In: Tagungsband 5. Deutscher IT-Sicherheitskongreß des BSI, SecuMedia Verlag, 1997, S. 61-76.

Sichere digitale Signatursysteme

Dirk Fox, Universität Siegen
fox@nue.et-inf.uni-siegen.de

Zusammenfassung

In diesem Beitrag werden drei digitale Signatursysteme vorgestellt, die etwa so effizient wie das RSA-Verfahren bzw. der DSS sind. Im Unterschied zu RSA und DSS sind alle drei Verfahren jedoch bewiesenermaßen nicht einmal bei einem adaptiven aktiven Angriff existentiell fälschbar, sofern eine wohldefinierte Komplexitätsannahme gilt. Der Beitrag schließt mit einem übersichtsartigen Vergleich der wichtigsten Eigenschaften dieser Verfahren mit denen des RSA-Signatursystems.

Stichworte

Adaptiver aktiver Fälschungsangriff, Cramer/Damgård-Signatursystem, digitale Signatursysteme, diskretes Logarithmusproblem, DSS, Dwork/Naor-Signatursystem, ElGamal-Signatursystem, existentielle Fälschung, Fälschungssicherheit, Faktorisierungsproblem, Gesetz zur digitalen Signatur, GMR-Signatursystem, RSA, Signaturverordnung.

1 Einleitung

Vom Bundeskabinett wurde am 11. Dezember 1996 im Rahmen des Entwurfes für ein „Informations- und Kommunikationsdienste-Gesetz“ ein „Gesetz zur digitalen Signatur“ (SigG, Art. 3 IuKD-G) an den Bundesrat weitergeleitet, in dem gesetzliche Anforderungen an eine Infrastruktur für digitale Signaturen formuliert wurden [IuKD_96]. Im Innenministerium wurde ein zugehöriger Verordnungsentwurf entwickelt [SigV_97]. Mit diesem Gesetzentwurf ist die Erwartung verbunden, daß digitalen Signaturen in absehbarer Zeit dieselbe rechtliche Bindungs- und Beweiskraft beigemessen wird, die heute allein eigenhändige Unterschriften besitzen. Um im allgemeinen Rechtsverkehr die Beweiskraft einer eigenhändigen Unterschrift zu erlangen, müssen die technischen Verfahren, die zur Realisierung digitaler Signaturen eingesetzt werden, sehr hohen Sicherheitsanforderungen, d.h. einer besonders strengen Unfälschbarkeitsdefinition genügen.

Seit der wegweisenden Veröffentlichung von Diffie und Hellman, die 1976 erstmals die Idee eines digitalen Signatursystems auf der Basis eines asymmetrischen Kryptoverfahrens vorstellten [DiHe_76], wurde inzwischen eine Vielzahl unterschiedlicher Verfahren zur Realisierung digitaler Signatursysteme entwickelt. Die wichtigsten darunter sind das verbreitete RSA-Signatursystem und das Verfahren von ElGamal [RiSA_78, ElGa_84]. Vor drei Jahren wurde das erste digitale Signatursystem, angelehnt an eine von Schnorr entwickelte Variante des ElGamal-Verfahrens, in den USA als *Digital Signature Standard* (DSS) genormt [Schn_91, NIST_94]. DSS und RSA sind inzwischen sehr weit verbreitet; es kann daher davon ausgegangen werden, daß diese beiden Verfahren auch in den nächsten Jahren in den meisten Anwendungen als digitales Signatursystem eingesetzt werden.

In Kapitel 2 wird gezeigt, daß die Fälschungssicherheit der überwiegenden Zahl digitaler Signatursysteme nicht einer strengen Unfälschbarkeitsdefinition genügen – RSA und DSS ein-

geschlossen. In den letzten Jahren wurden jedoch mehrere Verfahren entwickelt, die sogar, in einem bestimmten Sinne, beweisbar fälschungssicher sind. Drei wichtige Vertreter dieser Verfahren, für die die Sicherheitseigenschaft unter einer Komplexitätsannahme bewiesen wurde, und die zudem effizient implementierbar sind, werden in Kapitel 3 vorgestellt.

Kapitel 4 faßt die wesentlichen Eigenschaften der vorgestellten Verfahren vergleichend zusammen und schließt mit einem Ausblick auf weitere Entwicklungen in diesem Gebiet, das im Licht des derzeit im Rahmen des IuKD-G diskutierten Signaturgesetzes besonders wichtig werden wird.

2 Fälschungssicherheit digitaler Signatursysteme

Ein digitales Signatursystem, das im Kern eine *trapdoor*-Funktion verwendet, d.h. eine Funktion, deren Umkehrung nur mit Kenntnis eines Geheimnisses effizient möglich ist, wie das RSA-Signatursystem und der DSS, kann nicht unbedingt, d.h. informationstheoretisch fälschungssicher sein. Denn mit dem öffentlichen Prüfschlüssel könnte ein Fälscher, dem Zeit und Rechenleistung unbegrenzt zur Verfügung stünden, alle Signierschlüssel durchprobieren, bis er den richtigen gefunden hat. Solche digitalen Signatursysteme sind daher höchstens kryptographisch fälschungssicher: Ein Fälscher darf praktisch, d.h. mit begrenzter Rechenleistung, nicht in genügend kurzer Zeit gültige (neue) Signaturen erzeugen können.

Wünschenswert ist es nun, daß diese praktische Sicherheit eines digitalen Signatursystems auf einer möglichst gesicherten Annahme über die Komplexität eines bekannten Problems beruht, d.h. dem durchschnittlich erforderlichen Aufwand, um dieses Problem mit Hilfe eines Computers zu lösen. Der Zusammenhang zwischen der Komplexität des Problems und der Fälschungssicherheit des digitalen Signatursystems sollte zudem eine (bewiesene) Äquivalenzbeziehung sein: Ist das Problem nicht effizienter lösbar, dann darf auch das Fälschen einer Signatur nicht leichter sein (und umgekehrt).

