5 Verschlüsselung und Virtual Private Networks (VPN)

Ziel dieses Kapitels ist es, die praktische Anwendung und Implementierung von Virtual Private Networks (VPN) zu beschreiben. Im ersten Teil werden einige theoretische Grundlagen für einer sicheren Datenübertragung erläutert, für einen tieferen Einblick in das Thema Kryptographie muß jedoch auf weitere Fachliteratur wie [Schn 96] verwiesen werden.
Im zweiten Teil wird die Konfiguration eines VPNs mit FreeS/WAN unter Linux erklärt.

5.1 Begriffsdefinitionen und Ziele

Bevor wir uns näher mit der gesicherten Übertragung von Daten über ein Netzwerk beschäftigen ist es nötig, einige Begriffe zu definieren und festzulegen, was mit einer gesicherten Übertragung eigentlich genau erreicht werden soll.
Oft wird in Verbindung mit einem sicheren Datenverkehr einfach von "Verschlüsselung" (und was auch immer in den Einzelfällen darunter verstanden wird) gesprochen. Die Verschlüsselung ist allerdings nur ein Teilbereich einer wirklich sicheren Datenübertragung über ein nicht vertrauenswürdiges Medium.

Das Ziel der sicheren Datenübertragung ist es, eine in Klartext vorhandene Nachricht auf elektronischem Wege unter Einhaltung folgender Vorgaben vom Absender zum Empfänger zu transportieren:

1. **Wahrung der Vertraulichkeit** (Privacy):
   Dazu muß sichergestellt werden, daß die Klartext-Nachricht nur vom gewünschten Empfänger gelesen werden kann. Sollte ein Dritter die Nachricht abfangen oder mithören darf es ihm nicht möglich sein, die Nachricht im Klartext aus den übertragenen verschlüsselten Daten zu extrahieren. In besonderen Fällen darf schon die Tatsache, daß eine Nachricht übertragen wurde, nicht erkennbar sein.

2. **Wahrung der Datenintegrität** (Integrity):
   Wahrung der Datenintegrität bedeutet, daß die Nachricht auf ihrem Weg vom Absender zum Empfänger nicht zufällig oder beabsichtigt verändert werden kann. Dies beinhaltet sowohl Übertragungsfehler als auch Veränderungen durch Angriffe. Der Empfänger muß die Möglichkeit haben, eventuelle Manipulationen an der Nachricht zu erkennen.

3. **Garantie der Echtheit, Authentizität** (Authenticity):
   Der Empfänger muß den Absender der Nachricht eindeutig identifizieren können, um zu verhindern, daß ihm von einer fremden Person unter Vortäuschung einer falschen Identität eine Nachricht untergeschoben wird.

Zur Realisierung dieser Anforderungen werden meist folgende drei Verfahren eingesetzt:

1. **Verschlüsselung** (Encryption):
   Bei der Verschlüsselung wird eine in Klartext vorliegende Nachricht (Plaintext oder Cleartext) mit einem Verschlüsselungsalgorithmus (Chiffre, Cipher) unter Verwendung einer zusätzlichen Information (Schlüssel für die Verschlüsselung, Encryption Key) in eine Zeichenkombination (Ciphertext) überführt, aus welcher die ursprüngliche Klartext-Nachricht nur mithilfe einer weiteren Information (Schlüssel für die Entschlüsselung, Decryption Key) rekonstruierbar ist. Sind beide Schlüssel gleich spricht man von Symmetrischer, ansonsten von Asymmetrischer Verschlüsselung.

2. **Prüfsummenbildung** (Hashing):
   Mittels einer mathematischen Funktion wird aus der Nachricht eine Zeichenkette konstanter Länge (digitaler Fingerabdruck, Hash Value) generiert, die einmalig für diese Nachricht ist. Diese Prüfsumme wird zusammen mit der Nachricht übertragen. Der Empfänger kann durch Neuberechnung der Prüfsumme und Vergleich des Ergebnisses mit dem übermittelten Wert die Integrität der Nachricht verifizieren.

3. **Digitale Signaturen** (Digital Signature)
   Eine Digitale Signatur stellt das Analogon der menschlichen Unterschrift in der digitalen Welt dar. Sie dient also im Wesentlichen dazu, den Absender einer Nachricht eindeutig zu identifizieren.
   Die Digitale Signatur beinhaltet aber auch Mechanismen zur Garantie der Verbindlichkeit (Die Unterschrift kann nicht mehr abgestritten werden) und in gewissen Anwendungsfällen auch die Sicherstellung der einmaligen Verwendung von elektronischen Unterschriften.

Natürlich ist eine 100%ige Sicherheit bei keinem dieser Punkte erreichbar. Steht einem Angreifen genügend Rechenleistung und Zeit zur Verfügung lassen sich theoretisch alle praktisch einsetzbaren Algorithmen umgehen. Es läßt sich also nicht eindeutig eine Methode finden, die in allen Anwendungsfällen die beste ist. Vielmehr ist abzuwägen, welchen Wert die Informationen besitzen, welchen Aufwand man in die Sicherung der Informationen stecken möchte und welche Ressourcen bei potentiellen Angreifern vorhanden sein können. In den folgenden Abschnitten werden die Methoden für Verschlüsselung und Prüfsummenbildung sowie Digitale Signaturen und die dafür benötigten Zertifikate beschrieben.

5.2 Verschlüsselungsalgorithmen

Eine Möglichkeit der Verschlüsselung besteht darin, den Verschlüsselungsalgorithmus selbst geheim zu halten ("Security by Obscurity"). Nicht vollständig offengelegte Verschlüsselungsalgorithmen sollten jedoch immer mit Skepsis betrachtet werden, da die Geheimhaltung oft ein Indiz dafür ist, daß der Entwickler selbst nicht sehr überzeugt von der Sicherheit

seines Verfahrens ist. Ein geheimgehaltener Algorithmus in Form eines Computerprogrammes kann zudem immer durch Disassemblierung rekonstruiert werden. Und allzu oft haben sich zuvor geheimgehaltene Algorithmen nach Veröffentlichung als erschreckend schwach erwiesen.
Alle ernstzunehmenden Verschlüsselungsmethoden basieren auf offengelegten mathematischen Algorithmen, welche allein aufgrund ihrer mathematischen Eigenschaften als sicher gelten oder deren Sicherheit zumindest bisher noch nicht widerlegt werden konnte.

Es gibt im Wesentlichen zwei Arten von Verschlüsselungsverfahren (Cipher): Symmetrische und Asymmetrische Algorithmen. Hybride Verfahren nutzen zur Verbesserung der Relation von Sicherheit und Aufwand eine Kombination aus symmetrischer und asymmetrischer Verschlüsselung.

Obwohl nicht Gegenstand dieses Praktikums sei hier noch die **Steganographie** als weiteres Thema der Informationssicherung erwähnt. Aufgabe der Steganographie ist es, die blosse Existenz einer Botschaft zu verschleiern, inden diese in anderen Daten verborgen wird. Die Informationen können dadurch auch vor Datenstromanalysen (Analyse der Anzahl und Dauer von Verbindungen, Paketgrößen, übertragene Datenmengen usw.) geschützt werden. Zum Beispiel könnte eine (ggf. vorher komprimierte, verschlüsselte und signierte) Nachricht in den niederwertigen Bits (LSB, Least significant Bit) einer Grafik versteckt sein oder die Botschaft durch Variation statistischer Eigenschaften von auf den ersten Blick harmlosen Informationen (Subliminal Channels) übertragen werden.
Andere Anwendungsgebiete solcher Verfahren sind der Kopierschutz und digitale Wasserzeichen (Watermarking).

5.3 Symmetrische Verschlüsselung

Bei der Symmetrischen Verschlüsselung wird sowohl für die Verschlüsselung als auch für die Entschlüsselung der Daten derselbe Schlüssel (Secret Key) verwendet. Wird für jede einzelne Kommunikationsbeziehung ein eigener Secret Key verwendet, beinhaltet die Verschlüsselung somit gleichzeitig auch die Authentisierung.
Das Verfahren bringt mit sich, daß jeder, der diesen einen Schlüssel kennt, in der Lage ist, jede damit verschlüsselte Nachricht zu entschlüsseln oder Nachrichten im Namen des Besitzers des Schlüssels an die Kommunikationspartner zu versenden. Für die Sicherheit der Daten ist es also unerläßlich, dafür zu sorgen, daß der Schlüssel nicht in die falschen Hände gerät. Dies bringt mehrere Probleme mit sich:

- Für die Übertragung des Schlüssels muß ein gesicherter Transportweg vorhanden sein, z.B. die persönliche Übergabe oder der Postweg, notfalls auch das Telefon. Eine solche sichere Informationsübertragung ist jedoch oft nicht möglich, da die Anwendung von Verschlüsselung gerade in Fällen benötigt wird, wo es diesen sicheren Datenkanal eben nicht gibt.

- Für jede einzelne Kommunikationsbeziehung wird ein eigener geheimer Schlüssel benötigt. Dies bedeutet, daß für jeden Kommunikationspartner ein Schlüssel zu verwalten ist, der vor fremden Zugriff zu schützen ist. Die Anzahl der Schlüssel wird dabei schnell unüberschaubar. Sie beträgt $n(n-1)/2$ bei n Kommunikationspartner, die paarweise gesichert kommunizieren möchten.

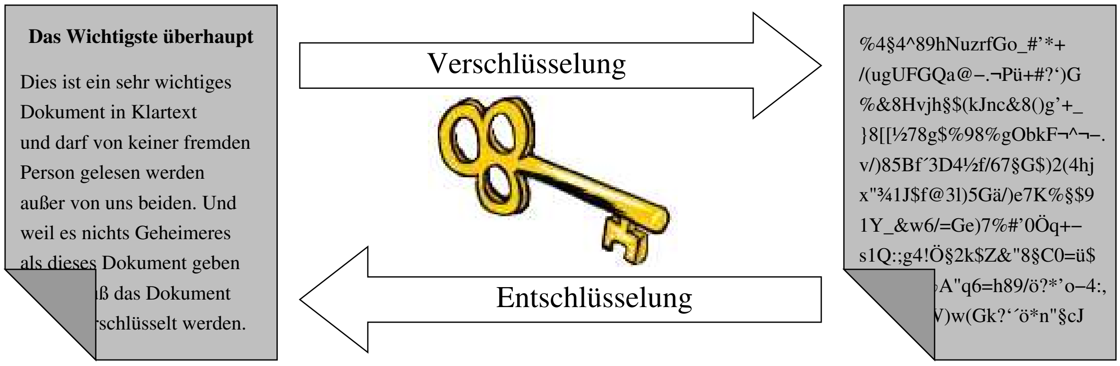

Abbildung 41: Symmetrische Verschlüsselung mit einem geheimen Schlüssel

Symmetrische Algorithmen werden in Block- (Block Ciphers) und Strom-Chiffren (Stream Ciphers) eingeteilt.

Block-Algorithmen führen eine bestimmte mathematische Operation auf einen Datenblock fester Länge l (typischerweise 64 Bit) aus. Zu beachten ist, daß die Blocklänge nichts über die Schlüssellänge und damit über die Sicherheit des Algorithmus aussagt und nicht damit verwechselt werden darf.
Im einfachen **ECB-Modus** (Electronic Code Book Mode) haben Block-Chiffren die Eigenschaft, daß bei wiederholter Verschlüsselung desselben Datenblocks der Länge l mit demselben Schlüssel immer ein gleicher verschlüsselter Datenblock derselben Länge l entsteht. Diese Information könnte von Angreifern genutzt werden, um Rückschlüsse auf den Inhalt der Nachricht zu ziehen oder einzelne Blöcke der verschlüsselten Nachricht zu ersetzen (**Block Replay**). Aus diesem Grund gibt es weitere Moden, die diese Probleme durch Verknüpfung des aktuell zu verschlüsselnden Blocks mit vorangegangenen Blöcken umgehen. Der erste Block der Nachricht wird dabei mit einem individuellen Initialisierungsvektor verknüpft, der am Beginn die Rolle des vorangegangenen Blocks übernimmt bzw. den Algorithmus initialisiert und damit verhindert, daß gleiche Nachrichten zu gleichen Chiffren führen.
Bei einigen Moden führt diese Rückkopplung dazu, daß nur ein Bitfehler in der verschlüsselten Nachricht den Inhalt der entschlüsselten Nachricht nach der Fehlerstelle unbrauchbar macht. Aus diesem Grund können weitere Synchronisationsinformationen verwendet werden, um solche Fehler korrigierbar zu machen.
Drei dieser Rückkopplungs-Moden sind **CBC** (Cipher Block Chaining), **CFB** (Cipher Feedback) und **OFB** (Output Feedback).

Strom-Algorithmen interpretieren die zu verschlüsselnden Daten als eine endliche Zeichenfolge und können auch einzelne Bits verschlüsseln. Das Problem des gleichen Verschlüsselungsergebnisses bei gleichen Klartextblöcken tritt bei ihnen somit nicht auf.

5.3.1 Data Encryption Standard (DES)

Der symmetrische DES-Verschlüsselungsalgorithmus wurde 1974 von der US-Regierung veröffentlicht und einige Jahre später als ANSI-Standard normiert.
DES ist ein Block-Algorithmus, der aus 64 Bit Klartext mittels einer effektiven Schlüssellänge von 56 Bit (64 Bit gesamt, davon acht Bit Paritätsprüfung) wiederum 64 Bit verschlüsselte Daten erzeugt. DES ist vor allem im Finanzsektor weit verbreitet (z.B. EC-Karten), bietet jedoch aufgrund der geringen Schlüssellänge (2^{56} mögliche Schlüssel) und einiger Angriffspunkte für mathematische Analysemethoden keinen ausreichenden Schutz mehr.

