Fälschungssicherheit digitaler Signaturen

Eine Übersicht

Dirk Fox

Mit der aktuellen Diskussion des Signaturgesetzes sind digitale Signatursysteme in das Licht der Öffentlichkeit gerückt. Die Idee digitaler Signaturen ist jedoch nicht neu: Sie geht auf eine grundlegende Veröffentlichung aus dem Jahr 1976 zurück.

Der Beitrag gibt eine Übersicht über die Sicherheitseigenschaften der heute verbreiteten Verfahren (RSA und DSA) und zieht Konsequenzen für deren Einsatz im elektronischen Rechtsverkehr.

Dipl.-Inform.
Dirk Fox

Wissenschaftlicher Mitarbeiter am Institut für Nachrichtenübermittlung, Universität Siegen

Arbeitsgebiete:
Kryptologie, Netzwerksicherheit.

1 Einführung

Die Idee digitaler Signaturen wurde erstmals im Jahr 1976 von Diffie und Hellman in dem wegweisenden Aufsatz *„New Directions in Cryptography“* veröffentlicht. Darin führen die Autoren das Prinzip asymmetrischer Kryptographie ein [DiHe_76].

Unter einem digitalen Signatursystem versteht man ein algorithmisches Verfahren, das dem Autor einer digitalen Nachricht – und zwar nur diesem – erlaubt, mit einem geheimzuhaltenden „Signierschlüssel“ einen zugehörigen Authentikator (digitale Signatur) zu berechnen. Dieser kann zusammen mit der Nachricht verschickt werden. Mit einem passendern, öffentlich bekannten „Prüfschlüssel“ kann der Authentikator und damit die Unverfälschtheit, d.h. die Integrität und Authentizität der Nachricht von jedermann überprüft und gegenüber Dritten belegt werden.

Es blieb Rivest, Shamir und Adleman vorbehalten, im Jahre 1978 das erste digitale Signatursystem zu konstruieren, das den von Diffie und Hellman geforderten Eigenschaften genügte [RSA_78]. Dieses nach den Autoren RSA genannte Verfahren ist inzwischen zum de-facto-Standard geworden.

Eine Vielzahl weiterer digitaler Signatursysteme wurde seitdem vorgeschlagen; einige dieser Verfahren konnten in kurzer Zeit gebrochen werden, d.h. es wurden effektive Methoden zur Fälschung von Signaturen gefunden. Durchsetzen konnte sich neben dem RSA-Verfahren das El’Gamal-Signatursystem [ElGa_84]. Ein an eine von Schnorr vorgeschlagene Modifikation angelehnte Variante wurde 1994 – als erstes und bisher einziges digitales Signatursystem – vom amerikanischen *National Institute of Standards and Technology* (NIST) als *Digital Signature Algorithm* (DSA) genormt [Schn_89, NIST_94].

Dieser Beitrag faßt die wichtigsten theoretischen und praktischen Aspekte der Fälschungssicherheit beider Verfahren zusammen. Es wird gezeigt, wie spezielle Fälschungsangriffe abgewehrt werden können. Abschließend werden Konsequenzen für den Einsatz digitaler Signatursysteme im elektronischen Rechtsverkehr vorgeschlagen.

2 Sicherheit

Ein digitales Signatursystem auf der Basis einer *trapdoor*-Funktion[1] – wie z.B. RSA und DSA – kann nicht unbedingt (oder informationstheoretisch) sicher sein:[2] Da ein Fälscher den öffentlichen Prüfschlüssel besitzt, könnte er, wenn ihm Zeit und Rechenleistung unbegrenzt zur Verfügung stünden, alle Signierschlüssel des endlichen Schlüsselraumes durchprobieren, bis er den passenden gefunden hat.[3]

Digitale Signatursysteme erreichen daher höchstens kryptographische Fälschungssicherheit: Es darf praktisch, d.h. mit begrenzter Rechenleistung nicht in genügend kurzer Zeit möglich sein, gültige (neue) Signaturen zu erzeugen.

Hinsichtlich der kryptographischen Fälschungssicherheit lassen sich vier verschiedene Ebenen unterscheiden.

- Die **erste Ebene** ist die der **Komplexität** des für eine Fälschung zu lösenden zugrundeliegenden Problems.

Eine Fälschung kann nie aufwendiger sein als ein zufälliges Durchprobieren aller mög-

[1] Funktion, deren Umkehrfunktion nur mit Kenntnis eines Geheimnisses berechnet werden kann.

[2] Das einzige unbedingt sichere (symmetrische) Kryptosystem ist Shannons *one-time-pad*: eine nur einmalig verwendbare, echt zufällige Bitfolge von mindestens der Länge der Nachricht, die mit dieser verknüpft wird [Kahn_67].

[3] Anders ist das bei *fail stop*-Signatursystemen: Sie besitzen viele passende Signierschlüssel, aber nur einer ist der richtige; der Signierer kann daher Fälschungen nachweisen [Pfit_96].

lichen Signierschlüssel und ist, da der Prüfalgorithmus effizient durchführbar sein muß, höchstens ein **NP**-Problem.[4]

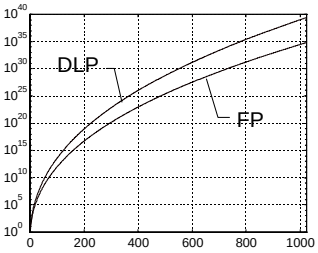
Für einige digitale Signatursysteme konnte gezeigt werden, daß eine Fälschung äquivalent zur Lösung eines anderen bekannten, gut untersuchten (meist zahlentheoretischen) Problems ist. Einige dieser Probleme, wie z.B. die Zerlegung großer Zahlen in ihre Primfaktoren (**Faktorisierungsproblem, FP**: Bestimme *x, y* zu *n* mit $n = x \bullet y$) oder die Bestimmung von Logarithmen in einem Restklassenring (**Diskretes Logarithmusproblem, DLP**: Finde *x* mit $a^x \bmod p = y$)[5] sind z.T. seit über 300 Jahren (verstärkt in den letzten Jahrzehnten) Gegenstand intensiver mathematischer Forschung. Es gilt daher als „gesicherte" Annahme, daß kein Lösungsalgorithmus der Komplexität **P** existiert[6] – wenn dies auch bis heute nicht bewiesen werden konnte.[7]

Die Fälschungssicherheit einiger digitaler Signatursysteme (auch von RSA und DSA) basiert auf weniger gut untersuchten Problemen. Deren polynomiale Unlösbarkeit wird daher im folgenden als „ungesicherte" Annahme bezeichnet.

