

### 2.3.4 Die Digitale Unterschrift \*

**Die digitale Unterschrift basiert auf asymmetrischen Verfahren, bei denen jedem Teilnehmer zwei unterschiedliche Schlüssel zugeordnet sind: ein öffentlicher und ein privater Schlüssel. Beim Unterschreiben bedient man sich des eigenen privaten Schlüssels.**

Bei der **digitalen Unterschrift** geht es darum, dass eine Person einem Datensatz weitere Daten (also eine Folge von Bits und Bytes) hinzufügt, die als persönliche Unterschrift aufgefasst werden können.

Herr E. Mustermann möchte eine von ihm verfasste Winword-Datei per E-Mail-Anhang verschicken. Dabei soll der Empfänger sicher sein können, dass die Datei wirklich von Herrn Mustermann stammt – er möchte die Datei *unterschreiben*.

Beispiel

Wie könnte in dem Beispiel eine Unterschrift aussehen?

Eine Möglichkeit wäre es sicher, aus der handschriftlichen Unterschrift per Scanner ein Bild (z. B. im GIF-Format) zu erzeugen (siehe Abb. 2.3-5) und diese Datei am Ende des Winword-Dokuments einzufügen.

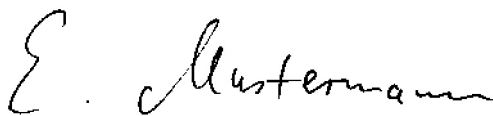


Abb. 2.3-5: Eine gescannte handschriftliche Unterschrift .

Das Problem hierbei: Jeder, der in den Besitz dieser Winword-Datei gerät, kann das GIF-Bild daraus extrahieren

und als »Unterschrift« von Herrn Mustermann unter eine beliebige andere Datei setzen. Diese Lösung muss also verworfen werden.

An digitale Unterschriften sind offenbar die folgenden Anforderungen zu stellen:

- Nur der Unterschreibende sollte die Unterschrift (also die betreffenden Bits und Bytes) herstellen können.
- Jeder soll feststellen können, ob die behauptete Unterschrift wirklich von demjenigen stammt, von dem dies behauptet wird.
- Es soll nicht möglich sein, eine echte digitale Unterschrift als Bitfolge zu kopieren und unter einen anderen Datensatz zu setzen, so dass dieser dann fälschlich auch als unterschrieben gilt.

Auf den ersten Blick macht besonders die dritte Forderung Sorgen: Wenn die digitale Unterschrift nichts als eine weitere Bitfolge ist, dann können diese Bits doch sicher kopiert und in böser Absicht unter ein anderes Dokument gesetzt werden – oder nicht? Die Lösung liegt darin, dass die digitale Unterschrift keine *feste* Bitfolge ist, sondern je nach unterschriebenem Datensatz variiert.

Wenn Sie sich nun die ersten beiden Anforderungen noch einmal vergegenwärtigen und auf der anderen Seite bereits »Asymmetrische Kryptosysteme« (S. 81) bearbeitet haben, wird Ihnen die entscheidende Idee für die Umsetzung digitaler Unterschriften sicher bereits gekommen sein:

- Der Unterschreibende verwendet seinen eigenen privaten Schlüssel, um aus dem zu unterschreibenden Datensatz die Unterschrift zu erzeugen – mit dem öffentlichen Schlüssel, der dazu passt, kann jeder die Unterschrift auf Echtheit überprüfen.

Wie in »RSA-Verfahren« (S. 88) nachzulesen, wird dieses (wie jedes asymmetrische) Verschlüsselungsverfahren angewendet, indem die Daten mit dem öffentlichen Schlüssel des Adressaten verschlüsselt werden. Dieser kann mit seinem geheimen Schlüssel wieder entschlüsseln. Beim RSA-Verfahren klappt dies auch umgekehrt (dies ist nicht bei jedem asymmetrischen Verschlüsselungsverfahren der Fall): Sie können vorliegende Daten mit Ihrem eigenen privaten Schlüssel »entschlüsseln«, obwohl sie gar nicht verschlüsselt waren – der dabei herauskommende »Datensalat« kann mit dem zugehörigen öffentlichen Schlüssel wieder in die Originaldaten verwandelt werden. Entscheidend ist: *Diesen* Datensalat können nur *Sie* erzeugt haben, insofern kann er als Ihre Unterschrift unter den Datensatz angesehen werden.