5.3.2 Triple-DES (3DES)

Durch spezielle Verknüpfung von drei DES-Durchläufen mit zwei verschiedenen Schlüsseln wird bei 3DES die effektive Schlüssellänge gegenüber DES auf 112 Bit verdoppelt. Dadurch soll die größte Schwachstelle von DES, der zu kurze Schlüssel, geschlossen werden. Der Aufwand für eine Brute Force-Attacke wird dadurch erheblich erhöht.

5.3.3 International Data Encryption Algorithm (IDEA)

IDEA wurde 1990 vom schweizer Unternehmen Ascom veröffentlicht und ist ein Block-Algorithmus mit 128 Bit Schlüssellänge und 64 Bit langen Datenblöcken. Der Algorithmus bietet aufgrund des langen Schlüssels einen sehr guten Schutz vor Brute Force-Angriffen und hat bisher auch keine bekannten Schwachstellen. Er ist patentiert, für die nicht-kommerzielle Anwendung jedoch kostenlos nutzbar.

5.3.4 Advanced Encryption Standard (AES)

AES entstand aus einer 1997 vom National Institute of Standards and Technology (NIST) veröffentlichten Ausschreibung für einen DES-Nachfolger. Von den 5 letzten Kandidaten (MARS, RC6, RIJNDAEL, Serpent, Twofish) wurde im Jahr 2000 der von zwei belgischen Kryptographen entwickelte RIJNDAEL-Algorithmus als AES-Algorithmus ausgewählt.
AES bietet drei verschiedene Schlüssellängen: 128, 192 and 256 Bit. Die großen Schlüssellängen dürften die Verschlüsselung mindestens für die nächsten 20 Jahre sichern, theoretische Schwachstellen wurden bisher noch keine bekannt. Die Block-Länge von AES beträgt 128 Bit.

5.4 Asymmetrische Verschlüsselung

Die neueren Asymmetrischen Algorithmen verwenden nicht mehr nur einen Schlüssel für die Ver- und Entschlüsselung sondern ein Schlüsselpaar. Daten, welche mit einem dieser beiden Schlüssel verschlüsselt wurden können nur vom dazugehörigen zweiten Schlüssel entschlüsselt werden.
Die Asymmetrische Verschlüsselung wird hauptsächlich so angewandt, daß einer dieser Schlüssel beim Anwender (bzw. Rechner, Gateway usw.) verbleibt (privater Schlüssel, Private Key) und geheimgehalten wird, der zweite wird frei an alle Kommunikationspartner verteilt (öffentlicher Schlüssel, Public Key).

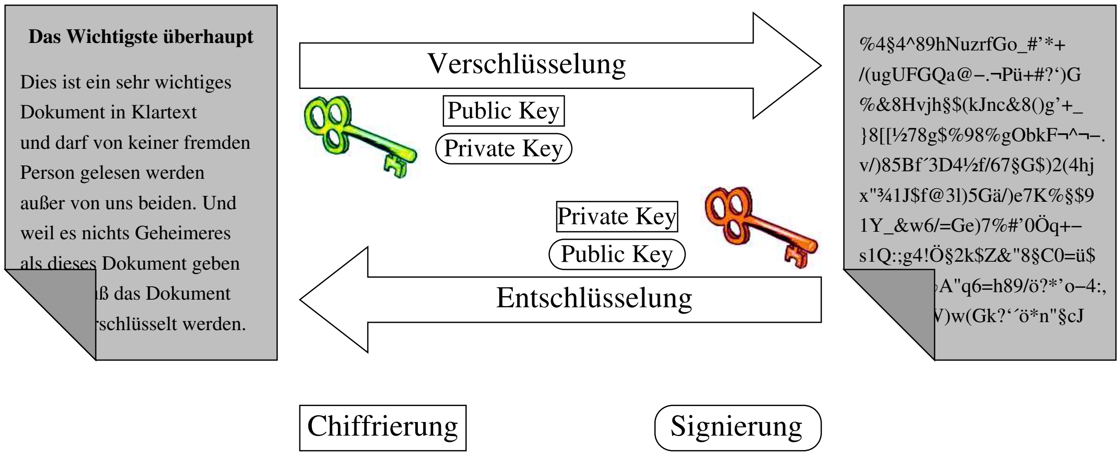

Abbildung 42: Asymmetrische Verschlüsselung mit zwei Schlüsseln

Bei Asymmetrischer Verschlüsselung stehen die beiden Schlüssel durch eine Einwegfunktion in einem komplexen mathematischen Zusammenhang. Einwegfunktionen sind zwar ohne besonderen Aufwand berechenbar, die Umkehrfunktion ist jedoch praktisch unlösbar. Z.B. ist die Multiplikation zweier großer Primzahlen eine relativ einfache, die Faktorisierung des Ergebnisses jedoch eine ungleich aufwändigere Operation.

Diese so genannten Public Key Algorithmen bringen folgende Vorteile und Möglichkeiten mit sich:

- Jeder Kommunikationspartner muß nur mehr einen einzigen Schlüssel, seinen Private Key, geheim halten und ihn auch nie an seine Kommunikationspartner übermitteln.
- An die Kommunikationspartner wird nur der Public Key übermittelt, der nicht geheimgehalten werden muß.
- Die Anzahl der Schlüssel steigt nur linear mit der Anzahl der Kommunikationsteilnehmer an.

- Jeder Besitzer des öffentlichen Schlüssels seines Partners kann diesem eine verschlüsselte Nachricht schicken, die nur von ihm mit dem passenden privaten Schlüssel entschlüsselt werden kann.
- Eine vom privaten Schlüssel verschlüsselte Nachricht ist zwar nicht vor Fremdzugriff geschützt, da sie von jedem Besitzer des öffentlichen Schlüssels entschlüsselt werden kann. Es ist aber sichergestellt, daß die Nachricht von einem bestimmten Absender stammt, da nur dieser den passenden privaten Schlüssel besitzt.

Asymmetrische Algorithmen können also sowohl zur Chiffrierung (Verschlüsselung mit dem Public Key, Entschlüsselung mit dem Private Key) als auch zur Signierung (Verschlüsselung mit dem Private Key, Entschlüsselung mit dem Public Key) eingesetzt werden, siehe Abbildung 42.

5.4.1 Diffie-Hellmann

Diffie und Hellmann veröffentlichten im Jahre 1976 den ersten Asymmetrischen Algorithmus. Dieser eigent sich nur zur Vereinbarung eines Sitzungsschlüssels für eine anschließende Symmetrische Verschlüsselung eignet, nicht jedoch für die Verschlüsselung beliebiger Nachrichten. Über den Austausch von zwei öffentlichen Schlüsseln können die beteiligten Geräte daraus den Sitzungsschlüssel berechnen, ohne diesen Schlüssel selbst übers Netz schicken zu müssen. Die Sicherheit des Verfahren basiert auf dem diskreten Logarithmus-Problem. Die genaue Funktionsweise von Diffie-Hellmann ist unter [Inc. 02b] beschrieben, eine Variante davon wurde in RFC 2631 [Resc 99] standardisiert. Diffie-Hellmann war in den USA patentiert, das Patent ist jedoch 1997 abgelaufen.
Die sogenannten Diffie-Hellmann-Gruppen bezeichnen die Länge eines für den Schlüsselaustausch verwendeten Parameters. Als sicher anzusehen sind die DH-Gruppen 2 (1024 Bit) und 5 (1536 Bit). Der Algorithmus wird in vielen Produkten für den Austausch eines symmetrischen Schlüssels verwendet.

5.4.2 RSA

Der nach seinen Entwicklern Rivest, Shamir und Adleman benannte RSA-Algorithmus zählt zu den am weitesten verbreiteten Asymmetrischen Algorithmen. Seine Sicherheit beruht auf der Schwierigkeit der Primzahlenzerlegung großer Zahlen.

Hier die prinzipielle Funktionsweise des Algorithmus nach [Inc. 02a]:

Ausgangspunkt für die Generierung des Schlüsselpaares sind zwei große, in ihrer Länge deutlich unterschiedliche Primzahlen[30] p und q sowie deren Produkt $n = pq$ (Modulus).

[30] Bei ähnlich langen Zahlen p und q wären diese relativ einfach aus n durch Ausprobieren von Zahlen um $n/2$ zu bestimmen.

Benötigt werden des Weiteren zwei Zahlen e und d (Öffentlicher und Privater Exponent), die folgenden Bedingungen entsprechen:

- e ist kleiner n und hat außer 1 keine gemeinsamen Teiler mit $(p-1)(q-1)$.
- $(ed-1)$ ist durch $(p-1)(q-1)$ teilbar.

Die Zahlen p und q müssen geheim gehalten oder nach Abschluß der Berechnungen vernichtet werden.

Eine Klartext-Nachricht m wird mit dem Public Key bestehend aus den beiden Zahlen e und n durch folgende Operation[31] verschlüsselt:

$c = m^e \ mod \ n$

Die verschlüsselte Nachricht c kann nach der Übertragung vom Private Key, den Zahlen d und n, wieder entschlüsselt werden:

$m = c^d \ mod \ n$

Zum digitalen Signieren (Signatur s) einer Nachricht m werden Public und Private Key vertauscht:

$s = m^d \ mod \ n$

bzw.

$m = s^e \ mod \ n$

Der RSA-Algorithmus war bis zum 20. September 2000 in den USA patentiert, ist heute jedoch frei benutzbar (siehe ebenfalls [Inc. 02a]).

5.5 Symmetrische und Asymmetrische Verschlüsselung im Vergleich

Ein entscheidender Nachteil von Asymmetrischen Algorithmen ist der vergleichsweise hohe Rechenaufwand aufgrund der zugrundeliegenden Modularen Arithmetik. Dieser ist um ein Vielfaches (je nach Schlüssellänge z.B. bis 100 mal) höher als bei Symmetrischen Algorithmen.

Ein weiterer Unterschied zwischen den beiden Verfahren liegt in der Schlüssellänge. Für beide Chiffren bedeutet zwar ein längerer Schlüssel auch eine sicherere Verschlüsselung und höheren Rechenaufwand, ein 128 Bit-langer Schlüssel ist jedoch für einen Symmetrischen Algorithmus sehr sicher, für einen Public Key-Algorithmus sehr schwach. Diese Unterschiede sind in den mathematischen Grundlagen der Verfahren begründet.

Während es im symmetrischen Fall gilt, den richtigen Schlüssel durch Ausprobieren aller möglichen Schlüssel (Brute Force-Attacke) zu finden kann bei asymmetrischen Verfahren

[31] Der *mod*-Operator hat als Ergebnis immer den Rest der Division.

zusätzlich noch der Public Key als Ausgangspunkt für die Bestimmung des passenden Private Keys verwendet werden. Es handelt sich hier im Wesentlichen um das Problem der Faktorisierung sehr großer Zahlen.
Die mathematischen Hintergründe sind bei Symmetrischer Verschlüsselung einfacher als bei Asymmetrischer Verschlüsselung. Bei letzterer ist es daher wahrscheinlicher, daß durch mathematische Analysemethoden, insbesondere durch Erkenntnisse über die Möglichkeit der Umkehrung von Einwegfunktionen, noch Möglichkeiten gefunden werden, den Algorithmus zu knacken. Zum Beispiel würde die Verfügbarkeit eines effizienten Faktorisierungsalgorithmus den asymmetrischen RSA-Algorithmus wirkungslos machen.
Der Schlüsselaustausch ist bei beiden Verfahren eine sehr sensible Angelegenheit. Während der geheime Schlüssel bei Symmetrischen Verfahren ohnehin über einen vertrauenswürdigen Weg erfolgen muß kann der Public Key der Asymmetrischen Verschlüsselung auch über ungesicherte Verbindungen übertragen werden. Trotz der Öffentlichkeit des Schlüssels muß dennoch sichergestellt werden, daß der Kommunikationspartner auch den richtigen Schlüssel erhält. Auf die Problematik des Austausches von öffentlichen Schlüsseln werden wir in Abschnitt 5.9 genauer eingehen.
Bei beiden Verfahren hängt die Sicherheit der Verschlüsselung wesentlich von der Verfügbarkeit von statistisch unabhängigen Zufallszahlen und somit von der Qualität des Zufallszahlengenerators (unter Linux `/dev/random`) des Rechners ab.

5.6 Hybride Verschlüsselungsverfahren

Aufgrund der genannten Eigenschaften der beiden Verschlüsselungverfahren werden sie häufig kombiniert eingesetzt. Bei diesen Hybriden Verfahren (siehe Abbildung 43) wird zuerst ein zufälliger Sitzungsschlüssel (Session Key) generiert, der meist nur für eine Sitzung oder einen beschränkten Zeitraum gültig ist. Die eigentlichen Nutzdaten werden mit diesem Sitzungsschlüssel verschlüsselt. Die symmetrisch verschlüsselten Nutzdaten werden zusammen mit dem über den Public Key des Empfängers verschlüsselten Session Key verschickt. Der Empfängers kann mit seinem Private Key den Session Key und damit wiederum die Nachricht entschlüsseln.

So kombiniert man einen sicheren Schlüsselaustausch mit einer schnellen Verschlüsselung. Zu beachten ist, daß die Schlüssel für beide Verfahren eine angemessene Länge haben müssen.

5.7 Kryptographische Prüfsummen

Zum Schutz der Datenintegrität werden ähnlich wie bei der Asymmetrischen Verschlüsselung Einwegfunktionen (One Way Hash Functions) verwendet. Die hier verwendeten Funktionen aktzeptieren als Eingabe eine Nachricht beliebiger Länge, erzeugen jedoch daraus immer eine Zeichenfolge konstanter Länge, z.B. eine 128 Bit-Folge, die Kryptographische Prüfsumme oder Hash.