- Die **zweite Ebene** der Sicherheitsbetrachtung betrifft die **Wahl des Sicherheitsparameters**.

Der Aufwand eines Fälschungsalgorithmus wird üblicherweise in Abhängigkeit von einem Sicherheitsparameter *l* ausgedrückt, der die Modullänge in bit angibt. Er sollte für alle Algorithmen zur systematischen Fälschung von Signaturen sehr schnell (z.B. exponentiell) ansteigen.

Die besten heute bekannten Algorithmen zur Lösung von FP und DLP haben einen asymptotisch ähnlichen Aufwand, der sub-exponentiell in *l* wächst (s. Graphik u. Abschnitte 3.2, 4.2). Daß keine besseren (nicht-polynomialen) Lösungsalgorithmen existieren ist angesichts der algorithmischen Verbesserungen der letzten 20 Jahre allerdings unwahrscheinlich.



Komplexität von DLP und FP[8]

Durch geeignete Festlegung von *l* kann eine Signaturfälschung praktisch unmöglich gemacht werden: Dazu wird für ein konkretes Signatursystem *l* mindestens so groß gewählt, daß eine Signaturfälschung mit den schnellsten bekannten Verfahren und der einem Fälscher vermutlich maximal zur Verfügung stehenden Rechenleistung in absehbarer Zukunft nicht möglich sein wird.

Dazu sind realistische Annahmen über die finanzielle Ausstattung eines Fälschers und die zukünftige Entwicklung der Informationstechnik (Rechenleistung, Parallelisierung) zu treffen.

- **Drittens** muß die Fälschungssicherheit in Abhängigkeit vom jeweiligen **Angreifermodell** bewertet werden.

Die erste Klassifikation von Fälschungsangriffen stammt von Goldwasser, Micali und Rivest [GMR_84]. Sie unterscheiden zwischen **passiven** Fälschungsangriffen, bei denen ein Fälscher keine oder endlich viele Nachrichten mit passender Signatur kennt, **aktiven**, bei denen er sich vorausgewählte Nachrichten vom Signierer digital signieren lassen kann, und **adaptiven**, bei denen er die Wahl der nächsten Nachricht, zu der er vom Signierer eine gültige Signatur erhält, nach Erhalt der vorausgegangenen angepaßt treffen kann.

Bei einer erfolgreichen Fälschung wird unterschieden, ob es sich um ein **vollständiges Brechen** handelt, d.h. die Bestimmung des Signierschlüssels, um ein **universelles Fälschen**, d.h. die Entwicklung eines äquivalenten Signierverfahrens, um eine **selektive Fälschung**, d.h. das Signieren einer bestimmten, vorausgewählten Nachricht oder um eine **existentielle Fälschung**, d.h. die Erzeugung einer gültigen neuen Signatur zu irgendeinem beliebigen (nicht notwendig sinnvollen) Text.

Wie noch genauer gezeigt wird, sind RSA- und DSA-Signaturen beispielsweise bei einem aktiven Angriff selektiv bzw. existentiell fälschbar.

- Und schließlich können **viertens** Fälschungen unter **Ausnutzung von Implementierungsfehlern** erfolgen.

Prüft der Empfänger z.B. bestimmte Randbedingungen nicht, kann das Fälschen von Signaturen sehr leicht sein.

3 RSA

Das RSA-Signatursystem wurde 1978 am MIT[9] entwickelt. Es ist heute das bekannteste und verbreitetste digitale Signatursystem. Das liegt zum einen an der vergleichsweise einfachen, leicht verständlichen Struktur, zum anderen wohl auch an der von der Firma RSA Data Security Inc. geförderten Integration von RSA in Standards und Sicherheitslösungen, wie Anhang A der ISO/IEC-Norm 9796 [ISO_91], den Internet-Standard *Privacy Enhanced Mail* (PEM), das Protokoll *Secure Socket Layer* (SSL) oder *Pretty Good Privacy* (PGP) (siehe z.B. [HoPo_94, Grim_96]).

3.1 Funktionsweise

Der Signierschlüssel eines RSA-Signatursystems ist ein geheimzuhaltender Wert *sk*. Eine RSA-Signatur *Sig* zu einer Nachricht *m* wird nun wie folgt berechnet:[10]

$$Sig = m^{sk} \bmod n$$

Dabei ist *n* Teil des öffentlichen Signatur-Prüfschlüssels und wird als Produkt zweier großer, ebenfalls geheimzuhaltender Primzahlen $n = p \bullet q$ gewählt. Die Prüfung einer RSA-Signatur ist mit dem passenden öffentlichen Schlüssel *pk* möglich, der so gewählt sein muß, daß $sk \bullet pk \bmod \phi(n) = 1$ gilt.[11] Ein solches *pk* ist gerade die multiplikative Inverse sk^{-1} modulo $\phi(n)$.[12] Jeder kann mit dem öffentlichen Schlüssel (*pk*, *n*) die Signatur *Sig* folgendermaßen prüfen:

$$m = Sig^{pk} \bmod n \ ?$$

Denn es gilt, falls *Sig* und *m* nicht verfälscht wurden, nach einem Satz von Euler:

[4] Komplexitätsklasse **NP**: Probleme mit Lösungsalgorithmen, die eine zufällig gewählte Lösung in polynomialer Zeit prüfen (**N**ichtdeterministisch **P**olynomial).

