

Sommersemester 2002

**Konzepte von Betriebssystem-
Komponenten: Schwerpunkt
Sicherheit (KVBK)**

Vortrag zum Thema:

„Symmetrische Verschlüsselung (DES, 3DES, AES) und
Schlüsselaustausch (Diffie-Hellman)“

Referent:

Florian Dachlauer
Mat.Nr. 1913556

Gliederung:

- 1 Symmetrische Verschlüsselungsverfahren
 - 1.1 Grundlagen
 - 1.1.1 Terminologie, Definitionen
 - 1.2 Grundlegende Anforderungen und Probleme
 - 1.2.1 Diskussion um Open-Source
 - 1.2.2 Anforderungen an kryptographische Verfahren
 - 1.2.2.1 Angriffsmöglichkeiten
 - 1.2.3 Schlüssel und Schlüsselaustausch
 - 1.2.3.1 Schlüsselraum
 - 1.2.3.2 Schlüsselaustausch
- 2. Betrachtung ausgewählter symmetrischer Verschlüsselungsverfahren
 - 2.1 One Time Pad
 - 2.2 DES
 - 2.2.1 Algorithmus
 - 2.2.2 Schnelligkeit
 - 2.2.3 Sicherheit
 - 2.3 3DES
 - 2.3.1 Algorithmus
 - 2.3.2 Schnelligkeit
 - 2.3.3 Sicherheit
 - 2.4 AES
 - 2.4.1 Entstehung
 - 2.4.2 Algorithmus
 - 2.4.3 Schnelligkeit
 - 2.4.4 Sicherheit
- 3. Ausblick auf asymmetrische Verschlüsselungsverfahren
 - 3.1 Public Key
 - 3.1.1 Kritische Betrachtung
 - 3.1.2 Diffie-Hellmann
 - 3.2 Verweis auf folgende Vorträge
- 4. Quellenangabe

- 1 Symmetrische Verschlüsselungsverfahren
- 1.1 Grundlagen
- 1.1.1 Terminologie, Definitionen

Zum besseren Verständnis der Zusammenhänge und zur präzisen Formulierung sind vor dem eigentlichen Einstieg in das Thema der symmetrischen Verschlüsselungsverfahren einige Begriffe zu erläutern.

Die **Kryptographie** beschäftigt sich mit der Sicherung von Nachrichten, indem diese verschlüsselt werden, während sich die **Kryptoanalyse** mit dem Reproduzieren der ursprünglichen Nachricht ohne Kenntnis des **Schlüssels** befasst. Die **Kryptologie** ist der allgemeine Begriff für die Lehre der Ver- und Entschlüsselung.

Schlüssel sind Zeichenketten, mit deren Hilfe nach vordefinierten Regeln **plaintext** (Klartext, die unverschlüsselte Nachricht) zu **ciphertext** (Chiffretext, die verschlüsselte Nachricht) verschlüsselt (**encryption**) bzw. aus ciphertext der plaintext wiedergewonnen werden kann (**decryption**).

Die zu übertragenden Nachrichten sollten die Anforderungen der **Geheimhaltung**, der **Integrität** und der **Verbindlichkeit** erfüllen.

Unter der Geheimhaltung versteht man, dass der Inhalt einer Nachricht Unbefugten verborgen bleibt. Ebenso darf eine zu übertragende Nachricht nicht auf ihrem Weg zum gewünschten Ziel verändert werden, wobei man hierbei von Integrität spricht. Der Sender der Nachricht wiederum sollte eindeutig bestimmbar sein, das Absenden der Nachricht also sollte das Kriterium der Verbindlichkeit erfüllen.

Kryptographische Algorithmen sind Berechnungsvorschriften zur Ver- und Entschlüsselung, weiter oben noch „Regeln“ genannt.

Bei **symmetrischen** Verfahren wird sowohl zur Ver- als auch zur Entschlüsselung derselbe Schlüssel verwendet, **asymmetrische** Verfahren benutzen verschiedene Schlüssel für encryption und decryption.