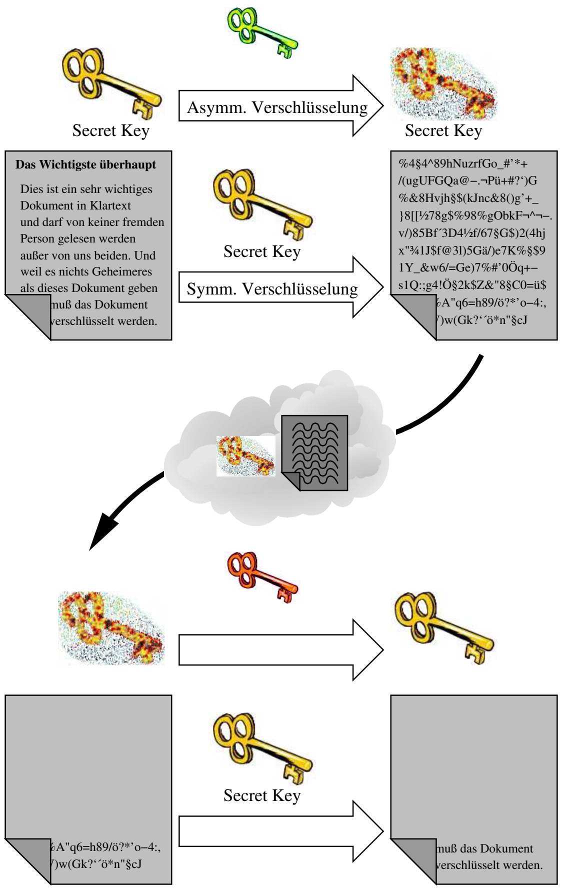

Abbildung 43: Hybride Verschlüsselung

Die hier verwendeten kryptographisch sicheren (kollisionsfreien) Einwegfunktionen müssen folgende Eigenschaften haben:

- Aus der Kryptographischen Prüfsumme darf die ursprüngliche Nachricht nicht rekonstruierbar sein.
- Da die Prüfsumme im Allgemeinen erheblich kürzer als die Nachricht selbst ist, gibt es zu einer Prüfsumme im Prinzip unendlich viele passende Nachrichten. Der Prüfsummenalgorithmus muß daher garantieren, daß die Wahrscheinlichkeit, innerhalb einer gewissen Zeitspanne eine zweite Nachricht mit derselben Prüfsumme zu finden, verschwindend gering ist (Kollisionsfreiheit).
- Jede Veränderung an der Nachricht ergibt eine vollständig andere, von der Prüfsumme der Original-Nachricht statistisch unabhägige Prüfsumme[32].

[32]Mathematisch ausgedrückt bedeutet dies, daß bei Änderung eines einzigen Bits der Nachricht für jedes einzelne Bit der Prüfsumme die Wahrscheinlichkeit einer Änderung bei genau 50% liegen muß.

Einige Prüfsummenfunktionen arbeiten mit einem zusätzlichen Schlüssel, der bei der Berechnung des Hash-Wertes benötigt wird und den Algorithmus damit um die Funktion der Authentisierung erweitert. Diese Art von Hashes werden als Message Authentication Code (MAC) bezeichnet. In RFC 2104 wird ein Algorithmus (HMAC, Keyed-Hashing for Message Authentication) beschrieben, der einen Hash-Algorithmus um diese Funktion erweitert. Im einfachsten Fall kann der letzte verschlüsselte Block eines Block-Algorithmus in einem der Rückkopplungs-Moden (siehe Abschnitt 5.3) als MAC verwendet werden. Der Initialisierungsvektor übernimmt dann die Rolle des Schlüssels.

5.7.1 Message Digest Nr. 5 (MD5)

Auf die Vorgängerversionen MD2 und MD4 folgte MD5 (siehe RFC 1321, [Rive 92]) als ein weit verbreiteter Hash-Algorithmus der Firma RSA Data Security. MD5 erzeugt eine 128-Bit-lange Kryptographische Prüfsumme.
Unter Linux steht das Kommando `md5sum`[33] zur Berechnung der MD5-Prüfsumme zur Verfügung:

```
linux:~# md5sum Nachricht.ps
6cbf4606139c25dd1f646b89c79e52b0  Nachricht.ps
linux:~#
```

Eine Variante von MD5 mit zusätzlichem Authentisierungsschlüssel nennt sich Keyed MD5 (RFC 1828).
Dem MD5-Algorithmus wurde in [Dobb 96] eine potentielle Schwachstelle nachgewiesen, er ist daher mit Vorsicht einzusetzen.

5.7.2 Secure Hash Algorithm, SHA-1

SHA-1 wurde vom amerikanischen NIST (National Institute for Standards and Technology) zusammen mit der NSA (National Security Agency) entwickelt. Die generierte Prüfsumme ist 160 Bit lang. Bei diesem Algorithmus gibt es keine bekannten Schwachstellen.
SHA-1 kann in Verbindung mit HMAC als MAC verwendet werden (Keyed SHA-1, HMAC-SHA-1, siehe RFCs 2104 und 2404).
Unter Linux berechnet der Befehl `sha1sum` die SHA-1-Prüfsumme einer Datei.

5.7.3 RIPEMD

Bei RIPEMD handelt es sich um eine Hash-Funktion, deren Design auf dem des MD4-Algorithmus aufbaut. Entwickelt wurde RIPEMD für das RIPE-Projekt der Europäischen Union, aufbauend auf Angriffe auf MD5.

[33] Bei Unix manchmal nur `md5`.

Eine erste 128-Variante stellte sich als unsicher heraus, was zur Entwicklung des wesentlich stärkeren RIPEMD-160 mit einer 160 Bit-langen Prüfsumme führte. Es sind auch Varianten mit 256 oder 320 Bit verfügbar. Der Algorithmus gilt bisher als sicher und auch seine Design-Kriterien sind im Gegensatz zu SHA-1 voll offengelegt.

Die RIPEMD-Algorithmen sind unter [Boss 99] genauer beschrieben und im Quellcode erhältlich.

5.8 Digitale Signaturen

Eine Digitale Signatur muß ähnlich wie eine manuelle Unterschrift folgende Anforderungen erfüllen:

- Verifizierbarkeit: Die Digitale Signatur muß einwandfrei auf Echtheit überprüft werden können.
- Die Signatur muß untrennbar mit dem signierten Dokument verbunden sein.
- Fälschungssicherheit: Die Signatur kann nur von der einen identifizierten Person stammen.
- Verbindlichkeit: Der Unterzeichner kann eine einmal geleistete Signierung nicht mehr abstreiten.
- Integrität: Der Inhalt eines einmal unterschriebenen Dokumentes kann nachträglich nicht mehr verändert werden.

Digitale Signaturen können jedoch nicht nur für Personen, sondern auch für Organisationen oder Rechnersysteme verwendet werden. Sie lassen sich durch die Kombination von Verschlüsselungs- und Hash-Algorithmen realisieren (siehe Abbildung 44).
Aus dem zu signierenden Dokument wird zunächst mit einem Hash-Algorithmus eine kryptographische Prüfsumme berechnet. Diese wird anschließend mit dem Private Key des Absenders verschlüsselt. Die unverschlüsselte Nachricht und die verschlüsselte Prüfsumme werden an den Empfänger übermittelt, der mit dem Public Key des Absenders die vom Absender generierte Prüfsumme entschlüsseln kann. Eine mit dem selben Hash-Algorithmus aus dem erhaltenen Dokument berechnete Prüfsumme wird dann mit der übermittelten Prüfsumme verglichen. Bei Übereinstimmung der Prüfsummmen ist sichergestellt, daß die Nachricht auf ihrem Weg nicht verändert wurde und auch wirklich vom richtigen Absender stammt, weil nur dieser den passenden Private Key besitzen kann. Dies stellt auch sicher, daß der Absender seine Digitale Unterschrift nicht mehr abstreiten kann. Stimmen die Prüfsummen nicht überein so ist die übermittelte Nachricht nicht vertrauenswürdig.
Prinzipiell könnte auch das Dokument selbst zur Signierung mit dem Private/Public Key des Absenders verschlüsselt/entschlüsselt werden. Aufgrund der Komplexität der Asymmetrischen Verschlüsselung würde der Vorgang aber zu lange dauern.

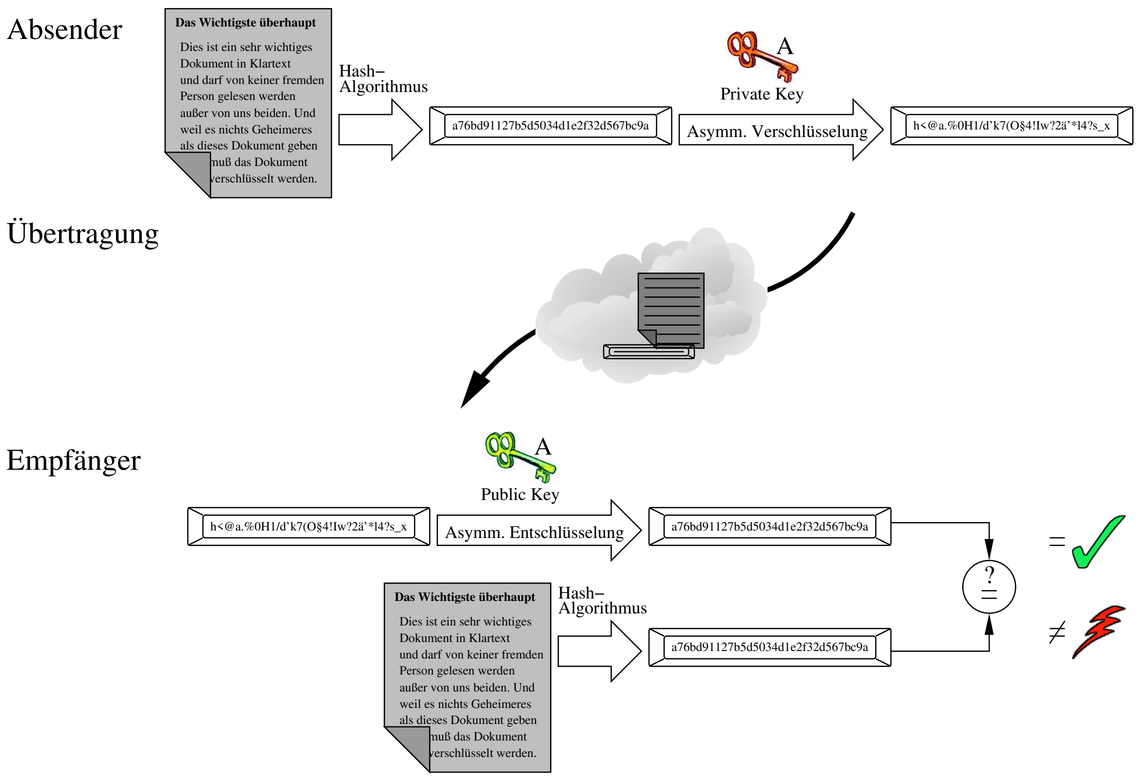

Abbildung 44: Digitale Signatur als Kombination aus Verschlüsselung und Prüfsummenbildung

Besteht zusätzlich zur Signierung noch die Notwendigkeit, das zu übermittelnde Dokument vor fremden Zugriff zu schützen kann dies über Hybride Verschlüsselung erfolgen.

Abbildung 45 zeigt zusammenfassend den vollständigen Ablauf der signierten und verschlüsselten Übertragung eines Dokumentes.

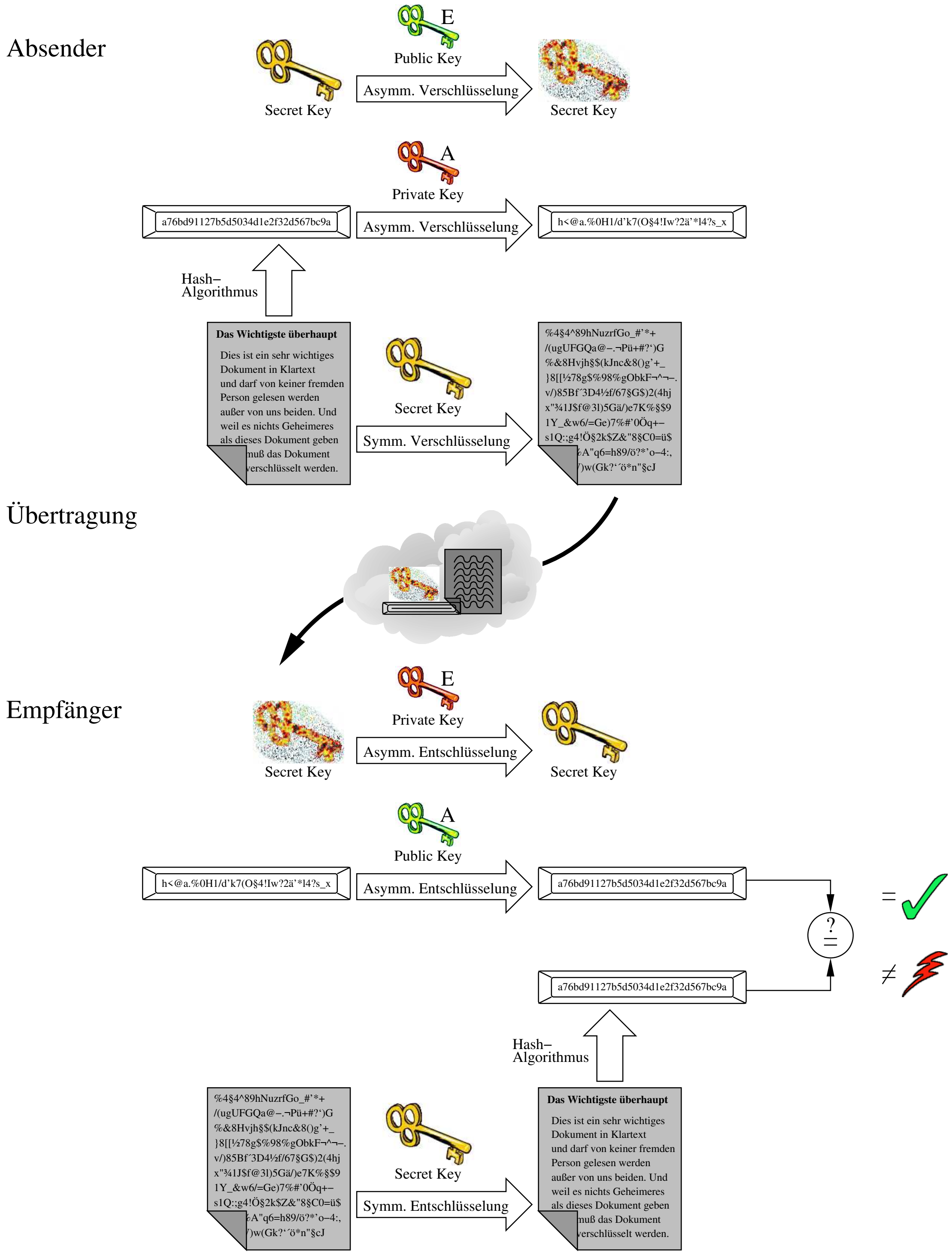

Abbildung 45: Signierte und verschlüsselte Übertragung einer Nachricht

5.9 Zertifikate

Die Sicherheit der Asymmetrischen Verschlüsselung und aller darauf aufbauenden Verfahren basieren auf zwei Voraussetzungen:

- Der Private Key ist eindeutig einem Besitzer zugeordnet und nur diesem bekannt
- Der Public Key paßt zum Private Key des jeweiligen Kommunikationspartners

Während die Problematik des ersten Punktes offensichtlich ist (kommt ein Dritter in den Besitz des Private Keys einer anderen Person kann er Dokumente in dessen Namen unterzeichnen und dessen Kommunikation entschlüsseln) ist die Problematik im Falle des Public Keys etwas komplizierter.
Ein Public Key ist zwar prinzipiell nichts Geheimes, es ist jedoch entscheidend, zu welchem Private Key er paßt. Dies ist aber nicht ohne weiteres festzustellen.
Gelingt es einem Angreifer, beim Austausch der Public Keys zwischen den Kommunikationspartnern die "richtigen" Public Keys durch seine eigenen zu ersetzen, könnte er die übertragenen Dokumente lesen.

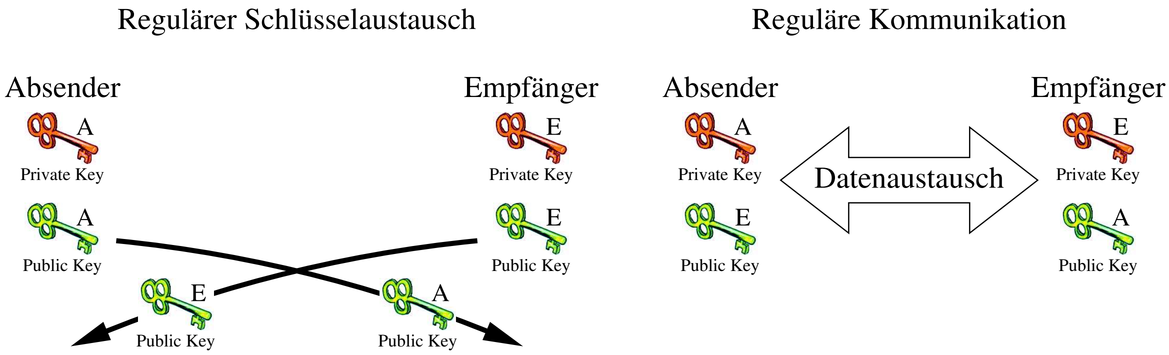

Abbildung 46: Reguläre Kommunikation mit Private/Public Key-Verschlüsselung

Gelingt es ihm zudem, sich in den Kommunikationspfad zwischen Absender und Empfänger einzuklinken und alle gesendeten Nachrichten abzufangen würde seine Präsenz nicht ohne weiteres bekannt und er könnte die Dokumente zusätzlich noch nach Belieben verändern. Diese Art von Man-in-the-Middle-Attacke ist in Abbildung 47 dargestellt:

Beim Schlüsselaustausch fängt der Hacker die Nachrichten mit den Public Keys von Absender und Empfänger ab und ersetzt darin die Schlüssel mit eigenen Public Keys. Anschließend stellt er die Nachrichten an die beiden Kommunikationspartner zu. Die mit diesen Schlüsseln verschlüsselten bzw. signierten Nachrichten werden auch vom Angreifer abgefangen. Dieser kann die Nachrichten entschlüsseln, lesen und verändern, und dann erneut verschlüsselt und signiert an den Empfänger weiterreichen. Da dieser die Nachricht mit seinem Private Key und dem vermeintlich richtigen Public Key des Absenders entschlüsseln und verifizieren kann wird er das Dokument für vertrauenswürdig halten.

Das Einklinken in den Kommunikationspfad könnte z.B. an einem Mailserver erfolgen, welchen die Mails der angegriffenen Personen durchlaufen. Möglich wäre auch, über DNS-Spoofing Mails oder andere verschlüsselte Verbindungen (HTTPS, SSH usw.) über einen dritten Server umzuleiten, auf dem der Angriff stattfinden kann.

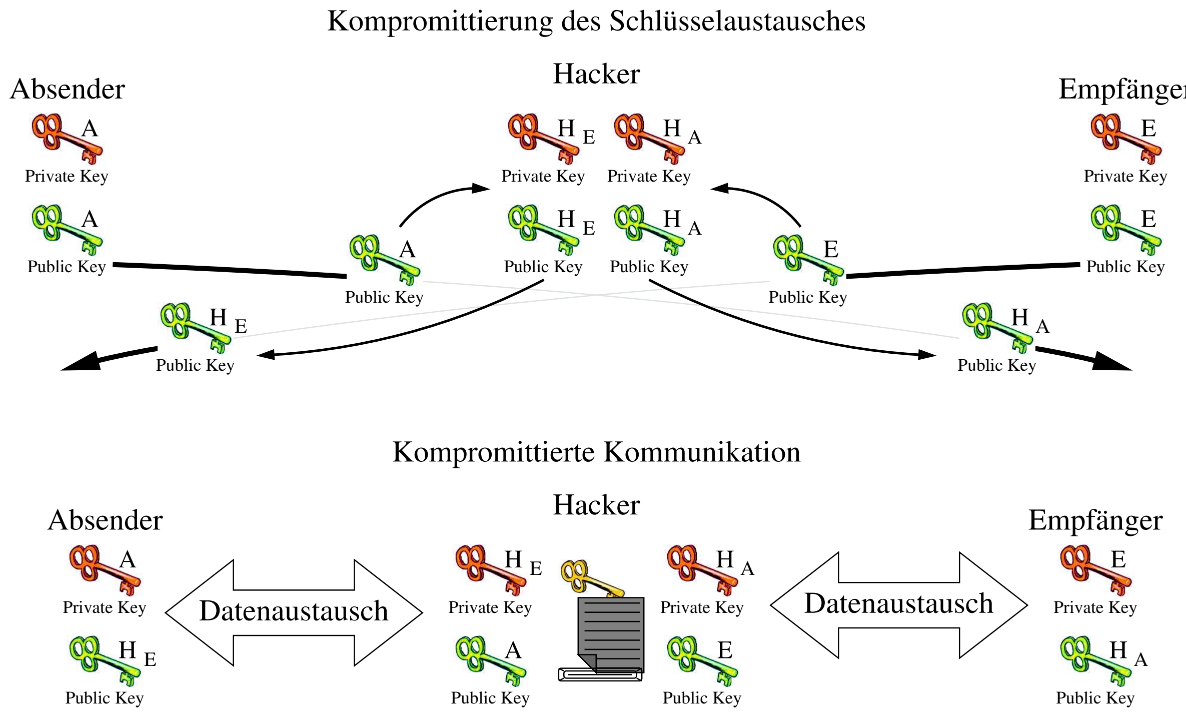

Abbildung 47: Man-in-the-Middle-Angriff auf die Private/Public Key-Verschlüsselung

Ein vertrauenswürdiger Austausch der öffentlichen Schlüssel ist also zentraler Bestandteil der Asymmetrischen Verschlüsselung und somit auch der Signierung. Diese gesicherte Schlüsselübergabe soll durch Zertifikate garantiert werden.
Ein Zertifikat ist ein öffentlicher Schlüssel verbunden mit der Beschreibung des Besitzers des Schlüssels, zusammen signiert von einer vertrauenswürdigen Instanz. Diese Instanz ist ganz allgemein eine Person oder Organisation, welcher die Kommunikationspartner vertrauen, eine so genannte Zertifizierungsinstanz (CA, Certification Authority oder Trust Center).

Die Generierung eines Zertifikates über eine CA wird in Abbildung 48 beschrieben.

Der Besitzer X.Y. des öffentlichen Schlüssels reicht diesen bei einer CA ein. Die CA überprüft die Identität von X.Y. und dessen öffentlichen Schlüssel. Sind alle Daten verifiziert signiert die CA den Public Key von X.Y. zusammen mit seinen persönlichen Daten und einigen CA- und Zertifikat-spezifischen Daten mit dem privaten Schlüssel der CA.
Das Zertifikat wird dann in einer für alle Kommunikationspartner zugänglichen Datenbank eingetragen. Jeder, der nun mit X.Y. kommunizieren möchte kann sich dessen Zertifikat

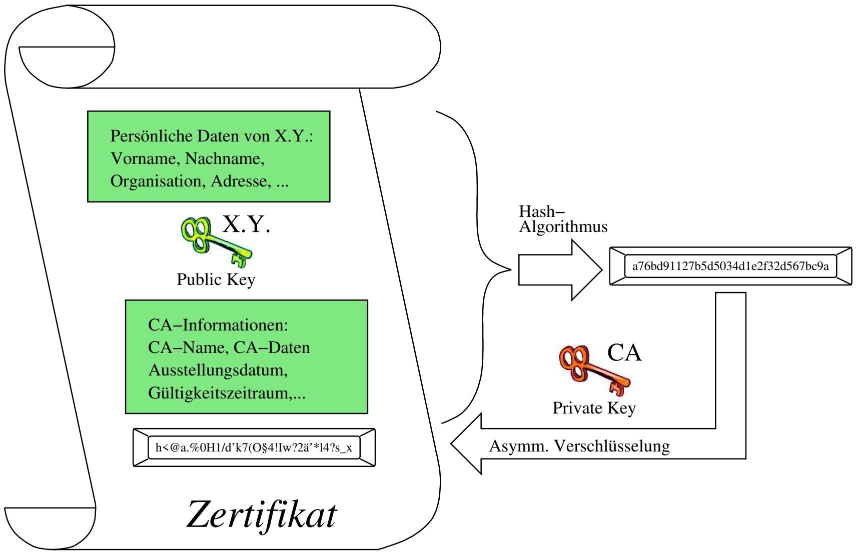

Abbildung 48: Generierung eines Zertifikates

aus dieser Datenbank holen und daraus den öffentlichen Schlüssel von X.Y. mit dem öffentlichen Schlüssel der CA entschlüsseln. Natürlich ist für diesen auch wieder sicherzustellen, daß er auch zur richtigen CA gehört. Da die Anzahl dieser CA-Schlüssel aber überschaubar ist gestaltet sich auch deren Verteilung und Verifizierung einfacher. Z.B. sind die Zertifikate und öffentlichen Schlüssel der gängigen CAs für Internet-Angebote in vielen Browsern integriert. Als Beispiel finden Sie in Abbildung 49 einen Auszug aus der CA-Zertifikatliste von Mozilla/Netscape.

5.9.1 X.509-Zertifikate

Der X.509-Standard definiert Rahmenbedingungen, zur Bereitstellung von Authentisierungsdiensten. Ein wichtiger Bestandteil ist die Spezifikation des X.509-Zertifikats, das sich inzwischen in der Praxis als Standard durchgesetzt hat. Das X.509-Zertifikat-Format in Version 3 ist in RFC 3280 beschrieben.

Zum Thema Zertifikate, Zertifizierungsstellen und Public Key Infrastruktur (PKI) gibt es noch viele weitere Aspekte, die wir im Rahmen dieses Praktikums jedoch nicht behandeln können. Für eine Vertiefung sei hier auf spezielle Fachliteratur wie z.B. [Aust 01] verwiesen.

5.10 GNU Privacy Guard (GnuPG)

GnuPG ist ein Werkzeug zur Signierung und Verschlüsselung von Daten. Es ist für alle gängigen Linux/Unix-Systeme frei verfügbar, es existieren aber auch Windows- und MacOS X-Versionen.

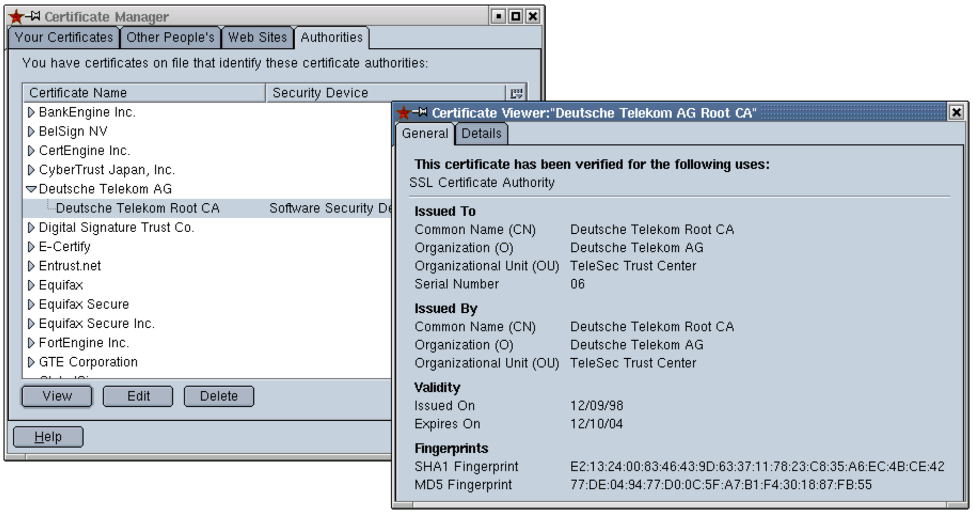

Abbildung 49: Zertifikate im Browser (Mozilla)

GnuPG wird über die Kommandozeile bedient. Zusätzlich gibt es grafische Oberflächen, über welche die wichtigsten Funktionen von GnuPG aufgerufen werden können.
Wir werden im Folgenden die grundlegenden Funktionen für die Signierung und Verschlüsselung von Dateien sowie für die Erzeugung von Schlüsselpaaren mit GnuPG und dem grafischen KDE-Programm KGPG beschreiben. KGPG unterstützt noch nicht alle Funktionen von GnuPG, bietet aber die Möglichkeit, GnuPG direkt von der Oberfläche aus im Kommandomodus zu starten.

