Digitale Signaturen - Teil 2

Asymmetrisches

von Dipl.-Inf. Robert Gehring

Asymmetrische Verschlüsselungsverfahren und kryptographische Hashfunktionen bilden das technische Fundament, auf dem digitale Signaturen aufgebaut sind. Eine nationale Zertifizierungsinfrastruktur bietet den logistischen Rahmen, das Signaturgesetz den notwendigen rechtlichen Rahmen, um den breiten Einsatz digitaler Signaturen in der Praxis von Vertragsabschlüssen zu ermöglichen. Das Zusammenspiel dieser drei Elemente zu beschreiben, ist Gegenstand des zweiten Teils unserer kleinen Reihe.

Am Anfang waren...

Für gewöhnlich werden Whitfield Diffie und Martin Hellman als die Urheber der Idee von der asymmetrischen Kryptographie im Jahr 1976 genannt. Neuere Literatur weist gleichberechtigt Ralph Merkle aus. Im Zuge der Freigabe von Geheimakten diverser Regierungen, sind noch Überraschungen zu erwarten. So kursierte vor wenigen Wochen die Meldung, daß ein ehemaliger britischer Geheimdienstangestellter bereits Anfang der 70'er Jahre auf die Public Key-Kryptographie, so lautet ein anderer Name für diese Art von Kryptographie, gestoßen sei. Die NSA (National Security Agency, ein Untergeheimdienst der CIA, weltweit größter Arbeitgeber für Mathematiker) war nach Äußerungen eines ehemaligen Direktors bereits 20 Jahre vor Diffie und Hellman auf die asymmetrische Kryptographie gestoßen. Beweise für die Wahrheit dieser Behauptung wurden allerdings nicht auf den Tisch gelegt. Da Kryptographie-Experten von einem Vorsprung von bis zu zwanzig Jahren der militärischen/geheimdienstlichen vor der zivilen/öffentlichen Forschung ausgehen, kann die Behauptung aber durchaus wahr sein.

Wie es auch immer gewesen sein mag, im Ergebnis der Beschäftigung mit den Arbeiten von Diffie und Hellman legten R. Rivest (R), A. Shamir (S) und L. Adleman (A) 1978 den ersten asymmetrischen Verschlüsselungsalgorithmus vor: RSA. Heute, 20 Jahre nach seiner Veröffentlichung und Patentierung (!), stellt RSA den de-facto-Standard für Public Key-Verschlüsselung dar. Als Vertreter der asymmetrischen Kryptographie wollen wir RSA ein wenig genauer betrachten und dabei typische Eigenschaften dieser Art Verschlüsselungstechnologie herausarbeiten.

Anmerkung: Der eine oder die andere mag sich über den scheinbar willkürlichen Gebrauch der Begriffe *asymmetrische Kryptographie* und *Public Key-Kryptographie* wundern. Zur Erläuterung kann man sagen, daß es sich um Synonyme für ein und dieselbe Art von Kryptographie handelt. Bei dieser Art kommen (mindestens) zwei unterschiedliche Schlüssel zum Einsatz. Aufgrund der besonderen Eigenschaften der Verschlüsselungsverfahren, muß nur einer der beiden Schlüssel geheim bleiben. Der andere Schlüssel kann veröffentlicht werden, ohne daß die Sicherheit des Verfahrens oder der Verschlüsselung darunter leidet. Daher bezeichnet man ihn als öffentlichen Schlüssel, englisch: „public key". Der andere Schlüssel bleibt Geheimnis des Inhabers und heißt deshalb geheimer oder privater Schlüssel (englisch: „private key").

Das Beispiel RSA

Asymmetrische Verfahren basieren auf schwer lösbaren mathematischen Problemen. In solche Probleme werden Hintertüren eingebaut, die eine Abkürzung zur Lösung in einem konkreten Fall darstellen. Das Problem, das die Entwickler von RSA aufgegriffen haben, ist die Faktorisierung. Diese klassische Aufgabe der Zahlentheorie, eine Zahl in ihre Primfaktoren zu zerlegen, ist für große Zahlen nur mit enormem Rechenaufwand zu lösen. Im Vokabular der Komplexitätstheorie heißen solche Probleme NP-vollständig. Im Falle der Faktorisierung neh-

