In: Müller, G.; Rannenberg, K.; Reitenspieß, M.; Stiegler, H. (Hrsg.): *Verläßliche IT-Systeme – Zwischen Key Escrow und elektronischem Geld*, Vieweg, Braunschweig 1997, S. 121-137..

Ein effizientes *und* sicheres digitales Signatursystem

Dirk Fox

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

Zusammenfassung

Das am 13. Juni 1997 vom Bundestag beschlossene Signaturgesetz weist digitalen Signatursystemen eine besondere Rolle bei der Entwicklung technischer Systeme für elektronischen Rechtsverkehr und digitalen Handel zu. Sowohl die Sicherheit (im Sinne von Unfälschbarkeit) als auch die Effizienz von Erzeugung und Prüfung digitaler Signaturen sind daher für die Wahl geeigneter digitaler Signaturverfahren von entscheidender Bedeutung.

Die heute verbreiteten Verfahren zur Erzeugung digitaler Signaturen, bspw. der DSS und das RSA-Verfahren, verwenden asymmetrische Kryptographie. Sie zeichnen sich durch hinreichende Effizienz aus, genügen aber keiner strengen Sicherheitsdefinition. Ohne die Verwendung von Hashfunktionen oder den Einsatz von Redundanzschemata können sowohl einzelne DSS- als auch RSA-Signaturen leicht gefälscht werden.

In diesem Beitrag wird ein an das GMR-Signatursystem angelehntes digitales Signatursystem vorgestellt, dessen Sicherheit auf der Schwierigkeit beruht, diskrete Logarithmen zu bestimmen. Das Verfahren erlaubt sehr effiziente Implementierungen, u.a. auf Elliptischen Kurven über endlichen Körpern, und genügt der stärksten Sicherheitsdefinition; es wird die Äquivalenz von Signaturfälschung und der Bestimmung diskreter Logarithmen bewiesen. Abschließend werden Aufwandsvergleiche mit herkömmlichen Signatursystemen angestellt und Meßergebnisse einer prototypischen Implementierung angegeben.

1 Einleitung

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 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 in den USA das erste digitale Signatursystem, angelehnt an eine Variante des ElGamal-Verfahrens von Schnorr, als *Digital Signature Standard* (DSS) genormt [Schn_91, NIST_94]. DSS und RSA sind inzwischen sehr weit verbreitet; es kann daher erwartet werden, daß diese beiden Verfahren auch in den nächsten Jahren in den meisten Anwendungen als digitales Signatursystem eingesetzt werden.

Mit dem Mitte Juni vom Bundestag verabschiedeten Signaturgesetz (SigG, Art. 3 IuKD-Gesetz) [IuKD_97] ist die Erwartung verbunden, daß digitalen Signaturen in absehbarer Zeit dieselbe rechtliche Bindungs- und Beweiskraft beigemessen wird, die heute allein eigenhändige Unterschriften besitzen. Dies ist eine der zentralen Voraussetzungen für die weitere Entwicklung des elektronischen Rechtsverkehrs. Um die Beweiskraft einer eigenhändigen Unterschrift zu erlangen, müssen die technischen Verfahren, die zur Realisierung digitaler Signaturen eingesetzt werden, allerdings sehr hohen Sicherheitsanforderungen genügen.

In Kapitel 2 wird nach einer Einführung des Sicherheitsmodells gezeigt, daß die heute verbreiteten digitalen Signatursysteme nicht im strengen Sinne unfälschbar sind: Bei einem aktiven Angriff können RSA- und DSS-Signaturen existentiell bzw. sogar selektiv gefälscht wer-

den. Kapitel 3 stellt drei digitale Signaturverfahren vor, für die die existentielle Unfälschbarkeit von Signaturen unter einer Komplexitätsannahme bewiesen ist, und die zudem effizient implementiert werden können; eingeschlossen das erste solche Verfahren von Goldwasser, Micali und Rivest (GMR) [GoMR_88, FoPf_91].

Eine neue Variante des GMR-Signatursystems, DLP-GMR, wird in Kapitel 4 vorgestellt. Dessen Konstruktion lehnt sich an das GMR-Signatursystem an; die Sicherheit dieses Verfahrens beruht jedoch nicht auf dem Faktorisierungs-, sondern dem diskreten Logarithmusproblem. Die Äquivalenz von Signaturfälschung und Bestimmung diskreter Logarithmen wird nachgewiesen (Kapitel 5); es folgen Vorschläge für eine effiziente Implementierung (Kapitel 6).

Kapitel 7 faßt die wesentlichen Eigenschaften der vorgestellten Verfahren vergleichend zusammen und schließt mit Meßergebnissen prototypischer Implementierungen.

2 Fälschungssicherheit digitaler Signatursysteme

Digitale Signatursysteme, die im Kern eine *trapdoor*-Funktion verwenden, d.h. eine Funktion, deren Umkehrung nur mit Kenntnis eines Geheimnisses effizient möglich ist, wie das RSA-Signatursystem und der DSS, sind höchstens **kryptographisch** fälschungssicher: Ein Fälscher darf praktisch, d.h. mit begrenzter Rechenleistung, nicht in hinreichend kurzer Zeit gültige (neue) Signaturen erzeugen können. Informationstheoretische Sicherheit ist prinzipiell nicht erreichbar, denn der öffentliche Prüfschlüssel ist bekannt: Ein Fälscher mit unbegenzter Zeit und Rechenleistung könnte alle Signierschlüssel durchprobieren, bis er den passenden fände.

Die Sicherheit eines digitalen Signatursystems sollte auf einer möglichst gesicherten Annahme über die Komplexität eines bekannten Problems beruhen, d.h. dem durchschnittlich erforderlichen Aufwand, um dieses Problem mit Hilfe eines Computers zu lösen. Der Zusammenhang von Komplexität des Problems und Fälschungssicherheit des Signatursystems sollte zudem eine Äquivalenzbeziehung sein: Das Fälschen einer Signatur darf nicht leichter sein als die Lösung des Problems (und umgekehrt).

Zu den heute in diesem Zusammenhang wichtigen, überwiegend zahlentheoretischen Problemen zählen vor allem die Zerlegung großer Zahlen in ihre Primfaktoren (**Faktorisierungsproblem**: Bestimme x, y zu n mit $n = x \bullet y$) und die Bestimmung von Logarithmen in einem Restklassenring (**Diskretes Logarithmusproblem**: Finde x mit $a^x \bmod p = y$). Beide sind seit Jahrzehnten Gegenstand intensiver mathematischer Forschung. Die Existenz effizienter Lösungsalgorithmen für hinreichend große Zahlen gilt als sehr unwahrscheinlich, auch wenn dies bis heute nicht bewiesen ist.

Eine Bestimmung der Fälschungssicherheit eines digitalen Signatursystems erfordert auch eine genaue Beschreibung des Angreifermodells. Nach der Klassifikation von Angriffstypen und Fälschungen von Goldwasser, Micali und Rivest besitzt ein Signatursystem die größte Fälschungssicherheit genau dann, wenn gilt [GoMR_84]:

> *Selbst bei einem adaptiven, aktiven Angriff, bei dem der Fälscher endlich oft zueiner Nachricht seiner Wahl die passende digitale Signatur 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 verbreiteten digitalen Signatursysteme wie das RSA-Verfahren [RiSA_78] und der *Digital Signature Standard* [NIST_94], genügen einem strengen Sicherheitsbegriff nicht. Erstens ist eine Fälschung nicht nachgewiesen äquivalent der Lösung des zugrundeligenden zahlentheoretischen Problems: So ist zwar das Fälschen einer RSA-Signatur nicht schwieriger als die Faktorisierung des Moduls n; die umgekehrte Aussage, daß RSA-Signaturen nur durch