5.10.1 Schlüsselgenerierung und -verwaltung

GnuPG wird über das Kommando gpg gestartet. Alle Daten werden standardmäßig im Verzeichnis .gnupg im Home-Verzeichnis des Benutzers ($HOME/.gnupg/) abgelegt.

Beim ersten Aufruf von KGPG (Kommando kgpg) oder gpg wird eine Beispiel-Konfigurationsdatei $HOME/.gnupg/gpg.conf erzeugt. Sind in $HOME/.gnupg/ noch keine Schlüsselpaare vorhanden fragt KGPG nach, ob ein neues Schlüsselpaar generiert werden soll (Siehe Abbildung 50).

Nach Eingabe aller Daten wird ein Passwort, das so genannte Mantra (kann auch Leerzeichen enthalten), benötigt, über welches der Zugriff auf den Private Key geschützt wird.
KGPG hängt sich unter KDE in die Kontrollleiste ein. Durch Anklicken des Icons können die einzelnen Funktionen von KGPG aufgerufen werden (Key Manager, Editor und weitere).

Auf der Kommandozeile wird ein neues Schlüsselpaar mit gpg --gen-key generiert. Nur auf der Kommandozeile ist es möglich, mit gpg --edit-key <User-ID> addphoto ein Foto im JPEG-Format zum Public Key, eine so genannte Photo-ID, hinzuzufügen.

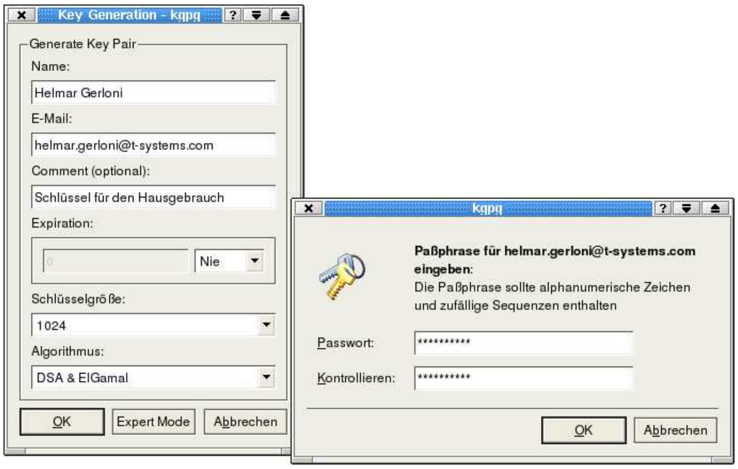

Abbildung 50: Erzeugung eines neuen Schlüsselpaares mit KGPG.

Jeder Schlüssel bzw. jedes Schlüsselpaar wird von GnuPG über eine User-ID identifiziert. Diese ergibt sich aus "Vorname Nachname (Kommentar) <e-mail@adresse.de>". Beim Aufruf von `gpg` reicht es aber aus, wenn für `<User-ID>` eine eindeutige Zeichenfolge aus der User-ID angegeben wird, also z.B. nur die E-Mail-Adresse oder der Nachname.
Zu beachten ist, daß bei GnuPG der Begriff "Schlüssel" (Key) als Oberbegriff für einen Private Key mit den dazugehörigen Public Keys, der User-ID und Photo-ID verwendet wird. Die Public und Private Keys selbst werden als Unterschlüssel (Subkeys) bezeichnet. Ein Schlüssel beinhaltet mindestens einen Public oder Private Key.
Um die Angelegenheit nicht zu kompliziert zu machen werden wir in unseren Beispielen davon ausgehen, daß jeder Schlüssel einen einzigen Public Key, maximal einen Private Key, eine User-ID und eine Photo-ID enthält.

Falls der Public Key des neuen Schlüsselpaares auf einem Keyserver bereitgestellt werden soll ist es sinnvoll, sofort ein Schlüsselwiderruf-Zertifikat (Revocation Certificate) zu erzeugen, um im Notfall, z.B. bei Verlust des Private Keys oder des Mantras, das Schlüsselpaar als ungültig markieren zu können. Da für die Generierung des Schlüsselwiderruf-Zertifikates der Private Key und das Mantra benötigt wird ist ein späterer Widerruf des Schlüsselpaar ohne fertiges Widerruf-Zertifikat nicht mehr möglich.
Ein Widerruf-Zertifikat zum Schlüssel `<eigene User-ID>` wird mit `gpg --output <Widerruf>.asc --gen-revoke <eigene User-ID>` in die Datei `<Widerruf>.asc` geschrieben. Diese Datei sollte an einem sicheren Ort gespeichert werden, da sie von einem Dritten dazu verwendet werden könnte, den eigenen Public Key als ungültig zu markieren.

Es können beliebig viele Schlüsselpaare (z.B. eines für die private, eines für die geschäftliche Kommunikation) generiert werden. Diese und alle anderen Schlüssel werden in den Dateien `$HOME/.gnupg/pubring.gpg` (der so genannte Schlüsselbund oder Keyring für die Public Keys) und `$HOME/.gnupg/secring.gpg` (Schlüsselbund für die Private Keys) abgelegt.

Zur Weitergabe des eigenen Public Keys inklusive Foto kann dieser im KGPG Key Ma-

nager über den Menüpunkt Schlüssel - Export Public Key... bzw. mit `gpg --output <MeinSchluessel>.asc --export --armor <User-ID>` im ASCII-Format in die Datei `<MeinSchluessel>.asc` exportiert werden.
Entsprechend müssen die Public Keys der Kommunikationspartner in den eigenen Schlüsselbund aufgenommen werden. Dies geschieht wiederum mit Schlüssel - Import Key... oder mit `gpg --import <Partnerschluessel>.asc` .
Vor Verwendung eines fremden Public Keys sollte unbedingt dessen Echtheit anhand seines Fingerabdrucks (Fingerprint) im Rahmen einer persönlichen Kontaktaufnahme verifiziert werden. Der Fingerabdruck kann über den Menüpunkt Schlüssel - Key Info oder mit `gpg --fingerprint <User-ID>` angezeigt werden. Der Public Key kann nach dieser Verifizierung als vertrauenswürdig gekennzeichnet werden, indem er mit einem eigenen Private Key signiert wird. Dies geschieht über Schlüssel - Sign Key... oder mit `gpg --local-user <eigene User-ID> --edit-key <fremde User-ID> sign` .

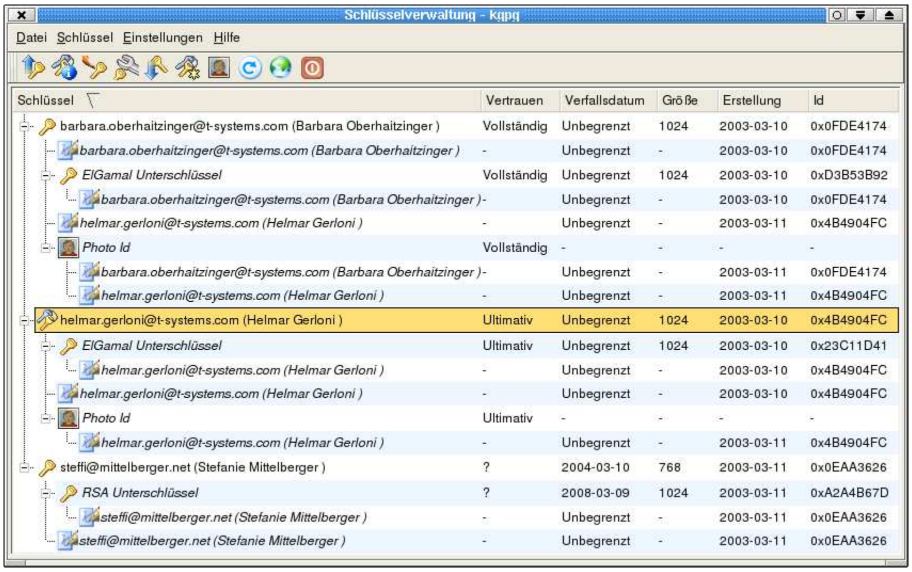

Abbildung 51: Schlüsselverwaltung mit KGPG.

Abbildung 51 zeigt die so entstandene Schlüsselliste von Helmar Gerloni im KGPG Key Manager. Man erkennt, daß alle Schlüssel standardmäßig vom jeweiligen Besitzer selbst signiert sind.
Das markierte, selbst generierte Schlüsselpaar enthält neben dem Private Key (nicht explizit dargestellt) den ElGamal-Public Key (Unterschlüssel) sowie die Photo-ID. Von Barbara Oberhaitzinger wurde sowohl ein Public Key als auch eine Photo-ID importiert. Beide wurden verifiziert und mit dem Private Key von Helmar Gerloni unterzeichnet. Von Stefanie Mittelberger ist ein Public Key vorhanden, der noch nicht von Helmar Gerloni unterzeichnet wurde und somit noch nicht vertrauenswürdig ist (? in der Spalte Vertrauen).

Alternativ kann mit `gpg --list-public-keys` bzw. `gpg --list-secret-keys` eine Liste der Schlüssel mit den dazugehörigen User-IDs und Photo-IDs ausgegeben werden:

```
helmar@linux:~$ gpg --list-public-keys
/home/helmar/.gnupg/pubring.gpg
--------------------------------
pub  1024D/4B4904FC 2003-03-10 Helmar Gerloni (Schlüssel für den Hausgebrauch) <helmar.gerloni@t-systems.com>
uid                            [jpeg image of size 3890]
sub  1024g/23C11D41 2003-03-10

pub   768D/0EAA3626 2003-03-11 Stefanie Mittelberger (Muggi) <steffi@mittelberger.net>
sub  1024R/A2A4B67D 2003-03-11 [verfällt: 2008-03-09]

pub  1024D/B43CB16C 2003-03-31 Barbara Oberhaitzinger (MEIN Schlüssel) <barbara.oberhaitzinger@t-systems.com>
uid                            [jpeg image of size 4243]
sub  1024g/F7474A1F 2003-03-31

helmar@linux:~$ gpg --list-secret-keys
/home/helmar/.gnupg/secring.gpg
--------------------------------
sec  1024D/4B4904FC 2003-03-10 Helmar Gerloni (Schlüssel für den Hausgebrauch) <helmar.gerloni@t-systems.com>
uid                            [jpeg image of size 3890]
ssb  1024g/23C11D41 2003-03-10

helmar@linux:~$
```

5.10.2 Verschlüsseln und entschlüsseln

Für das Verschlüsseln und Signieren von beliebigen Dateien dient bei KGPG der Editor. ASCII-Texte können damit auch bearbeitet und aus der Zwischenablage übernommen werden.
Für die Übermittlung einer verschlüsselten Nachricht `<Nachricht>.txt` an Helmar wird Barbara diese über Datei - Encrypt File... im KGPG-Editor auswählen. KGPG zeigt dann eine Liste aller Schlüssel an, die einen Public Key für die Verschlüsselung beinhalten. Nach Auswahl des Schlüssels von Helmar wird die verschlüsselte Nachricht in die Datei `<Nachricht>.txt.asc` geschrieben.
Helmar wird die an ihn übermittelte Datei `<Nachricht>.txt.asc` über Datei - Decrypt File... im Editor öffnen, im folgenden Dialog das Mantra für seinen Private Key angeben und die Klartext-Nachricht nach `<Nachricht>.txt` schreiben.
Alternativ können alle Aktionen auch aufgerufen werden, indem die betreffende Datei aus dem Dateimanager (Konqueror) auf das KGPG-Icon in der Kontrollleiste gezogen wird.
Auf der Kommandozeile wird eine Datei mit `gpg --encrypt --armor --recipient <fremde User-ID> <Nachricht>.txt` mit dem Public Key des Empfängers verschlüsselt und von diesem mit seinem Private Key durch `gpg --output <Nachricht>.txt --decrypt <Nachricht>.txt.asc` wieder entschlüsselt:

```
barbara@linux:~$ gpg --encrypt --armor --recipient helmar.gerloni@t-systems.com Nachricht.txt
barbara@linux:~$
```

Die Datei `Nachricht.txt.asc` wird übertragen.

```
helmar@linux:~$ gpg --output Nachricht.txt --decrypt Nachricht.txt.asc

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Helmar Gerloni (Schlüssel für den Hausgebrauch) <helmar.gerloni@t-systems.com>"
1024-Bit ELG-E Schlüssel, ID 23C11D41, erzeugt 2003-03-10 (Hauptschlüssel-ID 4B4904FC)

gpg: verschlüsselt mit 1024-Bit ELG-E Schlüssel, ID 23C11D41, erzeugt 2003-03-10
      "Helmar Gerloni (Schlüssel für den Hausgebrauch) <helmar.gerloni@t-systems.com>"
helmar@linux:~$
```

Soll eine verschlüsselte Nachricht an mehrere Empfänger gleichzeitig geschickt werden, muß sie mit dem Public Key eines jeden Empfängers verschlüsselt werden. Dazu können in KGPG mehrere Schlüssel gleichzeitig ausgewählt werden, auf der Kommandozeile wird dazu einfach die Option `--recipient <fremde User-ID>` mehrfach angegeben.