men Mathematiker an, daß es keinen schnellen Weg gibt, an die Primfaktoren zu gelangen. Allgemeiner formuliert, ist es nicht vorhersagbar, ob eine Zahl eine Primzahl ist, da es keine Funktion gibt, um Primzahlen zu berechnen. Man kann eine Zahl nur testen, um festzustellen, ob sie eine Primzahl ist.
Andere NP-vollständige Probleme sind z.B. die Bestimmung diskreter Logarithmen oder das „Superincreasing Subset-Sum"-Problem. Letzteres ist auch unter dem Namen „Merkles Rucksack" bekannt. Kurz beschrieben, stellt es sich folgendermaßen dar: Man nehme eine Menge von Stücken mit unterschiedlichem Gewicht und einen Rucksack, der eine Teilmenge dieser Stücke aufnehmen kann. Die Aufgabe lautet, diejenige Teilmenge aus den Stücken auszuwählen, die die Aufnahmekapazität des Rucksack exakt ausfüllt. Wer es einmal ausprobiert, wird feststellen, daß die Aufgabe mit zunehmender Anzahl an Stücken erheblich an Schwierigkeit gewinnt.

RSA setzt das Problem der Faktorisierung ein (nach Schneier 1996). Vor der Ver- und Entschlüsselung müssen passende Schlüssel generiert werden: Es werden zwei große Primzahlen benötigt. Da es kein Verfahren gibt, das große Primzahlen generiert, werden zwei Zahlen gewählt und mit einem geeigneten Verfahren auf die Primzahleigenschaft geprüft. Übliche Tests sind z.B. das Verfahren von Miller-Rabin und das Verfahren von Solovay-Strassen. Beide Verfahren stellen mit einer gewissen Wahrscheinlichkeit fest, ob eine gegebene Zahl eine Primzahl ist, oder nicht. Aufgrund der Einschränkung, daß mit Wahrscheinlichkeiten operiert wird, sind derartige Verfahren deutlich schneller als Faktorisierungsverfahren.

Aus beiden Primzahlen, in der Literatur üblicherweise mit p und q bezeichnet, wird das Produkt n berechnet: n = p * q. p und q müssen unterschiedliche Zahlen sein und sollten auch in der Länge um einige Größenordnungen differieren. Die Länge von n sei dabei k Bit. Normalerweise wird k vorgegeben und p und q so gewählt, daß n die Länge k hat, z.B. 512 oder 1024 Bit. n wird zum ersten Bestandteil des öffentlichen Schlüssels. Dann berechnet man das Produkt z der Vorgänger von p und q, die sog. Eulersche *phi*-Funktion: *z = phi(n) = (p - 1) * (q* - 1). Nun werden der geheime Schlüssel und der zweite Bestandteil des öffentlichen Schlüssels gewählt. Beide müssen so gewählt werden, daß sie folgende Bedingung erfüllen: e * d = 1 mod(*z*), e hat keinen gemeinsamen Teiler mit z (teilerfremd bzw. relativ prim).
Ob dabei e oder d als privater Schlüssel verwendet werden, ist im Prinzip egal. Der andere wird dann zum Bestandteil des öffentlichen Schlüssels. In der Literatur (z.B. bei [Schneier 1996]) steht e für gewöhnlich im öffentlichen Schlüssel und d ist der private Schlüssel. p und q müssen unbedingt geheim bleiben! Wenn man ganz sicher gehen will, sollte man sie vernichten. Warum? Würde ein potentieller Angreifer p und q kennen, so könnte er sich den Faktorisierungsaufwand sparen, um die Verschlüsselung zu brechen. Anschließend hat man folgende Schlüssel erhalten:

11 Privater Schlüssel: d
Öffentlicher Schlüssel: (e, n)

Man könnte auch (d, n) als den privaten Schlüssel bezeichnen. In der Literatur findet man es allerdings so, wie oben angegeben.