- 1.2 Grundlegende Anforderungen und Probleme
- 1.2.1 Diskussion um Open-Source

Zur Fragestellung, ob man den Quelltext eines Kryptographischen Algorithmus geheim halten oder jedem Interessierten öffentlich zugänglich machen soll, gibt es nicht nur in der Kryptologie oft heftige Diskussionen.

Einige mögliche Argumente, sowohl Pro als auch Contra, werden im Folgenden angeführt und bewertet.

Zum einen spricht für die Geheimhaltung eines Algorithmus, dass schnell und einfach Plagiate angefertigt werden könnten, was bei finanziellen Interessen z.B. nicht erwünscht ist. Auch wird oft befürchtet, dass Sicherheitslücken leichter entdeckt und ausgenutzt werden könnten. Dieses Argument lässt sich natürlich leicht umdrehen, indem man sich von offenem Code eine frühere und lückenlosere Fehlererkennung erhofft.

Des weiteren muss im Falle der Geheimhaltung jeder Mitarbeiter an diesem Projekt über dessen Verschwiegenheit kontrolliert und bei Austritt aus Projekt oder Firma der Algorithmus geändert werden, was äußerst aufwendig sein kann.

Die beiden wohl treffendsten Argumente für Offenlegung des Codes sind, dass jedes

Programm, das veröffentlicht wird, von Spezialisten oder begabten Programmierern disassembliert werden kann und so der Algorithmus ohnehin öffentlich bekannt würde.

Einzigster Ausweg wäre, das Verschlüsselungs-Programm nicht zu veröffentlichen, was aber in den meisten Fällen Ziel der Entwicklung ist und somit nur eine theoretische Lösung darstellt.

Besonders wichtig, vor allem bei Verschlüsselungsalgorithmen, ist das Vertrauen der Nutzer in die Sicherheit des zu nutzenden Verfahrens. Aus diesem Grund müssen Algorithmen ständiger Kontrolle unterliegen, mögliche Angriffe versucht und evtl. ausgeschlossen werden. Hierzu sollte der Quellcode offenliegen, um höchstmögliche Kontrolle zu erlangen und Fehlerlösungen schnell und effizient zu entwickeln.

Letztlich muss sich jeder Entwickler selbst seine Meinung bilden, wie man aber sehen wird, sind die meisten anerkannten Algorithmen open-source.

1.2.2 Anforderungen an kryptographische Verfahren

1.2.2.1 Angriffsmöglichkeiten

Es gibt mehrere Angriffsmöglichkeiten auf kryptographische Verfahren.

Bei vielen Chiffren, v.a. bei Tauschchiffren, können **analytische Verfahren** Aufschluss darüber geben, welches Zeichen des Chiffretextes einen bestimmten Zeichen im Klartext codiert. Hierfür werden u.a. die Häufigkeit der Buchstaben in verschiedenen Sprachen betrachtet.

Allgemein gibt es folgende Möglichkeiten, ein Verfahren zu analysieren:

Ein Angriff, bei dem der Angreifer versucht, ohne Kenntnis des Schlüssels aus der verschlüsselten Nachricht die ursprüngliche zu rekonstruieren, nennt man **Ciphertext-only-Angriff**.

Eine weitaus erfolgsversprechendere Methode wäre der **Known-Plaintext-Angriff**.

Hier kennt der Analytiker sowohl plaintext als auch ciphertext und kann so umfangreiche Erkenntnisse über die Sicherheit des Algorithmus erlangen. Diese Angriffsmethode ist oft gewollt und dient der Überprüfung des verwendeten Verfahrens.

Beim **Chosen-Plaintext-Angriff** ist es das Ziel, zu einem Stück plaintext den zugehörigen ciphertext zu generieren. Besonders **Public-Key-Verfahren** (Erklärung folgt in Gliederungspunkt 3.1) sind anfällig für solche Analyseverfahren, da der Angreifer den öffentlichen Schlüssel zur Verfügung hat und diesen zur Verschlüsselung nutzen kann.

Analog hierzu versucht man beim **Chosen-ciphertext-Angriff** aus einem vorgegebenen Chiffretext den zugehörigen Klartext zu ermitteln.