Faktorisierung von n gefälscht werden können, ist bis heute nicht bewiesen. DSS-Signaturen kann leicht fälschen, wer diskrete Logarithmen effizient berechnen kann; die Umkehrung dieser Aussage ist jedoch nicht nachgewiesen. Zweitens sind RSA-Signaturen, wenn weder ein Redundanzschema noch eine Hashfunktion verwendet wird, durch einen passiven Angriff existentiell, mit einem aktiven sogar selektiv fälschbar (siehe Übersicht in [Fox1_97]). DSS-Signaturen lassen sich bei einem aktiven Fälschungsangriff existentiell fälschen [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 oben) genügte [GoMR_84,GoMR_88]. Die zentrale Idee des GMR-Signatursystems ist, eine Signatur nicht nur abhängig von einem geheimen Signierschlüssel und der Nachricht, sondern zusätzlich auch von einem nur einmalig verwendbaren Zufallswert, Referenz genannt, zu berechnen. Ist diese Referenz authentisch, ist sie dem Zugriff eines Fälschers entzogen; selbst durch einen adaptiven aktiven Angriff gewinnt ein Fälscher keine verwertbaren Informationen.[1]

Die Authentisierung dieser Referenzen erfolgt beim GMR-Verfahren durch Anhängen an die Blätter eines binären Referenzenbaumes der Tiefe d; alle Referenzen inklusive der Hilfsreferenzen an den Knoten des Baumes werden (ebenso wie die Nachricht) bezüglich des jeweiligen Elternknotens mit einer digitalen Signatur authentisiert (siehe Bild 3-1). Damit ist die Zahl der mit einem Schlüsselpaar erzeugbaren digitalen Signaturen auf 2^d begrenzt.

Eine GMR-Signatur zu einer Nachricht m bezüglich einer Referenz Ref_i besteht also neben der Nachrichtensignatur aus allen Hilfsreferenzen mit den zugehörigen Signaturen auf dem Pfad des Referenzenbaums von der (öffentlichen) Wurzelreferenz R_ε bis zur i-ten Referenz Ref_i.

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 (DN) und das Verfahren von Cramer/Damgård (CD) [DwNa_94, CrDa_96], die beide die Signaturlänge durch die Verwendung von Referenzenbäumen mit jeweils l Nachfolgern je Knoten erheblich reduzieren (siehe Bild 3-2). Damit liegt die Baumtiefe, und so auch die Signaturlänge, um einen Faktor $\lfloor \log_2 l \rfloor$ 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 $d = 20$. Eine vergleichende Darstellung dieser drei Verfahren findet sich in [Fox3_97].

Es gibt noch einige weitere Verfahren, die unter einer wohldefinierten Komplexitätsannahme existentiell unfälschbar sind, wie z.B. *fail-stop*-Signaturen, eingesetzt als „normale“ Signaturen [Pfit_96, HePe_92], eine (dem in Kapitel 4 vorgestellten Verfahren ähnliche) GMR-Variante auf der Basis von „Signaturprotokollen“ [CrDa_94], die one-time-Signaturen von Merkle mit beweisbar kollisionsresistenten Hashfunktionen nach Damgard [Merk_87, Damg_88] oder das Signatursystem von Bos und Chaum [BoCh_92]. Die meisten dieser Signatursysteme sind aber für die Praxis nicht hinreichend effizient.

---

[1] Eine ähnliche Idee liegt dem ElGamal-Signatursystem zugrunde; siehe auch Kapitel 4 [ElGa_84].

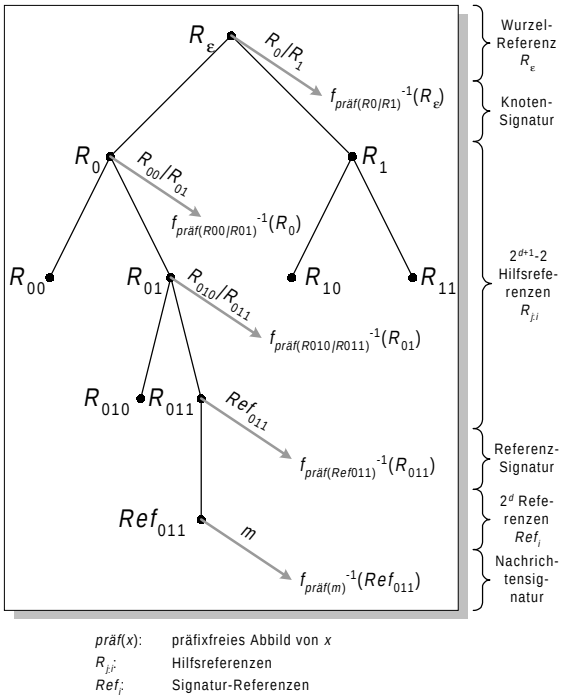

Bild 3-1: GMR-Referenzenbaum[2]

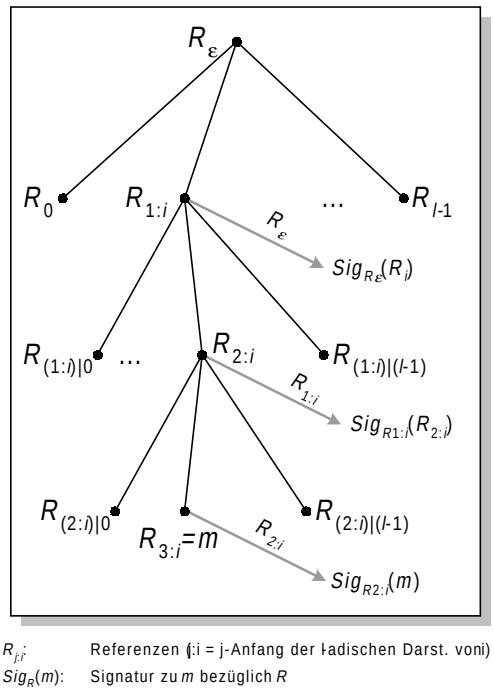

Bild 3-2: DN-Referenzenbaum

4 Das DLP-GMR-Signatursystem

Wie erwähnt ist der wesentliche Nachteil des GMR-Signatursystems mit $2(d+1)$ k bit die Länge der Signaturen. Das DN- und das CD-Verfahren verringern diese durch eine Vergrößerung der Baumbreite l. Dadurch läßt sich bei gleichbleibender Referenzenzahl mit einer um den Faktor $\lfloor \log_2 l \rfloor$ kleineren Baumtiefe d arbeiten; die Signaturänge verkürzt sich um denselben Faktor.

Eine andere Möglichkeit zur Verringerung der Signaturlänge besteht in der Verkürzung der Authentisierungen von Nachricht und Referenzen. Soll dabei die Fälschungssicherheit des Verfahrens erhalten bleiben, muß die Signierfunktion des GMR-Verfahrens gegen eine geeignete Funktion getauscht werden, die bei demselben Sicherheitsniveau kürzere Authentisierungen erzeugt. Dies gelingt mit Funktionen auf der Basis des diskreten Logarithmus-Problems (DLP), die Signaturen in Untergruppen kleinerer, primer Ordnung bilden, wie beispielsweise der DSA [NIST_94]. Sie erlauben zudem eine Implementierung auf Elliptischen Kurven über endlichen Gruppen mit einem um etwa einen Faktor 6 kürzeren Sicherheitsparameter k [FoRö_96]. Dabei steigt zugleich die Effizienz von Signier- und Prüffunktion deutlich.

Die zentrale Idee des DLP-GMR-Signatursystems beruht daher auf einer Ersetzung der Signierfunktion des originalen GMR-Signatursystem – dort aus klauenfreien Permutationenpaaren konstruiert – durch eine ElGamal-artige Funktion [ElGa_84]. Diese Auswahl der Signierfunktion fußt auf der Beobachtung, daß im originalen ElGamal-Signatursystem (ebenso in einigen Varianten wie dem DSS [NIST_94]), ein Teil der Signatur, nämlich der Wert [3]

$$r = \alpha^x \bmod p$$

---

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

[3] α ist Primitivwurzel in $\mathbf{Z}_p^*$, p Primzahl und x ein nur einmalig verwendbarer, zufälliger Wert aus $\mathbf{Z}_{p-1}$.

eine ähnliche Rolle spielt wie die Referenzen an den Blättern des GMR-Referenzenbaumes: Ein Angreifer kann ElGamal- bzw. DSS-Signaturen fälschen, wenn r nicht authentisch ist (oder der Signierer x wiederverwendet). Vereinfacht ausgedrückt authentisiert das im weiteren vorgestellte DLP-GMR-Verfahren die Signaturteile r ElGamal-artiger Signaturen durch die Verwendung eines GMR-artigen, binären Referenzenbaumes (d.h. $l = 2$).