[5] Der Operator *mod* („modulo") bestimmt den Divisionsrest von a^x / p.

[6] Komplexitätsklasse **P**: Algorithmen, deren Aufwand abhängig von der Eingabelänge *l* durch ein Polynom ausgedrückt werden kann (**P**olynomial).

[7] Es ist eine der großen, bis heute offenen Fragen der theoretischen Informatik, ob es überhaupt Probleme in **NP** gibt, die nicht in polynomialer Zeit gelöst werden können, d.h. ob $\mathbf{P} \neq \mathbf{NP}$ [AhHU_74].

[8] Eine Komplexität von 10^{40} Prozessoroperationen ist physikalisch nicht mehr realisierbar.

[9] *Massachusetts Institute of Technology*.

[10] Ist *m* größer *n*, wird üblicherweise ein Hashwert $h = hash(m)$ signiert [Dobb_97].

[11] $\phi(n)$ bezeichnet den Wert der Eulerschen ϕ-Funktion, die die Anzahl der zu *n* teilerfremden positiven ganzen Zahlen angibt. Ist $n = p \bullet q$ sind dies gerade $(p-1) \bullet (q-1)$ viele – *n* abzüglich der Anzahl aller ganzzahligen Vielfachen von *p* und *q* kleiner *n*.

[12] Die multiplikative Inverse ist modulo $\phi(n)$ eindeutig bestimmt, wenn *sk* und $\phi(n)$ teilerfremd sind.

$$Sig^{pk} \bmod n = (m^{sk})^{pk} \bmod n$$
$$= m^{sk \bullet pk \bmod \phi(n)} \bmod n = m.$$

3.2 Komplexität

Die Fälschungssicherheit von RSA-Signaturen beruht damit also auf der Schwierigkeit, einen Wert *Sig* zu finden, sodaß für ein bestimmtes *m* gilt: $Sig^{pk} \bmod n = m$, d.h. die *pk*-te Wurzel aus *m* modulo *n* zu ziehen. Klar ist: Wenn man die Primfaktoren von *n* kennt (d.h. Faktorisieren kann), dann läßt sich $sk = pk^{-1}$ modulo $\phi(n)$ leicht berechnen und das RSA-Signatursystem vollständig brechen.

Daher kann das Fälschungsproblem nicht schwieriger sein die Faktorisierung von *n*. Und deren Lösungskomplexität liegt mit dem besten heute bekannten Algorithmus, dem *number field sieve*, asymptotisch bei (s. Graphik) [13]

$$o(l) = e^{(c+o(1)) \bullet (l \cdot \ln 2)^{1/3} (\ln(l \cdot \ln 2))^{2/3}}$$

Die umgekehrte Aussage, daß RSA-Signaturen nur durch Faktorisierung des Moduls *n* gefälscht werden können, ist bis heute nicht bewiesen. Zwar kann gezeigt werden, daß sowohl das Finden eines passenden *sk'* als auch die Bestimmung von $\phi(n)$ zugleich das Faktorisieren von *n* erlauben, nicht aber, daß es keinen anderen Weg zum Fälschen von RSA-Signaturen gibt.

Daher beruht die Sicherheit des RSA-Verfahrens auf der (ungesicherten) Annahme, daß die Bestimmung einer gültigen Signatur *Sig*, d.h. der *pk*-ten Wurzel aus *m* modulo *n* (ohne Kenntnis von *sk* oder der Primfaktoren von *n*), eine genügend hohe Komplexität besitzt.

3.3 Sicherheitsparameter

1977 veröffentlichten Rivest, Shamir und Adleman einen mit RSA bezüglich einem 129stelligen Modul (l = 426 bit) verschlüsselten Text und setzten ein Preisgeld von 100 US-$ für die Entschlüsselung aus [Gard_77]. Mit den besten damals bekannten Algorithmen und einer Maschine mit einer – auch heute unrealistischen – Rechenleistung von 3 Mio. MIPS[14] schätzte Rivest die Faktorisierungszeit auf bis zu 4 Billiarden Jahren.

In den letzten zwanzig Jahren wurden Faktorisierungsalgorithmen erheblich verbessert. So erledigt die *elliptic curve method* das Problem mit einem 24-billiardstel dieses Aufwands – mit Rivests Wunder-Maschine in nur zwei Monaten [Lens_87].

Am 2. April 1994 gelang mit einer verteilten Implementierung des *quadratic sieve*-Algorithmus – unter Nutzung von 1600 Workstations im Internet, deren *Idle*-Zeiten zur Verfügung gestellt worden waren – nach knapp acht Monaten die Faktorisierung des 129stelligen Moduls. Die Rechenleistung summierte sich dabei auf geschätzte 4000 bis 6000 MIPS-Jahre [AGLL_94]. Mit einer ähnlichen Lösung unter Verwendung des asymptotisch effizienteren *general number field sieve*-Algorithmus' erscheint es möglich, daß sich 512 bit lange (154stellige) Moduli im Internet inzwischen ebenfalls in wenigen Monaten faktorisieren lassen.

Rivest, Shamir und Adleman empfahlen 1978 bereits 200 Dezimalstellen (665 bit) für *n* [RSA_78] – viele RSA-Implementierungen arbeiten jedoch heute noch mit einer Modullänge von nur 512 bit.

Soll es auch bei einer prognostizierten weiteren Verbesserung der Faktorisierungsalgorithmen in den nächsten 20-30 Jahren praktisch unmöglich sein, RSA-Signaturen durch Faktorisierung zu fälschen, muß heute eine Modullänge von wenigstens l = 768, besser l = 1024 bit (230-310 Dezimalstellen) gewählt werden.