Am häufigsten findet der **Brute-Force-Angriff** Anwendung. Es werden hierbei alle möglichen Schlüssel durchgetestet, bis der ciphertext entschlüsselt ist. Im nächsten Abschnitt wird man erkennen, wie zeitaufwendig diese Methode werden kann, wenn der **Schlüsselraum** groß genug ist.

Angriffsmethoden, die **physischer** Art sind seien hier nur am Rande erwähnt, da diese aus dem Bereich der Kryptoanalyse herausfallen.

1.2.3 Schlüssel und Schlüsselaustausch

1.2.3.1 Schlüsselraum

Wie sicher ein Schlüssel ist, d.h. wie viele Möglichkeiten es für diesen gibt und wie viel Aufwand ein möglicher Angreifer betreiben muss, um den Schlüssel zu erraten oder durch Analyse zu generieren, hängt vom gewählten **Schlüsselraum**, also der Menge aller möglichen Schlüssel, aus der ein spezifischer Schlüssel gewählt wird, ab. Da Schlüssel meist binär codierte Zeichenketten darstellen, lässt sich der Schlüsselraum einfach berechnen.

Ein 4-Bit Schlüssel wäre aus einem (2^4)-großen Schlüsselraum entnommen. Hier wäre keine große Sicherheit gegeben, da man maximal 16 mal testen muss um den richtigen Schlüssel zu finden.

Daher werden heute meist Schlüssel gewählt, die zwischen 128 und 1024 Bit lang sind. Man sieht, dass sich der Aufwand sehr schnell steigert und bei 1024 Bit schon recht hoch erscheint.

Dennoch gibt es effektive Angriffsmöglichkeiten, bei denen mit Hochleistungsrechnern oder ganzen verteilten Systemen versucht wird, trotz der großen Schlüsselräume die Sicherheit der Schlüssel zu testen.

1.2.3.2 Schlüsselaustausch

Wie man an den Ausführungen zu Schlüsselräumen erkennen kann, ist es möglich, den Aufwand des „Knackens“ eines verschlüsselten Textes im Rahmen der technischen Möglichkeiten beliebig hoch zu treiben, indem man den Schlüsselraum genügend groß wählt.

Da aber **symmetrische Verschlüsselungsverfahren** bei encryption und decryption mit demselben Schlüssel arbeiten, muss auch der Empfänger einer Nachricht den richtigen Schlüssel kennen.

Es stellt sich also das Problem, wie der Schlüssel sicher vor unbefugtem Zugriff zum Empfänger gelangt.

Ein weiteres Verschlüsseln des benötigten Schlüssels mit einem starken Verschlüsselungsverfahren hätte wenig Sinn, da auch hier wieder ein Schlüssel übertragen werden müsste.

Ebenso ist es eine unbefriedigende Lösung, den Schlüssel über ungesicherte Wege zu übertragen, da so die Gefahr groß wäre, dass ein Unbefugter den Schlüssel erfährt.

Das könnte zum Beispiel durch einen sogenannten **Man-in-the-middle-Angriff** geschehen, bei dem sich ein Angreifer zwischen die kommunizierenden Partner einschaltet und so bei ungesicherter Übertragung an den Schlüssel kommt.

Auch erscheinen **Vermittlungsstellen** (sog. **Trustcenter**) als potentiell Risiko, da sie selten Kontrollen unterliegen und dennoch diskrete Daten zugeschickt bekommen und weiterleiten.

Zusammenfassend lässt sich an diesem Punkt feststellen, dass zwar die Verschlüsselung an sich beinahe beliebig sicher werden kann, die größte Sicherheitslücke stellt der Austausch des Schlüssels zwischen den Partnern dar bzw. der Besitzer des Schlüssels selbst.

Lösungsansätze hierfür werden an geeigneten Stellen der folgenden Betrachtungen verschiedener Verschlüsselungsverfahren erläutert.

2 Betrachtung ausgewählter symmetrischer Verschlüsselungsverfahren

2.1 One Time Pad

Das One-Time-Pad (1917) stellt eine der sichersten Chiffrierverfahren dar. Die Erfinder Major J. Mauborgne und G. Vernam machen sich dabei einen „unendlich“ langen Schlüssel und Zufallszahlen zunutze. One-Time-Pad an sich stellt eine sog. Vigenère-Chiffre¹ dar.