Für eine etwas formalere Beschreibung des DLP-GMR-Verfahrens werden nun, in Anlehnung an die Notation in [GoMR_88], vier Phasen unterschieden:

Vorausberechnung

Der Sicherheitsparameter k wird festgelegt. Ein probabilistischer Polynomzeitalgorithmus $G(1^k)$ liefert ein Tripel (p, q, α) aus einer k bit langen Primzahl p, einer Primzahl q und der Primitivwurzel $\alpha := h^{(p-1)/q} \bmod p$ (mit $0 < h < p$, h Primitivwurzel von Z_p^*, und $\alpha > 1$), d.h. einem Generator der Untergruppe primer Ordnung q von $\boldsymbol{Z}_{p-1}$, es gilt also: $q|(p-1)$. Die Länge von q ist unabhängig von k; q sollte, um Schutz vor „Geburtstags"-Angriffen zu bieten, wie beim DSS mindestens 160 bit lang sein [NIST_94, Dobb_97].[4] G ist dabei so definiert und k so gewählt sind, daß es praktisch unmöglich ist, diskrete Logarithmen in $\boldsymbol{Z}_p$ bezüglich α zu bestimmen.[5]

Initialisierungsphase

Die Tiefe des Referenzenbaumes d, damit auch die maximale Anzahl möglicher Signaturen je Schlüssel (2^d), wird festgelegt. Nun wählt der Signierer seinen geheimen Signierschlüssel, das Tripel $(k_S, k_{SR}, r_\varepsilon)$, zufällig aus $\boldsymbol{Z}_q^*$. Zu diesem Signierschlüssel bestimmt er die zugehörigen Werte des öffentlichen Prüfschlüssels:[6]

$$k_V = \alpha^{k_S} \bmod p\,,\ k_{VR} = \alpha^{k_{SR}} \bmod p \text{ und } R_\varepsilon = \alpha^{r_\varepsilon} \bmod p \qquad (4.1)$$

und veröffentlicht $(p, \alpha, k_V, k_{VR}, R_\varepsilon)$ auf authentische Weise, z.B. unter Verwendung von Signaturschlüssel-Zertifikaten nach dem Signaturgesetz [Fox2_97].

Signieren

Die i-te Signatur (mit $0 \le i \le 2^d-1$) zu einer k bit langen Nachricht m berechnet der Signierer, indem er zunächst die Authentikatoren zu den Referenzen auf dem i-ten Pfad im Referenzenbaum (analog dem originalen GMR-Verfahren) bestimmt. Dazu startet er einen Suchalgorithmus $Path(i, d)$, der auf Eingabe von Signaturnummer i und Baumtiefe d den Pfad im geheimen Referenzenbaum von der Wurzel-Referenz r_ε zum i-ten Blatt $r_i := r_{d:i}$ zurückliefert (siehe Bild 4-1). Dieser „Referenzenpfad" besteht aus einer Liste von d geheimen Referenzen-Paaren $(r_{j:i|0}, r_{j:i|1})$, jeweils beiden Nachfolgeknoten einer Referenz $r_{j:i}$ an einem Knoten des Pfades, und einer geheimen Nachrichtenreferenz ref_i. Alle Referenzen $r_{j:i}$ und ref_i sind dabei zufällig gewählte (und paarweise verschiedene) Elemente aus $\boldsymbol{Z}_q^*$.[7]

---

4 Die Wahl der primen Untergruppe von $\boldsymbol{Z}_{p-1}$ hat beweistechnische Gründe; siehe Kapitel 5.

5 Eine formalere Definition dieser Annahme findet sich in Kapitel 5.

6 Der Sicherheitsbeweis erfordert die Verwendung zweier Schlüsselpaare; siehe Kapitel 5.

7 Der Index $j:i$ bezeichnet hier die höchstwertigen j Bits der Binärdarstellung von i, z.B.: 2:5 = 10.

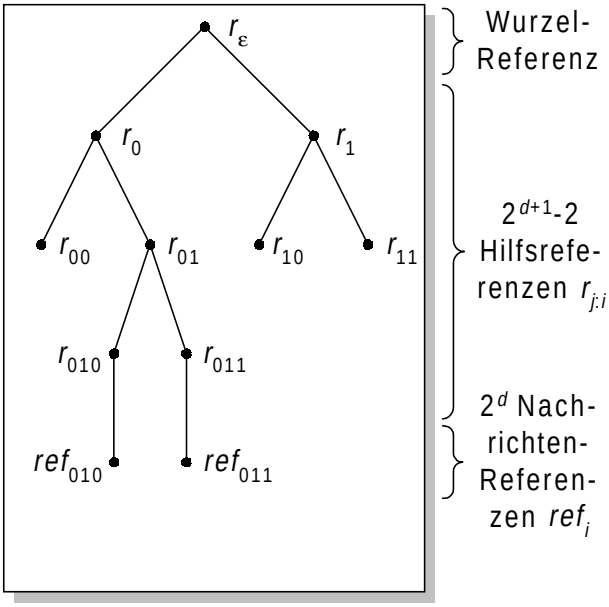

Bild 4-1: Referenzenbaum (Tiefe d=3)

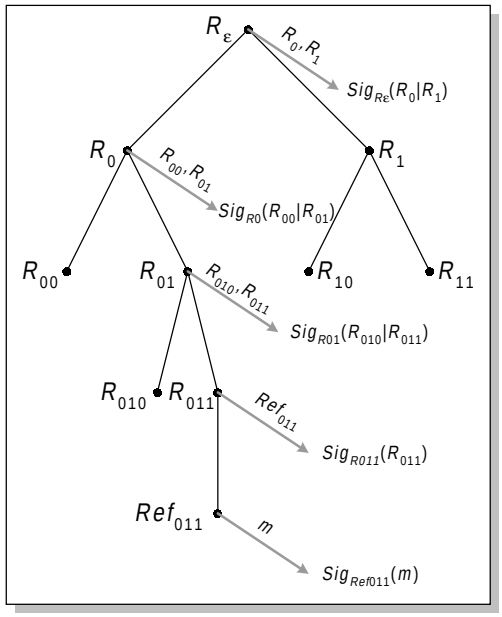

Bild 4-2: Authentisierungsbaum (Tiefe d=3)

Für jedes Paar geheimer Referenzen ($r_{j:i|0}$, $r_{j:i|1}$) bestimmt der Signierer nun die zugehörigen öffentlichen Referenzen $R_{j:i|0}$ und $R_{j:i|1}$ und authentisiert beide bezüglich deren Elternknoten $R_{j:i}$, indem er für $j = 0, ..., d\text{-}1$ berechnet (siehe auch Bild 4-2):

$$R_{j:i|0} = \alpha^{r_{j:i|0}} \bmod p\,,\ R_{j:i|1} = \alpha^{r_{j:i|1}} \bmod p \tag{4.2}$$

$$Sig_{R_{j:i}}(R_{j:i|0}, R_{j:i|1}) = k_{SR} \cdot hash(R_{j:i|0}|R_{j:i|1}) + r_{j:i} \bmod q \tag{4.3}$$

Auf dieselbe Weise erhält und authentisiert er die öffentliche Nachrichtenreferenz Ref_i bezüglich der Referenz R_i am i-ten Blatt des Authentisierungsbaums: [8]

$$Ref_i = \alpha^{ref_i} \bmod p \tag{4.4}$$

$$Sig_{R_i}(Ref_i) = k_{SR} \cdot hash(Ref_i) + r_{d:i} \bmod q \tag{4.5}$$

Schließlich berechnet er die Signatur zur Nachricht m bezüglich der Nachrichtenreferenz Ref_i:

$$Sig_{Ref_i}(m) = k_S \cdot hash(m) + ref_i \bmod q \tag{4.6}$$

Ist $m \geq q$, muß der Signierer die Nachricht m zuvor mit einer kollisionsresistenten Hashfunktion $hash(\ )$ [Dobb_97] auf einen Wert $hash(m) < q$ abbilden.