Die RSA-Verschlüsselung

Der Klartext wird in Blöcke zerlegt, die kürzer als n sind. Handelt es sich beim Klartext nicht um Zahlen oder Bitmuster, so muß man diese erst entsprechend aufbereiten. (Buchstaben könnte man z.B. durch ihre Stellung im Alphabet ersetzen.) Die einzelnen Blöcke sollten gleich lang sein, wozu man ggf. Nullen voranstellt. Ein solcher Block wird dann entsprechend der folgenden Formel verschlüsselt:

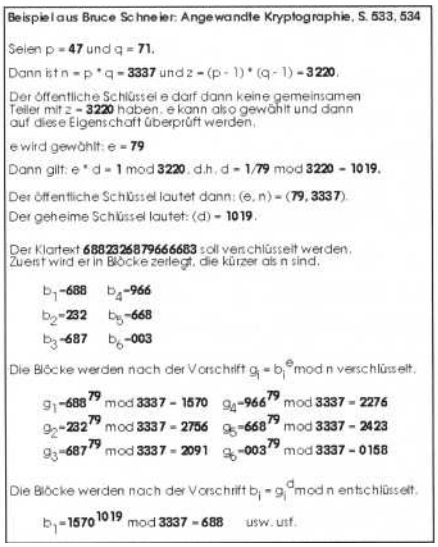
```
Geheimtextblock = (Klartextblock ^ e) mod n.
```

Die RSA-Entschlüsselung

Die Entschlüsselung erfolgt dann so:

```
Klartextblock = (Geheimtextblock ^ d) mod n.
```

Durch die Verschlüsselung mit dem öffentlichen Schlüssel ist sichergestellt, daß nur der berechtigte Empfänger, der als einziger über den geheimen (privaten) Schlüssel d verfügt, den Klartext wiederherstellen kann. Umgekehrt funktioniert es genauso.
Würden zwei Partner miteinander kommunizieren wollen und sollten sie Wert darauf legen, daß der Inhalt dieser Kommunikation geheim bliebe, so würden sie folgendermaßen handeln: Partner A generiert ein Paar RSA- Schlüssel, Partner B ebenfalls. Jeweils einen der generierten Schlüssel, den öffentlichen, übergeben sie wechselseitig. So sehen sich beide in der Lage, Nachrichten mit dem öffentlichen Schlüssel des Partners zu verschlüsseln. Nur der berechtigte Empfänger kann solche Nachrichten entschlüsseln, da er als einziger Zugriff zum passenden geheimen (privaten) Schlüssel hat.
In Abb. 1 findet sich ein durchgerechnetes, einfaches Beispiel für RSA-Verschlüsselung. Es handelt sich um eine erweiterte Version der



Beispiel aus Bruce Schneier: Angewandte Kryptographie, S. 533, 534

Seien $p = 47$ und $q = 71$.

Dann ist $n = p * q = 3337$ und $z = (p - 1) * (q - 1) = 3220$.

Der öffentliche Schlüssel e darf dann keine gemeinsamen Teiler mit $z = 3220$ haben. e kann also gewählt und dann auf diese Eigenschaft überprüft werden.

e wird gewählt: $e = 79$

Dann gilt: $e * d = 1 \bmod 3220$, d.h. $d = 1/79 \bmod 3220 = 1019$.

Der öffentliche Schlüssel lautet dann: $(e, n) = (79, 3337)$.

Der geheime Schlüssel lautet: $(d) = 1019$.

Der Klartext **688232687966683** soll verschlüsselt werden. Zuerst wird er in Blöcke zerlegt, die kürzer als n sind.

$b_1 = 688$ $b_4 = 966$

$b_2 = 232$ $b_5 = 668$

$b_3 = 687$ $b_6 = 003$

Die Blöcke werden nach der Vorschrift $g_i = b_i^e \bmod n$ verschlüsselt.

$g_1 = 688^{79} \bmod 3337 = 1570$ $g_4 = 966^{79} \bmod 3337 = 2276$

$g_2 = 232^{79} \bmod 3337 = 2756$ $g_5 = 668^{79} \bmod 3337 = 2423$

$g_3 = 687^{79} \bmod 3337 = 2091$ $g_6 = 003^{79} \bmod 3337 = 0158$

Die Blöcke werden nach der Vorschrift $b_i = g_i^d \bmod n$ entschlüsselt.

$b_1 = 1570^{1019} \bmod 3337 = 688$ usw. usf.

Abb. 1: Beispiel einer RSA-Verschlüsselung

Rechnung, die im Standardwerk [Schneier 1996] zu finden ist.

Die asymmetrische Verschlüsselung erscheint so elegant, warum wird sie nicht immer eingesetzt, wenn es um Geheimhaltung geht? Es wird Zeit, über die Nachteile zu sprechen.