Die Verschlüsselung erfolgt durch eine XOR-Verknüpfung jedes Nachrichtenbits mit dem jeweils dem Index entsprechenden Schlüsselbits².

Dieses Vorgehen stellt aufgrund der absolut zufälligen Codierung jedes Bits eine nahezu perfekte Art der Verschlüsselung dar, da ein möglicher Angreifer bei perfekten Zufallszahlen keine statistischen Möglichkeiten hat, auf einzelne Codierungen zu schließen. Auch die Kenntnis mehrerer Ciphertexte bietet keine Angriffsmöglichkeit, da die Schlüssellänge unendlich lang gewählt und somit jede Codierung nur in einer spezifischen Nachricht Anwendung findet.

Allerdings kann dieses Verfahren zwar in der Theorie funktionieren, in der Praxis stellen sich folgende Probleme:

Digitale Computer können keine echten Zufallszahlen erzeugen. Zwar existieren zu deren Erzeugung viele Algorithmen, die jedoch immer noch sog.

Pseudozufallszahlenfolgen generieren.

Man bedient sich zur Erzeugung von Zufallszahlen meist also keiner Berechnungsvorschriften, sondern physikalischer Prozesse wie Messen eines thermischen Rauschens oder dem radioaktiven Zerfall.

Auch der „unendlich“ lange Schlüssel stellt in der Praxis ein Hindernis dar. Es müssten lange und identische Folgen von Zufallszahlen bei beiden kommunizierenden Instanzen gespeichert werden, was ineffizient und fehleranfällig wird.

Einen möglichen Lösungsansatz lieferte hier M. Rabin. Er schafft eine globale Instanz, die kontinuierlich Zufallszahlen erzeugt und für jeden User zugänglich macht. Da dieses Versenden mit sehr hoher Frequenz geschieht, wird es für Angreifer schwer, die relevanten Bits, die ein Anwender zum Verschlüsseln seiner Nachricht verwendet hat, zu speichern.

Dem Empfänger muss also nur mitgeteilt werden, ab welchem Zeitpunkt die Bitfolge zum Verschlüsseln aus dem Strom gewählt wurde.

Wie zu erwarten war, kommt auch bei diesem Lösungsansatz Kritik auf, die Diskretion der zentralen Instanz müsste gesichert sein, man erwartet hohen Soft- und Hardwareaufwand zur Realisierung und letztlich muss immer noch eine Nachricht, nämlich die des Startzeitpunktes der Schlüsselentnahme, un- oder schwach verschlüsselt übertragen werden, womit das Schlüsselaustauschproblem weiterhin existent ist.

¹ Nach Blaise de Vigenère, basiert auf der Verschiebechiffre

² Siehe Anhang G1

2.2 DES

2.2.1 Algorithmus

Der **Data-Encryption-Standard** funktioniert allgemein folgendermaßen: Ein Klartext, geteilt in 64-Bit-Blöcke, wird durch eine **Eingangspermutation** gemischt, danach werden 16 „Runden“ durchlaufen, in denen je ein Teilschlüssel aus dem Schlüssel S generiert wird, am Ende wird eine **Schlusspermutation** durchgeführt, die invers zur Eingangspermutation ist.

In jeder **Runde** wird der 64-Bit-Block in 2 Blöcke zu je 32 Bit „L“ und „R“ geteilt. $R(i)$ ³ wird in der nächsten Runde unverändert als $L(i+1)$ übernommen, auf den jeweiligen Teilschlüssel $T(i)$ und $R(i-1)$, wird eine Funktion f angewendet. Das Ergebnis dieser Funktion wird mit $L(i-1)$ durch XOR verknüpft und bildet $R(i+1)$. Dieser Vorgang wird in jeder der 16 Runden wiederholt.