5.10.3 Signieren und Signaturen prüfen

Die Signierung von Dateien verläuft analog zur Verschlüsselung. Über Signatur - Signatur generieren wird die zu signierende Datei geöffnet. Nach Auswahl des eigenen Private Keys wird die zur Nachricht `<Nachricht>.txt` passende Signatur nach `<Nachricht>.txt.sig` geschrieben. Nach der Übertragung beider Dateien `<Nachricht>.txt` und `<Nachricht>.txt.sig` kann der Empfänger über Signatur - Signatur überprüfen die Datei `<Nachricht>.txt` öffnen. KGPG sucht den passenden Public Key im Schlüsselbund aus und gibt nach der Prüfung eine Meldung aus, ob die Signatur in `<Nachricht>.txt.sig` zur Nachricht paßt und von wem die Signatur stammt.
Mit GnuPG erfolgt die Signierung über `gpg --output <Nachricht>.txt.sig --armor --local-user <eigene User-ID> --detach-sig <Nachricht>.txt`, die Prüfung der Signatur über `gpg --verify <Nachricht>.txt.sig`:

```
barbara@linux:~$ gpg --output Nachricht.txt.sig --armor --local-user barbara.oberhaitzinger@t-systems.com --detach-sig Nachricht

You need a passphrase to unlock the secret key for
user: "Barbara Oberhaitzinger (Allgemeiner Schlüssel) <barbara.oberhaitzinger@t-systems.com>"
1024-bit DSA key, ID B43CB16C, created 2003-03-31

barbara@linux:~$
```

Die Dateien `Nachricht.txt` und `Nachricht.txt.sig` werden übertragen.

```
helmar@linux:~$ gpg --verify Nachricht.txt.sig
gpg: Unterschrift vom Fre 04 Apr 2003 11:35:18 CEST, DSA Schlüssel ID B43CB16C
gpg: Korrekte Unterschrift von "Barbara Oberhaitzinger (Allgemeiner Schlüssel) <barbara.oberhaitzinger@t-systems.com>"
gpg:                    alias "[jpeg image of size 4243]"
helmar@linux:~$
```

Um mit KGPG eine Datei sowohl zu verschlüsseln als auch zu signieren müssen die beiden ober beschriebenen Schritte einzeln durchgeführt werden. Übertragen werden dann die beiden Dateien `Nachricht.txt.asc` (verschlüsselte Nachricht) und `Nachricht.txt.sig` (zur Nachricht passende Signatur).
Mit GnuPG gibt es die Möglichkleit, beide Schritte zu kombinieren:

```
barbara@linux:~$ gpg --encrypt --armor --recipient helmar.gerloni@t-systems.com --local-user barbara.oberhaitzinger@t-systems.co

You need a passphrase to unlock the secret key for
user: "Barbara Oberhaitzinger (Allgemeiner Schlüssel) <barbara.oberhaitzinger@t-systems.com>"
1024-bit DSA key, ID B43CB16C, created 2003-03-31

barbara@linux:~$
```

Die Datei `Nachricht.txt.asc` enthält nun sowohl die verschlüsselte Nachricht als auch die Signatur. Nur diese Datei muß übertragen werden.

```
helmar@linux:~$ gpg --output Nachricht.txt --decrypt Nachricht.txt.asc

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Helmar Gerloni (Schlüssel für den Hausgebrauch) <helmar.gerloni@t-systems.com>"
1024-Bit ELG-E Schlüssel, ID 23C11D41, erzeugt 2003-03-10 (Hauptschlüssel-ID 4B4904FC)

gpg: verschlüsselt mit 1024-Bit ELG-E Schlüssel, ID 23C11D41, erzeugt 2003-03-10
      "Helmar Gerloni (Schlüssel für den Hausgebrauch) <helmar.gerloni@t-systems.com>"
gpg: Unterschrift vom Fre 04 Apr 2003 11:55:25 CEST, DSA Schlüssel ID B43CB16C
gpg: Korrekte Unterschrift von "Barbara Oberhaitzinger (Allgemeiner Schlüssel) <barbara.oberhaitzinger@t-systems.com>"
gpg:                    alias "[jpeg image of size 4243]"

helmar@linux:~$
```

5.11 Internet Protocol Security (IPSEC)

Um nicht nur eine applikationsspezifische Sicherung der Datenübertragung zu ermöglichen wurden mit IPSEC Standards geschaffen, die das IP-Protokoll um entsprechende Funktionen erweitern (RFC 2401 bis RFC 2410). Dadurch kann eine sichere Übertragung der Daten schon auf der Vermittlungsschicht für alle darauf aufbauenden Protokolle und Anwendungen erfolgen. IPSEC ist Bestandteil von IPv6, aber auch für IPv4 verfügbar. Wir werden IPSEC nur in Verbindung mit IPv4 behandeln.
Die wesentlichen Erweiterungen des IP-Protokolls durch IPSEC sind AH und ESP.

5.11.1 IP Authentication Header (AH)

Für die Garantie von Datenintegrität und Authentizität sowie bei Verwendung von Sequenznummern zum Schutz vor Replay-Attacken kann zwischen IP-Header und dem Header der Transportschicht (meist TCP oder UDP) der so genannte IP Authentication Header (AH) eingefügt werden (siehe Abbilung 52 und RFC 2402 [KeAt 98]). AH beinhaltet keine Funktionen zum Schutz der Vertraulichkeit, also keine Verschlüsselung und unterliegt deshalb auch keinen Export-Beschränkungen.
AH schützt alle Teile des IP-Paketes, die sich auf dem Weg durchs Netz nicht mehr ändern dürfen. Die Teile des IP-Headers (z.B. TTL), die von den durchlaufenen Routern geändert werden können, werden von AH nicht berücksichtigt.

Für die Prüfsummenbildung und die Authentisierung verwendet AH standardmäßig die Keyed-Variante des MD5-Algorithmus mit mindestens 128 Bit Schlüssellänge.

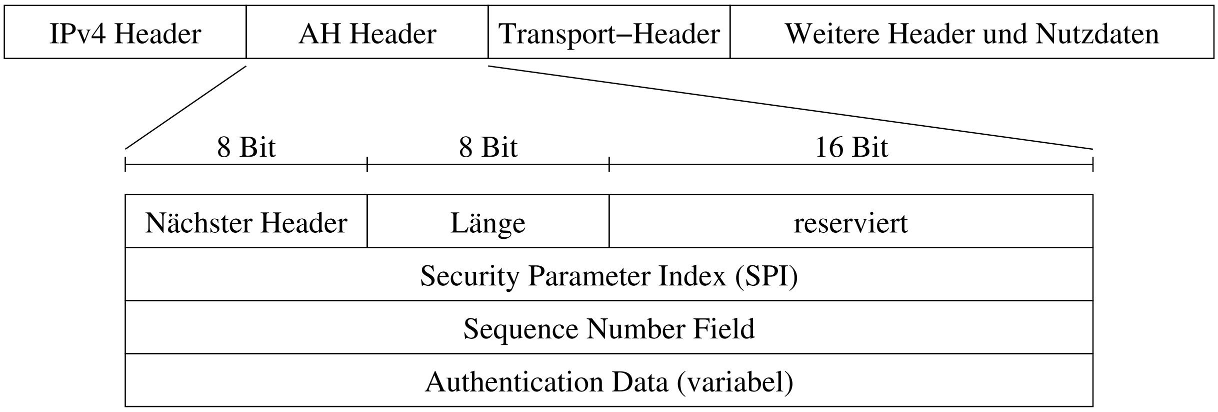

Abbildung 52: IP Authentication Header von IPSEC

Neben der vorgeschriebenen Unterstützung von Keyed MD5 kann AH implementierungsabhängig auch andere Hash- und Authentisierungsalgorithmen unterstützen und damit auch die Verbindlichkeit der Datenpakete realisieren[34] (z.B. mit RSA).
AH wird im IP-Header mit der Protokoll-Nummer 51 im Feld "Protokoll der Transportschicht" (Abbildung 5) identifiziert. Da dieses Feld somit nicht mehr für die Identifikation des folgenden Transportschicht-Protokolls zur Verfügung steht, gibt es im AH das Feld "Nächster Header", welches diese Aufgabe übernimmt. Das Längen-Feld gibt die Anzahl der 32 Bit-Worte an, die auf die 16 reservierten Bits folgen.
AH kann sowohl für Punkt-zu-Punkt-Verbindungen zwischen zwei Clients, zwischen einem Client und einem Gateway oder zwischen zwei Gateways eingesetzt werden, die zwei sichere Netze über ein unsicheres Netz miteinander verbinden.

5.11.2 IP Encapsulating Security Payload (ESP)

Ein weiterer Sicherheitsmechanismus bei IPSEC ist der IP Encapsulating Security Payload (ESP) Header (RFC 1827 [Atki 95b]) mit der Protokollnummer 50.
ESP beinhaltet immer eine Datenverschlüsselung, kann aber, abhängig vom verwendeten Algorithmus, auch zum Schutz von Datenintegrität und Authentizität verwendet werden. Wie bei AH ist über Sequenznummern auch der Schutz vor Wiedereinspielen von Datagrammen realisierbar.
ESP kennt zwei verschiedene Betriebsmodi. Im **Tunnel-Mode** werden die vollständigen IP-Pakete gesichert und die daraus entstehenden Daten mit neuen Klartext-IP-Headern versehen. Der Tunnel-Mode wird zwischen Gateways eingesetzt, um zwei vertrauenswürdige Netzwerke über ein fremdes Netz (Internet) miteinander zu verbinden. In diesem Modus

[34] Alle diese Funktionen sind natürlich nur möglich, wenn vor Anwendung von AH die Schlüssel auf sichere Weise ausgetauscht wurden, siehe Kapitel 5.11.4.

werden alle Informationen (IP-Adressen, Flags usw.) aus den internen Netzen verschlüsselt und mit einem neuen IP-Header aus dem externen Netz versehen, damit die Daten dort transportiert werden können.
Im **Transport-Mode** werden nur die IP-Daten verschlüsselt, der IP-Header bleibt Klartext. Die Pakete können dann direkt an die Router weitergeleitet werden. Dieser Modus wird für Punkt-zu-Punkt-Verbindungen zwischen Clients benötigt.
ESP bringt abhängig von der gewählten Verschlüsselung höhere Durchlaufzeiten und Anforderungen an die Rechenleistung der beteiligten Geräte mit sich. Dabei ist der Tunnel-Mode aufgrund der Verschlüsselung und Neuadressierung ganze IP-Pakete deutlich aufwändiger. Router, welche die ESP-verschlüsselten Pakete nur weiterleiten müssen, sind davon nicht betroffen.
Standardmäßig muß von allen ESP-Implementierungen als symmetrischer Verschlüsselungsalgorithmus DES im CBC-Mode unterstützt werden. Ähnlich wie bei AH können aber auch weitere Algorithmen integriert sein, ebenso kann auch ESP auf beliebigen Kombinationen von Clients und Gateways eingesetzt werden.
Wie bei allen Verschlüsselungsprodukten kann der Einsatz von ESP in gewissen Ländern Einschränkungen unterliegen.

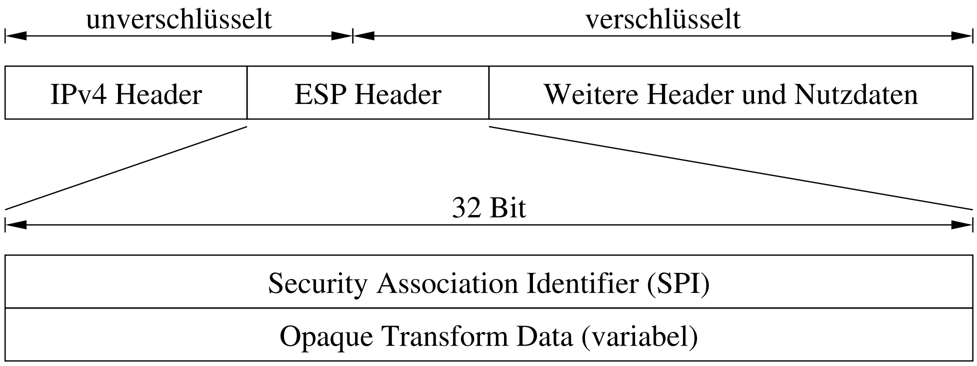

Abbildung 53: Encapsulating Security Payload Header von IPSEC

AH und ESP können auf verschiedene Weise kombiniert oder auch einzeln verwendet werden. Keine der beiden Methoden gewährt jedoch Schutz vor bestimmten Angriffen durch Datenstromanalysen. Außerdem werden in den RFCs keine Vorschriften bzgl. des Schlüsselaustausches gemacht.

5.11.3 Security Association (SA)

Sowohl für AH als auch für ESP wird die so genannte Security Association (SA, siehe RFC 1825 [Atki 95a]) zwingend benötigt. Die SA beinhaltet die verschiedene Konfigurationsparameter der zu sichernden Verbindung. Diese Parameter müssen auf beiden Endpunkten der Verbindung gleich konfiguriert sein und variieren je nach Art der verwendeten Sicherungsmechanismen und der konkreten Implementierung, da einige Parameter optional sind.