Für jeden Authentisierungsschritt im Referenzenbaum sind demnach zwei Exponentiationen in $\boldsymbol{Z}_p^*$ (allerdings mit lediglich 160 bit langen Exponenten) und eine Multiplikation in $\boldsymbol{Z}_q$ erforderlich (ohne Berücksichtigung des Aufwands der Hashfunktion). Die Authentisierung der Nachricht erfordert eine weitere Multiplikation in $\boldsymbol{Z}_q$. Damit liegt der Aufwand (ohne Optimierung) bei[9]

$$2d \cdot \mathrm{E}_k + (d{+}1) \cdot \mathrm{M}_{|q|} \tag{4.7}$$

Der tatsächlich notwendige Rechenaufwand für die Bestimmung einer Signatur liegt erheblich darunter; er ist durchschnittlich sogar unabhängig von der Baumtiefe d (siehe Kapitel 6).

---

8 Wie im originalen GMR-Signaturverfahren sind hier aus beweistechnischen Gründen zusätzliche authentische Referenzen Ref_i für die Nachrichtensignaturen („*bridge-items*“ genannt) erforderlich.

9 Wesentliche Operationen sind Exponentiation (E), Multiplikation (M) und Inversenbestimmung (I).

Die Länge einer Signatur wächst allerdings linear mit der Tiefe d des Authentisierungsbaums. Ohne weitere Optimierungen setzt sich eine Signatur (ohne Nachricht) zusammen aus $2d$ Referenzen $R_{j:i}$ ($j > 0$), der Nachrichtenreferenz Ref_i, den zugehörigen d+1 Signaturen sowie der Nachrichtensignatur, zusammen also

$$(2d+1)\,k + (d+2)\,160 \text{ bit} \tag{4.8}$$

Prüfen

Um die Gültigkeit einer Signatur zu überprüfen, beginnt der Empfänger bei der Nachrichtensignatur $Sig_{Ref_i}(m)$ und prüft, ob gilt:

$$\alpha^{Sig_{Ref_i}(m)} = k_V{}^{hash(m)} \cdot Ref_i \bmod p \tag{4.9}$$

$$\alpha^{Sig_{R_{d:i}}(Ref_i)} = k_{RV}{}^{hash(Ref_i)} \cdot R_{d:i} \bmod p \tag{4.10}$$

Anschließend wird nach (4.10) die Authentizität der Nachrichtenreferenz Ref_i geprüft, und schließlich testet der Empfänger, ob für alle j mit $0 \le j \le d$-1 gilt:

$$\alpha^{Sig_{R_{j:i}}(R_{j:i|0},R_{j:i|1})} = k_{RV}{}^{hash(R_{j:i|0}|R_{j:i|1})} \cdot R_{j:i} \bmod p \tag{4.11}$$

Sind alle Gleichungen erfüllt, akzeptiert er die Signatur.

In jedem Überprüfungsschritt müssen zwei Exponentiationen (mit je 160 bit langen Exponenten) und eine Multiplikation in $\boldsymbol{Z}_p^*$ durchgeführt werden (ohne Berücksichtigung der kollisionsresistenten Hashfunktion). Für den gesamten Pfad summiert sich der Berechnungsaufwand einer Signaturprüfung auf

$$2\,(d+2)\cdot \mathrm{E}_k + (d+2)\cdot \mathrm{M}_k \tag{4.12}$$

5 Fälschungssicherheit des DLP-GMR-Verfahrens

Die Fälschungssicherheit des in Kapitel 4 vorgestellten DLP-GMR-Signatursystems genügt der strengen Anforderung von Goldwasser, Micali und Rivest [GoMR_88], d.h. es ist existentiell unfälschbar auch unter einem adaptiven aktiven Angriff – vorausgesetzt, die folgende Komplexitätsannahme gilt:

> **Diskretes-Logarithmus-Problem (DLP)**: Sei k ein Sicherheitsparameter. Gegeben eine Primzahl p der Länge k bit, α eine Primitivwurzel, die eine Untergruppe primer Ordnung von $\boldsymbol{Z}_{p-1}$ erzeugt, und q die Ordnung dieser Untergruppe; (p, q, α) sind zufällig gewählt von einem Generierungsalgorithmus $G(1^k)$. Gegeben sei weiter ein zufällig gewählter Wert y aus der von α erzeugten Untergruppe von $\boldsymbol{Z}_{p-1}$. Dann gilt für hinreichend große k: Es gibt keinen Polynomzeitalgorithmus, der mit nicht-vernachlässigbarer Erfolgswahrscheinlichkeit $\log_\alpha y$ bestimmt, d.h. ein $x \in \boldsymbol{Z}_q$ findet, sodaß gilt: $\alpha^x \bmod p = y$.

Der Beweis für diese Behauptung wird im folgenden durch Widerspruch geführt: Es wird gezeigt, daß ein Angreifer, dem es mit nicht-vernachlässigbarer Wahrscheinlichkeit gelingt, auch nur eine Signatur zu fälschen, den diskreten Logarithmus $\log_\alpha y$ einer beliebigen, vorgegebenen Zahl y aus der von α erzeugten primen Untergruppe von $\boldsymbol{Z}_{p-1}$ bestimmen kann.

Im Kern verwendet der Beweis einen „simulierten Signierer“, der von einem tatsächlichen Signierer nicht unterschieden werden kann. Wie im Beweis von Goldwasser, Micali und Rivest [GoMR_88] werden zwei verschiedenen Simulationen mit jeweils dem „halben Geheimnis“ konstruiert; zu einer dieser beiden fälscht der Angreifer dann eine Signatur. Daraus läßt sich

dann ein Widerspruch zum DLP herleiten. Die Simulation wird dabei abhängig vom Typ der Signaturfälschung (siehe unten) ausgewählt.

Sei k ein (geeigneter) Sicherheitsparameter. Gegeben seien eine Primzahl p der Länge k bit, die prime Ordnung q einer Untergruppe von $\boldsymbol{Z}_{p-1}^{*}$ und eine Primitivwurzel α, die die multiplikative Untergruppe von $\boldsymbol{Z}_p^{*}$ der Ordnung q erzeugt, sowie ein zufällig gewählter Wert y aus der von α erzeugten primen Untergruppe von $\boldsymbol{Z}_{p-1}$. Dann können die beiden folgenden „Simulationen" eines Signierers konstruiert werden, die von einem realen Signierer ununterscheidbar sind:

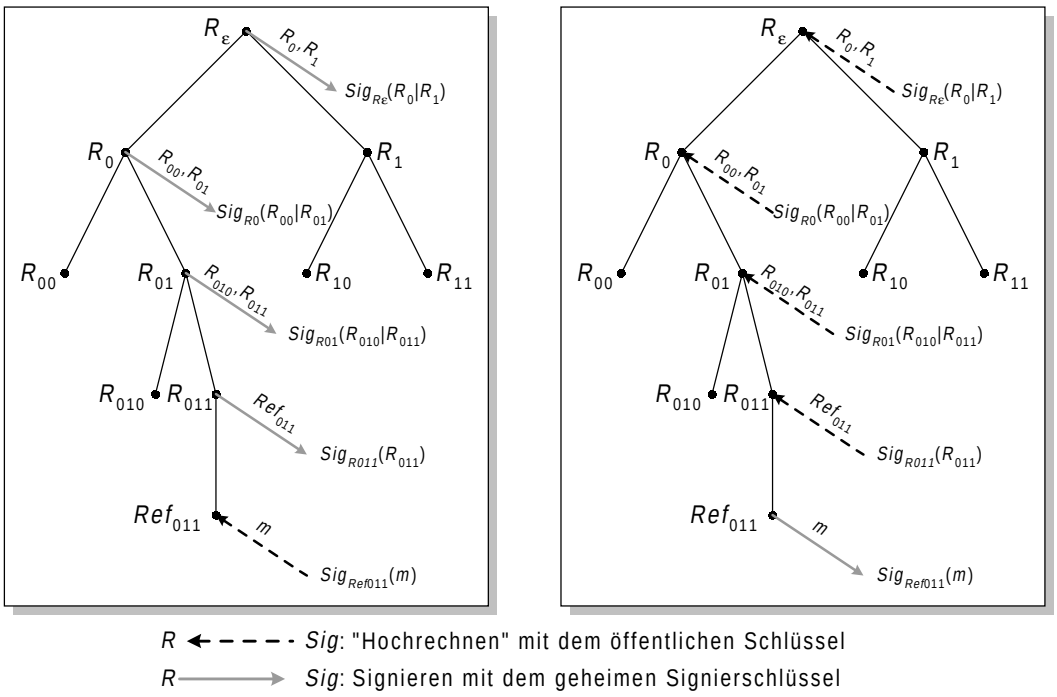

Bild 5-1: „Top down"-Simulation *Bild 5-2: „Bottom up"-Simulation*

„Top down"-Simulation

Für die Konstruktion einer „*top down*"-Simulation wählen wir, wie in der in Kapitel 4 beschriebenen Initialisierungsphase, ein geheimes Schlüsselpaar (k_{SR}, r_ε) und bestimmen daraus die zugehörigen öffentlichen Schlüssel-Werte (k_{VR}, R_ε). Als öffentlichen Schlüssel k_V wählen wir den *zufälligen* Wert y aus der von α erzeugten primen Untergruppe von $\boldsymbol{Z}_{p-1}$ und veröffentlichen (p, α, k_V, k_{VR}, R_ε). Das Ergebnis dieses Schlüsselgenerierungsprozesses ist nicht von dem der in Kapitel 4 beschriebenen Initialisierungsphase zu unterscheiden, da α ein Generator der multiplikativen Untergruppe von $\boldsymbol{Z}_p^{*}$ der Ordnung q ist und daher Zufälligkeit und Verteilung von y und $\alpha^{k_S} \bmod p$ übereinstimmen.

Soll der so „simulierte Signierer" nun eine digitale Signatur zu einer Nachricht m erzeugen, konstruieren wir zunächst, genau wie im Falle eines realen Signieres, den i-ten Pfad im Authentisierungsbaum „*top down*" (siehe Bild 5-1) von der Wurzelreferenz R_ε bis zum i-ten Blatt $R_{d:i}$: Alle benötigten geheimen Referenzen $r_{j:i}$ werden zufällig aus $\boldsymbol{Z}_q^{*}$ gewählt; die zugehörigen Referenzen $R_{j:i}$ werden wie in (4.2) bestimmt und jeweils paarweise nach (4.3) authentisiert.

Abweichend vom Signiervorgang in Kapitel 4 wird nun die Nachrichtensignatur $Sig_{Ref_i}(m)$ zufällig aus $\boldsymbol{Z}_q^*$ gewählt und die dazu passende Nachrichtenreferenz Ref_i wie folgt bestimmt:

$$Ref_i = \alpha^{Sig_{Ref_i}(m)} \cdot k_V^{-hash(m)} \bmod p \qquad (5.1)$$

Unabhängig davon, ob die Berechnung der Referenz Ref_i „aufwärts" oder „abwärts" erfolgt, bleibt die Verteilung von Ref_i erhalten. Schließlich wird Ref_i wie in (4.5) authentisiert.

Die resultierende Signatur ist ununterscheidbar von der eines „realen" Signierers. Der einzige, für den Sicherheitsbeweis wichtige Unterschied ist, daß der Signierer jede Signatur ohne Kenntnis des geheimen Schlüsselteils k_S und der Nachrichtenreferenzen ref_i erzeugt.

„Bottom up"-Simulation

Die zweite Simulation beginnt mit der Wahl von k_S als geheimem Schlüssel und der Berechnung des zugehörigen öffentlichen k_V. Anders als bei einem „realen" Signierer wird als öffentlicher Schlüssel-Parameter k_{VR} der *zufällige* Wert y aus der von α erzeugten primen Untergruppe von $\boldsymbol{Z}_{p-1}$ gewählt. Dies ist, wie bei der „*top down*"-Simulation, ununterscheidbar von der Bestimmung von k_{VR} aus einem zufällig gewählten k_{SR}. Dann wird, abweichend vom Ablauf der Initialisierungsphase eines tatsächlichen Signierers, vor der Veröffentlichung des öffentlichen Schlüssel-Tupels der gesamte Authentisierungsbaum von unten nach oben („*bottom up*") vorausberechnet. Dazu werden zunächst die geheimen Nachrichtenreferenzen ref_i zufällig aus $\boldsymbol{Z}_q^*$ gewählt und die zugehörigen öffentlichen Referenzen Ref_i nach (4.4) bestimmt.

Anschließend werden die „inneren" Referenzen $R_{j:i}$ und die zugehörigen Authentisierungen $Sig_{R_{j:i}}(R_{j:i|0}, R_{j:i|1})$ ohne Kenntnis von k_{SR} und der geheimen Referenzen $r_{j:i}$ des Referenzenbaumes wie folgt bestimmt (siehe auch Bild 5-2): Für jede Nachrichtenreferenz Ref_i wird eine Signatur $Sig_{R_{d:i}}(Ref_i)$ zufällig aus $\boldsymbol{Z}_q^*$ gewählt und R_i daraus berechnet:

$$R_{d:i} = \alpha^{Sig_{R_{d:i}}(Ref_i)} \cdot k_{VR}^{-hash(Ref_i)} \bmod p \qquad (5.2)$$

Schließlich werden für $j = d\text{-}1, ..., 0$ jeweils paarweise zwei Referenzen $R_{j:i|0}$ und $R_{j:i|1}$ „aufwärts" signiert, indem die Signatur $Sig_{R_{j:i}}(R_{j:i|0}, R_{j:i|1})$ zufällig gewählt und die zugehörige Referenz $R_{j:i}$ daraus folgendermaßen berechnet wird:

$$R_{j:i} = \alpha^{Sig_{R_{j:i}}(R_{j:i|0}, R_{j:i|1})} \cdot k_{VR}^{-hash(R_{j:i|0}|R_{j:i|1})} \bmod p \qquad (5.3)$$

Der letzte Berechnungsschritt (für $j = 0$) liefert die Wurzelreferenz R_ε, die nun zusammen mit den anderen Schlüsselparametern als öffentlicher Schlüssel (p, α, k_V, k_{VR}, R_ε) veröffentlicht werden kann.

Diese Schlüsselgenerierung ist ununterscheidbar von der in Kapitel 4 beschriebenen Initialisierungsphase, da die Verteilung der zufällig gewählten Signaturen $Sig_{R_{j:i}}$ durch (5.2) bzw. (5.3) nicht verändert wird; die Referenzen $R_{j:i}$ sind also ebenso zufällig wie bei einer Berechnung aus zufällig gewählten $r_{j:i}$.

Soll nun eine i-te Nachricht m signiert werden, sucht der „simulierte Signierer" alle auf dem i-ten Pfad von der Wurzelreferenz R_ε bis zur Nachrichtenreferenz Ref_i liegenden Referenzen und zugehörigen Signaturen heraus. Dies ist ebenfalls ununterscheidbar von einem „realen" Signierer, denn die Verteilung der Signaturen und Referenzen ändert sich nicht durch die Vorausberechnung. Abschließend wird die Nachrichtensignatur $Sig_{Ref_i}(m)$ zur Referenz Ref_i gemäß (4.6) berechnet und die gesamte Signatur ausgegeben.

Die resultierende Signatur ist nicht unterscheidbar von der eines tatsächlichen Signierers. Der für den Sicherheitsbeweis wichtige Unterschied ist, daß der Signierer weder den geheimen Schlüsselteil k_{SR} noch eine der Referenzen $r_{j:i}$ kennt.

Nun können wir den folgenden zentralen Satz beweisen (die Umkehrung gilt trivialerweise):

> **Theorem**: Das in Kapitel 4 vorgestellte digitale Signatursystem ist existentiell unfälschbar selbst unter einem adaptiven, aktiven Angriff, wenn das Diskrete-Logarithmus-Problem gilt.