Die Funktion f erweitert in der sog. **Expansionspermutation** die 32 Bit zu 48 und XOR-verknüpft diese mit dem 48 Bit langen Teilschlüssel $T(i)$. Es folgt die **S-Box-Transformation**, bei der zuerst 6er Blöcke gebildet werden, die danach in eine **S-Box** eingegeben werden. Die **S-Boxes** bilden gemeinsam ein 32-Bit-Wort, das die **P-Box-Transformation** durchläuft.⁴

Die Teilschlüssel $T(i)$ werden gebildet, indem durch eine sog. **Schlüsselpermutation** der 64-Bit-Schlüssel S auf 56 Bit verkleinert und in zwei Hälften geteilt wird. In jeder Runde i werden die beiden Hälften der Vorgängerrunde abhängig vom Rundenindex i um ein oder 2 Bits nach links verschoben. Diese bilden in der Folgerunde die beiden Hälften. Die Kompressionspermutation bildet nun aus dem 56 Bit-Schlüssel einen 48-Bit-langen Teilschlüssel, der als Eingabe für f dient.

2.2.2 Schnelligkeit

Da DES nur einfache logische und arithmetische Funktionen benutzt, können Software-Implementierungen bis zu 1.000.000 Blöcken pro Sekunde verschlüsseln. Hardwareimplementierungen erreichen Raten um die 10 Gigabit pro Sekunde.

2.2.3 Sicherheit

Da der Algorithmus open-source veröffentlicht wurde, erfüllt er das **Kerhoffs-Prinzip**⁵. Er kann ständiger Kontrolle unterliegen und auf Sicherheit getestet werden. Weiterhin lässt sich anmerken, dass bei DES der sogenannte **Lawineneffekt** auftritt, d.h. eine minimale Änderung am Klartext hat eine große Änderung des ciphertexts zur Folge. Dieser Effekt trägt zur Sicherheit bei, da analytische Verfahren kompliziert werden und somit die decryption ohne Kenntnis des Schlüssels erschwert wird.

³ Index

⁴ Siehe Anhang G2

⁵ nach A.Kerchoffs „Die Sicherheit eines Verschlüsselungsverfahrens darf nur von der Geheimhaltung des Schlüssels abhängen, nicht von der Geheimhaltung des Algorithmus“

Bei bestimmten Schlüsseln weist DES Schwächen auf, z.B. bei Schlüsseln, in denen nur Nullen oder Einsen vorkommen und somit jeder generierte Teilschlüssel trotz Permutationen identisch ist.

Insgesamt ist der Schlüsselraum groß genug, dass ein Brute-Force-Angriff nahezu aussichtslos erscheint.

Dennoch schaffte es Distributed.net 1999 mit einem Netz aus 100.000 Computern und einem Hochleistungsrechner einen knapp 90 Byte langen Chiffretext in knapp einem Tag zu knacken. Schon Jahre vorher wurde mit viel enormerem Zeitaufwand eine DES-Verschlüsselung dechiffriert.

Heute existieren Superrechner, die im Mittel etwa 5 Tage benötigen, um einen passenden Schlüssel zu generieren.

Aus diesem Grund blieb DES nicht unbestritten und man verlangte nach einer Verbesserung.

2.3 3DES

2.3.1 Algorithmus

Eine Verbesserung erreichte man mit der Entwicklung des **Triple-DES**. Hierbei wird der DES-Algorithmus **dreimal hintereinander aber mit zwei Schlüsseln** angewendet.

Man bemerkte, dass ein mögliches „2-DES“ für **Meet-in-the-Middle-Angriffe** anfällig war, der Kryptoanalytiker müsste also nur doppelten Aufwand betreiben, um dieses Verfahren zu hintergehen, da er bei einem **Known-Plaintext-Angriff** mit verschiedenen Schlüsselpaaren nur vergleichen müsste, ob Entschlüsselung eines Chiffretextes mit zweitem Schlüssel und Verschlüsselung des Plaintextes mit erstem Schlüssel identisch sind.

Zum Algorithmus an sich ist in 2.2 nachzulesen.

2.3.2 Schnelligkeit

Software-Implementierungen sind aufgrund der mehrfachen Anwendung etwas langsamer als bei DES.

Hardwareimplementierungen können mehrere hundert Megabit pro Sekunde ver- oder entschlüsseln.