3.4 Fälschungsangriffe

Da Signier- und Prüfalgorithmus bei RSA übereinstimmen, ist eine existentielle Fälschung von RSA-Signaturen sogar allein mit dem öffentlichen Prüfschlüssel *pk* möglich: Zu jeder beliebigen, z.B. zufällig gewählten Signatur *Sig* kann eine passende Nachricht *m* bestimmt werden [Hors_85]:

$$m = Sig^{pk} \bmod n$$

Mit großer Wahrscheinlichkeit ist ein solches *m* keine sinnvolle Nachricht. Zu beliebigen Bitfolgen wie bspw. Schlüsseln können auf diese Weise jedoch gültige Signaturen erzeugt werden.

Kann man den Signierer zum Signieren zweier vorgegebener Nachrichten bringen (aktiver Angriff), können RSA-Signaturen zu sinnvollen Nachrichten selektiv gefälscht werden. Dazu wählt ein Fälscher zunächst eine Nachricht *m*, z.B. „Ich schulde A DM 1000,-". Diese Nachricht zerlegt er nun in zwei (unsinnige) Nachrichten *m'* und *m"* mit $m' \bullet m'' \bmod n = m$. Dann läßt er *m'* und *m"* vom Signierer unterschreiben; er erhält *Sig'* und *Sig"*. Nun bestimmt er eine gültige Signatur für *m*:

$$Sig = Sig' \bullet Sig'' \bmod n.$$

Der Fälschungsangriff von Moore benötigt nur eine „Hilfs"-Signatur [Denn_84]: Der Fälscher wählt *Sig**, berechnet $m^* = Sig^{*pk} \bmod n$ und läßt $m \bullet m^*$ unterschreiben. Er erhält $Sig' = (m \bullet m^*)^{sk} \bmod n$. Nun kann er die gesuchte Signatur *Sig* zu *m* bestimmen:

$$Sig = Sig' \bullet Sig^{*-1} \bmod n$$
$$= m^{sk} \bullet m^{*sk} \bullet Sig^{*-1} \bmod n = m^{sk} \bmod n.$$

Diese existentiellen Fälschungsangriffe beruhen auf der mathematischen Struktur des RSA-Verfahrens: Das Produkt zweier Signaturen ist gleich der Signatur zum Produkt der Nachrichten (**Multiplikativität**). Durch zwei Maßnahmen lassen sich solche Angriffe vereiteln:

- die Hinzufügung von Redundanz, um „gültige" von „ungültigen" Nachrichten *m* unterscheiden zu können; ungültige Nachrichten werden nicht signiert; oder
- die Verwendung einer kollisionsresistenten Hashfunktion *hash*, mit der ein „digitaler Fingerabdruck" $h = hash(m)$ der Nachricht *m* erzeugt und dieser signiert wird (s. [Dobb_97], in diesem Heft).

Dies zerstört die Multiplikativität von RSA-Signaturen.

3.5 Implementierung

Eine wichtige Rolle für die Fälschungssicherheit einer RSA-Implementierung spielt die Generierung der Primfaktoren *p* und *q* (Schlüsselgenerierung). Ursprünglich wurde die Wahl **starker Primzahlen** gefordert, um die Anwendbarkeit spezieller Faktorisierungsalgorithmen zu verhindern. Sie sollten die folgenden Eigenschaften besitzen [Gord_85, Rula_93]:

- *p-1* und *q-1* sollten einen großen Primfaktor *p'* bzw. *q'*, und *p'-1*, *q'-1*, *p+1* sowie *q+1* einen großen Primfaktor besitzen (Schutz vor **Iterationsangriffen**);
- *ggT(p-1, q-1)* sollte klein sein und
- die Längen von *p* und *q* sollten sich um wenigstens 20 bit unterscheiden (Schutz vor **Quadratdifferenzangriffen**).

Es gibt noch weitere, sehr seltene Fälle, die einen Faktorisierungsangriff erleichtern. So

[13] *o(l)* gibt die Zahl der durchschnittlich erforderlichen Prozessoroperationen abhängig von der Eingabelänge *l* an. Für die beste mir bekannte Variante ist c ≈ 1,562 [LeLe_93].

[14] MIPS: *millions of instructions per second*. Zum Vergleich: Ein mit 100 MHz getakteter Pentium-Prozessor erreicht etwa 166 MIPS.

darf sich z.B. unter den Primfaktoren von *p-1, q-1, p+1* und *q+1* keine kleine Fibonacci-Zahl befinden, sonst liefert die Iteration $c_n := c_{n-1}^{-1} + 1 \bmod n$ schnell einen Primfaktor $p = ggT(c, n) > 1$ von n [Hube_91].

Diese spezielle Wahl der Primzahlen hat angesichts neuer Faktorisierungsalgorithmen an Bedeutung verloren; daher wird empfohlen, stattdessen eine größere Schlüssellänge zu wählen [Schn_96].

Als problematisch könnte sich noch die Verwendung eines kleinen öffentlichen Prüfschlüssels *pk* erweisen, wie er zur Beschleunigung der Signaturprüfung in vielen Implementierungen eingesetzt wird.[15] Denn öffentliche Exponenten mit weniger als 32 bit Länge gelten für das RSA-Verschlüsselungssystem nach neueren Ergebnissen als unsicher [Pata_95, CFPR_96].

Wichtig ist natürlich auch, daß ein Schlüsselpaar nicht mehrfach vergeben wird. Dies ist dann vernachlässigbar unwahrscheinlich, wenn die Primzahlen zufällig und gleichverteilt gewählt werden.