Beweis: Angenommen, es gäbe einen probabilistischen Polynomzeitalgorithmus ***A***, der mit nicht-vernachlässigbarer Erfolgswahrscheinlichkeit $P(k) \geq 1/Q(k)$ eine DLP-GMR-Signatur (für unendlich viele $k > k_0$ und ein Polynom Q) fälschen kann.

Es wird nun geziegt, daß wir dann einen probabilistischen Polynomzeitalgorithmus ***B*** konstruieren können, der (unter Verwendung von ***A***) den diskreten Logarihmus eines beliebigen, vorausgewählten Wertes y aus der von α erzeugten primen Untergruppe von $\boldsymbol{Z}_{p-1}$ mit nicht-vernachlässigbarer Erfolgswahrscheinlichkeit bestimmt. Dies ist ein Widerspruch zu unserer Komplexitätsannahme, dem DLP.

Hat Algorithmus ***A*** eine DLP-GMR-Signatur gefälscht, muß einer der folgenden drei „Fälschungstypen“ vorliegen:

- **Typ 1**: Die Signatur enthält eine Nachrichtensignatur $Sig_{Ref_i}(m)$ zu einer Nachricht m, die noch nicht signiert wurde, bezüglich einer Nachrichtenreferenz Ref_i, die bereits authentisiert wurde. Das bedeutet, daß bereits eine Nachricht bezüglich Ref_i signiert wurde.

- **Typ 2**: Die Signatur enthält **entweder** die Authentisierung $Sig_{R_{j:i}}(R_{j:i|0}, R_{j:i|1})$ zweier Referenzen $R_{j:i|0}$ und $R_{j:i|1}$ (mit $0 \leq j < d$) **oder** $Sig_{R_{d:i}}(Ref_i)$, jeweils bezüglich einer bereits authentisierten Referenz $R_{j:i}$ im Authentisierungsbaum. Dabei ist es für den weiteren Beweis unerheblich, ob $R_{j:i}$ bereits für eine Authentisierung „genutzt“ wurde oder noch „frei“ ist.

- **Typ 3**: Der Fälscher konnte eine Kollision für die Hashfunktion *hash*( ) finden und so eine Nachrichtensignatur für eine neue Nachricht m^* oder (im Authentisierungsbaum) eine Authentisierung für eine neue Referenz Ref_i^* oder $R_{j:i}^*$ verwenden.

Daher gilt für den Fälschungsalgorithmus ***A*** einer der folgenden drei Fälle:

- **Fall 1**: Für unendlich viele k kann Algorithmus ***A*** Signaturen durch einen Typ-1-Fälschungsangriff mit einer Erfolgswahrscheinlichkeit $P_1(k) > 1/3 \cdot Q(k)$ fälschen.

- **Fall 2**: Für unendlich viele k kann Algorithmus ***A*** Signaturen durch einen Typ-2-Fälschungsangriff mit einer Erfolgswahrscheinlichkeit $P_2(k) > 1/3 \cdot Q(k)$ fälschen.

- **Fall 3**: Für unendlich viele k kann Algorithmus ***A*** Signaturen durch einen Typ-3-Fälschungsangriff mit einer Erfolgswahrscheinlichkeit $P_3(k) > 1/3 \cdot Q(k)$ fälschen.

Trifft Fall 3 zu, liegt direkt ein Widerspruch zum DLP vor, sofern eine Hashfunktion eingesetzt wurde, für die die Äquivalenz von Kollisionsresistenz und DLP nachgewiesen ist, z.B. eine der von Damgård vorgeschlagenen Funktionen [Damg_87].[10]

---

[10] Wird aus Effizienzgründen eine andere Hashfunktion (z.B. RIPEMD-160, SHS-1 [DoBP_96, NIST_95]) verwendet, dann beruht die Sicherheit des Verfahrens auch auf der Kollisionsresistenz dieser Hashfunktion.

Für die Fälle 1 und 2 können wir nun, abhängig vom jeweils vorliegenden Fall, zwei Varianten des folgenden probabilistischen Polynomzeitalgorithmus konstruieren, der mit nichtvernachlässigbarer Wahrscheinlichkeit diskrete Logarithmen in $\boldsymbol{Z}_p^*$ bestimmt:

Algorithmus *B*:

- **Eingabe**: Tripel (p, q, α) aus einer Primzahl p der Länge k bit, einer Primitivwurzel α (erzeugendes Element der multiplikativen Untergruppe der Ordnung q in $\boldsymbol{Z}_p^*$) und der Ordnung q dieser Untergruppe; sowie ein (beliebiger) Wert y aus der von α erzeugten primen Untergruppe von $\boldsymbol{Z}_{p-1}$.

Wie in der Initialisierungsphase (siehe Kapitel 4) wählen wir nun eine Baumtiefe d. Nun lassen wir, abhängig vom vorliegenden Fall, den Fälschungsalgorithmus ***A*** gegen einen der simulierten Signierer laufen:

- **Fall 1**: Der Fälschungsalgorithmus ***A*** wird gegen die „*bottom up*"-Simulation gestartet.

Nach der Initialisierungsphase der Simulation liefert der „simulierte Signierer" dem Angreifer (Algorithmus ***A***) endlich viele Signaturen ($< 2^d$) zu von diesem gewählten Nachrichten. Liefert ***A*** nun eine Signaturfälschung von Typ 1, dann gilt für eine i-te Nachrichtensignatur $Sig_{Ref_i}(m_1)$ bezüglich einer Referenz Ref_i zu einer (bisher nicht signierten) Nachricht m_1:

$$Ref_i = \alpha^{Sig_{Ref_i}(m_1)} \cdot y^{-hash(m_1)} \bmod p \qquad (5.4)$$

Da die Referenz Ref_i schon authentisiert wurde, muß die Simulation zuvor schon eine Nachricht m_2 bezüglich Ref_i signiert haben; es gilt also:

$$Ref_i = \alpha^{Sig_{Ref_i}(m_2)} \cdot y^{-hash(m_2)} \bmod p \qquad (5.5)$$

Damit folgt:

$$\log_\alpha y = \frac{Sig_{Ref_i}(m_1) - Sig_{Ref_i}(m_2)}{hash(m_1) - hash(m_2)} \qquad (5.6)$$

- **Fall 2**: Der Fälschungsalgorithmus ***A*** wird gegen die „*top down*"-Simulation gestartet.

Nach der Initialisierungsphase liefert die Simulation dem Angreifer endlich viele Signaturen ($< 2^d$) zu von ***A*** gewählten Nachrichten. Bestimmt ***A*** eine Signaturfälschung von Typ 2, dann gilt **entweder** für eine Referenz $R_{j:i}$ auf dem Pfad einer i-ten Signatur im Authentisierungsbaum:

$$R_{j:i} = \alpha^{Sig_{R_{j:i}}(R_{j:i|0}^*, R_{j:i|1}^*)} \cdot y^{-hash(R_{j:i|0}^* | R_{j:i|1}^*)} \bmod p \qquad (5.7)$$

bezüglich zweier „neuer", d.h. nicht von der Simulation bezüglich $R_{j:i}$ authentisierter Referenzen $R_{j:i|0}^*$ und $R_{j:i|1}^*$, **oder**: für eine Referenz $R_{d:i}$ am i-ten Blatt des Authentisierungsbaums:

$$R_{d:i} = \alpha^{Sig_{R_{d:i}}(Ref_i^*)} \cdot y^{-hash(Ref_i^*)} \bmod p \qquad (5.8)$$

bezüglich einer „neuen", d.h. vom simulierten Signierer nicht bezüglich R_i authentisierten Nachrichtenreferenz Ref_i^*. Da die Simulation in der Initialisierungsphase bereits den gesamten Authentisierungsbaum vorausberechnet hat, existiert in beiden Fällen bereits eine Signatur bezüglich $R_{j:i}$ bzw. $R_{d:i}$. Es gilt also analog Fall 1 **entweder**:

$$\log_\alpha y = \frac{Sig_{R_{j:i}}(R_{j:i|0}^*, R_{j:i|1}^*) - Sig_{R_{j:i}}(R_{j:i|0}, R_{j:i|1})}{hash(R_{j:i|0} | R_{j:i|1}) - hash(R_{j:i|0}^* | R_{j:i|1}^*)} \qquad (5.9)$$

oder:

$$\log_\alpha y = \frac{Sig_{R_{d:i}}(Ref_i^*) - Sig_{R_{d:i}}(Ref_i)}{hash(Ref_i^*) - hash(Ref_i)} \qquad (5.10)$$

- **Ausgabe**: Diskreter Logarithmus $\log_\alpha y$.