2.3.3 Sicherheit

Der Schlüsselraum hat sich durch die Einführung von 3DES verdoppelt, was zusätzliche Sicherheit bietet. Dennoch liegt der DES-Algorithmus zugrunde, der wie in 2.2.3 erwähnt mehrfach geknackt wurde. Aus diesem Grund wurde ein **Nachfolge-Standard** gesucht, der weniger anfällig sein sollte für Angriffe.

Natürlich ist nach wie vor das Schlüsselaustauschproblem existent.

2.4 AES

2.4.1 Entstehung

Die Entwicklung des **Advanced-Encryption-Standard-Algorithmus** wurde durch das NIST⁶ offen ausgeschrieben. AES sollte **lizenzfrei und offen** verfügbar sein, **variable Schlüssel- und Blocklängen** unterstützen, **Sicherheit** gegen Angriffe bieten, **mathematisch nachprüfbar** sein, möglichst **einfach** gestaltet und **effizienter als 3DES** sein.

Es meldeten sich anfangs 15 „Kandidaten“, die in der Folgezeit geprüft wurden. 1999 blieben noch 5 Algorithmen für die Endausscheidung übrig⁷.

Ausgewählt wurde der Algorithmus **Rijndael**, entwickelt von **John Daemen** und **Vincent Rijmen**.

2001 wurde AES offiziell zum „**federal information processing standard**“ erklärt. Auch die anderen Algorithmen finden heute vielfach Anwendung.

2.4.2 Algorithmus

Je nach Festsetzen der Schlüssellänge und Blockgröße durchläuft AES unterschiedlich viele Runden⁸. Zwischenergebnisse werden hier als Zustände bezeichnet.

Vor jeder Runde wird der Schlüssel mit dem jeweiligen Zustand XOR-verknüpft.

Das daraus resultierende Ergebnis dient als Eingabe für die Folgerunde.

Jede Runde (ausser der letzten, bei der die dritte Funktion wegfällt) besteht aus drei Funktionen,

- **ByteSub**
- **ShiftRow**
- **MixColumn.**

Die mathematische Betrachtung dieser Funktionen wird dem Interessierten überlassen.

Aus dem ursprünglichen Schlüssel wird durch **Schlüsselexpansion** ein Teilschlüssel mehr als Runden durchlaufen werden gebildet und die Länge des Schlüssels errechnet sich aus Teilschlüsselanzahl multipliziert mit der Blocklänge.

2.4.3 Schnelligkeit

Auf heutigen Rechnern können Geschwindigkeiten bis zu 200 Mbit/s erreicht werden. AES ist also schneller als DES und genügt somit der anfangs gestellten Forderung. Da AES leicht parallelisierbar ist, können Softwareimplementierungen sehr effizient vorgenommen werden.

⁶ „National Institute of Standards and Technology“

⁷ MARS, RC6, Rijndael, Serpent und Twofish

⁸ Siehe Anhang G3

Auch Hardwareimplementierungen sind mit AES effizient möglich, da nur XOR und Shift-Operationen verwendet werden. Die möglichen Datenraten liegen bei etwa 1 Gigabit/s.

2.4.4 Sicherheit

Wegen des relativ „jungen“ Alters des Algorithmus können zwar noch keine absolut verlässlichen Erkenntnisse vorgelegt werden, die Sicherheit von AES ist aber im Vergleich zu Konkurrenzalgorithmen sehr hoch.

Es ist „resistent“ gegen **differentielle und lineare Kryptoanalyse**, was von Entwicklern theoretisch untersucht wurde.

Auch die Konkurrenten starteten Angriffe auf Rijndael, was aber bislang erfolglos blieb.

Das Vertrauen der Anwender in AES steigt dadurch natürlich und man erwartet auch in naher Zukunft keinen vergleichbaren Standard.

Was auch durch Einführung von AES nicht gelöst werden konnte ist das **Schlüsselaustauschproblem**. Nach wie vor liegen die höchsten Sicherheitsrisiken darin.

3. Ausblick auf asymmetrische Verschlüsselungsverfahren

3.1 Public Key

3.1.1 Kritische Betrachtung

Zur Lösung dieses Problems entwickelte man das Prinzip des „Public-Key“.

Man verwendet also zwei Schlüssel, einen öffentlich bekannten und einen geheimen und persönlichen.

