GI-Fachtagung Verläßliche Informationssysteme (VIS'91), März 1991, Darmstadt, Informatik-Fachberichte 271, Springer-Verlag, Heidelberg 1991, 289-301

Fail-stop-Signaturen und ihre Anwendung

Birgit Pfitzmann, Michael Waidner

Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe
Postfach 6980, D-W7500 Karlsruhe 1, Bundesrepublik Deutschland

Kurzfassung: Die Unfälschbarkeit konventioneller digitaler Signaturen beruht zwangsläufig auf komplexitätstheoretischen Annahmen, d.h. selbst die sichersten Systeme können durch einen unerwartet mächtigen Angreifer gebrochen werden. Daher führen wir Fail-stop-Signaturen ein: Sie sind so unfälschbar wie die besten konventionellen Signaturen, aber wenn doch eine Signatur gefälscht wird, kann der angebliche Unterzeichner unbedingt (d.h. ohne jegliche Annahmen) die Fälschung beweisen, mit beliebig hoher Wahrscheinlichkeit.

Wir konstruieren konkrete Fail-stop-Signatursysteme, die sogenannten Versteckssysteme, aus beliebigen kollisionsfreien Paaren von Permutationen. Als Spezialfall ergibt sich ein relativ praktikables System, in dem Fälschen so schwer ist wie Faktorisierung.

Ausführlich werden Anwendungen in digitalen Zahlungssystemen betrachtet, auf Anwendungen auf zuverlässige Verteilung wird verwiesen.

1 Einleitung

Motivation: Digitale Signaturen sind ein wichtiger kryptograhischer Grundbaustein für sichere Systeme [DiHe_76]. Sie sind dadurch gekennzeichnet, daß der Empfänger einer Signatur jeden beliebigen Dritten von ihrer Gültigkeit überzeugen kann (im Gegensatz zu Authentikationscodes [GiMS_74], wo der Empfänger nur für sich selbst sicher ist, daß eine Nachricht vom behaupteten Unterzeichner stammt). Deswegen sind sie insbesondere in allen Anwendungen nötig, wo Rechtssicherheit herrschen soll, d.h. unterschriebene Nachrichten evtl. einem Gericht vorgelegt werden müssen, z.B. in digitalen Zahlungssystemen.

Die Sicherheit digitaler Signaturen hat in letzter Zeit einige Aufmerksamkeit erfahren, denn das bekannteste System RSA [RSA_78] ist bekanntlich in reiner Form sehr anfällig gegen aktive Angriffe (d.h. im wesentlichen das Bilden neuer Signaturen aus vorher erhaltenen, ohne den eigentlichen Schlüssel zu brechen) (s. z.B. [Denn_84]), so daß z.B. zu Redundanzprädikaten gegriffen werden muß, über deren Sicherheit noch weniger bekannt ist als über RSA selbst [DaPr_85, EkHV_88, Gira_88, JoCh_86]. Gleiches gilt für das System aus [Elga_85], s. [GoMR_88]. Manche anderen Systeme wurden ganz gebrochen (s. z.B. [Odly_84, EAKM_86, BrDe_86]).

In [GoMR_88] wurde daher die optimale Sicherheit für (konventionelle) digitale Signatursysteme definiert und Systeme mit dieser Sicherheit konstruiert. Mit **„konventionelles Signatursystem“** bezeichnen wir eines gemäß dieser Definition (darunter fällt auch RSA mit Redundanzprädikat, nur ist eben seine Sicherheit bisher nicht bewiesen). Insbesondere veröffentlicht jeder Teilnehmer einen öffentlichen Schlüssel $\ddot{O}$, und ein S gilt als seine Signatur unter eine Nachricht N genau dann, wenn es einen Test $test(\ddot{O}, N, S)$ besteht. Dies impliziert, daß das Fälschen von Signaturen komplexitätstheoretisch innerhalb von NP liegt.