Asymmetrische Verschlüsselungsverfahren sind so langsam, daß sie für die Verschlüsselung großer Datenmengen nicht in Frage kommen. Das gilt zumindest für den Fall, daß die Antwortzeiten bei (Online-)Transaktionen in einem akzeptablen Rahmen gehalten werden sollen. Geht man von der Zielstellung aus, ein digitales Pendant zur Unterschrift schaffen zu wollen, kommt es nicht auf Geheimhaltung an. Vielmehr kommt es darauf an, daß der Inhalt des unterschriebenen Dokuments nicht nachträglich manipuliert werden kann.

Man nutzt dazu die Eigenschaft der asymmetrischen Verschlüsselung aus, daß Geheimtexte nur erfolgreich entschlüsselt werden können, wenn die Daten unverändert geblieben sind. Jede Veränderung eines einzigen Bits im Geheimtext würde den Versuch der Entschlüsselung scheitern lassen. Nachträgliche Veränderungen im ursprünglichen Klartext lassen sich durch einen einfachen Vergleich mit dem Ergebnis der Entschlüsselung des Geheimtextes ebenfalls augenblicklich feststellen.

Um die zu sichernden Datenmengen klein zu halten, wird nicht das komplette Dokument verschlüsselt, sondern ein Hashwert des Dokuments. Durch das Hashing wird ein Komprimat des Klartextes, z.B. eines Vertrages, erzeugt. Dieses ist ausreichend klein, um praktisch verzögerungsfrei verschlüsselt werden zu können. Dazu können jedoch nicht beliebige Hashfunktionen verwendet werden, sondern nur spezielle: kollisionsfreie Einweg-Hashfunktionen *(collision free one-way hashfunctions).* Anderenfalls würde man einem potentiellen Angreifer das Geschäft sehr erleichtern.

Kollisionsfreie Einweg-Hashfunktionen

Ein zu signierendes elektronisches Dokument wird als Eingabe der kollisionsfreien Einweg-Hashfunktion (kurz: KEH) verwendet. Die KEH liefert als Ausgabe einen Hashwert fester Länge, meist ein paar Bytes. Diese Bytes enthalten eine eindeutige Aussage über das Dokument. Jedes im Dokument geänderte Bit führt zu einem anderen Hashwert, so daß jede Manipulation am Dokument aufgedeckt würde, wenn man den ursprünglichen Hashwert mit dem Hashwert des veränderten Dokuments vergleichen würde. Ebenso ist es praktisch unmöglich, zwei sinnvolle, aber unterschiedliche Dokumente zu finden, die denselben Hashwert haben. Diese Eigenschaft der Hashfunktionen bezeichnet man als Kollisionsfreiheit. Um sich die Bedeutung der Kollisionsfreiheit vor Augen zu führen, sollte man ein paar Gedanken über die Struktur elektronischer Dokumente anstellen.

Wenn man die Zeichen, aus denen ein Dokument besteht, klassifizieren würde, könnte man drei Kategorien unterscheiden: den eigentlichen Inhalt, die Formatierungsinformationen und „Spaces", d.h. lückenfüllende Zeichen. Wie das in der Praxis aussieht, wird an einem ApplixWord-Dokument gezeigt (siehe Abb. 2)

Aus diesen drei Kategorien sind für den Inhalt eines Vertrages nur Zeichen aus der ersten Kategorie relevant. Eine Hashfunktion, auf das Dokument angewandt, bezieht jedoch alle drei Kategorien ein. Einem potentiellen Angreifer bieten sich daher etliche Schwachstellen dar, die bei Verwendung von reinem ASCII-Text im Gegensatz zu formatiertem Text nicht vorhanden wären. Im Falle einer nur schwach kollisionsresistenten

plixWord-Dokument

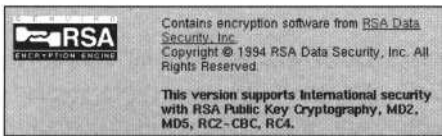


Abb.3: Auch Netscape setzt MD5-Hashfunktionen ein

Hashfunktion könnten diese, wie im Beispiel gezeigt, ausgenutzt werden. Um derartigen Angriffen vorzubeugen, besteht die Forderung nach der Kollisionsfreiheit.

Die Einweg-Eigenschaft haben Hashfunktionen, wenn man aus dem Hashwert nicht den ursprünglichen Inhalt des Dokuments errechnen kann. Einweg-Hashfunktionen sind verlustbehaftet. Die Kombination aus Einweg-Eigenschaft und Kollisionsfreiheit machen solche Hashfunktionen so wertvoll, wenn es um digitale Signaturen geht. Ein klassisches Beispiel für Hashfunktionen dieser Art ist MD5, die unter anderem auch in der Netscape Communicator-Suite eingesetzt wird.

