Grundlagen der Verschlüsselung und Authentifizierung (2)

Ausarbeitung im Seminar „Konzepte von Betriebssystem-Komponenten“

Benjamin Klink

21. Juli 2010

Inhaltsverzeichnis

1 Einleitung

Allgemein geht es in dieser Arbeit um das grundsätzliche Problem, dass sich zwei Teilnehmer A und B - oder auch mit *Alice* und *Bob* bezeichnet - verschlüsselte Nachrichten schicken möchten. Hierbei kann man zwei Gattungen von Verschlüsselung unterscheiden: Die symmetrische Verschlüsselung, die genau einen Schlüssel verwendet und die asymmetrische Verschlüsselung, die mit einem Schlüsselpaar arbeitet.

Ich möchte in dieser Arbeit die **asymmetrische** Verschlüsselung beleuchten.
Mein Kollege Michael Fiedler geht im ersten Teil dieses Themas auf die symmetrische Verschlüsselung und allgemeinen Grundlagen der Kryptographie ein.

Meine Arbeit führt im folgenden Abschnitt die asymmetrische Verschlüsselung im Vergleich zur symmetrischen ein und erläutert die Kernidee, die das erst ermöglicht hat. Im Anschluss daran möchte ich einen sehr bekannten und verbreiteten Vertreter dieser Gattung — das RSA-Verfahren — vorstellen und erklären.

Der nächste große Abschnitt ist das Diffie-Hellman-Verfahren, welches einen sehr interessanten Aspekt der Public-Key-Kryptographie abdeckt:
die Schlüsselvereinbarung über einen unsicheren Kanal

Der vorletzte Abschnitt ist der Erzeugung von Zufallszahlen gewidmet und soll einen kleinen Überblick der Möglichkeiten liefern um das Thema Verschlüsselung abzurunden.

Und ganz zum Schluss erwähne ich noch Probleme und Angriffsmöglichkeiten der vorgestellten Verfahren und was man dagegen tun kann.

2 Asymmetrische Verschlüsselung

Dieser Abschnitt beschäftigt sich mit den Grundlagen und der allgemeinen Funktionsweise der asymmetrischen Verschlüsselung. Dabei soll in diesem Zusammenhang auf die Bedeutung von Einwegfunktionen eingegangen werden.

Als prominentestes Beispiel der asymmetrischen Verschlüsselung werde ich anschließend das RSA-Verfahren näher erläutern und dieses anhand eines kleinen Beispiels demonstrieren.

2.1 Definition

Die asymmetrische Verschlüsselung hat im Gegensatz zur symmetrischen Verschlüsselung einen wesentlichen Unterschied:

Anstatt eines gemeinsamen Schlüssels, den beide Kommunikationspartner benötigen und der sowohl zu Ver- als auch zur Entschlüsselung verwendet wird, existiert nun ein *Schlüsselpaar*.

- Ein *öffentlicher Schlüssel* k_e der nur für die Verschlüsselung verwendet wird
- Ein zugehöriger *privater Schlüssel* k_d der wieder zum Entschlüsseln dient

Dieses auch als *„Public-Key-Kryptographie“* bekannte Konzept wurde erstmals 1976 von W. Diffie und M. Hellman vorgestellt. Es bietet folgende Vorteile:

– Der Schlüssel k_e kann, wie der Name schon sagt, öffentlich bekannt gegeben werden.
– Man benötigt also nur ein Schlüsselpaar, damit einem beliebig viele Leute per k_e verschlüsselte Nachrichten schicken können, die nur man selbst wieder entschlüsseln kann.

Der allgemeine Ablauf lässt sich wie folgt beschreiben:

Der Empfänger *Bob* erzeugt sich ein Schlüsselpaar und veröffentlicht den *öffentlichen Schlüssel* k_e. Der Sender *Alice* will nun *Bob* eine verschlüsselte Nachricht schicken und holt sich deshalb k_e. Damit verschlüsselt sie die Nachricht M und schickt das Resultat (der Chiffretext C) an *Bob*.

Dieser wendet nun die Entschlüsselung mit Hilfe seines geheimen, *privaten Schlüssels* k_d an und erhält wieder die Originalnachricht M'. (M und M' sind natürlich identisch, der Strich dient lediglich zur logischen Unterscheidung)

2.2 Einwegfunktionen

Einwegfunktionen sind *der* wesentliche Bestandteil der asymmetrischen Verschlüsselung: Sie sind auf der einen Seite dafür zuständig, dass mit ihnen eine Nachricht effizient verschlüsselt werden kann. Auf der anderen Seite aber auch, dass ein Angreifer selbst bei Kenntnis der Verschlüsselungsfunktion und des (öffentlichen) Schlüssels nur mit sehr großem Aufwand die verschlüsselte Nachricht knacken kann, was der Umkehrung der Funktion entspräche.