Beispiel

Wenn wie im letzten Beispiel beschrieben die »Entschlüsselung mit RSA« als digitale Unterschrift fungiert, gibt es ein grundsätzliches Problem:

- Die Unterschrift ist genauso lang wie die Originaldaten (gemeint ist: besteht als Datensatz aus der gleichen Anzahl von Bytes).

Für ein Winword-Dokument aus 100 Seiten bestünde also die Unterschrift aus noch einmal derselben Datenmenge! Da dies nicht wünschenswert ist, wendet man *vor dem Unterschreiben* auf die zu unterschreibenden Daten erst einmal eine Komprimierungsfunktion an, die man in diesem Kontext **Hashfunktion** nennt. Eine solche Hashfunktion muss selbstverständlich wiederum gewisse Anforderungen erfüllen, damit diese nicht einen Schwachpunkt bei den digitalen Unterschriften darstellt – dazu gibt es einen eigenen Baustein: »Hashfunktionen« (S. 108).

Die Erstellung einer digitalen Unterschrift ist nun in Abb. 2.3-6 beschrieben. Wie Sie sehen, nennt man das Ergebnis der Komprimierung der Originaldaten auch *Message Digest*. Dieser Datensatz ergibt, durch Anwendung des jeweils eingesetzten asymmetrischen Verfahrens mit dem privaten Schlüssel des Unterschreibenden, die digitale Unterschrift. Diese Unterschrift kann an die ursprünglichen Daten (z. B. die 100 Seiten Winword-Datei) angefügt werden.

Man beachte: Die Originaldaten können nicht aus der digitalen Unterschrift allein rekonstruiert werden – sie bilden vielmehr mit der Unterschrift *zusammen* den unterschriebenen Datensatz. Auch sind die Originaldaten zunächst einmal nicht verschlüsselt – es ist hier nur die Rede von der digitalen Unterschrift gewesen. Selbstverständlich kann man das unterschriebene Dokument *zusätzlich* noch verschlüsseln (mit irgendeinem Verschlüsselungsverfahren), jedoch hat dies nichts mit der Funktionalität der digitalen Unterschrift zu tun.

In Abb. 2.3-7 ist der Ablauf bei der Prüfung der digitalen Unterschrift dargestellt – der Prüfende muss den öffentlichen Schlüssel derjenigen Person einsetzen, die die Unterschrift angeblich erstellt hat. Da die Hashfunktion allgemein bekannt (d. h. in den entsprechenden Geräten implementiert) ist, besteht die Prüfung in dem Test, ob »Hashen« der Originaldaten und Anwendung des öffentlichen Schlüssels auf die (angebliche) Unterschrift denselben Datensatz ergeben.

Zertifikate &  
PKI

Damit dies alles funktioniert, muss natürlich sichergestellt sein, dass man sich immer die *korrekten öffentlichen Schlüssel* besorgen kann – auch hier könnte einem eine Fälschung untergeschoben werden, so dass man eine gefälschte digitale Unterschrift als echt ansieht! Für dieses Problem gibt es die Zertifikate:

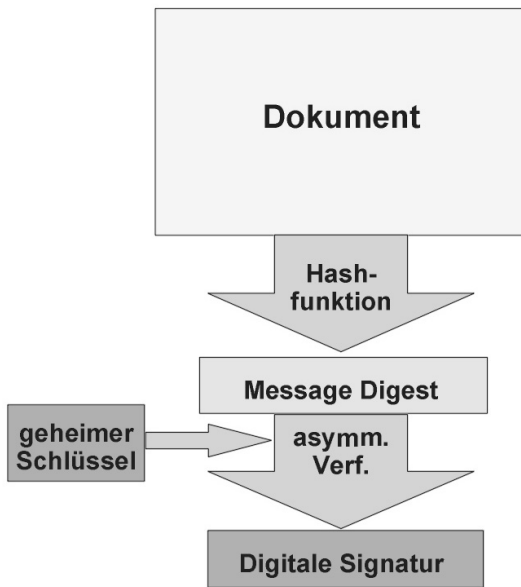


Abb. 2.3-6: So funktioniert die digitale Unterschrift.