Mit dem öffentlichen Schlüssel können zwar Nachrichten an den Besitzer des Schlüssels verschlüsselt werden, nicht aber entschlüsselt, da man hierfür den geheimen Schlüssel benötigt.

Somit wäre das Problem des Schlüsseltausches gelöst, da jeder Einblick in den „public Key“ eines Kommunikationspartners hat.

Der geheime Schlüssel dient wie schon angemerkt zum Entschlüsseln der mit dem öffentlichen Schlüssel verschlüsselten Nachricht.

Das Schlüsselaustauschproblem ist mit diesem Verfahren also befriedigend gelöst. Dennoch bietet auch Public-Key Angriffsmöglichkeiten beispielsweise durch einen **Chosen-Plaintext-Angriff**.

Das Prinzip sei hiermit kurz angesprochen, ist jedoch Inhalt des folgenden Vortrags.

3.1.2 Diffie-Hellmann

Dieser Algorithmus, entwickelt von Whitfield Diffie und Martin Hellman, versucht das Austauschen eines geheimen Schlüssels ohne die oben angesprochenen Risiken zu lösen.

Die Kommunikationspartner vereinbaren eine feste, große Primzahl n und eine Zahl z . Diese beiden Zahlen können auf unsicheren Kanälen übertragen werden, da sie öffentlich bekannt sein dürfen.

Nun wählt Kommunikationspartner K1 eine große Zufallszahl x und generiert das Ergebnis der Rechnung

$$X := z^x \bmod n$$

Der Schlüssel (X, g, n) stellt den öffentlichen Schlüssel von K1 dar.

Kommunikationspartner K2 wählt ebenso eine große Zufallszahl y und erhält als Ergebnis

$$Y := z^y \bmod n$$

Der Schlüssel (Y, g, n) ist also der öffentliche Schlüssel von K2.

K1 errechnet den geheimen Schlüssel folgendermaßen:

$$k := Y^x \bmod n$$

K2 errechnet den geheimen Schlüssel mit

$$h := X^y \bmod n$$

Es gilt hierbei, dass

$$k = h$$

Um bei Kenntnis der Zahlen n, z, X, Y auf die geheimen Schlüssel schließen zu können, müsste man das sog. Problem des diskreten Logarithmus lösen, das äußerst aufwendig ist.

Wichtig ist bei diesem Verfahren, dass die Sicherheit von der Wahl der Zahlen n und z abhängt, wobei n eine möglichst große Primzahl sein muss und z eine möglichst große Untergruppe der ganzen Zahlen erzeugen soll:

$$z^1, z^2, \dots, z^{n-1}$$

3.2 Verweis auf folgende Vorträge

Genauere Besprechung asymmetrischer Verschlüsselungsverfahren folgen im nächsten Vortrag, hier sollte nur eine Überleitung geschaffen werden.

Schneier, Bruce: **Applied Cryptography**, Wiley, 1996

Selke, Gisbert: **Kryptographie, Verfahren, Ziele, Einsatzmöglichkeiten**, O`Reilly, 2000