Kommen wir nun zu Frage: *Was sind eigentlich Einwegfunktionen?*

Unter einer Einwegfunktion versteht man eine Funktion z.B. $y = f(x)$ die in die eine Richtung einfach zu berechnen ist, aber die Berechnung ihrer Umkehrfunktion $x = f^{-1}(y)$ sehr schwer bzw. unmöglich ist.

Ist es sogar eine Einwegfunktion mit „Falltür“, dann bedeutet dies, dass unter Kenntnis einer geheimen Zusatzinformation z die Berechnung von $x = f^{-1}(y, z)$ doch wieder leicht möglich ist.

Das bedeutet für uns, dem Benutzer ist es leicht möglich die Funktion auszuwerten; ist er auch noch Besitzer der geheimen Zusatzinformation, so kann er auch wieder die Umkehrfunktion berechnen. Der Angreifer hat jedoch diese Zusatzinformation nicht und kann deshalb nur mit erheblichem Aufwand die Funktion umkehren.

Die Existenz von Einwegfunktionen ist zwar bislang nicht bewiesen, aber es existieren sehr starke Kandidaten dafür. Auf zwei davon möchte ich näher eingehen, da wir sie in den nachfolgenden Abschnitten noch benötigen werden:

- $y = x^e \bmod n$ mit $n = p \cdot q$, dem Produkt zweier Primzahlen p und q; e sei ein beliebiger Exponent

  Hierbei handelt es sich um eine Einwegfunktion mit Falltür. Als Umkehrfunktion müssten wir die Quadratwurzel modulo n berechnen, was genau so schwer wie die Faktorisierung von unserem n ist (Siehe [1]. Die Faktorisierung von großen Zahlen ist im Allgemeinen jedoch sehr schwierig. Die geheime Zusatzinformation z ist hier die konkrete Faktorisierung von n in p und q.

  Diese Einwegfunktion wird in der RSA-Verschlüsselung eingesetzt.

- $y = g^x \bmod p$, g ist eine beliebige Basis, p ist eine Primzahl

  Hierbei handelt es sich um eine normale Einwegfuntion. Die Umkehrfunktion ist der sogenannte *diskrete* Logarithmus zu der bisher kein effizientes Berechnungsverfahren bekannt ist.

  Diese Einwegfunktion wird im Diffie-Hellman-Verfahren verwendet.

2.3 RSA

Das RSA-Verfahren ist eines der bekanntesten asymmetrischen Verschlüsselungsverfahren. Es wurde 1978 von R. Rivest, A. Shamir und L. Adleman erfunden, als diese ursprünglich versuchten die *Public-Key-Kryptographie* zu widerlegen [1]. Die Hauptidee liegt in der Verwendung der Einwegfunktion $y = x^e \bmod n$ (siehe Abschnitt 2.2).

Das Verfahren besteht aus drei Teilen: der Schlüsselpaar-Erzeugung (die Vorbereitung des Ganzen), der Verschlüsselung und der Entschlüsselung

- Schlüsselpaar-Erzeugung
  - Man benötigt zwei große, zufällige Primzahlen p und q und berechnet damit $n = p \cdot q$ (von der Größenordnung her sollte $p \cdot q > 10^{200}$ sein)
  - Dann bestimmt man eine zufällige Zahl e die teilerfremd zu $\varphi(n) = (p-1)(q-1)$ ist[1], also

    $$ggT\left(\varphi(n), e\right) = 1$$

  - Berechne nun eine Zahl d, die folgende Bedingung erfüllt[2]:

    $$d \cdot e \bmod \varphi(n) = 1$$

  - $k_d = \{\text{d, n}\}$ ergibt den (streng geheimen) *privaten Schlüssel*
  - $k_e = \{\text{e, n}\}$ ergibt den *öffentlichen Schlüssel*; dieser kann nun weitergegeben werden
- Verschlüsselung
  - Die Nachricht muss in Teilblöcke $M < n$ unterteilt werden
  - Die Blöcke werden nun einzeln nach folgender Formel verschlüsselt (dazu wird der *öffentlichen Schlüssel* benötigt):

    $$C = M^e \bmod n$$

  - Die verschlüsselten Blöcke C (der Chiffretext) können nun versendet werden
- Entschlüsselung
  - Die Entschlüsselung funktioniert analog zur Verschlüsselung, nur wird hier als Parameter der *private Schlüssel* verwendet:

    $$M' = C^d \bmod n$$

  - Die einzelnen Blöcke werden wieder zur ganzen Nachricht zusammengesetzt

[1] $\varphi(n)$ bezeichnet die EULERsche φ-Funktion, siehe z.B. [3]
[2] beispielsweise mit Hilfe des *erweiterten euklidischen Algorithmus*, siehe [2]

Warum funktioniert das Verfahren?