Ein **Zertifikat** ist – verkürzt ausgedrückt – eine Art »Beglaubigung« für einen öffentlichen Schlüssel. Technisch steckt dahinter, dass eine vertrauenswürdige Instanz (z.B. der Bundesinnenminister) den öffentlichen Schlüssel einer Person A digital unterschreibt – der gesamte Datensatz aus öffentlichem Schlüssel von A, Unterschrift des Bundesinnenministers und einigen weiteren Daten ist dann das Zertifikat dieses öffentlichen Schlüssels. In »Zertifikate« (S. 137) sind der Aufbau und die Verwendung von Zertifikaten genauer erläutert.

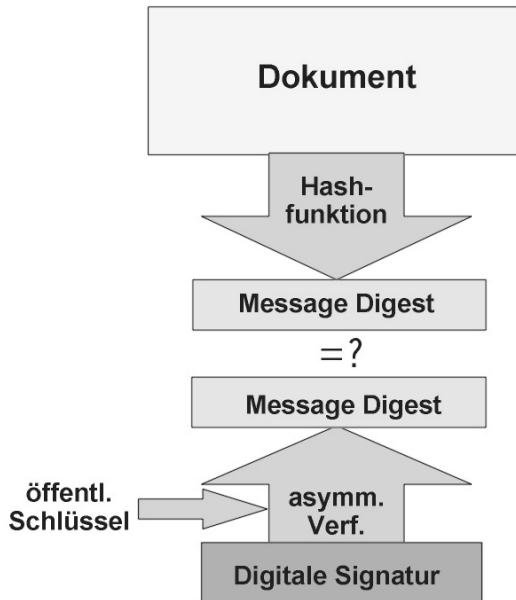


Abb. 2.3-7: Verifikation einer digitalen Unterschrift.

Der Vollständigkeit halber sollen drei weitere Begriffe erwähnt werden, die in diesem Kontext wichtig sind:

- Instanzen, die Zertifikate ausstellen, werden **Zertifizierungsstellen** genannt. Wenn man die Digitale Signatur entsprechend dem deutschen Signaturgesetz (mit höchster Sicherheitsstufe) verwenden will, muss man bei einer akkreditierten Zertifizierungsstelle ein Zertifikat beantragen.
- In der Praxis gibt es eine hierarchische Struktur von Zertifizierungsstellen, so dass ein öffentlicher Schlüssel auch von einer Instanz zertifiziert sein kann, deren öffentlicher Schlüssel wiederum von einer höheren Instanz zertifiziert ist usw. Es sind aber auch andere (nicht-hierarchische) Ge-

flechte möglich. Allgemein spricht man von einer **Public Key Infrastruktur** (kurz: PKI).

- Die privaten Schlüssel werden in der Regel auf einer Chipkarte PIN-geschützt aufbewahrt. Die Instanzen, die passende Schlüsselpaare erzeugen, die Chipkarten an die Benutzer aushändigen usw. werden **Trust Center** genannt. In der Regel agiert ein Trust Center auch als Zertifizierungsstelle.

Sollen digital unterschriebene elektronische Dokumente archiviert werden, so stellen sich einige zusätzliche Probleme – in »Archivierung« (S. 103) ist dies kurz erläutert.

Wie bereits erwähnt, kann für digitale Unterschriften beispielsweise das **RSA**-Verfahren verwendet werden. Die USA haben einen eigenen *Digital Signature Standard* (DSS) festgelegt, der einen *Digital Signature Algorithm* (**DSA**) verwendet. DSA basiert auf dem Problem des diskreten Logarithmus und wird in einem eigenen Wissensbaustein vorgestellt – siehe »DSA« (S. 112).