Die Erzeugung der Primzahlen *p* und *q* basiert bei den meisten Implementierungen jedoch nicht auf echten Zufallswerten, sondern der Ausgabe eines Pseudozufallszahlengenerators.[16] Ist dieser Generator schlecht gewählt oder die Startsequenz manipulier- oder ausforschbar, kann die Schlüsselgenerierung von einem Fälscher nachvollzogen und das Verfahren so total gebrochen werden. Dies gelang 1995 zwei Studenten der Universität Berkley mit einer SSL-Implementierung des Netscape-Browsers (Version 1.1): Der „zufällige" Startwert setzte sich aus der Prozeß-ID und der Systemzeit zusammen [GoWa_96].

4 DSA

Im Jahr 1992 wurde vom amerikanischen *National Institute of Standards and Technology* (NIST) eine von der *National Security Agency* (NSA)[17] entwickelte Variante des El'Gamal-Signatursystems als *Digital Signature Algorithm* (DSA) zur Standardisierung vorgeschlagen. Nach einer intensiven öffentlichen Diskussion des Verfahrens unter Beteiligung vieler Kryptologen wurde der Algorithmus 1994 als *Digital Signature Standard* (DSS) verabschiedet [NIST_94].

4.1 Funktionsweise

Das DSA-Verfahren arbeitet mit zwei Moduli, einer großen Primzahl *p* und einer 160 bit langen Primzahl *q*, die *p-1* teilt, sowie einem Generator *g* des endlichen Körpers *GF(q)*.[18] Der Signierschlüssel ist ein Wert $0 < sk < q$; den öffentlichen Prüfschlüssel *pk* bestimmt man daraus mit:

$$pk = g^{sk} \bmod p.$$

Für jede DSA-Signatur wählt ein Signierer einen zufälligen, nur einmal verwendbaren Wert $0 < k < q$ und bestimmt damit die Signatur (r, s) zu einem Hashwert $h = SHA(m)$ der Nachricht *m*, bestehend aus zwei 160 bit-Werten:[19]

$$r = (g^k \bmod p) \bmod q$$
$$s = (k^{-1} \bullet (h + sk \bullet r)) \bmod q$$

Der Empfänger kann diese Signatur prüfen, indem er $t = s^{-1}$ modulo *q* bildet und mit dem öffentlichen Prüfschlüssel *pk* des Signierers testet, ob gilt:

$$r = (g^{h \bullet t} \bullet pk^{r \bullet t} \bmod p) \bmod q \ ?$$

Durch Vorausberechnungen läßt sich der Aufwand für die Erzeugung einer Signatur auf zwei modulare Multiplikationen reduzieren [Fox_93].

4.2 Komplexität

Will man den Signierschlüssel *sk* aus *pk* berechnen, erfordert dies beim DSA die Bestimmung eines Diskreten Logarithmus, denn $pk = g^{sk} \bmod q$. Der Aufwand des schnellsten heute bekannten allgemeinen Algorithmus' zur Lösung des Diskreten Logarithmusproblems, dem *general number field sieve* [Gord_93], ist vergleichbar der Lösung des Faktorisierungsproblems (s. Graphik):[20]

$$o(l) = e^{(c+o(1)) \bullet (l \cdot \ln 2)^{1/3} (\ln(l \cdot \ln 2))^{2/3}}$$

Allerdings ist auch für den DSA nicht nachgewiesen, daß eine Signaturfälschung die Lösung des DLP voraussetzt.[21] Möglicherweise kann ein Fälscher ein passendes Paar (r, s) zu gegebenem *h* auf anderem Wege effizienter bestimmen – wenn auch bis heute nicht bekannt ist, wie.

4.3 Sicherheitsparameter

Der Standard läßt für die Wahl des Sicherheitsparameters *l*, hier der Länge des primen Moduls *p* in bit, die Wahl unter allen Vielfachen von 64 von 512 bis 1024 bit. Das Modul *q* der Untergruppe ist auf 160 bit festgelegt.

Mit dem *general number field sieve*-Algorithmus benötigt die Lösung eines 512-bit-DLP ca. $5{,}3 \bullet 10^{21}$ MIPS-Jahre, die eines 1024-bit-DLP etwa $1{,}5 \bullet 10^{32}$ MIPS-Jahre. Selbst eine Million parallel rechnender „Rivest-Wunder-Maschinen" benötigen dafür noch $1{,}8 \bullet 10^{9}$ bzw. $4{,}8 \bullet 10^{19}$ Jahre. Eine Modullänge von wenigstens $l = 512$ bit sollte daher nach heutigem Wissen für die nächsten 20-30 Jahre genügen.

4.4 Fälschungsangriffe

Auch DSA-Signaturen sind ohne Kenntnis eines Paares aus Hashwert *h* und Signatur (r, s) existentiell fälschbar. Aus zwei beliebigen Werten *X* und *Y* kann er einen Hashwert *h'* mit passender Signatur (r', s') berechnen [ElGa_84]:

$$r' = g^X \bullet pk^{-Y} \bmod p$$
$$s' = r \bullet X^{-1} \bmod (p-1)$$
$$h' = Y \bullet s' \bmod (p-1)$$

Sofern SHA-1 kollisionsresistent ist, sollte es jedoch praktisch unmöglich sein, eine zu *h'* passende Nachrit *m'* zu finden mit $SHA(m') = h'$ [NIST_95, Dobb_97].

[15] Üblich ist die 4. Fermatzahl: $2^{16}+1$.

[16] Solche Algorithmen erzeugen aus einer (möglichst echt zufälligen) Startsequenz eine unvorhersagbare, zufällig „aussehende" Bitfolge.

[17] Der amerikanische Nachrichtendienst mit der angeblich größten konzentrierten Rechenleistung der Welt. Wurde lange auch *No Such Agency* genannt [Barl_92].