Folgende Parameter werden für den Einsatz von AH bzw. ESP von RFC 1825 zwingend vorgeschrieben:

- Verwendeter Authentisierungsalgorithmus und ggf. Mode für AH
- Verwendete(r) Schlüssel für den Authentisierungsalgorithmus bei AH
- Verschlüsselungsalgorithmus und verwendeter Mode für ESP
- Verwendete(r) Schlüssel für den Verschlüsselungsalgorithmus bei ESP
- Parameter für die optionale Verwendung der Kryptographischen Synchronisation und des Initialisierungsvektors für den Verschlüsselungsalgorithmus (ESP)

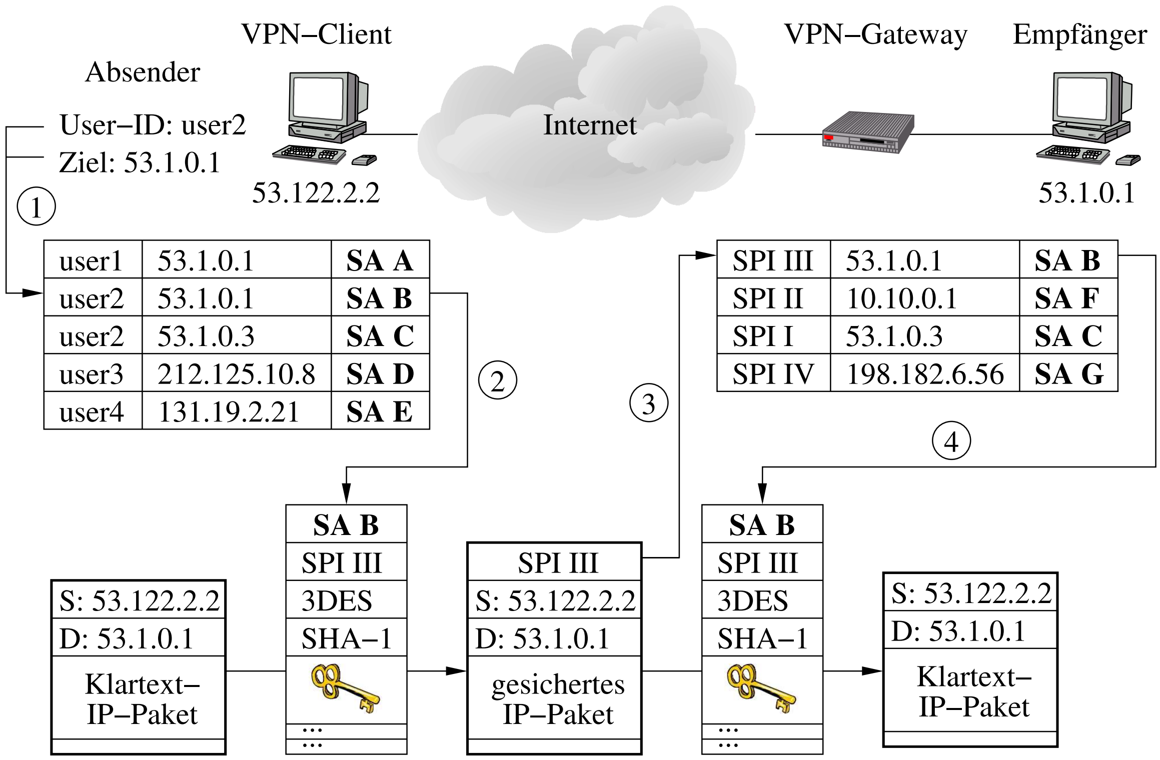

Abbildung 54: Auswahl der Security Association

Abbildung 54 stellt beispielhaft die Funktionsweise der SA beim Datenaustausch von einem Absender auf einem IPSEC-fähigen VPN-Client zu einem nicht-IPSEC-fähigen Gerät hinter einem IPSEC-VPN-Gateway dar.

Die SA, welche zur Verschlüsselung bzw. der Prüfsummenbildung für die vom IPSEC-Client `53.122.2.2` ausgehenden Pakete verwendet werden muß, wird über die Ziel-IP-Adresse `53.1.0.1` und über die User-ID `user2` des Prozesses, welcher das Paket generiert hat,

eindeutig identifiziert (Schritt 1). Bei einer rein Rechner-basierten Verbindungssicherung wird für alle User-IDs für eine bestimmte Ziel-IP-Adresse immer dieselbe SA ausgewählt, die SA-Unterscheidung basiert dann nur mehr auf der Ziel-IP-Adresse.
Das zu versendende Paket wird nun mit den in der SA (SA B) angegebenen Verfahren gesichert, sowohl bei AH als auch bei ESP wird in den entprechenden Header der so genannte Security Parameter Index (SPI) eingetragen (Schritt 2).
Das Paket wird nun über das öffentliche Internet zum VPN-Gateway geroutet. Dieses bestimmt über den im Paket enthaltenen SPI und über die Ziel-IP-Adresse eindeutig die passende SA (Schritt 3) und damit gleichzeitig die für die Entschlüsselung und Verifizierung zu verwendenden Algorithmen (Schritt 4). Das Klartext-Paket kann nun im sicheren Netz an den Empfänger mit der IP-Adresse `53.1.0.1` zugestellt werden.
Für eine bidirektionale Verbindung gibt es immer zwei SAs, eine für jede Richtung. Auch haben Verbindungen von einem Gerät zu unterschiedlichen Zielen immer auch unterschiedliche SAs, bei Unterscheidung zwischen den User-IDs kommen zusätzlich verschiedene SAs pro User dazu.

5.11.4 SA-Synchronisation und Schlüsselaustausch

In den bisherigen IPSEC-Spezifikationen wurden keine Aussage darüber gemacht, wie die Kommunikationspartner die gemeinsamen SA-Sicherheitsparameter (Schlüssel, verwendete Algorithmen usw.) auszuhandeln haben. Eine von vielen Produkten gebotene Möglichkeit besteht darin, alle Sicherheitsparameter manuell zu konfigurieren.
Über spezielle Protokolle ist aber auch die automatische Konfiguration der Parameter möglich. Die beteiligten Rechner teilen sich dabei alle unterstützten Algorithmen und Parameter (z.B. DES, 3DES, AES; MD5, SHA-1; Unterstützung für eine CA usw.) mit und einigen sich dann auf von beiden Systemen unterstützte Standards.

Mit dem **Internet Security Association and Key Management Protocol (ISAKMP)** ist ein Vorschlag von der Internet Engineering Task Force (IETF) zur Umsetzung dieser Funktion vorhanden. Über ISAKMP kann vor der eigentlichen Datenübertragung eine gemeinsame SA ausgehandelt werden, außerdem beinhaltet es Mechanismen zum Schlüsselmanagement und zur Authentisierung.

ISAKMP macht zwar Vorgaben für die Verwaltung, Erzeugung, Änderung und Löschung von SAs, die Generierung und der Austausch der Schlüssel muß jedoch von anderen Protokollen erledigt werden. Dafür steht mit **Oakley** wiederum ein zu ISAKMP passendes Protokoll zur Verfügung, welches für die Schlüsselvereinbarung den Diffie-Hellman-Algorithmus verwendet.

Ein weiteres Protokoll für Schlüsselaustausch und -management, **SKEME**, ist unter [Kraw 95] beschrieben. Es unterstützt den Schlüsselaustausch über Asymmetrische Verschlüsselungsverfahren sowie die Benutzung von CAs und die manuelle Schlüsselinstallation.

Das **Simple Key Management for Internet Protocols (SKIP)** ist eine weniger verbreitete Alternative zur Kombination ISAKMP/Oakley. Es wurde von Ashar Aziz und Whitfield Diffie bei Sun Microsystems entwickelt. ISAKMP/Oakley ist das leistungsfähigere Protokoll, SKIP allerdings einfacher zu handhaben. SKIP verwendet zum Schlüsselaustausch ebenfalls eine Variante des Diffie-Hellman-Algorithmus. Allerdings wird nicht für jede Verbindung ein neuer Schlüssel ausgehandelt, sondern jedes Paket mit einem eigenen Schlüssel chiffriert, der dem Paket in verschlüsselter Form angehängt wird. Die Chiffrierung dieses Schlüssel erfolgt wiederum mit einem Langzeitschlüssel, der vor Beginn der Kommunikation von den Partnern ausgehandelt worden ist.

Internet Key Exchange Protocol (IKE)

IKE ist eine Kombination aus ISAKMP/Oakley/SKEME und wird in RFC 2409 beschrieben. Das hybride IKE-Protokoll unterstützt allerdings nicht alle Funktionen von Oakley und SKEME. IKE wird von vielen gängigen Verschlüsselungsprodukten in Verbindung mit IPSEC eingesetzt.

Für die Aushandlung des Secret Keys für die symmetrische Datenverschlüsselung bietet IKE zwei Möglichkeiten:

- **Manueller Schlüsselaustausch**: Der geheime Schlüssel wird auf einem sicheren Weg (Post, Telefon, PGP-E-Mail usw.) ausgetauscht und dann manuell in die beiden für die IPSEC-Kommunikation vorgesehenen Geräten eingetragen.
- **Automatische Schlüsselaushandlung** über die Verwendung von Asymmetrischer Verschlüsselung.

Während die prinzipielle Problematik des manuellen Schlüsselaustausches schon in Kapitel 5.3 beschrieben wurde bietet die automatische Schlüsselaushandlung noch einen weiteren Vorteil: Der symmetrische Schlüssel kann ohne weiteren Aufwand periodisch neu ausgehandelt werden. Kommt ein Angreifer z.B. über eine Brute Force-Attacke in den Besitz des Secret Keys kann er die Kommunikation so nur lange mithören, bis ein neuer Secret Key ausgehandelt wird. Die Gültigkeitsdauer des Secret Keys wird von der Lebensdauer der beiden IPSEC SAs (eine SA pro Kommunikationsrichtung) bestimmt. Diese liegt typischerweise im Stundenbereich und wird oft als **IPSEC SA Lifetime** bezeichnet.

Während ein manueller Schlüsselaustausch auch gleich die Authentisierung beinhaltet muß der automatischen Schlüsselaushandlung eine Authentifizierung über einen anderen Mechanismus vorangehen. Dafür bietet IKE wiederum zwei Möglichkeiten:

- **Shared Secret**: Das Shared Secret (manchmal auch Pre-Shared Secret) ist ähnlich wie ein gewöhnliches Passwort eine geheime Zeichefolge, auf die sich die Administratoren der beiden VPN-Geräte verständigen und die sie in die Geräte eintragen.
  Das Shared Secret hat ähnliche Nachteile wie der manuelle Austausch des Secret Keys. Da das Shared Secret jedoch nicht für die Verschlüsselung, sondern nur für

die Authentifizierung der Geräte verwendet wird entschärft sich die Problematik etwas. Einmal kann mit dem Shared Secret der Verkehr nicht entschlüsselt werden. Der Secret Key kann weiterhin periodisch geändert werden. Zweitens gehören zur Authentifizierung von Geräten oft noch andere Parameter wie IP-Adressen und/oder DNS-Namen, die zumindest eine schwache Authentifizierung beinhalten. Ein Shared Secret ist einfach zu konfigurieren, jedoch vom Sicherheitsstandpunkt her zu vermeiden.

- **Signatur und Zertifikat**: Die sicherste Methode der Authentifizierung ist die Verwendung einer Digitalen Signatur in Verbindung mit einem Zertifikat einer CA. Der Aufwand dafür ist allerdings um einiges größer als die Verwendung der beiden anderen Methoden.

Ähnlich wie bei der IPSEC SA müssen auch für IKE-Paramater periodisch neu ausgetauscht werden. Die Gültigkeit dieser Parameter wird als **ISAKMP SA Lifetime** oder auch **IKE SA Lifetime** bezeichnet. Auch diese liegt in der Regel im Bereich von einer bis mehreren Stunden.

Die Aushandlung der ISAKMP SA geschieht in der sogenannten **Phase one** von IKE, darauf aufbauend wird in der **Phase two** die IPSEC SA ausgehandelt.

Für die Aushandlung der Parameter benutzt IKE standardmäßig den UDP-Port 500.

5.12 IPSEC und FreeS/WAN

FreeS/WAN ist eine freie IPSEC-Implementierung unter Linux für IPv4. Sie unterliegt **nicht** den US-Exportbestimmungen für Verschlüsselungssoftware, da die Entwickler streng darauf achten, daß nur Quellcode von nicht-US-Bürgern ins Paket einfließt. Aus Sicherheitsgründen wurde mittlerweile auch einige von den RFCs geforderten Funktionen deaktiviert, insbesondere die DES-Unterstützung. Als z.Z. einziger Verschlüsselungsalgorithmus wird 3DES verwendet, dazukommen soll die AES-Verschlüsselung. Als Authentisierungsalgorithmen werden Keyed MD5 und Keyed SHA-1 unterstützt.

FreeS/WAN besteht aus zwei Hauptteilen:

- **Kernel IP Security (KLIPS)** implementiert AH und ESP im Linux-Kernel.
- **Pluto** ist ein Daemon, welcher über IKE die SA-Parameter und die Schlüssel aushandelt.

Außerdem stehen verschiedene Scripten und Programme zur Administration der Pakete zur Verfügung.

Nach der Installation der SuSE 8.0-Pakete `freeswan-1.95_0.9.8-69` und `km_freeswan-1.95_0.9.8-69` mit `Yast` kann FreeS/WAN gestartet und dessen Initial-Status abgefragt werden:

```
linux:~ # /etc/init.d/ipsec start
ipsec_setup: Starting FreeS/WAN IPsec 1.95...
ipsec_setup:                                           done
linux:~ # ipsec whack --status
000 interface ipsec0/eth0 10.50.187.55
000
000
linux:~ #
```

FreeS/WAN beinhaltet eine ganze Reihe von Funktionen und Konfigurationsmöglichkeiten für verschiedenste Netzwerktopologien. Wir werden uns hier auf die Beschreibung eines Gateway-Gateway-VPN anhand der Beispiel-Topologie aus Abbildung 55 beschränken.
Vor der Konfiguration einer VPN-Verschlüsselung sollten alle benötigten Dienste vorher ohne Verschlüsselung konfiguriert und (mit harmlosen Daten) getestet werden. Dies ist unbedingt ratsam, da aufgrund der Komplexität der IPSEC-Konfiguration alle anderen Fehlerquellen soweit wie möglich beseitigt sein sollten.