#### 2.3.4.1 Archivierung digitaler Dokumente \*

Zunehmend wird in der Industrie wie im öffentlichen Sektor dazu übergegangen, wichtige Dokumente in digitaler Form zu archivieren. Dies trägt der Tatsache Rechnung, dass heute die meisten Dokumente ohnehin in digitaler Form vorliegen, zudem ist eine digitale Archivierung platz- und kostensparend. Um Manipulationen zu verhindern bzw. zu erkennen, müssen wichtige Dokumente dazu digital signiert werden.

Sinn einer Archivierung ist es, auch nach längerer Zeit noch auf die entsprechenden Dokumente zugreifen zu können. So gibt es beispielsweise im deutschen Gesundheitswesen die Vorschrift, Dokumente 30 Jahre lang aufzubewahren.

Probleme  
digitaler  
Archivierung

In dieser grundsätzlichen Situation steht man nun vor zwei Problemen:

- Gängige Computerprogramme und Dateiformate werden im Laufe der Zeit immer wieder verändert, so dass darauf geachtet werden muss, dass archivierte Dateien später noch gelesen werden können.
- Die Sicherheit und Unmanipulierbarkeit einer digitalen Unterschrift kann nicht auf Jahrzehnte im Voraus garantiert werden, da Computer immer leistungsfähiger werden (und so Systeme möglicherweise »geknackt« werden können) und die zugrunde liegenden kryptologischen Verfahren durch neue mathematische Erkenntnisse kompromittiert werden können.

Beispiel-  
lösung  
ArchiSig

In einem unter Federführung des Fraunhofer-Instituts für Sichere Informationstechnologie – SIT – durchgeführten und vom Bundeswirtschaftsministerium geförderten Projekt namens **ArchiSig** wurde ein Verfahren entwickelt, welches die erneute Signierung digital archivierter Dokumente erlaubt und zudem leicht in bestehende Dokumenten-Management-Systeme integriert werden kann. Das entsprechende entwickelte Softwarepaket trägt den Namen **ArchiSoft**.



Abb. 2.3-8: ArchiSig-Logo.

Die Funktionsweise von ArchiSig bzw. ArchiSoft beruht im Grunde auf zwei »Tricks«:

- Die Aktualisierung erfolgt durch **Zeitstempel**, d. h. durch digitale Unterschriften des gesamten Dokuments einschließlich früherer Signaturen (»Übersignierung«), in denen der Zeitpunkt ihrer Erstellung mit festgehalten ist.

- Damit mehrere Dokumente zusammen übersigniert werden können, werden diese in einem Baum hierarchisch angeordnet, und es werden Hashwerte (siehe dazu auch »Hashfunktionen« (S. 108)) so gebildet, dass nur der zur Wurzel des Baumes gehörende Datensatz mit einem Zeitstempel versehen werden muss.

#### 2.3.4.2 Blinde Signaturen \*\*

**Blinde Signaturen** dienen dazu, digitale Unterschriften für Daten zu erzeugen, ohne dass der Unterschreibende diese Daten zur Kenntnis nimmt oder speichert. Dies kann bei digitalem Geld oder bei Online-Wahlen verwendet werden.

Wenn Sie etwas unterschreiben, wollen Sie selbstverständlich wissen, *was* Sie unterschreiben. Es gibt jedoch auch Anwendungsszenarien, bei denen von dieser Selbstverständlichkeit abgewichen wird.

Unter einem Verfahren zur Erzeugung blinder Signaturen versteht man ein Protokoll, in dem eine Instanz *A* einem Unterzeichner *B* einen Datensatz so vorlegen kann, dass

blinde  
Signaturen

- *B* diesen Datensatz nicht zu sehen bekommt und
- *A* in den Besitz der gültigen digitalen Unterschrift von *B* unter diesen Datensatz gerät.

Wenn man im täglichen Leben etwas »blind« unterschreibt, liegt dies in der Regel an einer gewissen Nachlässigkeit: Das Dokument ist zu unwichtig, als dass sich die Mühe des sorgfältigen Lesens lohnen würde. Anders bei der digitalen blinden Signatur: Der Unterschreibende soll das Dokument gar nicht lesen *können*, selbst wenn er es wollte! Da auf der anderen Seite die digitale Unterschrift immer vom zu unterschreibenden Datensatz abhängt (siehe dazu »Digitale Un-

terschrift« (S. 97)), scheint dies auf den ersten Blick ein Widerspruch zu sein.