– Nach dem Satz von FERMAT-EULER[3] gilt für M und n teilerfremd: $M^{\phi(n)} \bmod n = 1$
– $d \cdot e \bmod \varphi(n) = 1 \Longleftrightarrow \exists k : \; d \cdot e = k \cdot \varphi(n) + 1$
– Betrachten wir die Hintereinanderusführung von Ver- und Entschlüsselung:

$$M' \equiv (M^e)^d \equiv M^{e \cdot d} \equiv M^{k \cdot \varphi(n)+1} \equiv \left(M^{\varphi(n)}\right)^k \cdot M \equiv 1^k \cdot M \equiv M \pmod{n}$$

Somit erhält der Empfänger nach dem Entschlüsseln auch wirklich wieder die Originalnachricht zurück.

2.4 RSA - ein kleines Beispiel

Damit das Verfahren etwas anschaulicher wird, möchte ich es an einem kleinen Beispiel demonstrieren:

Bob möchte anderen anbieten ihm verschlüsselte Nachrichten zu schicken. Dazu muss er zuerst ein Schlüsselpaar generieren. Den öffentlichen Schlüssel kann er dann z.B. über einen *Key-Server*[3] für jeden zugreifbar machen.

Schlüsselpaar-Erzeugung:
B
Wählt zufällig $p = 11$ und $q = 13$
$\Rightarrow n = 143$, $\varphi(n) = 120$
Wählt zufällig $e = 23$ (teilerfremd zu 120)
Löst $d \cdot 23 \bmod 120 = 1$: $\Rightarrow d = 47$

$k_d = \{47, 143\}$
Veröffentlicht $k_e = \{23, 143\}$

Alice will Bob nun ein verschlüsselte Nachricht schicken, nämlich den Text „KvBK“. Dabei gehen wir aus, dass die ASCII-Codierung verwendet wird, d.h. „KvBK“ entspricht der Zahlenfolge „75 118 66 75“.

Verschlüsselung von „75 118 66 75“:
A
Holt sich $k_e = \{23, 143\}$

Verschlüsselt jedes Zeichen einzeln:
„K“: $75^{23} \bmod 143 = 69$
„v“: $118^{23} \bmod 143 = 105$
„B“: $66^{23} \bmod 143 = 66$
„K“: $75^{23} \bmod 143 = 69$

Versand an B

[3]Ein *Key-Server* ist in das elektronische Äquivalent eines Telefonbuches für öffentliche Schlüssel

Entschlüsselung von „69 105 66 69“:
B
Benutzt dazu seinen privaten Schlüssel $k_d = \{47, 143\}$:

$69^{47} \bmod 143 = 75 \;\hat{=}$ „K“
$105^{47} \bmod 143 = 118 \;\hat{=}$ „v“
$66^{47} \bmod 143 = 66 \;\hat{=}$ „B“
$69^{47} \bmod 143 = 75 \;\hat{=}$ „K“

Anmerkung: Die Primzahlen $p = 11$ und $q = 13$ sind für den Praxisgebrauch natürlich viel zu klein! Das n ließe sich einfach durch Ausprobiernen aller Möglichkeiten in einem Bruchteil einer Sekunde faktorisieren und dann der private Schlüssel berechnen...

Außerdem wird in der Praxis auch nicht die Nachricht selber verschlüsselt, sondern nur ein Sitzungsschlüssel; siehe Abschnitt 5.

3 Diffie-Hellman

Nehmen wir an, wir stehen vor dem Problem, dass wir über einen Kanal, den jemand abhören könnte, mit jemanden vertrauliche Informationen austauschen wollen. Der allgemeine Ansatz ist natürlich, dass wir unsere Daten verschlüsseln müssen.

Nur was ist, wenn ich mit meinem Kommuniktionspartner weder einen geheimen (symmetrischen) Schlüssel ausgemacht habe, noch einer von uns beiden einen (asymmetrischen) öffentlichen Schlüssel bereit stellt?