Darüber hinaus beruhen alle bekannten Signatursysteme auf allgemein für richtig gehaltenen, aber unbewiesenen komplexitätstheoretischen Annahmen. Der ursprüngliche Vorschlag aus [GoMR_88], GMR genannt, beruht auf der Existenz kollisionsfreier Paare von (Einweg-)Permutationen mit Falltür (claw-free pairs of trap-door permutations). Diese Annahme konnte zunächst auf beliebige Falltür-Einweg-Funktionen (trap-door one-way functions) abgeschwächt werden [BeMi_88], später auf beliebige Einweg-Funktionen [NaYu_89]. Um Signaturen praktisch zu verwenden, muß man eine spezielle Funktion auswählen und

hoffen, daß sie die Einweg-Eigenschaft besitzt. Meist nimmt man die Schwierigkeit des Faktorisierens oder des diskreten Logarithmus in gewissen Gruppen an. (Das effizienteste System ist immer noch der auf Faktorisierung beruhende Spezialfall von GMR aus [GoMR_88]. Sein Aufwand ist vergleichbar mit dem von RSA.)

Fail-stop-Signaturen verbessern die Sicherheit auf andere Weise: Sie ermöglichen es angeblichen Unterzeichnern unbedingt (d.h. ohne jegliche Annahme, nicht einmal die, daß Angreifer nur polynomiale Algorithmen ausführen können), Fälschungen zu beweisen (mit beliebig hoher Wahrscheinlichkeit). Vorteile sind:

- Sobald eine Fälschung auftritt, kann das Signatursystem abgebrochen oder der Sicherheitsparameter erhöht werden.
- Das Risiko gefälschter Signaturen kann beliebig aufgeteilt werden. Meist wird man festlegen, daß eine Signatur ungültig wird, sobald ein Fälschungsbeweis gezeigt wird. Dann sind die Unterzeichner unbedingt sicher.

Wir fordern auch, daß unter kryptographischen Annahmen Fälschungen überhaupt nicht auftreten können, und Unterzeichner nicht fälschlich behaupten können, eine Fälschung sei aufgetreten. Somit ist die Sicherheitsdefinition von Fail-stop-Signaturen echt stärker als die eines konventionellen Signatursystems.

ANMERKUNGEN: Es gab Vorschläge für konventionelle Signaturen, die besagten, daß Unterzeichner ja merken, wenn ihre Signaturen gefälscht werden, oder auch wenn sie ihre geheimen Schlüssel verlieren, und daß sie in diesen Fällen das Signatursystem abbrechen dürfen. Dies würde jedoch vollständige Unsicherheit für die Empfänger von Signaturen bedeuten, da ein Unterzeichner einfach behaupten könnte, eine Fälschung sei aufgetreten, um eine lästig gewordene Signatur abzuleugnen.

Fail-stop-Signaturen haben zusätzliche praktische Vorteile in öffentlichen Anwendungen wie Zahlungssystemen: Erstens ist es zweifelhaft, ob man Personen verpflichten könnte, an Systemen teilzunehmen, wo sie sich auf kryptographische Annahmen verlassen müssen, dazu i.allg. mit Sicherheitsparametern, die sie nicht selbst wählen dürfen. Benutzt man Fail-stop-Signaturen und konventionelle Signaturen zusammen (s. Kap. 3), so kann man den Kunden solcher Systeme unbedingte Sicherheit garantieren.

Zweitens könnten bei konventionellen Signaturen selbst dann, wenn die kryptographischen Annahmen zutreffen, Kunden unwiderlegbar behaupten, ihre Signatur sei gefälscht worden, was beträchtliche öffentliche Unsicherheit auslösen könnte.

