeigt die Probleme der Anomalie-Erkennung auf, da eine Vorabentscheidung über den Normalzustand meist nur schwer oder gar nicht möglich ist.
- *Signatur-basierte Erkennung:* Hierbei werden bekannte Angriffe spezifiziert und das System mit Hinblick auf diese vorgegebenen Muster untersucht. Nicht bekannte Angriffe können auf diese Weise allerdings nicht erkannt werden.

Lokation der Überwachung

- *Host-basierte Erkennung:* Das IDS befindet sich hierbei auf dem zu überwachenden Host. Dadurch wird zwar ein tieferer Einblick in das Hostsystem selbst möglich, allerdings ist keine globale Netzwerksicht möglich.
- *Netzwerk-basierte Erkennung:* Das IDS befindet sich hier im Netzwerk und kann den gesamten Verkehr überwachen. Detaileinblicke auf Vorgänge einzelner Hosts sind hier nicht möglich.

Um beide Vorteile zu verbinden werden sog. hybride IDS eingesetzt, welche dann allerdings untereinander koordiniert und die anfallenden Informationen miteinander kombinieren müssen.

IDS Beispiele

SNORT - signaturbasiertes Network IDS

- `alert tcp any any -> any 80 (content:! "GET";)` → Alle TCP-Verbindungen auf Port 80 die kein „GET “ enthalten, provozieren einen Alarm
- `alert tcp any any -> any 21 (msg:"FTP ROOT" content:"USER root"; nocase;)` → Ein Root-Login Versuch über FTP wird als Angriff erkannt
- `alert tcp any any -> any 80 (content:"foo"; content:"evil"; http_cookie;)` → Sollte in einem Cookie der String „evil “ enthalten sein, wird ein Alarm geworfen (statt cookie kann auch header verwendet werden)

BRO

- Skripte für Anomalie-Erkennung können definiert werden
- Beispiel: Erkennung eines FTP Brute-Force Angriffs (BRO Manual)
- Dies basiert auf der Überschreitung der zulässigen Schwellwerte für Fehlversuche beim Login innerhalb einer bestimmten Zeitspanne
- Es werden sowohl falsche Passwörter als auch Nnutzernamen betrachtet
- Für diese Analysen kann auch auf Logdaten der Rechner zurückgegriffen werden

Literaturhinweise und Quellenangaben I

Die Inhalte und Abbildungen dieses Kapitels wurden mit Hilfe folgender Quellen erstellt:
[6, 2, 5, 3, 1, 4]

[1] Buchmann, Johannes: Einführung in die Kryptographie.
Springer, Berlin, 2003.

[2] Eckert, Claudia: IT-Sicherheit - Konzepte, Verfahren, Protokolle (6. Aufl.).
Oldenbourg, 2009, ISBN 978-3-486-58999-3.

[3] Ertel, Wolfgang: Angewandte Kryptographie. 30 Aufgaben.
Hanser Fachbuchverlag, August 2003, ISBN 3446223045.

[4] Kruegel, Christopher, Fredrik Valeur und Giovanni Vigna: Intrusion Detection and Correlation - Challenges and Solutions, Band 14 der Reihe Advances in Information Security.
Springer, 2005, ISBN 978-0-387-23398-7.
http://dx.doi.org/10.1007/b101493.

[5] Schneier, Bruce: Angewandte Kryptographie - Protokolle, Algorithmen und Sourcecode in C: der Klassiker.
Pearson Education, 2006, ISBN 978-3-8273-7228-4.

[6] Tanenbaum, Andrew: Computer Networks.
Prentice Hall Professional Technical Reference, 4th Auflage, 2002, ISBN 0130661023.