Gut, eine Möglichkeit wäre jetzt sich eben schnell ein Schlüsselpaar zu erzeugen... Ein leicht anderer Ansatz wurde 1976 zusammen mit dem in Abschnitt 2.1 vorgestellten „Public-Key“-Konzept von Diffie und Hellman mitgeliefert.

Zuerst einigen sich die beiden Teilnehmern eine Basis g und eine Primzahl p.
Das geschieht noch unverschlüsselt und somit öffentlich. Dann folgt folgendes:

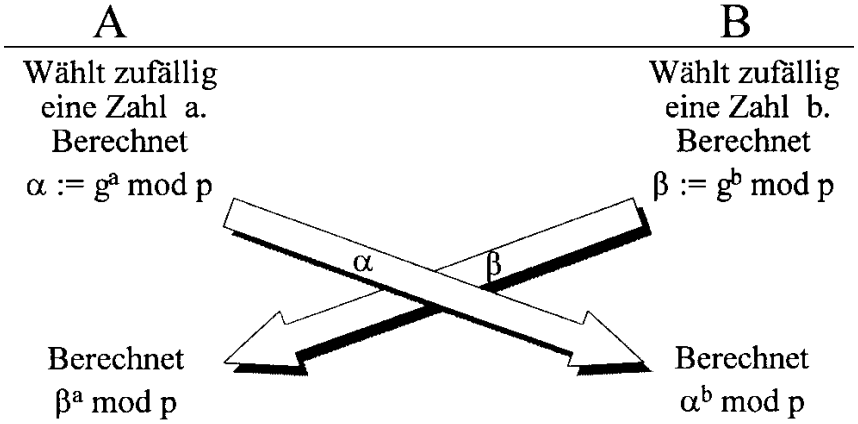

Abbildung 1: Das Diffie-Hellman-Verfahren zur Schlüsselvereinbarung (aus [1])

Die Wahl der Zufallszahlen und die Berechnungen geschehen jeweils im Geheimen. Nur die Ergebnisse der Berechnungen α und β werden (noch immer öffentlich) ausgetauscht.

Die letzte Berechnung führt dann zum Schlüssel k, der auch als „gemeinsames Geheimnis“ bezeichnet wird. Dieses k kann nun als Schlüssel für ein symmetrisches Verfahren dienen, damit sich die beiden Teilnehmer verschlüsselt unterhalten können.

Warum liefert dieses Verfahren überhaupt für beide Teilnehmer das gleiche k?

– Das wird leicht ersichtlich, wenn man sich die Berechnungen von A und B ansieht:

$$k = \beta^a \bmod p = (g^b)^a \bmod p = (g^a)^b \bmod p = \alpha^b \bmod p$$

Warum ist das Verfahren sicher, selbst wenn ein Angreifer den kompletten Datenfluss zwischen A und B belauscht?

- Es werden lediglich g, p, α und β übertragen, nicht aber die geheimen Zahlen a oder b
- $\alpha = g^a \bmod p$ ist zwar einfach zu berechnen. Der Rückweg, also a mit Hilfe des diskreten Logarithmus von α zur Basis g ist aber sehr schwierig (Eigenschaft der Einwegfunktion, siehe Abschnitt 2.2).
- Ohne a oder b kann der Schlüssel k nicht berechnet werden.

4 Zufallszahlen

Möglichkeiten zum Erzeugen von Zufallszahlen [4]:

- Erzeugung durch Hardware
  - Messung der Zeitabstände radioaktiven Zerfalls
  - jegliches analoges „Signalrauschen“ (→ z.B. Mikrophon oder Kamera)
  - usw. ...
- Erzeugung durch Software
  - Systemzeit
  - Zeitabstände zwischen Tastendruck oder Mausbewegung
  - Betriebssystem-Parameter wie z.B. Prozessorauslastung, Netzwerkverkehr, ...
- Verwendung eines Pseudo-Zufallszahlen-Generator
  - Erzeugt mit Hilfe einer mathematischen Funktion eine Folge zufällig aussehender Zahlen
  - Benötigt einen Startwert (Seed)
  - Die Schwierigkeit besteht in der Nicht-Vorhersagbarkeit der Folge

In der Praxis wird auch gerne eine Kombination verschiedener Zufallsquellen der ersten beiden Punkte genommen. Somit kann man die „Zufälligkeit“ weiter erhöhen. Dazu benötigt man noch eine geeignete Mischfunktion, um die einzelnen Resultate geschickt zu verknüpfen.

Für kryptographische Zwecke sollten nach Möglichkeit so wenig Pseudo-Zufallszahlen wie möglich eingesetzt werden, da laut Codierungstheorie die Entropie (ein Maß der „Zufälligkeit“) der gesamten Zahlenfolge nur maximal gleich der Entropie des verwendeten Startwerts ist.