Verwandte Arbeiten: Parallel zu dieser Arbeit wurden „unbedingt sichere Signaturen“ („unconditionally secure signatures“) entwickelt [ChRo_90]. Dies ist eine vorzügliche Idee, aber diese Signaturen unterscheiden sich ziemlich von konventionellen: Jeder Teilnehmer hat eine andere Testfunktion für Signaturen, und sie hängt zusätzlich davon ab, entlang wievieler Vorgänger eine Signatur empfangen wurde (d.h. ob direkt vom Unterzeichner, ob vom ersten Empfänger usw.). Zudem sind aktive Angriffe auf Empfänger möglich. Deswegen können unbedingt sichere Signaturen, im Gegensatz zu Fail-stop-Signaturen, nur in speziellen Protokollen eingesetzt werden, wo jeder Empfänger die Anzahl seiner Vorgänger kennt und wo aktive Angriffe auf Empfänger eingeschränkt werden. Das Hauptsystem aus [ChRo_90] entspricht dem Fall mit nur einem beabsichtigten Empfänger bei Fail-stop-Signaturen (s. Kap. 2). Es ist bisher nicht bekannt, wie lang die Signaturen im allgemeinen Fall sein müssen. (Die Maßnahmen aus [ChRo_90, Kap. 5] helfen nur, wenn man sicher ist, daß der Unterzeichner ehrlich ist.) In vielen Fällen werden Fail-stop-Signaturen effizienter sein als unbedingt sichere Signaturen. Zudem ist der Schlüsselaustausch für unbedingt sichere Signaturen ein komplexes Protokoll.

Eine genaue Sicherheitsdefinition und der Beweis unserer Fail-stop-Signaturen finden sich in [Pfit_89, PfWa_90], unsere BA-Protokolle (und ein erstes, aber ineffizientes Fail-stop-Signatursystem) in [WaPf_89, PfWa1_91]. Ein spezieller Aspekt der Schlüsselerzeugung wurde in [Bleu_90, BlPW_90] diskutiert.

2 Konstruktion von Fail-stop-Signaturen

2.1 Grundidee

Die Grundidee für Fail-stop-Signaturen ist, daß es bei gegebenem öffentlichen Schlüssel $\ddot{O}$ viele mögliche geheime Schlüssel G gibt, die zu $\ddot{O}$ gehören könnten; und daß für jede Nachricht N diese verschiedenen geheimen Schlüssel viele verschiedene mögliche Signaturen ergeben (selbst nach einem adaptiven aktiven Angriff, d.h. nachdem der Angreifer mehrere Signaturen seiner Wahl erhalten hat). Ein Unterzeichner hat jedoch davon nur einen einzigen geheimen Schlüssel G^* und kann nur eine einzige Signatur S^* effizient berechnen. Das Vorzeigen zweier verschiedener Signaturen zu ein und derselben Nachricht N und ein und demselben öffentlichen Schlüssel $\ddot{O}$ gilt als Fälschungsbeweis. (In konkreten Systemen werden als Fälschungsbeweis Teile dieser zwei Signaturen genügen.) Man sieht leicht, daß ein System mit diesen Eigenschaften ein sicheres Fail-stop-Signatursystem ist:

1. Selbst wenn ein Fälscher *unbeschränkt* viel rechnen kann, ist das äußerste, was er tun kann, alle möglichen Signaturen für eine Nachricht N (bzgl. $\ddot{O}$) zu bestimmen. Er kann jedoch nur raten, welche davon S^* ist. Mit hoher Wahrscheinlichkeit rät er falsch und wählt eine andere Signatur S. Sobald also der angebliche Unterzeichner mit S konfrontiert wird, kann er sowohl S als auch S^* vorzeigen, was als Fälschungsbeweis gilt.
2. Andererseits wird gezeigt werden, daß es *komplexitätstheoretisch* sogar für den Unterzeichner unmöglich ist, einen Fälschungsbeweis zu berechnen, wenn keine Fälschung aufgetreten ist (unter einer kryptographischen Annahme wie der Schwierigkeit des Faktorisierens). Solange also diese Annahme zutrifft, arbeitet das Fail-stop-Signatursystem genau wie ein konventionelles Signatursystem, ohne daß irgendein Fälschungsbeweis auftritt. (Dies impliziert, daß auch keine Fälschungen auftreten, denn jene würden gemäß Punkt 1. zu Fälschungsbeweisen führen.)

Daß es "viele" mögliche Signaturen gibt, heißt z.B. 2^{σ}, wobei der Aufwand des Signatursystems nur polynomial in σ ist.