Die Digitale Signatur

Soll ein Dokument digital signiert werden, so wird zuerst sein Hashwert berechnet. Dieser Hashwert wird mit einer asymmetrischen Verschlüsselungsmethode unter Zuhilfenahme des privaten, geheimen Schlüssels verschlüsselt. Auf diese Art und Weise hat man die Signatur erhalten, die den Inhalt des Dokuments mit der Identität des Schlüsselinhabers verknüpft. Die nächste Grafik zeigt den Ablauf.

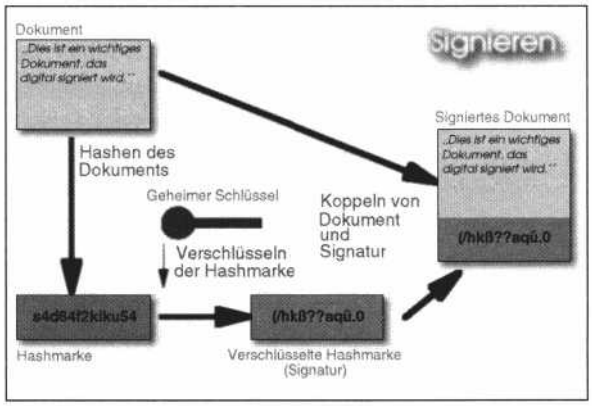


Abb. 4: Signieren eines Dokuments

Angriff auf schwach kollisionsresistente Hashfunktion

Phase 1

Man verwende zur Dokumentenerstellung ein beliebiges Formatierungsprogramm, wie z.B. eine Textverarbeitung ä la WordPerfect, um den Vertragstext zu erstellen. Um sich den Betrug zu erleichtern, sollte man von den angebotenen Formatierungsfunktionen eifrig Gebrauch machen, d.h. nicht zögerlich beim Einsatz von zwei Dutzend Fonts in mehreren Schriftgrößen sein. Auch Umrahmungen und Hervorhebungen sind geeignet. Kurz gesagt geht es darum, redundante Informationen in den Text „hineinzuschummeln". Den fertigen, ansehnlich gestalteten Text zeigt man dem Vertragspartner und läßt ihn das Dokument elektronisch signieren. Dazu wird das Dokument gehasht und der gewonnene Hashwert asymmetrisch verschlüsselt. Der Unterzeichner wähnt sich danach in Sicherheit. Eine Demonstration des Signaturtests kann diesen Eindruck noch verstärken.

Phase 2

Man ändere den Vertragsinhalt so, wie es einem paßt. Die Änderung sollte einem Außenstehenden nicht zu sehr auffallen. Für eine bestellte Pizza etwa ein paar Tausend Mark zu verlangen, wäre eine Dummheit. Anstelle von 9,- DM jedoch 11,- DM zu verlangen, dürfte -außer beim Kunden- keine besondere Verwunderung hervorrufen.

Dann hasht man das geänderte Dokument und vergleicht dessen Hashwert mit der entschlüsselten Signatur des ursprünglichen Vertrages. Dazu benötigt man bekanntlich nur den öffentlichen Schlüssel des Unterzeichners. Beim Vergleich stellt man fest, daß die Hashwerte differieren. Sollten sie gleich sein - Gratulation, die Fälschung ist geglückt. Dieser einfachste Fall ist leider ziemlich unwahrscheinlich, so daß weitere Arbeit zu verrichten ist.

Man ändere jetzt die Formatierungsinformationen bzw. die „Spaces", von denen es im Dokument nur so wimmelt, ein wenig und mache den Hashen-Vergleichen-Test erneut. So bietet es sich an, sukzessive alle Leerzeichen durch kursive Leerzeichen zu ersetzen. Oder man fügt die Steuerzeichen für „Fettdruck ein" und „Fettdruck aus" unmittelbar nacheinander ins Dokument ein. Der Phantasie sind kaum Grenzen gesetzt. Bei der Ansicht des geänderten Dokuments mit dem entsprechenden Textverarbeitungsprogramm, fallen Manipulationen dieser Art gar nicht auf. Die Bytefolgen, d.h. der tatsächliche Inhalt des Dokuments, ändern sich dabei erheblich, mithin erhält die Hashfunktion eine andere Eingabe.