[18] Ein Generator *g* der Ordnung *p* erzeugt mit $g^i \bmod p$, $i = 1, ..., p-1$ alle Werte aus *GF(p)*.

[19] Die Hashfunktion *secure hash algorithm* (SHA-1) wurde ebenfalls vom NIST genormt [NIST_95, Dobb_97].

[20] Der beste mir bekannte Wert für *c* stammt von Schirokauer: $c \approx 1{,}902$ [Gord_92].

[21] Für die (ursprüngliche) Variante von Schnorr ist dies hingegen bewiesen [PoSt_96].

4.5 Implementierung

Ein simpler Fälschungsangriff auf den DSA schiebt dem Empfänger $g = 1$ unter. Gelingt dies, kann man Signaturen zu beliebigem *pk* leicht fälschen [Vaud_96]:

$$r = (pk^k \bmod p) \bmod q$$
$$s = (k^{-1} \bullet r) \bmod q$$

Denn dann gilt für die Signaturprüfung:

$$(pk^{rt \bmod q} \bmod p) \bmod q =$$
$$(pk^k \bmod p) \bmod q = r.$$

Eine Implementierung muß daher die „Generatoren" $g = 1$ und $g = 0$ abweisen. Prüft die Implementierung außerdem nicht, ob $r < p$, dann kann ein Angreifer eine Signatur zu h^* selektiv fälschen. Zu gegebenem h, (s, r) bestimmt er [Blei_96]:

$$u = h^* \bullet h^{-1} \bmod (p-1)$$

Dann gilt: $g^{h^*} = pk^{ru} \bullet r^{su} \pmod p$. Nun kann die passende Signatur (s^*, r^*) leicht so bestimmt werden, daß gilt:[22]

$$s^* = s \bullet u \bmod (p-1),$$
$$r^* = r \pmod p \text{ und } r^* = r \bullet u \pmod{p-1}.$$

Schwieriger zu entdecken ist der folgende Angriff: Der Fälscher schiebt dem Empfänger $g = pk^a \bmod p$ mit $0 < \alpha < p$ unter; dabei ist *pk* der Prüfschlüssel eines anderen Benutzers. Dann kann er in dessen Namen Signaturen erzeugen [Vaud_96]:

$$r = (pk^k \bmod p) \bmod q$$
$$s = k^{-1} \bullet (\alpha \bullet h + r) \bmod q$$

Wichtig ist auch, daß der Wert k nur für eine einzige Signatur verwendet wird. Benutzt ein Signierer k ein zweites Mal (dies erkennt ein Fälscher daran, daß die Werte r übereinstimmen), kann *sk* bestimmt werden, denn mit h_1, h_2, (s_1, r) und (s_2, r) erhält man ein i.a. eindeutig lösbares Gleichungssystem [ElGa_84, Pete_96]:

$$s_1 = k^{-1} \bullet (h_1 + sk \bullet r) \bmod q$$
$$s_2 = k^{-1} \bullet (h_2 + sk \bullet r) \bmod q$$

Eine weitere Implementierungsschwäche des DSA wurde 1996 veröffentlicht: Da die Hashfunktion SHS-1 eine 160 bit lange Ausgabe erzeugt, gibt es Hashwerte $h = SHS(m)$, die größer sind als q. Signiert wird jedoch der Divisionsrest $h' = h \bmod q$. Damit lassen sich Kollisionen konstruieren – zwei Nachrichten mit derselben DSA-Signatur [Vaud_96].

[22] Den Wert für r^* liefert der Chinesischer Reste-Algorithmus (CRA): Er bestimmt ein x mit $x = x_1 \bmod p_1$ und $x = x_2 \bmod p_2$.

5 Fälschungssichere Signaturen

Im stärksten Sinne fälschungssicher sind nach der Klassifikation in [GMR_84] Signatursysteme, die selbst bei adaptiven aktiven Angriffen nicht einmal existentiell gefälscht werden können. Das erste Verfahren, daß diese Eigenschaft besitzt, wurde von Goldwasser, Micali und Rivest 1984 (in einer ausgearbeiteten Form 1988) vorgeschlagen [GMR_84, GMR_88]. Für das GMR-Verfahren ist die Äquivalenz von existentieller Fälschung einer Signatur und Faktorisierung des Moduls n bewiesen.

Die zentrale Idee ist, jede Signatur nicht nur von Nachricht und Schlüssel, sondern zusätzlich von einer einmaligen Zufallszahl (Referenz) abhängen zu lassen. Anders als beim El'Gamal-Verfahren wird diese Zufallszahl mit Hilfe eines Referenzen-Baumes authentisiert und damit dem Zugriff eines Fälschers entzogen.

Mit einigen Verbesserungen von Goldreich und weiteren Optimierungen ist das Verfahren durchschnittlich so aufwendig wie RSA, allerdings sind die Signaturen deutlich länger [Gold_86, FoPf_91].

Inzwischen wurden weitere Verfahren mit derselben Sicherheitseigenschaft, jedoch kürzeren Signaturen vorgeschlagen [DwNa_94, CrDa_96, auch *fail-stop*-Varianten aus Pfit_96].

6 Zusammenfassung

- Die Fälschungssicherheit von RSA- und DSA-Signaturen beruht auf unbewiesenen, z.T. erst wenig untersuchten komplexitätstheoretischen Annahmen. Das bedeutet nicht, daß RSA und DSA unsicher *sind*, wohl aber, daß sie sich als unsicher erweisen könnten.
- Unter bestimmten Umständen (zu kurze Schlüssel, spezielle Angriffe, Implementierungsfehler) sind Fälschungen einzelner RSA- und DSA-Signaturen sehr leicht möglich. Implementierungen müssen gegen solche Angriffe immun sein.
- Es gibt digitale Signatursysteme, für die selbst eine existentielle Fälschung ebenso schwierig ist wie die Lösung eines gut untersuchten zahlentheoretischen Problems. Die Fälschungssicherheit dieser Verfahren beruht also auf einer gesicherten kryptographischen Annahme.