Bis heute konnte nur für einige wenige der vielen vorgeschlagenen digitalen Signatursysteme diese Äquivalenz bezüglich eines gut untersuchten mathematischen Problems gezeigt werden. Die heute in diesem Zusammenhang wichtigen (zahlentheoretischen) Probleme sind die Zerlegung großer Zahlen in ihre Primfaktoren (Faktorisierungsproblem: Bestimme x, y zu n mit $n = x \bullet y$), die Bestimmung von Logarithmen in einem Restklassenring (Diskretes Logarithmusproblem: Finde x mit $a^x \bmod p = y$), und die Bestimmung von q-ten Wurzeln in Z_n^* (Wurzelproblem: Finde x mit $a^x \bmod n = y$ und $n = p \bullet q$; p, q Primzahlen). Sie sind schon lange, insbesondere in den letzten Jahrzehnten Gegenstand intensiver mathematischer Forschung. Die Existenz ausreichend effizienter Lösungsalgorithmen für große Zahlen gilt daher als sehr unwahrscheinlich, auch wenn dies bis heute nicht bewiesen ist.

Um einen Sicherheitsbeweis für ein digitales Signatursystem zu führen, muß auch das Angreifermodell genauer beschrieben werden. Die Klassifikation von Angriffstypen und Fälschungserfolgen für digitale Signatursysteme stammt von Goldwasser, Micali und Rivest [GoMR_84]. Danach besitzt ein Signatursystem die größte Fälschungssicherheit, für das genau dann, wenn die jeweilige Komplexitätsannahme zutrifft, gilt:

Selbst beim stärksten Fälschungsangriff, einem adaptiven, aktiven Angriff, bei dem der Fälscher zu endlich vielen Nachrichten seiner Wahl nach und nach die passenden digitalen Signaturen erhält, gelingt diesem nicht einmal eine existentielle Fälschung, d.h. die Erzeugung

einer einzigen gültigen neuen digitalen Signatur zu irgendeiner beliebigen, nicht notwendig sinnvollen Nachricht.

Die heute gebräuchlichen digitalen Signatursysteme wie insbesondere das RSA-Verfahren und der DSS, genügen dieser strengen Sicherheitsanforderung nicht. So ist zwar leicht zu zeigen, daß das Fälschen einer RSA-Signatur nicht schwieriger sein kann als die Faktorisierung des Moduls *n*. Die umgekehrte Aussage, daß RSA-Signaturen nur durch Faktorisierung des Moduls *n* gefälscht werden können, ist bis heute jedoch nicht bewiesen. Außerdem sind RSA-Signaturen ohne die Verwendung eines Redundanzschemas oder einer Hashfunktion schon mit einem passiven Angriff existentiell, mit einem aktiven sogar selektiv fälschbar (siehe Übersicht in [Fox1_97]). Für den DSS ist leicht zu zeigen, daß Signaturen fälschen kann, wer weiß, wie man diskrete Logarithmen effizient berechnet. Die Umkehrung dieser Aussage gilt jedoch nicht. Außerdem sind DSS-Signaturen bei einem aktiven Fälschungsangriff existentiell fälschbar [ElGa_84].

3 Existentiell unfälschbare digitale Signatursysteme

Goldwasser, Micali und Rivest stellten 1984, in einer Überarbeitung 1988 das erste digitale Signatursystem vor, das ihrer eigenen, strengen Sicherheitsanforderung (siehe Kapitel 2) genügte. Das Verfahren galt lange Zeit als für praktische Zwecke ungeeignet, obwohl es mit einer Reihe von Verbesserungen eine sehr effiziente Implementierung erlaubt [Gold_86, FoPf_91]. Ein Nachteil des Verfahrens ist jedoch die erhebliche Signaturlänge.

Inzwischen wurden zwei interessante Modifikationen des GMR-Signatursystems vorgeschlagen, das Dwork/Naor-Verfahren und das Verfahren von Cramer/Damgård [DwNa_94, CrDa_96], die beide die Signaturlänge erheblich reduzieren. Alle drei Verfahren werden in den folgenden Abschnitten vorgestellt.

Es gibt noch einige weitere Verfahren, die existentielle Unfälschbarkeit unter einer wohldefinierten Komplexitätsannahme besitzen, wie *fail-stop*-Signaturen, eingesetzt als „normale“ Signaturen [Pfit_96, HePe_92], eine GMR-Variante unter Verwendung von interaktiven Protokollen [CrDa_94] oder das Signatursystem von Bos und Chaum [BoCh_92]. Diese Signatursysteme sind aber für die Praxis derzeit weit weniger geeignet als die drei im folgenden beschriebenen. Eine effiziente, auf diskreten Logarithmen basierende Variante des GMR-Verfahrens ist in Entwicklung, derzeit aber noch unveröffentlicht [Fox2_97]

3.1 Das GMR-Signatursystem

Goldwasser, Micali und Rivest bewiesen 1988, daß das von ihnen vorgestellten Signatursystem ihrer strengen Sicherheitsdefinition genügt, sofern Familien klauenfreier Permutationenpaare (*claw free permutation pairs*) existieren [GoMR_88]. Aus einem klauenfreien Permutationenpaar wird dazu eine nicht nur vom geheimen Signierschlüssel *sk*, sondern auch von der zu signierenden Nachricht *m* abhängige Signierfunktion f_m^{-1} gebildet. Mit dieser wird bezüglich einer authentischen Referenz *Ref* die Signatur zu einer Nachricht *m* berechnet:

$$Sig_{Ref}(m) = f_m^{-1}(Ref) \qquad (3.1)$$

Zur Prüfung dieser Signatur wird mit der aus dem öffentlichen Schlüssel *pk* gebildeten Funktion f_m überprüft, ob gilt:

$$Ref = f_m(Sig_m(Ref)) \; ? \qquad (3.2)$$

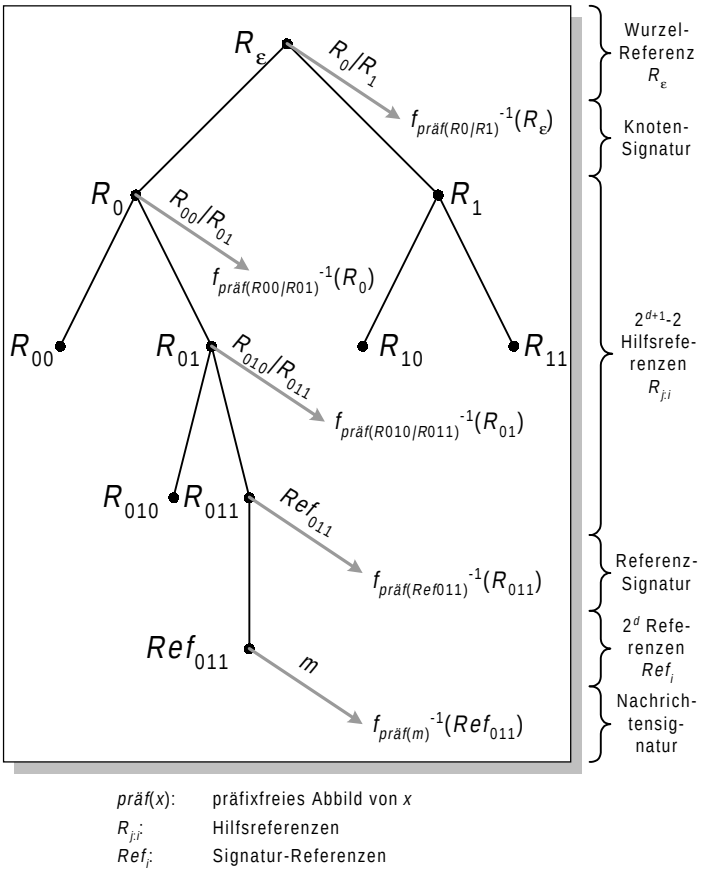


Bild 3-1: GMR-Referenzenbaum[1] mit Baumtiefe d=3

Das GMR-Verfahren kommt ohne eine separate Hashfunktion *hash*(°) aus, da die Konstruktion von f_m^{-1} und f_m beliebig lange Nachrichten m zuläßt. Die Aufgabe der Hashfunktion wird dabei implizit von Signier- und Prüffunktion erledigt. Dies gelingt durch eine bitweise Auswertung der Nachricht: Zu einem klauenfreien Permutationenpaar g_1 und g_2 wird die Signierfunktion f_m^{-1} durch Konkatenation gebildet:

$$f_m^{-1}(Ref) = g_{m0}^{-1}(g_{m1}^{-1}(...(g_{mr\text{-}1}^{-1}(Ref))...)),\ m = m_0...m_{r-1},\ m_i \in \{0, 1\} \quad (3.3)$$

Analog wird die Prüffunktion f_m aus g_0 und g_1 mit umgekehrter Auswerterichtung der m_i gebildet:

$$f_m(Sig_{Ref}(m)) = g_{mr\text{-}1}(g_{mr\text{-}2}(...(g_{m0}(Sig_{Ref}(m)))...)) \quad (3.4)$$

Die Authentizität der nur einmalig verwendbaren Referenzen *Ref* wird durch das Anhängen der Referenzen an die Blätter eines Binärbaumes (Referenzenbaumes) der Tiefe d sichergestellt, dessen Knoten mit je einer Hilfsreferenz $R_{j:i}$ markiert werden. Je zwei benachbarte Hilfsreferenzen werden durch eine Signatur bezüglich des Elternknotens authentisiert (siehe Bild 3-1).[2] Authentizität und einmalige Verwendung der Referenzen Ref_i stellen die existentielle Unfälschbarkeit sicher [GoMR_88].

---

1 Jede Nachricht muß vor dem Signieren präfixfrei abgebildet werden; näheres siehe [GoMR_88, FoPf_91].

2 Die Notation j:i bezeichnet den j-Anfang von i, d.h. die j ersten Bits der Binärdarstellung von i mit $0 \le j \le d$, $0 \le i < 2^d$, und 0:$i = \varepsilon$ und d:$i = i$.

Eine Signatur zu einer Nachricht *m* besteht also neben der Nachrichtensignatur (3.1) bezüglich einer Referenz Ref_i aus allen Hilfsreferenzen mit den zugehörigen Signaturen auf dem Pfad zur *i*-ten Referenz Ref_i. Das sind (ohne die Wurzelreferenz R_ε, die als Teil des öffentlichen Schlüssels *pk* authentisch bekanntgegeben wird) *d* Paare von Hilfsreferenzen $(R_{j:i|0}, R_{j:i|1})$, mit $j = 0, ..., d\text{-}1$; *d* zugehörige Signaturen $Sig_{Rj:i}(R_{j:i|0}, R_{j:i|1})$; die Referenz Ref_i und deren Signatur $Sig_{Ri}(Ref_i)$. Da Ref_i und jeweils eine der Hilfsreferenzen aus jedem der *d* Paare bei der Signaturprüfung nach (3.2) ohnehin berechnet werden, genügt es, *d* Hilfsreferenzen als Teil der Signatur zu verschicken. Insgesamt hat eine GMR-Signatur bei einem Sicherheitsparameter *k* und Baumtiefe *d* also eine Länge von:

$$2(d+1)\ k \text{ bit}$$

Von Goldwasser, Rivest und Micali wurde eine Familie von Permutationenpaaren vorgestellt, deren Klauenfreiheit auf dem Faktorisierungsproblem beruht. Der Aufwand von (g_1, g_2) ist jeweils eine Multiplikation, der von (g_1^{-1}, g_2^{-1}) entspricht etwa einer Exponentiation. Die Berechnung und das Prüfen einer Signatur kann durch eine Reihe von Optimierungen und Vorausberechnungen jedoch erheblich beschleunigt werden (Details finden sich in [Gold_86, FoPf_91]); der Aufwand reduziert sich damit auf:[3]

Signieraufwand (opt.): E + M

Prüfaufwand (opt.): $2(d+1)\cdot k$ M

Der Prüfaufwand kann reduziert werden, wenn ein Empfänger die jeweils letzte von einem Signierer erhaltene Signatur speichert: Beim Empfang der nächsten Signatur ist dann nur noch ein Teil des Authentisierungspfades zu prüfen.