Ist die Hashfunktion nicht ausreichend kollisionsresistent, wird man -bei vertretbarem Aufwand- ein Dokument generieren können, dessen Hashwert mit demjenigen des ursprünglichen Vertrages übereinstimmt. Dann kann man zu Phase 3 übergehen.

Phase 3

Man präsentiere die gefälschte Rechnung. Falls der zur Zahlung Aufgeforderte den Inhalt bestreitet, lasse man ihn das Dokument erneut signieren und die Signaturen vergleichen.Da kein Unterschied festzustellen ist, wird es für ihn schwierig sein, die Zahlung zu umgehen.

Anhand dieser Signatur kann überprüft werden, ob der Inhalt des Dokuments unverändert geblieben ist, und ob ein bestimmter Schlüsselinhaber die Signatur erzeugt hat. Dazu wird die Signatur mit dem öffentlichen Schlüssel des -vorgeblichen- Signatars [Anmerkung: Der Unterzeichner eines Vertrages -und darum geht es ja hier- heißt Signatar, laut Wörterbuch.] entschlüsselt. Heraus kommt der Hashwert des Ursprungsdokuments. Dann wird das vorliegende Dokument mit derselben Hashfunktion erneut gehasht. Der so resultierende Hashwert wird mit dem entschlüsselten Hashwert verglichen. Gibt es keine Differenzen, so kann man sicher sein, daß das Dokument unverletzt und der Besitzer des zum benutzten öffentlichen Schlüssel passenden privaten Schlüssels der Unterzeichner des Dokuments sein muß. Dieser Test wird in der folgenden Grafik gezeigt.

Noch steht allerdings die Frage nach der Identität des Schlüsselinhabers im Raum. Zwar wissen wir, daß er im Besitz des Schlüssels ist, mit dessen Hilfe das Dokument signiert wurde. Aber wer ist er? Offensichtlich benötigen wir irgendeine Art von Identitätsnachweis von ihm. Sollte der Unterzeichner greifbar sein, kann man leicht nach Ausweispapieren fragen. Schwierig wird es jedoch, wenn die fragliche Person am anderen Ende der Leitung sitzt, über die das signierte Dokument gekommen ist. Online-Pizzabestellung, Fernleihe oder Video über's Internet wären denkbare Einsatzfelder.

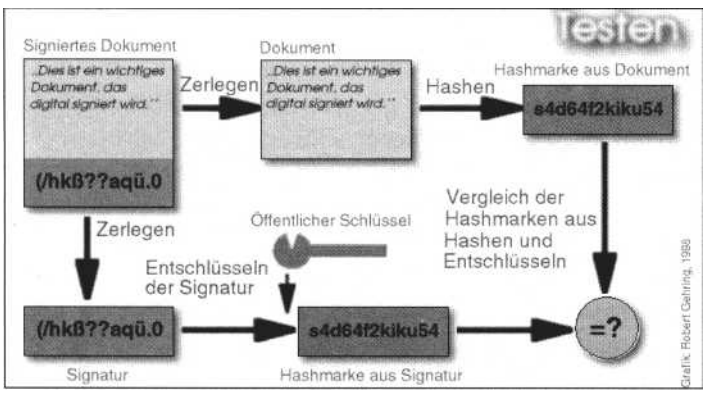


Abb. 5: Testen ob ein Dokument verändert wurde

Zertifizierung

Eine Lösung des Problems mit der Identität besteht darin, einen vertrauenswürdigen Dritten (Trusted Third Party - TTP) zu fragen, der einem die Identität bestätigen kann. Diese TTP fungiert als eine Art Makler. Sie läßt sich von einer Person den Nachweis erbringen, daß diese Person im Besitz eines privaten Schlüssels ist, zu dem sie den öffentlichen Schlüssel vorlegen muß. Dazu reicht ein einfacher Ver- und Entschlüsselungstest, bei dem beide Schlüssel zum Einsatz kommen.

Ebenfalls muß die Person einen Identitätsnachweis erbringen, z.B. indem ein Personalausweis vorgelegt wird. Wurden Schlüsselbesitz und Identität erfolgreich nachgewiesen, so stellt die TTP ein Zertifikat aus, in dem mindestens der öffentliche Schlüssel und ein Vermerk über die Identität des Inhabers aufgeführt sind.