Ertel, Wolfgang: **Angewandte Kryptographie**, Fachbuchverlag Leipzig, 2001

Buchmann, Johannes: **Einführung in die Kryptographie**, Springer-Verlag, 2001

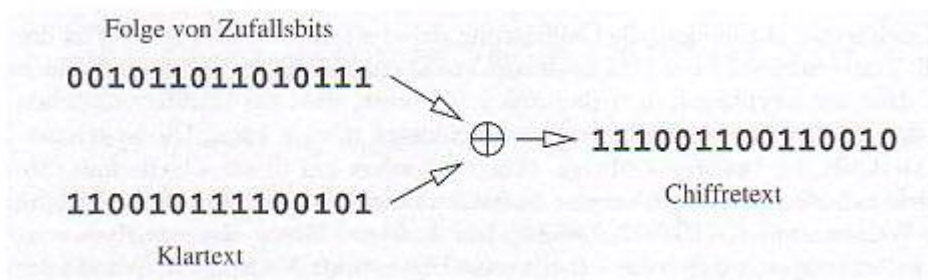
<http://www.fitug.de/ulf/krypto>

<http://www.iks-jena.de/mitarb/lutz/security/cryptfaq/>

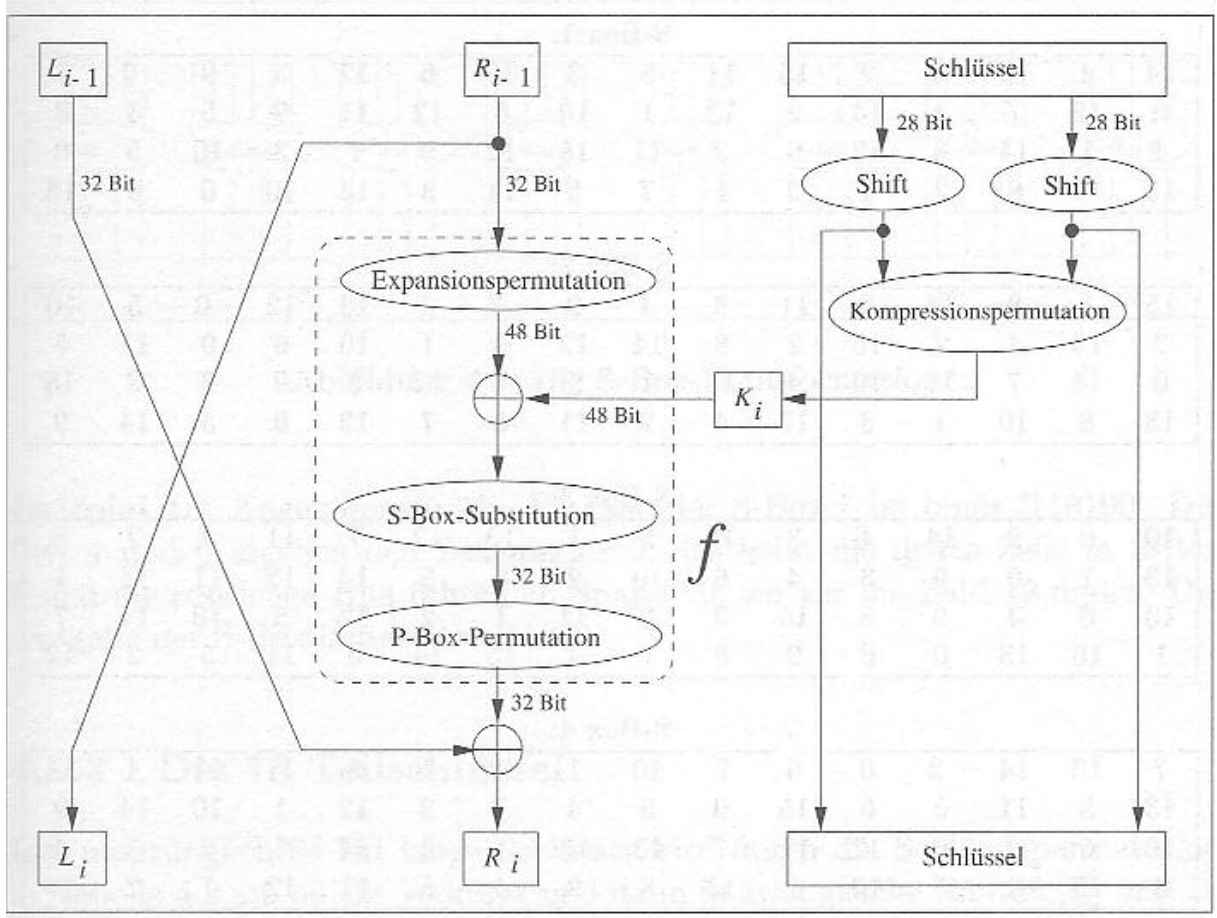
<http://www.unix-ag.uni-kl.de/~conrad/krypto/Krypto.html>

Anhang

(G1)



(G2)



(G3)

r	$b = 128$	$b = 192$	$b = 256$
$k = 128$	10	12	14
$k = 192$	12	12	14
$k = 256$	14	14	14

r = Anzahl der Runden
 k = Schlüssellänge
 b = Blockgröße