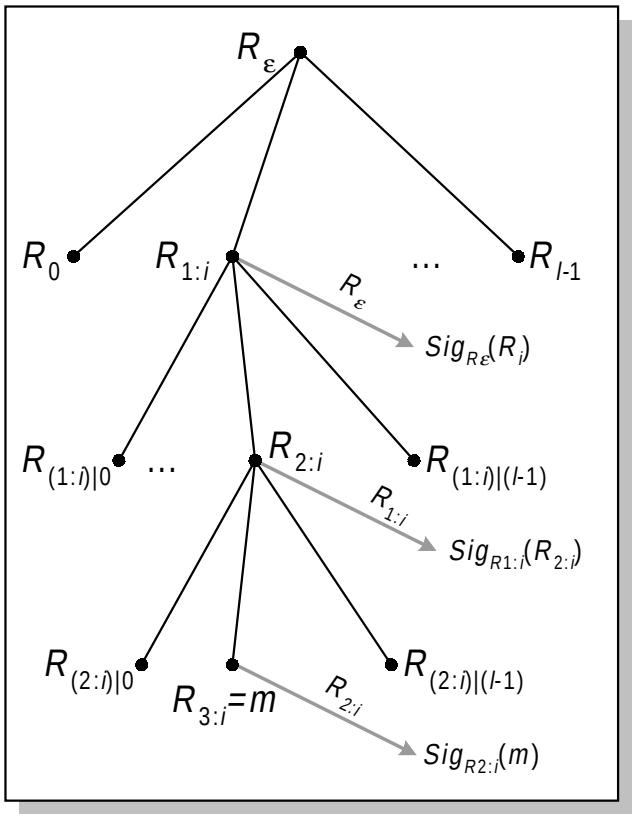
3.2 Das DN-Signatursystem

Von Dwork und Naor wurde 1994 ein im ebenso strengen Sinne, d.h. auch bei adaptiven aktiven Angriffen existentiell unfälschbares Signatursystem vorgestellt [DwNa_94], das den zentralen Nachteil des GMR-Signatursystems, die erhebliche Signaturlänge, deutlich verringert. Das Funktionsprinzip des DN-Systems gleicht dem des GMR-Verfahrens. Auch hier werden Einmal-Signaturen bezüglich Referenzen berechnet, die wiederum mit Hilfe eines Hilfsreferenzenbaumes authentisiert werden. Wie bei GMR hängt daher auch beim DN-System die Signaturlänge von der Baumtiefe *d* ab, d.h. logarithmisch von der Anzahl maximal möglicher Signaturen je Schlüssel. Statt eines Binärbaums wird jedoch ein Baum mit *l* Nachfolgern je Knoten verwendet; *l* ist dabei gleich dem Sicherheitsparameter *k* des Systems (also der Schlüssellänge in bit).

Damit liegt die Baumtiefe, und so auch die Signaturlänge, um einen Faktor $\lfloor \log_2 k \rfloor$ unter der des GMR-Verfahrens. Beispielsweise genügt bei einem Sicherheitsparameter von $k = 1024$ bit im DN-System eine Baumtiefe $d = 2$, um über eine Million Signaturreferenzen authentisieren zu können. GMR benötigt für dieselbe Referenzenzahl eine Baumtiefe von $d = 20$.

---

[3] Die zentralen arithmetischen Operationen zur Berechnung und Prüfung von Signaturen sind Exponentiation (E), Multiplikation (M) und Inversenbestimmung (I) in einem endlichen Körper.



$R_{j:i}$: Referenzen (j:i = j-Anfang der l-adischen Darst. von i)
$Sig_R(m)$: Signatur zu m bezüglich R

Bild 3-2: Referenzenbaum des DN-Systems für d=2

Die Knoten des Baumes werden analog GMR mit Hilfsreferenzen $R_{j:il}$ markiert (siehe Bild 3-2). Als Index $j{:}i_l$ der Referenzen dient hier der j-Anfangsabschnitt (j = 0, ..., d) der l-adischen Entwicklung der Signaturnummer i, gezählt vom äußersten linken Blatt nach rechts. Die jeweils letzte Ziffer bezeichnet also die Nummer der Referenz bezüglich des jeweiligen Elternknotens. Anders als im GMR-System werden hier die Blätter des Baumes mit den Nachrichten m (bzw. deren Hashwerten $hash(m)$, falls $|m|>k$) markiert; es gilt also $R_{d:il} = m$.

Das Verfahren arbeitet mit Schlüsselpaaren (pk, sk) aus einem öffentlichen Prüfschlüssel $pk = (n, R_\varepsilon)$ und geheimen Signierschlüssel $sk = (p, q)$ mit Primzahlen p, q und $n = p{\cdot}q$, $|n| = k$. R_ε bezeichnet (analog GMR) die Wurzel des Hilfsreferenzenbaumes.

Daneben benötigt DN zwei systemweit gültige Listen mit l Einträgen: eine Menge von Primzahlen $\boldsymbol{P} = \{p_0, ..., p_{l-1}\}$ und eine Menge von Zufallszahlen $\boldsymbol{X} = \{x_0, ..., x_{l-1}\}$ aus $\boldsymbol{Z}_n^*$.

Für die Authentisierung der Hilfsreferenzen an den Knoten und der Nachrichten m an den Blättern des Referenzenbaumes wird die folgende Funktion verwendet:

$$Sig_{R_{j:il}}(R_{(j+1):i_l}) = (R_{j:i_l} \cdot \prod_{h=0}^{l-1} x_h^{\beta_h})^{p_o^{-1}} \bmod n \quad (\text{mit } j = 0, ..., d\text{-}1) \qquad (3.5)$$

Dabei bezeichnet β_h das h-te Bit von $R_{(j+1):il}$, d.h. es wird für jede Nachfolge-Hilfsreferenz von $R_{j:il}$ ein anderes Produkt aus den l Zufallswerten $x_0, ..., x_{l-1}$ gebildet. Der Exponent p_o^{-1} wird aus der systemweit gültigen Primzahlmenge $\boldsymbol{P}$ anhand der Nummer o der Nachfolge-

Hilfsreferenz ausgewählt; d.h. der letzten Ziffer des (j+1)-Anfangs der l-adischen Entwicklung von i.