Wer nun eine erhaltene Signatur nachprüfen will, kann sich an die TTP wenden und den öffentlichen Schlüssel des -vorgeblichen- Signatars, sowie dessen Identität erfragen. Paßt der öffentliche Schlüssel, so wurde die Signatur von der Person, die solches von sich behauptet, erstellt. Die bestellte Pizza könnte demnach auf die Reise zum Empfänger gehen. Den Ablauf von Bestellung und Rechnungslegung zeigt die nächste Grafik.

Doch was wäre, wenn der fiktive Pizzabesteller seinerseits vorspiegelt, die TTP zu sein? Er würde sich selbst eine falsche Identität bestätigen und könnte unter dieser Bestellungen vornehmen. Irgendwann würde das sicherlich bemerkt, dann wäre es mit dem Schlemmen vorbei. Aber vor der Bezahlung könnte sich der Betrüger drücken.

Um dem vorzubeugen, wird die Zertifizierungsstelle, d.h. die TTP, ihrerseits zertifiziert. In einem mehrstufigen Zertifizierungssystem sinkt die Wahrscheinlichkeit eines erfolgreichen Betrugsversuches. Nun ist es nicht notwendig, ein hierarchisches Zertifizierungssystem aufzubauen, um Sicherheit zu gewährleisten. PGP baut auf eine netzartige Struktur, das „Web of Trust", und hat damit beachtlichen Erfolg. Die „amtlichen" Vorschläge -national wie international- sehen allerdings samt und sonders Hierarchien vor (z.B. X.509). Das deutsche Signaturgesetz bildet da keine Ausnahme.

Um eine landes- oder netzweite Versorgung mit solchen Diensten zu gewährleisten, muß eine entsprechende Infrastruktur aufgebaut werden. Sie sollte ausfallsicher und gegen Angriffe (Betrugsversuche) gesichert sein. Nur wenn die Zertifizierungsinfrastruktur eine vergleichbare Sicherheit aufweist, wie die eingesetzten kryptographischen Verfahren, kann davon ausgegangen werden, daß digitale Signaturen Beweiskraft haben. Den rechtlichen Rahmen dafür soll das Signaturgesetz (SigG) schaffen.

Das Signaturgesetz

Beschlossen wurde das Signaturgesetz als Artikel 3 des Multimediagesetzes (Informations- und Kommunikationsdienstegesetz - IuKDG). Sein Ziel ist in Paragraph

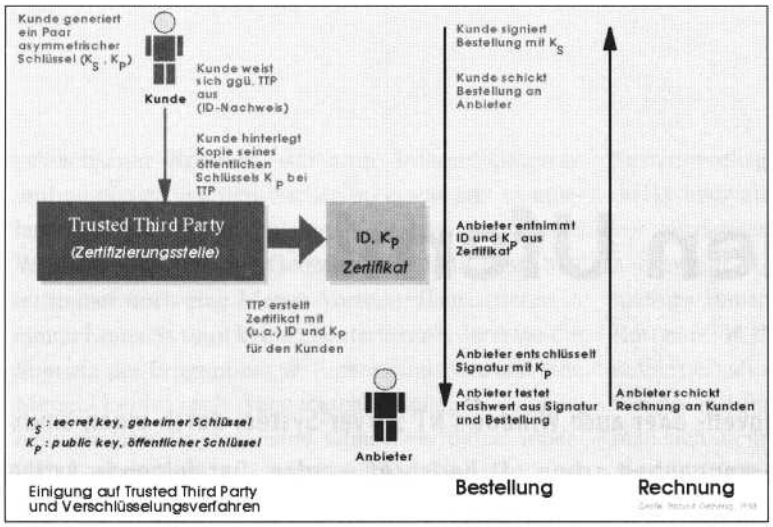


Abb. 6: Das Prinzip der Trusted Third Party

| definiert:

Zweck des Gesetzes ist es, Rahmenbedingungen, für digitale Signaturen zu schaffen, unter denen diese als sicher gelten und Fälschungen digitaler Signaturen oder Verfälschungen von signierten Daten zuverlässig festgestellt werden können (Pani Abs. 1 SigG).