Beide Varianten von Algorithmus ***B*** berechnen also für unendlich viele $k > k_0$ und ein Polynom Q diskrete Logarithmen zu einem gegebenen y aus der von α erzeugten primen Untergruppe von $\boldsymbol{Z}_{p-1}$ mit der Wahrscheinlichkeit:

$$P(k) > 1/3 \cdot Q(k) \qquad \text{(q.e.d.)}$$

6 Effiziente Implementierung des DLP-GMR-Systems

Die Speicheranforderungen einer DLP-GMR-Signatur und der Aufwand für das Signieren einer Nachricht lassen sich mit einigen Optimierungen gegenüber einer „naiven" Implementierung signifikant verringern.

Signieren

Durch Vorausberechnung aller Referenzen $R_{j:i}$ des Authentisierungsbaums aus zufällig gewählten $r_{j:i}$ kann der größte Teil des Rechenaufwands bei der Bestimmung einer digitalen Signatur, nämlich je Signatur bis zu $2d+1$ Exponentiationen, eingespart werden. Der Speicheraufwand für die vorausberechneten $2^{d+1}-1$ Referenzen $R_{j:i}$ und 2^d Referenzen $r_{d:i}$ liegt bei:

$$(2^{d+1}-1)\cdot k + 2^d \cdot |q| \text{ bit} \qquad (6.1)$$

Jede Referenz $R_{j:i}$ im Authentisierungsbaum kann darüberhinaus 2^{d-j}-mal verwendet werden. Durch die Speicherung von maximal d Signaturen $Sig_{R_{j:i}}(R_{j:i|0}|R_{j:i|1})$ der Länge k bit kann der Rechenaufwand weiter reduziert werden: Durchschnittlich sind je Signatur, wie in [FoPf_91] für das GMR-Signatursystem gezeigt wurde, nur die Berechnung von drei Authentisierungen, zusammen also drei Multiplikationen in $\boldsymbol{Z}_q^*$ erforderlich.[11]

Berechnet der Signierer auch noch die Nachrichtenreferenz Ref_i und deren Authentisierung $Sig_{R_{d:i}}(Ref_i)$ voraus, z.B. in „*idle*"-Zuständen des Rechners, genügt die *online*-Bestimmung der Nachrichtensignatur $Sig_{Ref_i}(m)$ bezüglich Ref_i – eine einzige Multiplikation in $\boldsymbol{Z}_q^*$:

$$1\ \mathrm{M}_{|q|} \qquad (6.2)$$

Der Speicherbedarf für die vorausberechneten Werte kann erheblich reduziert werden, denn es genügt, nur jeweils die für die nächste Signatur erforderlichen Referenzen und Authentisierungen vorauszuberechnen und zu speichern. Das sind zusammen d Referenzenpaare ($R_{j:i|0}$, $R_{j:i|1}$); hinzu kommen maximal d, durchschnittlich $d/2$ Referenzen $r_{j:i}$, die noch keine Nachfolgeknoten besitzen, und daher für deren Authentisierung aufbewahrt werden müssen, sowie die Nachrichtenreferenzen ref_i und Ref_i. Weiter werden insgesamt $d+1$ Authentisierungen vorausberechnet und gespeichert. Damit summiert sich der (durchschnittliche) Speicherbedarf beim Signierer (ohne geheime Schlüssel) auf:

$$(3d+2)\cdot k + (d/2+1)\cdot |q| \text{ bit} \qquad (6.3)$$

[11] Es ist leicht zu sehen, daß für alle Signaturen mit ungeradem i, also jede zweite, die Authentisierungen aller Referenzen inklusive der Blätter des Authentisierungsbaums wiederverwendet werden können; nur Ref_i und die Nachricht m müssen *online* signiert werden.

Verifizieren

Wie von Goldreich für das GMR-Signatursystem vorgeschlagen und auch im Verfahren von Cramer und Damgård genutzt, kann die Länge einer Signatur weiter verringert werden [Gold_86, CrDa_96]: Durch eine geringfügige Änderung der Signaturprüfung erhält der Empfänger alle Referenzen auf dem Authentisierungspfad, $R_{j:i}$ (für $j = 0, ..., d$), $R_{d:i}$ und Ref_i, als Zwischenergebnisse der Prüfung (ähnlich der „*bottom up*"-Konstruktion der Simulationen in Kapitel 5):

$$Ref_i = \alpha^{Sig_{Ref_i}(m)} \cdot k_V^{-hash(m)} \bmod p \qquad (6.4)$$

$$R_{d:i} = \alpha^{Sig_{R_{d:i}}(Ref_i)} \cdot k_{VR}^{-hash(Ref_i)} \bmod p \qquad (6.5)$$

$$R_{j:i} = \alpha^{Sig_{R_{j:i}}(R_{j:i|0}, R_{j:i|1})} \cdot k_{VR}^{-hash(R_{j:i|0}|R_{j:i|1})} \bmod p \qquad (6.6)$$

Zuletzt vergleicht der Empfänger $R_{0:i}$ mit dem öffentlichen R_ε. Daher genügt es, statt der Referenzenpaare in einer Signatur nur die Referenz mitzuschicken, bezüglich der keine nachfolgenden Referenzen authentisiert werden. Die Länge einer Signatur reduziert sich damit auf:

$$2 \cdot (d+1) \cdot |q| \text{ bit} \qquad (6.7)$$

Die Berechnung der in (6.4), (6.5) und (6.6) je Referenz erforderlichen „Doppelexponentiation" kann ebenso wie beim DSS wesentlich effizienter als die Berechnungen (4.9), (4.10) und (4.11) implementiert werden [Fox_93].

Implementierung auf elliptischen Kurven

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 (k = 160 statt 1024 bit, d.h. Faktor 6) arbeiten und erlauben so sehr effiziente Implementierungen [FoRö_96].

Varianten des vorgeschlagenen Signatursystems

Die Signierfunktion des vorgeschlagenen Signatursystems verwendet eine Variante des ElGamal-Verfahrens. Eine Übersicht und Verallgemeinerung von verschiedenen Varianten des ElGamal-Verfahrens wurde 1994 von Horster, Petersen und Michels veröffentlicht [HoPM_94]. Durch eine „Permutation" der Exponenten kann das vorgeschlagene Verfahren auf ebensolche Weise verallgemeinert werden, ohne seine Sicherheitseigenschaft zu verlieren.