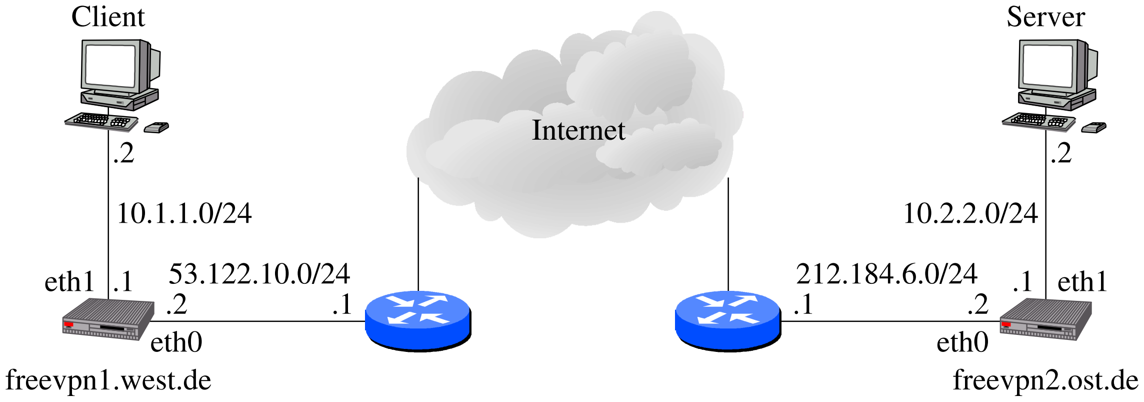

Abbildung 55: FreeS/WAN-Konfigurationsbeispiel

Der Client mit der IP-Adresse `10.1.1.2` möchte über die beiden VPN-Gateways und das Internet mit dem Server `10.2.2.1` kommunizieren. Die Default-Routen der beiden Endgeräte zeigen in unserem einfachen Beispiel auf die internen IP-Adressen der Gateways, diese wiederum benötigen eine Default-Route zum Internet. Auf den Gateways muß zusätzlich das Routing für fremde Pakete im Kernel aktiviert sein. Da beide internen Netze mit privaten IP-Adressen arbeiten müßte für einen Verbindungstest ohne VPN an den Gateways NAT eingerichtet sein, für die VPN-Verbindung wird dies nicht benötigt.
Vor dem Verbindungsaufbau müssen sich die beiden Gateways authentisieren. Dies geschieht über IKE mit den beiden bekannten Methoden, Shared Secret-Authentisierung oder über Public und Private Key mit RSA. Das Shared Secret bzw. der Public und Private Key des Gateways werden in der Datei `/etc/ipsec.secrets` abgelegt, der Public Key

des Kommunikationspartners in `/etc/ipsec.conf`[35].
In unserem Beispiel betrachten wir nur die RSA-Authentisierung von IKE. Dafür muß ein Public/Private Key-Paar fürs Gateway generiert werden. Bei SuSE wurde dies schon während der Installation von FreeS/WAN erledigt. Mit

```
freevpn1:~ # ipsec newhostkey > /etc/ipsec.secrets
getting 128 random bytes from /dev/random...
looking for a prime starting there (can take a while)...
found it after 170 tries.
getting 128 random bytes from /dev/random...
looking for a prime starting there (can take a while)...
found it after 184 tries.
swapping primes so p is the larger...
computing modulus...
computing lcm(p-1, q-1)...
computing d...
computing exp1, exp1, coeff...
output...

freevpn1:~ # chmod 600 /etc/ipsec.secrets
freevpn1:~ #
```

kann jederzeit ein neues Schlüsselpaar generiert werden. Das alte Schlüsselpaar wird dabei allerdings überschrieben.
FreeS/WAN benutzt für die Unterscheidung der Netze, welche miteinander kommunizieren sollen, die Bezeichnungen `left` und `right`, die Zuordnung kann beliebig sein. In unserem Beispiel ist das Netz `10.1.1.0/24` `left`, das Netz `10.2.2.0/24` ist `right`. Die Bezeichnung der Netze ist damit auf beiden VPN-Gateways gleich[36]. In unserem Beispiel entstehen sogar identische Konfigurationsdateien `/etc/ipsec.conf` (Abbildung 56) für beide VPN-Gateways.
Nun folgt der Austausch der Public Keys. Dazu kann mit dem Kommando `showhostkey` der eigene Public Key aus `/etc/ipsec.secrets` angezeigt werden:

```
freevpn1:~ # ipsec showhostkey --left
        # RSA 2048 bits   freevpn1   Fri Jul  5 15:22:34 2002
leftrsasigkey=0sAQO1Z8FCXzhuup5h312zdFKuNmTJ/3JusVniLA2cGPO56uX65g464iVTM
bCMQ6T/o/QDHhrxLVTgCn5Zft9+jHfMVlX12OuoO9G3cv6yAvXSnBhuyiVbB43FfvVH89DMv2
p3rNYqMyZom3HipDas6AljvHVX/r5DQ2aeOIB8rqrTERsudD7rdZ40DFgoBSzVLP0zELAE6IO
```

[35] Altenativ kann dieser Public Key auch aus dem Secure DNS, eine um Authentisierungsdienste erweiterte DNS-Variante, bezogen werden.

[36] Auf `freevpn1.west.de` ist `left` also das lokale interne, `right` das entfernte Netz, für `freevpn2.ost.de` ist es genau umgekehrt. Deshalb werwendet FreeS/WAN für die Unterscheidung der Netze auch nicht "local" und "remote".

```
4NgZeEqRvY3lThgtdIA53rjUjGo2xHNP6SpTaBtsp/qW/XFb0J8MF2WJ2TReV2p5syM00WVxC
x5Capap2k5W8DGaSzzMqxT32G3NG8h8HYHMtFfg3/YK1gpEF0gplUOQWFhc2fquC/Y3x
freevpn1:~ #
```

Die Ausgaben der Optionen `--left` und `--right` sind prinzipiell gleich, sie unterscheiden sich nur durch die Einträge `leftrsasigkey` bzw. `rightrsasigkey` für die Übernahme in die Datei `/etc/ipsec.conf` (siehe Abbildung 56).

Dieser Public Key muß dem Kommunikationspartner über einen vertrauenswürdigen Weg mitgeteilt werden. Auf die Verteilung der Public Keys in Zertifikaten über eine CA können wir allerdings nicht eingehen. Wir gehen hier einfach davon aus, daß die Administratoren der Gateways die Möglichkeit haben, sich die Schlüssel sicher zu übergeben, z.B. über eine PGP-E-Mail oder einem persönlichen Treffen.
FreeS/WAN unterstützt eine Reihe weiterer Schlüssel-Formate, die von anderen Verschlüsselungsprodukten benötigt werden. Insbesondere können Public Keys auch aus X.509 Zertifikaten extrahiert werden.

Die gesamte Konfigurationsdatei für unser Beispiel ist in Abbildung 56. Weitere Optionen sind in den Man-Pages zu `ipsec.conf` beschrieben.

Nach `config setup` folgen die allgemeinen, nicht verbindungsspezifischen Einstellungen von FreeS/WAN.
`interfaces` gibt die Liste aller Interfaces an, über welche VPN-Verbindungen zustandekommen sollen. In unserem Beispiel ist dies das Interface, über welches die Default-Route des Gateways läuft. Statt mit `%defaultroute` könnte in unserer Topologie das Interface auch mit `interfaces=''ipsec0=eth0''` identifiziert werden.
Auf `plutoload` und `plutostart` folgt eine Liste von Verbindungen, die nach einem Reboot des Rechner automatisch geladen bzw. gestartet werden sollen. `%search` bewirkt, daß dafür die Einträge `auto` im Abschnitt `conn` ausgewertet werden.
Auf `conn name` folgen alle Einstellungen für die Verbindung namens `name`. `%default` bezeichnet dabei Standardeinstellungen für alle VPN-Verbindungen. Sie können im verbindungsspezifischen `conn`-Abschnitt überschrieben werden.
`authby=rsasig` gibt als Authentisierungsmethode RSA-Public/Private Key vor.
`auto=start` bewirkt in Verbindung mit `plutoload=%search` und `plutostart=%search` den sofortigen Neuaufbau der VPN-Verbindung nach einer Unterbrechung.
Bei `leftid` bzw. `rightid` stehen die Namen der beiden Gateways zur gegenseitigen Identifikation. Das den Namen vorangestellte `@` verhindert aus Sicherheitsgründen eine DNS-Auflösung.
`left` bzw. `right` bezeichnen die externen Interfaces der beiden VPN-Gateways, also die beiden Enden des VPN-Tunnels. `leftnexthop` und `rightnexthop` bezeichnen die Router-Adressen, über welche der VPN-Tunnel geführt werden soll. Auch diese Parameter können über den Wert `%defaultroute` an die Default Routen der Gateways gebunden werden (näheres dazu siehe `man ipsec.conf`). `leftsubnet` und `rightsubnet` geben die Netze an, zwischen denen die Pakete über den Tunnel geführt werden sollen. Ohne die Angabe dieser beiden Parameter werden nur von den Gateways selbst kommende Pakete getunnelt. Die

Subnetze müssen nicht wie in unserem Beispiel direkt am VPN-Gateway angeschlossen sein sondern können auch über Intranet-Router erreichbar sein. Zu beachten ist, daß von der Verbindung `conn west-ost` nur der Verkehr zwischen den Netzen `10.1.1.0/24` und `10.2.2.0/24` getunnelt wird. Insbesondere wird der Verkehr zwischen den Gateways selbst nicht gesichert. Dafür muß ggf. eine eigene Verbindung (`conn`) konfiguriert werden.

Die Konfiguration erlaubt nun schon einen vollständigen VPN-Tunnel für die zu schützenden Netze über die VPN-Gateways. Alle anderen Parameter werden von den Gateways automatisch über IKE ausgehandelt.
Mit weiteren Parametern können allerdings auch Vorgaben für die von IKE auszuwählenden Protokolle und Parameter gemacht werden.
`auth` gibt vor, ob die Authentisierung über ESP (Standard) oder AH erfolgen soll. Die Gültigkeitsdauern der Schlüssel sowie der gesamten SA können mit `keylife` bzw. `ikelifetime` konfiguriert werden.
`compress=yes` aktiviert die Komprimierung vor der Verschlüsselung für die VPN-Verbindung.

Neben dem Gateway-Gateway-VPN unterstützt FreeS/WAN auch die Client-Gateway-Konfiguration ("Road Warrior" remote Access).
Eine spezielle Funktion von FreeS/WAN bezeichnen die Entwickler als "Opportunistic encryption". FreeS/WAN erlaubt damit ohne spezifische Konfiguration und ohne PKI die Verschlüsselung von beliebigen Verbindungen zwischen beliebigen Rechnern, allein aufgrund der Informationen im Secure DNS (DNSSEC), eine spezielle Version des Domain Name Service (ab BIND Version 9), die es erlaubt, in den DNS-Datenbanken auch Authentisierungsinformationen, insbesondere Public Keys, abzulegen.

```
# Allgemeine Einstellungen
config setup
  # Interface für die VPN-Verbindung
  interfaces=%defaultroute
  plutoload=%search
  plutostart=%search
# Standardeinstellungen fuer alle Verbindungen
conn %default
  authby=rsasig
  auto=start
# VPN West<->Ost
conn west-ost
  leftid=@freevpn1.west.de
  leftrsasigkey=0sAQO1Z8FCXzhuup5h312zdFKuNmTJ/3JusVniLA2cGPO56uX65g464iVTM
  bCMQ6T/o/QDHhrxLVTgCn5Zft9+jHfMVlX12OuoO9G3cv6yAvXSnBhuyiVbB43FfvVH89DMv2
  p3rNYqMyZom3HipDas6AljvHVX/r5DQ2aeOIB8rqrTERsudD7rdZ40DFgoBSzVLP0zELAE6IO
  4NgZeEqRvY3lThgtdIA53rjUjGo2xHNP6SpTaBtsp/qW/XFb0J8MF2WJ2TReV2p5syMOOWVxC
  x5Capap2k5W8DGaSzzMqxT32G3NG8h8HYHMtFfg3/YK1gpEF0gplUOQWFhc2fquC/Y3x
  rightid=@freevpn2.ost.de
  rightrsasigkey=0sAQOmalxxvEVM4b9JoYAnRTn/oGF9uYGIP/tiqQALL2t4selJZNOlYvY3
  l3O3OmKvGCGxS+ylBTyvKp4MN92hRLTraykOwzJ5Cy/9G3SFkzHt2zbLJdcbPIM7ygFwlUR5h
  1ejRrtcKGg1w847Oc5vaIxZpO8JbKFqL+1nMKatQ9lP7plAxFqomd9YRJxpZJEptNoZOduOHU
  /LN0UuZIlFexYPBHzq8fZeqBYNghk6mk6D8IGqCaBk7/cS86AQ1Bl6xoII0sIVoAvZhGtVgDZ
  JRD8P+syi5KpZseRxHHbz6K+98L3WnDGUmHadrrZNGW7dijKpxVt01SRZ/RKVinBckiPv
  left=53.122.10.2
  leftnexthop=53.122.10.1
  leftsubnet=10.1.1.0/24
  right=212.184.6.2
  rightnexthop=212.184.6.1
  rightsubnet=10.2.2.0/24
```

Abbildung 56: Konfigurationsdatei `/etc/ipsec.conf`

5.13 Praktische Aufgaben

5.13.1 FreeS/WAN

- Deaktivieren Sie den Firewall.
- Installieren Sie FreeS/WAN mit `YaST` FreeS/WAN auf ihrem Rechner und starten Sie den IKE-Daemon.
- Verschlüsseln Sie den gesamten Verkehr zwischen Ihrem Rechner und dem jeweiligen Partner-Rechner (`pcsec1` zu `pcsec2`, `pcsec3` zu `pcsec4` usw.).