7 Fazit

Aus der vorausgegangenen Betrachtung der Fälschungssicherheit digitaler Signaturen sollten hinsichtlich ihres Einsatzes im elektronischen Rechtsverkehr die folgenden Konsequenzen gezogen werden:

- **Erstens**: Eine geeignete Festlegung des Sicherheitsparameters l kann vor einem totalen Brechen des Verfahrens schützen. Dabei sind (prognostizierte) Verbesserungen der Fälschungsalgorithmen, Anforderungen an die Gültigkeitsdauer einer digitalen Signatur und die weitere Entwicklung von Rechenleistung zu berücksichtigen. Diese Wahl sollte von Experten regelmäßig in nicht zu langen Zeitabständen überprüft werden.
- **Zweitens**: Einen Schutz vor existentiellen Fälschungen bei Verfahren wie DSA und RSA bieten kollisionsresistente Hashfunktionen [Dobb_97]. Damit hängt die Fälschungssicherheit des Signatursystems allerdings auch von der Kollisionsresistenz der verwendeten Hashfunktion ab. Diese sollte daher ebenfalls festgelegt und die Wahl regelmäßig von Experten geprüft werden.
- **Drittens**: Jede Implementierung eines digitalen Signatursystems sollte vor dem praktischen Einsatz daraufhin untersucht werden, ob sie gegen alle bekannten Fälschungsangriffe immun ist. Dazu sind geeignete Tests zu entwickeln.
- **Viertens**: Trotz der großen Verbreitung von RSA und der Standardisierung des DSA gehört die Zukunft möglicherweise Signaturverfahren, die auch bei adaptiven aktiven Angriffen nicht existentiell gefälscht werden können und für die die Äquivalenz von Fälschung und einer gesicherten komplexitätstheoretischen Annahme mathematisch gezeigt ist. Protokolle und Anwendungen für den Einsatz im elektronischen Rechtsverkehr sollten daher so ausgelegt werden, daß sie diese Verfahren zukünftig unterstützen können.

Dank

Einige Korrekturen und sehr wertvolle Verbesserungsvorschläge zu diesem Beitrag verdanke ich Michael Hortmann und Holger Petersen.

Literatur

[AGLL_94] Atkins, Derek; Graff, Michael; Lenstra, Arjen K.; Leyland, Paul C.: *The magic words are squeamish ossifrage*. Proceedings of Asiacrypt '94, Australien 1994, S. 219-229.

[AhHU_74] Aho, Alfred V.; Hopcroft, John E.; Ullman, Jeffrey D.: *The Design and Analysis of Computer Algorithms*. Addison Wesley, Massachusetts 1974.

[Barl_92] Barlow, John Perry: *Decrypting the Puzzle Palace*. Communications of the ACM, Vol. 35, No. 7, 1992, S. 25-31.

[Blei_96] Bleichenbacher, Daniel: *Generating ElGamal signatures without knowing the secret key*. In: Maurer, U. (Hrsg.): Proceedings of Eurocrypt '96, LNCS 1070, Springer, Berlin 1996, S. 10-18.

[CFPR_96] Coppersmith, Don; Franklin, Matthew; Patarin, Jaques; Reiter, Michael: *Low-exponent RSA with related messages*. In: Maurer, U. (Hrsg.): Proceedings of Eurocrypt '96, LNCS 1070, Springer, Berlin 1996, S. 1-9.

[CrDa_96] Cramer, Ronald; Damgård, Ivan Bjerre: *New Generation of Secure and Practical RSA-Based Signatures*. Koblitz, N. (Hrsg.): Proceedings of Crypto '96, LNCS 1109, Springer, Berlin 1996, S. 173-185.

[Denn_84] Denning, Dorothy E.: *Digital Signatures with RSA and Other Public-Key Cryptosystems*. Communications of the ACM, Vol. 27, No. 4, 1984, S. 388-392.

[DiHe_76] Diffie, Whitfield; Hellman, Martin E.: *New Directions in Cryptography*. IEEE Transactions on Information Theory, Vol. IT-22, No. 6, 1976, S. 644-654.

[Dobb_97] Dobbertin, Hans: *Digitale Fingerabdrücke. Sichere Hashfunktionen für digitale Signatursysteme*. DuD 2/97 (in diesem Heft).

[DwNa_94] Dwork, Cynthia; Naor, Moni: *An Efficient Existentially Unforgeable Signature Scheme and ist Applications*. In: Desmedt, Y. G. (Hrsg.): Proceedings of Crypto '94, LNCS 839, Springer, Berlin 1994, S. 234-246.

[ElGa_84] ElGamal, Taher: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*. In: Blakley, G.R.; Chaum, D. (Hrsg.): Proceedings of Crypto '84, LNCS 196, Springer, Berlin 1995, S. 10-18.

[FoPf_91] Fox, Dirk; Pfitzmann, Birgit: *Effiziente Software-Implementierung des GMR-Signatursystems*. In: Pfitzmann, A.; Raubold, E. (Hrsg.): Verläßliche Informationssysteme. Proceedings der Fachtagung VIS '91, Informatik Fachberichte Nr. 271, Springer, Heidelberg 1991, S. 329-345.

[Fox_93] Fox, Dirk: *Der 'Digital Signature Standard': Aufwand, Implementierung und Sicherheit*. In: Weck, G.; Horster, P. (Hrsg.): Verläßliche Informationssysteme, Proceedings der GI-Fachtagung VIS '93, Vieweg, Braunschweig 1993, S. 333-352.

[Gard_77] Gardner, Martin: *Mathematical games. A new kind of cipher that would take millions of years to break*. Scientific American, August 1977, S. 120-124.