Wenn sich jedoch die Verwendung eines Pseudo-Zufallszahlen-Generator nicht vermeiden lässt (da z.B. deutlich mehr Zufallszahlen benötigt werden, als die Quellen hergeben), dann nur mit einem *echt* zufälligen Startwert (generiert aus obigen Verfahren) und nur mit Hilfe eines „Kryptographisch sicheren Zufallszahlengenerators“ (zur kurzen Übersicht sei auf [5] verwiesen).

5 Schluss und Ausblick

Zum Schluss dieser Arbeit möchte ich noch einen kritischen Blick auf die vermittelten Verfahren werfen, indem ich zum einen auf einen Nachteil der asymmetrischen Verschlüsselung eingehe und zum anderen eine bislang noch nicht erwähnte Problematik der Authentifizierung ansprechen.

Zunächst zur asymmetrischen Verschlüsselung:
Ein Nachteil ist, dass die Schlüssellängen in aller Regel deutlich länger als die symmetrischer Verfahren sein müssen, um gleiche Sicherheit zu gewähren. Im Falle von RSA liegt dies darin begründet, dass die Sicherheit eng mit der Schwierigkeit der Faktorisierung des n zusammenhängt. Das Auffinden der Primfaktoren ist jedoch im Regelfall etwas einfacher als das Ausprobieren von n Schlüsseln eines symmetrischen Verfahrens.

So benötigt man für RSA eine Schlüssellänge von 3072 bit, um eine äquivalente Sicherheit zu erreichen wie der symmetrische Standard AES-128 (128 bit Schlüssellänge). Oder etwa $n = 15360$ bit, um mit AES-256 (also 256 bit) mitzuhalten [6, S. 63].

Das macht jedoch die Verwendung der asymmetrischen Verfahren sehr langsam, weswegen sie in der Praxis hauptsächlich nur zur Schlüsselvereinbarung genutzt werden:

Es wird zu Beginn der Kommunikation per asymmetrischer Verschlüsselung ein sogenannter *Sitzungsschlüssel* ausgetauscht, der dann ab dem Zeitpunkt an für die restliche Kommunikation als Schlüssel eines symmetrischen Verfahrens dient.

Nun zur allgemeinen Problematik der Authentizität:
Bei den beiden Verfahren RSA und Diffie-Hellman haben wir bisher nur gesehen, dass sie gegen Angreifer, die *mitlauschen*, sicher sind.

Was passiert aber, falls ein Angreifer direkt ins Geschehen eingreift?
Dann benötigen wir zusätzliche Werkzeuge um die Authentizität sowohl der Nachricht, als auch der Teilnehmer sicherzustellen.

Für Ersteres braucht man Dinge wie „kryptographische Hashfunktionen“, um eine sichere Prüfsumme der Nachricht zu erstellen mit der man auch absichtliche Manipulation von außen erkennen kann. Desweiteren „digitale Signaturverfahren“, um den Absender zu bestimmmen. [1],[2],[4]

Zweiteres ist deshalb so wichtig, da sonst ein Angreifer die Rolle eines Kommunikationspartners übernehmen kann und dann die „sichere“ Verbindung mit dem Angreifer aufgebaut wird – der dann natürlich alles entschlüsseln kann, was gesendet wird.

Dem interessierten Leser sei an dieser Stelle die Lektüre des ersten Teils „Grundlagen der Verschlüsselung und Authentifizierung“ von meinem Kollegen M. Fiedler empfohlen.

Literatur

[1] A. Beutelspacher: „Moderne Verfahren der Kryptographie. Von RSA zu Zero-Knowledge“, 1995, Verlag Vieweg, Braunschweig/Wiesbaden

[2] J. Buchmann: „Einführung in die Kryptographie“, 4. Auflage, 2008, Springer-Verlag, Berlin/Heidelberg

[3] I.N. Bronstein, K.A. Semendjajew, G. Musiol, H. Mühlig: „Taschenbuch der Mathematik“, 7. Auflage, 2008, Verlag Harri Deutsch, Frankfurt am Main

[4] Folien zur Vorlesung Systemsicherheit/Netzwerksicherheit, Kapitel 4-6 `www7.informatik.uni-erlangen.de/~dressler/lectures/netzwerksicherheit-ws0708/`

[5] `http://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator` letztes Zugriffsdatum: 21.07.2010

[6] NIST: „Recommendation for Key Management – Part 1“, 2007 `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf` letztes Zugriffsdatum: 21.07.2010