Var.	Signaturberechnung	Aufwand	Signaturprüfung	Aufwand
1	$Sig_R(m)=(m-k_S)\cdot r^{-1}$	$I_{\|q\|} + M_{\|q\|}$	$\alpha^m = k_V \cdot R^{Sig_R(m)}$	$2\,E_k + M_k$

2	$Sig_R(m)=(m-r)\cdot k_S^{-1}$	$M_{\|q\|}$	$\alpha^m = k_V^{Sig_R(m)} \cdot R$	$2\,E_k + M_k$
3	$Sig_R(m)= k_S + r \cdot m$	$M_{\|q\|}$	$\alpha^{Sig_R(m)} = k_V \cdot R^m$	$2\,E_k + M_k$
4	$Sig_R(m)=(1-k_S \cdot m)\cdot r^{-1}$	$I_{\|q\|} + 2\,M_{\|q\|}$	$\alpha = k_V^{\,m} \cdot R^{Sig_R(m)}$	$2\,E_k + M_k$
5	$Sig_R(m)=(1-r \cdot m)\cdot k_S^{-1}$	$2\,M_{\|q\|}$	$\alpha = k_V^{Sig_R(m)} \cdot R^m$	$2\,E_k + M_k$

Tabelle 6-1: Varianten des vorgeschlagenen digitalen Signatursystems

Tabelle 6-1 zeigt fünf Varianten der in Kapitel 4 vorgestellten Signierfunktion. Die letzte Spalte gibt jeweils die erforderlichen arithmetischen Operationen an. Der tatsächliche Aufwand der einzelnen Operationen hängt von dem jeweils verwendeten endlichen Körper ab; das Verhältnis zwischen den einzelnen Operationen unterscheidet sich beispielsweise in $GF(p)$, $GF(2^n)$, $E(GF(p))$ und $E(GF(2^n))$ erheblich; siehe z.B. [Mene_93, FoRö_96]. [12]

Die multiplikativen Inversen, die für die Signaturberechnung in den Varianten 1 und 4 benötigt werden, können vorausberechnet werden. Die Berechnung der multiplikativen Inversen des geheimen Schlüssels k_S wurde nicht berücksichtigt, da diese während der Schlüsselgenerierung einmalig vorab erfolgen kann; es genügt für diese Varianten, k_S^{-1} statt k_S zu speichern.

7 Vergleichende Zusammenfassung und Ausblick

Das in Kapitel 4 vorgestellte DLP-GMR-Signatursystem ist auch bei adaptiven aktiven Angriffen existentiell unfälschbar; die Äquivalenz von Signaturfälschung und Lösung des DLP wurde in Kapitel 5 bewiesen. Kapitel 6 zeigt, daß das Verfahren sehr effiziente Implementierungen erlaubt: Mit Vorausberechnungen läßt sich die Erzeugung einer Signatur auf eine modulare Multiplikation reduzieren. Ein für ausgewählte Anwendungen („Scheckbuch“, „TAN-Liste“) möglicherweise interessanter Aspekt ist die endliche Zahl von Signaturen je Schlüsselpaar.

Verfahren	**DSS**	**RSA**	**GMR**	**DN**	**CD**	**DLP-GMR**
Signaturen	beliebig	belieb.	2^d	k^d	l^d	2^d
Signieren	$2\,M_{\|q\|}$	E_k	$E_k + M_k$	$E_k + 80\,M_k$	$2\cdot E_k + M_k$	$M_{\|q\|}$
$\|k_S\|$ [bit]	k bit	k bit	$3(d+1)\cdot k$ bit	$(d+k)\cdot k$ bit	$(d+l+1)\cdot k$ bit	$3(d+4)\cdot k$ bit
$\|Sig\|$ [bit]	$2\|q\|$ bit	k bit	$2(d+1)\cdot k$ bit	$d\cdot k$ bit	$(d+1)\cdot k$ bit	$2(d+1)\cdot\|q\|$ bit
Prüfen	$2\,E_k + I_{\|q\|} + 2\,M_{\|q\|}$	E_k	$2(d+1)\cdot k\,M_k$	$d\cdot E_k + d\cdot 80\,M_k$	$(d+1)\cdot(2E_k + M_k)$	$2(d+1)\cdot E_k + (d+1)\cdot M_k$
$\|k_V\|$ [bit]	$3k+\|q\|$ bit	$2\cdot k$ bit	$2\cdot k$ bit	$2(k+1)\cdot k$ bit	$3\cdot k$ bit	$5\cdot k$ bit

Tabelle 7-1: Aufwandsvergleich digitaler Signatursysteme

Tabelle 7-1 gibt den durchschnittlichen Aufwand des DLP-GMR-Signatursystems im Vergleich zu den in Kapitel 3 vorgestellten digitalen Signatursystemen, dem RSA-Verfahren und

[12] *GF* bezeichnet ein Galois-Feld, *E* eine Elliptische Kurve über einem endlichen Körper.

dem DSS an. Dabei sind die Kosten für die Generierung der Zufallswerte und der Hashfunktion nicht berücksichtigt – bei Verwendung kryptographisch starker Verfahren kann dies deutlichen Zusatzaufwand verursachen [BlBS_86, Damg_87]. Vorausgenerierte Zufallszahlen und die Hashfunktionen SHS-1 und RIPEMD-160 fallen hingegen nicht ins Gewicht.

Tabelle 7-2 gibt Ergebnisse von Testläufen prototypischer Implementierungen der betrachteten Verfahren unter Verwendung einer optimierten Langzahlarithmetik [Fox_91] auf einem 166 MHz-Pentium-PC wieder. Die verwendete Implementierung des CD-Verfahrens nutzt nicht alle möglichen Optimierungen; das RSA- und das DN-Signatursystem arbeiten mit nur 32 bit langen öffentlichen Exponenten. Die Messungen des DLP-GMR-Verfahrens beziehen sich auf eine Implementierung über GF(p) – von einer Implementierung über E(GF(q)) dürfen weitere Effizienzsteigerungen erwartet werden.

Verfahren	**DSS**	**RSA**	**GMR**	**DN**	**CD**	**DLP-GMR**
k [bit]	768	1024	1024	1024	1024	768
Signaturen	bel.	bel.	$2^{10} = 1024$	$l^1 = 1024$	$l^1 = 1024$	$2^{10} = 1024$
Sign [s]	« 0,01	0,39	0,39	0,45	0,77	« 0,01
$\|k_S\|$ [kbyte]	0,09	0,13	4,13	128	128,25	3,94
$\|Sig\|$ [byte]	40	128	2.816	128	256	440
Verify [s]	0,52	0,06	25,8	0,17	5,7	9,1
$\|k_V\|$ [kbyte]	0,3	0,25	0,25	256,25	0,38	0,47

Tabelle 7-2: Messergebnisse einer PC-Implementierung (Pentium-PC, 166 MHz)

Dank

Birgit Pfitzmann verdanke ich die entscheidende Anregung zu dem hier vorgeschlagenen Verfahren und sehr viele wertvolle und hilfreiche Diskussionen und Kommentare zu Vorfassungen dieses Beitrags. Für viele kleine, wichtige Korrekturen gilt mein Dank Heike Neumann. Die Implementierung des DN-Verfahrens besorgte Niko Schweitzer, und das CD-Verfahren programmierte Wasili Wasilewski.

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.

DwNa_95 Dwork, Cynthia; Naor, Moni: *An Efficient Existentially Unforgeable Signature Scheme and ist Applications*. Technical Report CS95-28, Weizmann Institute, 1995.

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, Verlag Vieweg, Braunschweig 1993, S. 333-352.

Fox1_97 Fox, Dirk: *Fälschungssicherheit digitaler Signaturen*. Datenschutz und Datensicherheit (DuD) 2/97, S. 5-10.

Fox2_97 Fox, Dirk: *Signaturschlüssel-Zertifikat*. Gateway, Datenschutz und Datensicherheit (DuD), 2/97, S. 106.

Fox3_97 Fox, Dirk: *Sichere digitale Signatursysteme*. In: Bundesamt für Sicherheit in der Informationstechnik (Hrsg.): Tagungsband 5. Deutscher IT-Sicherheitskongreß des BSI, SecuMedia Verlag, 1997, S. 61-76.

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.

HoPM_94 Horster, Patrick; Petersen, Holger; Michels, Markus: *Meta-ElGamal signature schemes*. Proceedings of the 2nd ACM Conference on Computer and Communications Security, Fairfax, 1994, S. 96-107.

IuKD_96 *Informations- und Kommunikationsdienste-Gesetz (IuKD-G)*. Beschluß des Bundestages vom 13.06.97, Bundestagsdrucksache 13/7934.

Mene_93 Menezes, Alfred J.: *Elliptic Curve Public Key Cryptosystems*. SECS 234, Kluwer Academic Publishers, Massachusetts, 1993.

Merk_87 Merkle, Ralf C.: *A Digital Signature based on a Conventional Encryption Function*. In: Pomerance, C. (Hrsg.): Proceedings of Crypto ‘87, LNCS 293, Springer, Berlin 1998, S. 369-378.

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.

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.