Diese Authentisierung einer Referenz oder einer Nachricht ist unabhängig von allen anderen Nachfolgern der Knotenreferenz, bezüglich der der Authentikator berechnet wird. Das bedeutet, daß (anders als im GMR-Verfahren) die „Geschwister" einer Referenz nicht als Teil der Signatur mitgeschickt werden müssen.

Die Berechnung der Signatur setzt also die Kenntnis der multiplikativen Inversen von p_o modulo $\phi(n)$ voraus:[4] Die systemweit gültige Liste $\boldsymbol{P}$ ist daher eine Art Menge gemeinsamer öffentlicher Schlüssel p_o, zu der jeder Signierer passend zu seinem geheimen Schlüssel (p, q) die multiplikativen Inversen $p_0^{-1} \bmod \phi(n)$, ..., $p_{l-1}^{-1} \bmod \phi(n)$ erzeugt.[5] Diese Menge multiplikativer Inversen kann auch vorausberechnet und als Teil des geheimen Signierschlüssels sk gespeichert werden. Damit vergrößert sich der Speicheraufwand für den Signierschlüssel jedoch von k auf $(l+1)\cdot k$ bit (für k = 1024 also von 128 Byte auf 128 KB).

Der Aufwand für die Erzeugung einer DN-Signatur hängt linear von der Baumtiefe d ab: Je Ebene sind eine modulare Exponentiation, eine Inversenbestimmung (die vorausberechnet werden kann) und $\eta(R_{(j+1):il})+1$ Multiplikationen erforderlich;[6] bei zufälliger Wahl der Hilfsreferenzen also im Mittel $k/2$:

Signieraufwand: $d\cdot$E + $d\cdot k/2$ M + $d\cdot$I

Speichert der Signierer die Signaturen der Hilfsreferenzen auf dem Pfad im Authentisierungsbaum als Teil des Signierschlüssels, ist je Signatur nur noch die Authentisierung der Nachricht bezüglich der Referenz $R_{(d-1):il}$ erforderlich. Die Signaturen der Hilfsreferenzen $R_{j:il}$ kann er l^{d-j}-mal wiederverwenden. Bei einem Sicherheitsparameter von k = 1024 und Baumtiefe d = 2 genügt die Speicherung von einer Hilfsreferenz-Signatur mit der Länge 128 Byte, um nur bei jeder 1024sten Nachricht eine Signatur von einer neuen Hilfsreferenz im Baum berechnen zu müssen.

Der Signieraufwand läßt sich durch den Einsatz einer Hashfunktion weiter verringern. Ist $|m| > k$ bit, muß m ohnehin zunächst auf einen Hashwert abgebildet werden. In [CrDa_96] wird vorgeschlagen, zusätzlich die Referenzen zu hashen. Damit reduziert sich die Zahl der Multiplikationen auf einen vom Sicherheitsparameter l unabhängigen Wert. Die heute zusammen mit RSA und DSS eingesetzen Hashfunktionen SHS-1 [NIST_95] und RIPEMD-160 [DoBP_96, BoDP_97] erzeugen eine 160 bit lange Ausgabe; kürzere Hashwerte sind anfällig für „Geburtstags-Angriffe" (siehe z.B. [Dobb_97]). Die Zahl der durchschnittlich je Signatur erforderlichen Multiplikationen sinkt damit auf $d\cdot 80$. Der Aufwand zur Erzeugung einer Signatur läßt sich damit näherungsweise auf den einer RSA-Signatur drücken:

Signieraufwand (opt.): E + 80 M

Unschön an der Verwendung von Hashfunktionen ist, daß die Unfälschbarkeit der Signaturen damit auch von der Kollisionsresistenz dieser Funktionen abhängt – und über diese Eigenschaft läßt sich für die heute bekannten, effizienten Hashfunktionen keine mathematisch gesi-

---

4 Die **Eulersche Phi-Funktion** $\phi(n)$ gibt die Anzahl der zu n teilerfremden ganzen Zahlen kleiner n an; für $n = p\cdot q$ gilt $\phi(n) = (p-1)\cdot(q-1)$.

5 Die gemeinsame Verwendung der p_i verringert die Menge an öffentlichen Schlüsseln, erfordert jedoch Primalität der p_i: Die multiplikativen Inversen p_i^{-1} wären auch für zu $\phi(n)$ teilerfremde p_i eindeutig bestimmt.

6 $\eta(x)$ gibt das **Hamming-Gewicht**, d.h. die Zahl der Einsen in der Binärdarstellung von x an.

cherte Aussage machen (wenn es auch intensive kryptoanalytische Untersuchungen gibt, die erwarten lassen, daß sich wenigstens für SHS-1 und RIPEMD-160 nicht leichter als durch „brute-force“ Kollisionen finden lassen [Dobb_97]).

Eine DN-Signatur setzt sich also zusammen aus den Hilfsreferenzen und zugehörigen Signaturen auf dem durch i_l bezeichneten Pfad der Länge d im authentisierten Referenzenbaum:

$$Sig(m_i) = \{R_{1:i_l}, Sig_{R\varepsilon}(R_{1:i_l}), R_{2:i_l}, Sig_{R_{1:i_l}}(R_{2:i_l}), \ldots, Sig_{R_{(d-1):i_l}}(m_i)\} \quad (3.6)$$

Die Länge der Signatur (ohne Nachricht) liegt mit $(2d\text{-}1)\ k$ bit deutlich unter der einer GMR-Signatur. Für $k = l = 1024$ und $d = 2$ ist sie aber noch immer etwa 3mal so lang wie eine RSA-Signatur.