Dazu wird definiert, was eine digitale Signatur ist:

Eine digitale Signatur im Sinne dieses Gesetzes ist ein mit einem privaten Signaturschlüssel erzeugtes Siegel zu digitalen Daten, das mit Hilfe eines zugehörigen öffentlichen Schlüssels, der mit einem Signaturschlüssel-Zertifikat einer Zertifizierungsstelle oder der Behörde nach §3 versehen ist, den Inhaber des Signaturschlüssels und die Unverfälschtheit der Daten erkennen läßt (Par. 2 Abs. 1).

Andere Definitionen, z.B. zu Zertifikaten und Zeitstempeln, folgen. Es schließen sich Vorschriften zur Zertifizierung, zum Inhalt von Zertifikaten und zum Umgang mit ihnen an. Die Regulierungsbehörde für Telekommunikation und Post (RegTP) wird als Wurzelinstanz eingesetzt, die privat betriebene Zertifizierungsstellen zertifiziert. Die nationale Zertifizierungsinfrastruktur für Deutschland wird demnach zweistufig ausfallen: Oben die RegTP, unten die Zertifizierungsstellen.

Weiterhin finden sich im Gesetz Datenschutzvorschriften und die Ermächtigung der Bundesregierung, weitere Einzelheiten in einer Verordnung festzulegen. Technische Einzelheiten soll die RegTP in einem Maßnahmenkatalog veröffentlichen.

Die rechtliche Regelung zur digitalen Signatur ist mithin ein dreistufiger Prozeß: Zuoberst wurde vom Bundestag das Signaturgesetz beschlossen. Dazu war eine Parlamentsmehrheit nötig. Um die Signaturverordnung (SigV) zu beschließen, kommt die jeweilige Bundesregierung zum Zuge. Der nachgeordnete Maßnahmenkatalog wird von der RegTP vorgelegt, die ihrerseits das Bundesamt für Sicherheit in der Informationstechnik (BSI) mit der Ausarbeitung beauftragt hat. Maßgabe war, internationale Regelungen so weit als möglich zu beachten.

Im Gesetz wird festgelegt, daß die Verwendung von Verfahren zur Signaturerzeugung freigestellt ist, wenn es nicht anders verlangt wird. Gegenwärtig sind Juristen dabei, bestehende gesetzliche Regelungen daraufhin zu durchforsten, ob an die Stelle einer eigenhändigen Unterschrift eine digitale Signatur -nach dem Signaturgesetz- treten kann. Um sich die Aufgabe anschaulich zu machen, sollte man sich vergegenwärtigen, daß in über 3000 Vorschriften nach der Schriftform, d.h. der eigenhändigen Unterschrift, verlangt wird. Von Fall zu Fall wird in Zukunft gleichberechtigt eine digitale Signatur möglich sein.

Bis dahin sind jedoch noch einige Schwierigkeiten zu bewältigen. Zum einen existiert noch keine Zertifizierungsstelle im Sinne des Gesetzes. Erst im Herbst ist damit zu rechnen, daß die Wurzelinstanz, d.h. die Zertifizierungsstelle der RegTP die Arbeit aufnimmt. Im nächsten Frühjahr folgen dann erste, privatwirtschaftliche Zertifizierungsstellen. Noch ist unklar, welche Kosten bei der Inanspruchnahme von Zertifizierungsdienstleistungen anfallen werden. Auch die Haftung im Schadensfall ist noch nicht geregelt. Aber das alles soll in der nächsten Folge genauer beleuchtet werden.

Zusammenfassung

In dieser Folge wurde kurz und knapp dargestellt, woher die asymmetrische Kryptographie stammt und auf welchen Prinzipien sie basiert. Es wurde gezeigt, wie durch die Kombination mit kollisionsfreien Einweg-Hashfunktionen Signaturen zu elektronischen Dokumenten erzeugt werden können. Die Frage der Identifizierung der Unterzeichner wurde angesprochen und die Zertifizierung als mögliche Antwort vorgestellt. Das Signaturgesetz mit der Signaturverordnung und dem Maßnahmenkatalog wurde als nationaler Ansatz eingeführt.

DER AUTOR

Robert Gehring hat Informatik und Philosophie an der TU Berlin studiert. Seit der Linux-Version 0.9.12 ist er mit dabei, entwickelte diverse Software und setzte dieses Betriebssystem auch ausgiebig für seine Sudien/Diplomarbeit ein. Zu erreichen ist er unter *rag@zblmath.FIZ-Karlsruhe.DE*.