[Gold_86] Goldreich, Oded: *Two Remarks concerning the Goldwasser-Micali-Rivest Signature Scheme*. In: Odlyzko, A.M. (Hrsg.): Proceedings of Crypto '86, LNCS 263, Springer, Berlin 1986, S. 104-110.

[GMR_84] Goldwasser, Shafi; Micali, Silvio; Rivest, Ronald L.: *A „Paradoxical" Solution to the Signature Problem*. In: Proceedings of 25th Symposium on Foundations of Computer Science (FOCS), 1984, S. 441-448.

[GMR_88] Goldwasser, Shafi; Micali, Silvio; Rivest, Ronald L.: *A Digital Signature Scheme Secure against Adaptive Chosen Message Attacks*. SIAM Journal on Computing, Vol. 17, No. 2, 1988, S. 281-308.

[Gord_84] Gordon, J.A.: *Strong Primes are easy to find*. In: Beth, T.; Cot, N.; Ingemarsson, J. (Hrsg.): Proceedings of Eurocrypt '84, LNCS 209, Springer, Berlin 1995, S. 216-223.

[Gord_92] Gordon, Daniel M.: *Designing and Detecting Trapdoors for Discrete Log Cryptosystems*. In: Brickell, E. (Hrsg.): Proceedings of Crypto '92, LNCS 740, Springer, Berlin 1993, S. 66-75.

[Gord_93] Gordon, Daniel M.: *Discrete Logarithms in GF(p) using the Number Field Sieve*. SIAM Journal on Discrete Mathematics, Vol. 6, No. 1, 1993, S. 124-138.

[GoWa_96] Goldberg, Ian; Wagner, David: *Randomness and the Netscape Browser*. Dr. Dobb's Journal, 1/96, S. 66-70.

[Grim_96] Grimm Rüdiger: *Kryptoverfahren und Zertifizierungsinstanzen*. DuD 1/96, S. 27-36.

[HoPo_94] Horster, Patrick; Portz, Michael: *Privacy Enhanced Mail: Ein Standard zur Sicherung des elektronischen Nachrichtenverkehrs im Internet*. DuD 8/94, S. 434-442.

[Hors_85] Horster, Patrick: *Kryptologie*. Reihe Informatik, Bd. 47, Bibliographisches Institut, Mannheim 1985.

[Hube_91] Huber, K.: *Some Considerations concerning the Selection of RSA Moduli*. In: Davies, D. W. (Hrsg.): Proceedings of Eurocrypt '91, LNCS 547, Springer, Berlin 1991, S. 147-150.

[ISO_91] International Organization for Standardization (ISO): *Information technology – Security techniques – Digital signature scheme giving message recovery*. International Standard ISO/IEC 9796, Genf 1991.

[Kahn_67] Kahn, David: *The Codebreakers. The story of secret writing*. Macmillan, New York, 1967.

[LeLe_93] Lenstra, A.; Lenstra, H.: *The Development of the Number Field Sieve*. Lecture Notes in Mathematics 1554, Springer, New York 1993.

[Lens_87] Lenstra, H.W.: *Factoring Integers with Elliptic Curves*. Ann. of Mathematics, No. 126, 1987, S. 649-673.

[NIST_94] National Institute of Standards and Technology (NIST): *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication 186 (FIPS-PUB), 19. Mai 1994.

[NIST_95] National Institute of Standards and Technology (NIST): *Secure Hash Standard (SHS-1)*. Federal Information Processing Standards Publication 180-1 (FIPS-PUB), 17. April 1995.

[Pata_95] Patarin, Jacques: *Some serious Protocol Failures for RSA with Exponent e of less than ≅ 32 bits*. Presented at Conference of Cryptography, CIRM Luminy, France, September 1995.

[Pete_96] Petersen, Holger: *Digitale Signaturverfahren auf der Basis des diskreten Logarithmusproblems und ihre Anwendungen*. Dissertation, TU Chemnitz-Zwickau, Shaker Verlag, Aachen 1996.

[Pfit_96] Pfitzmann, Birgit: *Digital Signature Schemes. General Framework and Fail-Stop Signatures*. LNCS 1100, Springer 1996.

[PoSt_96] Pointcheval, David; Stern, Jacques: *Security Proofs for Signature Schemes*. In: Maurer, U. (Hrsg.): Proceedings of Eurocrypt '96, LNCS 1070, Springer, Berlin 1996, S. 387-398.

[RSA_78] Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard: *A Method for obtaining Digital Signatures and Public Key Cryptosystems*. Communications of the ACM, Vol. 21, No. 2, 1978, S. 120-126.

[Rula_93] Ruland, Christoph: *Informationssicherheit in Datennetzen*. DataCom-Verlag, Bergheim 1993.

[Schn_89] Schnorr, Claus P.: *Efficient Identification and Signatures for Smart Cards*. In: Brassard, G. (Hrsg.): Proceedings of Crypto '89, LNCS 435, Springer, Berlin 1990, S. 239-252.

[Schn_96] Schneier, Bruce: *Applied Cryptography. Second edition.* John Wiley & Sons, New York 1996.

[Vaud_96] Vaudenay, Serge: *Hidden Collisions on DSS*. In: Koblitz, N. (Hrsg.): Proceedings of Crypto '96, LNCS 1109, Springer, Berlin 1996, S. 83-88.

Stichworte

Angreifermodell, Asymmetrische Kryptographie, digitale Signatur, Diskretes Logarithmusproblem, DSA, DSS, elektronischer Rechtsverkehr, existentielle Fälschung, Fälschungssicherheit, Faktorisierungsproblem, GMR, Hashfunktion, Komplexität, kryptographische Sicherheit, RSA, selektive Fälschung, Signatur, SHS-1, starke Primzahlen, totales Brechen, universelles Brechen.