Die Signatur kann von einem Empfänger mit dem öffentlichen Schlüssel n und den systemweit gültigen Mengen $\boldsymbol{P}$ und $\boldsymbol{X}$ geprüft werden. Für die Hilfsreferenzen und die Nachricht $m_i = R_{d:il}$ muß gelten:

$$Sig_{R_{j:i_l}}(R_{(j+1):i_l})^{p_o} \bmod n = R_{j:i_l} \cdot \prod_{h=0}^{l-1} x_h^{\beta_h} \bmod n \ ? \quad (3.7)$$

Die Fälschungssicherheit des DN-Signatursystems beruht also (ebenso wie das RSA-Verfahren) auf der Schwierigkeit, p_o-te Wurzeln modulo n zu bestimmen. Für das Verfahren selbst wurde die Äquivalenz von existentieller Unfälschbarkeit und der Gültigkeit dieser „RSA-Annahme“ bewiesen.

Der Aufwand zum Prüfen einer Signatur hängt ebenfalls linear von der Baumtiefe d ab: Je Ebene sind eine modulare Exponentiation und $\nu(R_{(j+1):il})+1$ Multiplikationen erforderlich, also im Mittel $k/2$:

Prüfaufwand: $d\cdot$E + $d\cdot k/2$ M

Werden Nachricht und Referenzen vor dem Signieren mit einer Hashfunktion auf einen Hashwert reduziert, sinkt die Zahl der Multiplikationen wie beim Signieraufwand auf einen von k unabhängigen Wert, für SHS-1 oder RIPEMD-160 auf 80.

Prüfaufwand (opt.): $d\cdot$E + $d\cdot$80 M

Speichert der Empfänger zusätzlich die Hilfsreferenzen $R_{j:il}$ und die zugehörigen Signaturen der jeweils letzten Signatur eines Signierers, kann der Signierer auf die Versendung bereits bekannter Hilfsreferenzen verzichten und der Empfänger auf die Prüfung eines Teils des Authentisierungspfades. Im besten Fall, wenn der Signierer Signaturen nur an den einen Empfänger sendet, ist lediglich jede l-te Signatur länger als eine RSA-Signatur.

Der Prüfaufwand sinkt dabei im besten Fall auf eine Exponentiation und 80 Multiplikationen. Bei optimaler Implementierung liegt der Prüfaufwand so nur geringfügig über dem zur Prüfung einer RSA-Signatur. Auch hier kann noch, wie beim RSA-Verfahren üblich, durch die Verwendung kurzer Primzahlen p_i mit kleinem Hamming-Gewicht $\eta(p_i)$ der Rechenaufwand der Exponentiation erheblich gesenkt werden.[7] Die zusätzlich je Prüfung erforderlichen 80 Multiplikationen fallen dann wieder stärker ins Gewicht.

Von Cramer und Damgård wurde vorgeschlagen, wie in GMR die Referenzen des Authentisierungsbaums während der Prüfung zu berechnen, anstatt sie als Teil der Signatur zu verschicken [CrDa_96] (siehe Abschnitt 3.3). Dadurch verringert sich die Signaturlänge auf

[7] Nach [Pata_95] sollten die öffentlichen Exponenten p_i mindestens 32 bit lang sein.

$d \cdot k$ bit

Sie ist damit nur noch d-mal so lang wie eine RSA-Signatur. Einziger, aber zentraler verbleibender Nachteil des DN-Signatursystems ist die große Zahl öffentlicher Parameter mit einem Speicherbedarf von insgesamt $2k^2$ bit.

3.3 Das CD-Signatursystem

Cramer und Damgård stellten 1996 eine Modifikation des Dwork-Naor-Signatursystems vor, die mit einer geringeren Zahl von Initialwerten auskommt, und bei der die Zahl der Nachfolger je Knoten im Authentisierungsbaum, d.h. die „Baumbreite“ l, unabhängig vom Sicherheitsparameter k gewählt werden kann [CrDa_96].

Das CD-Verfahren arbeitet wie das DN-System mit einem Referenzen-Authentisierungsbaum mit l Nachfolgern je Knoten (siehe Bild 3-2). l kann unabhängig, also auch größer als der Sicherheitsparameter k gewählt werden, d.h. die für eine vorgegebene Anzahl maximal möglicher Signaturen je Schlüssel erforderliche Baumtiefe d (und damit die Signaturenlänge) kann nach Maßgabe des zur Verfügung stehenden Speichers für die öffentlichen Parameter (z.B. bei einer SmartCard-Implementierung) eingestellt werden.

Anders als das DN-System kommt das Verfahren mit einer systemweit gültigen Menge $\boldsymbol{P} = \{q, p_0, ..., p_{l-1}\}$ von l+1 Primzahlen aus $\boldsymbol{Z}_n^*$ aus. Der geheime Signierschlüssel $sk = (p, q)$ mit Primzahlen p, q und $n = p \cdot q$, $|n| = k$ muß so gewählt sein, daß $\phi(n)$ teilerfremd zu allen Elementen in $\boldsymbol{P}$ ist.

Der öffentliche Prüfschlüssel $pk = (n, E, h, R_\varepsilon)$ umfaßt eine Liste von (kleinen) Exponenten $\boldsymbol{E} = \{e, e_0, ..., e_{l-1}\}$, für die gilt: e ist der kleinste ganzzahlige, positive Exponent mit $\omega := q^e > n$, und für $0 < i < l$-1 ist e_i der kleinste ganzzahlige, positive Exponent mit $v_i := p_i^{ei} > n$; weiter einen zufällig gewählten Wert $h \in \boldsymbol{Z}_n^*$, und R_ε bezeichnet auch hier die Referenz an der Wurzel des Authentisierungsbaumes.