Eine Lösung, die auf den bekannten Kryptologen David Chaum zurück geht und sich des **RSA**-Verfahrens bedient (siehe »RSA-Verfahren« (S. 88)), wird nun beschrieben.

blinde Signatur  
mit RSA

Der Datensatz  $m$  (es kann sich etwa um ein Dokument oder eine Nachricht handeln) einer Instanz (Person)  $A$  soll von  $B$  blind signiert werden. Die RSA-Parameter von  $B$  seien  $n$ ,  $e$  (öffentlicher Schlüssel) und  $d$  (geheimer Schlüssel).

$A$  wählt zunächst eine Zufallszahl  $r$ , die teilerfremd zu  $n$  ist, und bildet die Potenz mit  $e$  modulo  $n$ . Der resultierende Wert

$$x = m \cdot r^e \pmod{n}$$

wird  $B$  zur Unterschrift vorgelegt. (Man beachte, dass  $B$  nicht in der Lage ist, von  $x$  auf  $m$  zu schließen.)  $B$  unterschreibt nun  $x$  mit seinem geheimen Schlüssel, d. h. durch Potenzierung mit  $d$ :

$$y = x^d \pmod{n}$$

Das Ergebnis  $y$  wird an  $A$  zurückgegeben.  $A$  bildet nun

$$z = y \cdot r^{-1} \pmod{n}.$$

$z$  ist tatsächlich das durch  $B$  unterschriebene  $m$ , wie leicht nachzurechnen ist:

$$\begin{aligned} z &= y \cdot r^{-1} = x^d \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} \\ &= m^d \cdot r^{ed} \cdot r^{-1} = m^d \cdot r \cdot r^{-1} = m^d \pmod{n} \end{aligned}$$

Eine überzeugende Anwendung blinder Signaturen ist die Erzeugung elektronischer Münzen: Die Echtheit einer Münze wird durch die blinde Signatur der ausgebenden Bank gewährleistet; da die Inhaltsdaten von Münze zu Münze variieren und die Bank sich diese Daten nicht gemerkt hat, kann mit der elektronischen Münze – wie vom üblichen Bargeld gewohnt – anonym eingekauft werden.

Beim Schema für elektronische Münzen nach David Chaum besitzt die Bank für jeden Münzwert einen RSA-Schlüsselsatz  $(e, d, n)$  wie oben. Ferner muss es eine allgemeine Vereinbarung geben, nach der man eine echte elektronische 5-Euro-Münze erkennt; beispielsweise mag vereinbart sein, dass nach der Entschlüsselung mit dem öffentlichen Schlüssel  $e$  der Bank der herauskommende Datensatz aus zwei identischen Hälften besteht.

elektronische  
Münzen

Angenommen, Kunde  $A$  möchte von der Bank  $B$  eine 5-Euro-Münze erhalten. Er wählt nun zufällig eine Zahl  $w$  (sozusagen das »Rohmaterial« der Münze), die als Bitfolge aus zwei identischen Hälften besteht und als Zahl kleiner als  $n$  ist.  $A$  lässt  $w$  von der Bank  $B$  blind signieren und erhält die Münze  $z$ . Mit dieser Münze kann  $A$  anonym einkaufen, und jeder kann mit dem öffentlichen Schlüssel von  $B$  (also der Bank) die Echtheit der Münze überprüfen.

Bei einem Münzsystem wie im Beispiel beschrieben muss es zusätzlich einen Mechanismus geben, der sicherstellt, dass eine bestimmte Münze nicht mehrmals ausgegeben werden kann. Ein solches Münzsystem ist vor einigen Jahren von der Deutschen Bank unter dem Namen **Ecash** erprobt worden, jedoch wurden die Versuche mangels genügender Akzeptanz durch die Kunden wieder eingestellt.

Blinde Signaturen können auch bei **elektronischen Wahlen** eingesetzt werden – sie dienen dort dazu, anonyme echte Wahlzettel zu erzeugen. Damit auch die Stimmabgabe anonym ist, können **MIXe** eingesetzt werden (siehe »Das Mix-Konzept« (S. 326)).