Für die Authentisierung der Hilfsreferenzen an den ersten d-1 Knoten auf dem i-ten Pfad im Authentisierungsbaum wird die folgende Funktion verwendet:

$$Sig_{R_{j:i_l}}(R_{(j+1):i_l}) = (R_{j:i_l} \cdot h^{R_{(j+1):i_l}})^{v_{(j+1):i_l}^{-1}} \bmod n \quad (\text{mit } j = 0, ..., d\text{-}2) \qquad (3.8)$$

Die Signatur zu Nachrichten m an den Blättern des Referenzenbaumes wird berechnet als:

$$Sig_{R_{(d-1):i_l}}(m) = (R_{(d-1):i_l} \cdot h^m)^{\omega^{-1}} \bmod n \qquad (3.9)$$

Damit sind je Signatur d-1 innere Knoten und ein Blatt des Referenzenbaums (eine Nachricht m) zu authentisieren, jeweils mit einem Aufwand von zwei Exponentiationen, einer Multiplikation und einer Inversenbestimmung, zusammen also:

Signieraufwand: $2 \cdot d \cdot \mathrm{E} + d \cdot \mathrm{M} + d \cdot \mathrm{I}$

Alle Inversen v_0^{-1}, ..., v_{l-1}^{-1}, $\omega^{-1} \bmod (p\text{-}1)$ können vom Signierer vorausberechnet werden (Speicherbedarf: $(l+1) \cdot k$ bit). Speichert der Signierer die ersten d-1 Authentisierungen der jeweils letzten Signatur (Speicherbedarf: $(d\text{-}1) \cdot k$ bit), kann er diese für die nächste Signatur meist wiederverwenden. Nur jede l-te Signatur muß er wenigstens eine, höchstens d-2 neue Authentisierungen berechnen. Diese können in *Idle*-Zuständen vorausberechnet werden. Damit reduziert sich der Signieraufwand auf:

Signieraufwand (opt.): 2·E + M

Der Aufwand für jeweils eine der beiden Exponentiationen kann erheblich reduziert werden, indem die Referenzen der inneren Knoten des Referenzenbaumes, wie für das DN-Signatursystem beschrieben, vor dem Signieren mit einer Hashfunktion auf einen 160-bit-Wert abgebildet werden. Auch die Kosten der Multiplikationen sinken dadurch etwas.

Eine Signatur besteht damit aus d Authentisierungen für innere Knoten (Referenzen) und Blätter (Nachrichten) auf einem Pfad des Authentisierungsbaumes. Die Referenzen müssen dabei nicht als Teil der Signatur verschickt werden, da sie im Verlauf der Prüfung wiedergewonnen werden können ((3.11), (3.12)). Eine CD-Signatur hat damit eine Länge von $d \cdot k$ bit.

$$Sig(m_i) = \{Sig_{R_\varepsilon}(R_{1:i_l}), Sig_{R_{1:i_l}}(R_{2:i_l}), \ldots, Sig_{R_{(d-1):i_l}}(m_i)\} \qquad (3.10)$$

Beim Prüfen der Signatur einer Nachricht kann zugleich die Referenz $R_{(d\text{-}1):il}$ zurückgewonnen werden, indem der Prüfer berechnet [CrDa_96]:

$$R_{(d-1):i_l} = (Sig_{R_{(d-1):i_l}}(m))^{\omega} \cdot h^{-m} \bmod n \qquad (3.11)$$

Alle weiteren Baumreferenzen erhält er, indem er nacheinander für $j = d\text{-}2, \ldots, 0$ berechnet:

$$R_{j:i_l} = (Sig_{R_{j:i_l}}(R_{(j+1):i_l}))^{v_{j:i_l}} \cdot h^{-R_{(j+1):i_l}} \bmod n \qquad (3.12)$$

Schließlich akzeptiert er die Signatur, wenn die zuletzt berechnete Referenz $R_{0:il} = R_\varepsilon$ ist. Der Aufwand liegt dabei insgesamt bei (wenn $h^{-1} \bmod n$ vorausberechnet wird):

Prüfaufwand: $2d$·E + d·M

Werden die Referenzen vor dem Signieren mit einer Hashfunktion abgebildet, sinkt auch beim Prüfen der Aufwand der Hälfte aller Exponentiationen deutlich. Insgesamt liegt er allerdings über dem Prüfaufwand für DN-Signaturen.

Cramer und Damgård schlugen außerdem vor, statt des öffentlichen Parameters h zwei Werte h_1 und h_2 zu wählen, den jeweils zu authentisierenden Wert β (Referenz oder Nachricht) in zwei $k/2$ bit lange Werte β_1 und β_2 aufzuteilen und jeweils $h_1^{\beta 1} \cdot h_2^{\beta 2}$ zu berechnen. Damit wächst der öffentliche Schlüssel um k bit; der Aufwand für diese „Doppelexponentiation“ kann bei geschickter Implementierung auf weniger als die Hälfte der ursprünglichen vollen Exponentiation gesenkt werden [Fox_93]. Dieser Effizienzgewinn kommt sowohl Signier- als auch Prüffunktion zugute.

Der Speicheraufwand für die öffentliche Liste aus l+1 k-bit-Primzahlen kann erheblich verringert werden, indem die ersten l+1 kleinsten Primzahlen gewählt werden. Deren Generierung kann so schnell erfolgen, daß sich eine Speicherung erübrigt. Allerdings steigt dadurch der Speicherbedarf für $\boldsymbol{E}$ ein wenig; jedoch kann auch $\boldsymbol{E}$ schnell bestimmt werden.

4 Vergleichende Zusammenfassung und Ausblick

Die folgende Tabelle gibt einen Überblick zum Aufwand der drei in Kapitel 3 diskutierten sicheren digitalen Signatursysteme im Vergleich zum RSA-Verfahren. Nicht berücksichtigt in den Aufwandsangaben sind jeweils die Kosten für die Generierung der Zufallswerte und der Hashfunktion – bei Verwendung kryptographisch starker Verfahren kann dies einen deutlichen Zusatzaufwand bedingen [BlBS_86, Damg_87]; vorausgenerierte Zufallszahlen und die Hashfunktionen SHS-1 und RIPEMD-160 fallen hingegen praktisch nicht ins Gewicht.

Verfahren	RSA	GMR	DN	CD
Signaturanzahl	unbegrenzt	2^d	k^d	l^d
Signieren (opt.)	E	E + M	E + 80 M	2·E + M
Länge *sk*	k bit	$3(d+1)\cdot k$ bit	$(d+l)\cdot k$ bit	$(d+l+1)\cdot k$ bit
Signaturlänge	k bit	$2(d+1)\cdot k$ bit	$d\cdot k$ bit	$d\cdot k$ bit
Prüfen (opt.)	E	$2(d+1)\cdot k$ M	d·E + d·80 M	$2d$·E + d·M
Länge *pk*	$2\cdot k$ bit	$2\cdot k$ bit	$2(k+1)\cdot k$ bit	$3\cdot k$ bit

Alle bis heute vorgeschlagenen, auch bei adaptiven aktiven Angriffen existentiell unfälschbaren digitalen Signatursysteme, für die die Äquivalenz von Fälschbarkeit und Widerspruch zu einer wohldefinierten Komplexitätsannahme mathematisch gezeigt ist, und die zudem effizient und speichersparend implementiert werden können, setzen die Gültigkeit der RSA-Annahme oder der Faktorisierungsannahme voraus.

Angesichts der ständigen Entwicklung verbesserter Faktorisierungsalgorithmen mit subexponentiellem Aufwand sind jedoch digitale Signatursysteme wünschenswert, die Sicherheit gegen adaptive aktive Fälschungsangriffe auf der Basis des diskreten Logarithmusproblems (DLP) bieten. Denn für die Lösung des DLP auf Elliptischen Kurven sind bis heute keine subexponentiellen Algorithmen bekannt. DLP-basierte digitale Signatursysteme können daher über Elliptischen Kurven mit erheblich kürzeren Schlüssel- und Signaturlängen arbeiten und erlauben so sehr effiziente Implementierungen [FoRö_96].

Eine sehr effiziente Variante des GMR-Verfahrens, deren Sicherheit auf der Gültikeit des diskreten Logarithmusproblems beruht, ist derzeit in Entwicklung [Fox2_97].

Literatur

BlBS_86 Blum, L.; Blum, M.; Schub, M.: *A Simple Unpredictable Pseudo-Random Number Generator*. SIAM J. Computing, 15/2, 1986, S. 364-383.

BoCh_92 Bos, Jurjen; Chaum, David: *Provably Unforgeable Signatures*. In: Brickell, E.F. (Hrsg.): Proceedings of Crypto ‘92, LNCS 740, Springer, Berlin 1993, S. 1-14.

BoDP_97 Bosselaers, Antoon; Dobbertin, Hans; Preneel, Bart: *The RIPEMD-160 Cryptographic Hash Function*. Dr. Dobb’s Journal, 1/97, S. 24-28.

CrDa_94 Cramer, Ronald; Damgård, Ivan Bjerre: *Secure Signature Schemes based on Interactive Protocols*. BRICS report series, RS-94-29, September 1994, Aarhus University.

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.

Damg_87 Damgård, Ivan Bjerre: *Collision free Hash Functions and Public Key Signature Systems*. In: Chaum, D.; Price, W.L. (Hrsg.): Proceedings of Eurocrypt ‘87, LNCS 304, Springer, Berlin 1988, S. 203-216.

DiHe_76 Diffie, Whitfield; Hellman, Martin E.: *New Directions in Cryptography*. IEEE Transactions on Information Theory, Bd. IT-22, Nr. 6, 1976, S. 644-654.

Dobb_97 Dobbertin, Hans: *Digitale Fingerabdrücke. Sichere Hashfunktionen für digitale Signatursysteme*. Datenschutz und Datensicherheit (DuD), 2/97, S. 18-23.

DoBP_96 Dobbertin, Hans, Bosselaers, Antoon; Preneel, Bart: *RIPEMD-160: A Stengthened Version of RIPEMD*. Fast Software Encryption, LNCS 1039, Springer, Berlin 1996, S. 71-82.

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, Heidelberg 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.

FoRö_96 Fox, Dirk; Röhm, Alexander W.: *Effiziente Digitale Signatursysteme auf der Basis Elliptischer Kurven*. In: Horster, P. (Hrsg.): Digitale Signaturen. Proceedings der Arbeitskonferenz Digitale Signaturen 96, vieweg-Verlag, Braunschweig 1996, S. 201-220.

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, DuD-Fachbeiträge 16, Verlag Vieweg, Braunschweig 1993, S. 333-352.

Fox1_97 Fox, Dirk: *Fälschungssicherheit digitaler Signaturen*. Datenschutz und Datensicherheit (DuD) 2/97, S. 69-74.

Fox2_97 Fox, Dirk: *Ein effizientes GMR-Signatursystem auf der Basis des diskreten Logarithmusproblems*. Unveröffentlichte Arbeit, Universität Siegen, Februar 1997.

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.

GoMR_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.

GoMR_88 Goldwasser, Shafi; Micali, Silvio; Rivest, Ronald L.: *A Digital Signature Scheme Secure against Adaptive Chosen Message Attacks*. SIAM Journal on Computing, Bd. 17, Nr. 2, 1988, S. 281-308.

HePe_92 Heyst, E. van; Pedersen, T.P.: *How to make efficient fail-stop signatures*. In: Rueppel, R.A. (Hrsg.): Proceedings of Eurocrypt ‘92, LNCS 658, Springer, Berlin 1993, S. 366-377.

IuKD_96 *Informations- und Kommunikationsdienste-Gesetz (IuKD-G)*. Beschluß des Bundeskabinetts vom 11.12.96, Bundesratsdrucksache 966/96. Dokumentiert in: Datenschutz und Datensicherheit (DuD), 1/97, S. 38-45.

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.

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

RiSA_78 Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard: *A Method for obtaining Digital Signatures and Public Key Cryptosystems*. Communications of the ACM, Bd. 21, Nr. 2, 1978, S. 120-126.

Schn_91 Schnorr, Claus P.: *Efficient Signature Generation by Smart Cards*. Journal of Cryptology, Bd. 4, Nr. 3, 1991, S. 161-174.

SigV_97 *Verordnung zur digitalen Signatur (Signaturverordnung – SigV)*. Stand: 12.01.97. Dokumentiert in: Datenschutz und Datensicherheit (DuD), 2/97, S. 102-105.