2.2 Ein konkretes Fail-stop-Signatursystem: das Verstecksystem

Im folgenden stellen wir informell ein spezielles Fail-stop-Signatursystem, *Verstecksystem* genannt, vor. Es beruht auf LAMPORT's *Einmal-Signaturen* [DiHe_76, Merk_88] und einer neuen Sorte Einweg-Funktionen, sogenannten *bündelnden Funktionen*. Zusätzlich wurden mehrere Effizienzverbesserungen für Einmal-Signaturen von MERKLE so angepaßt, daß sie sowohl kryptographische Sicherheit als auch die Möglichkeit für Fälschungsbeweise garantieren. Die Konstruktion wird stufenweise erklärt.

Eine formale Beschreibung findet sich, zusammen mit einem Sicherheitsbeweis, in [Pfit_89, PfWa_90].

2.2.1 Grundbaustein: Bündelnde Funktionen und Einmal-Signaturen

Konventionelle Einmal-Signaturen: Konventionelle Einmal-Signaturen (one-time signatures) waren der allererste Vorschlag für ein digitales Signatursystem. Sie erlauben das Unterschreiben einer a priori festgelegten Anzahl Bits, etwa m. Der Unterzeichner wählt zunächst eine Einweg-Funktion (one-way function) f, i.allg., indem er einen Schlüssel V zu einer Funktionsfamilie wählt (z.B. DES).

Wir schreiben im folgenden Schlüssel immer als obere Indizes, also z.B. f^V für die Funktion f mit Schlüssel V.

Als geheimen Schlüssel G wählt er dann eine Liste von $2 \bullet m$ zufälligen Werten aus dem Definitionsbereich von f^V, etwa

$$G = ((r_{1,0}, r_{1,1}),\ldots, (r_{m,0}, r_{m,1})).$$

Dann berechnet er die Bilder $\ddot{o}_{i,b} := f^V(r_{i,b})$

Der öffentliche Schlüssel besteht aus V und der Liste dieser Bilder:

$$\ddot{O} = (V, (\ddot{o}_{1,0}, \ddot{o}_{1,1}),\ldots, (\ddot{o}_{m,0}, \ddot{o}_{m,1})).$$

Als Signatur, daß das i-te Bit einen bestimmten Wert $b \in \{0, 1\}$ hat, gilt $r_{i,b}$. (Der geheime Schlüssel ist also einfach eine Liste aller potentiellen Bit-Signaturen.) Der Empfänger eines Tripels (i, b, s), wobei s die Signatur sein soll, d.h. im korrekten Fall $s = r_{i,b}$, testet, ob $f^V(s) = \ddot{o}_{i,b}$.

Definition und Einsatz bündelnder Funktionen: Um zunächst Einmal-Signaturen mit Fälschungsbeweisen zu erhalten, definieren und konstruieren wir eine spezielle Art Einweg-Funktionen: Bündelnde Funktionen sind Funktionen, bei denen jedes Bild mindestens 2^σ Urbilder hat, die aber im kryptographischen Sinne kollisionsfrei sind, d.h. kryptographisch ist es unmöglich, zwei Urbilder mit demselben Bild zu finden. Solche Funktionen wurden bereits in [ChFN_88] informell definiert und verwendet, allerdings ohne eine Konstruktion anzugeben.

Verwenden wir eine bündelnde Funktion als f in Einmal-Signaturen, so erhält ein Angreifer aus $\ddot{O}$ keinerlei Information, welches der mindestens 2^σ Urbilder der Unterzeichner ursprünglich als $r_{i,b}$ wählte und zur Berechnung des Bildes $\ddot{o}_{i,b}$ benutzte. Wenn er eine Signatur fälscht, berechnet der Angreifer für mindestens ein $\ddot{o}_{i,b}$ ein Urbild, zu dem der richtige Unterzeichner noch nicht $r_{i,b}$ veröffentlicht hat (weil er Bit i noch gar nicht unterschrieben hat, oder mit dem anderen Wert 1-b). Somit wählt der Angreifer als Signatur für dieses Bit mit Wahrscheinlichkeit mindestens $1\text{-}2^{-\sigma}$ ein Urbild $r_{i,b}^* \neq r_{i,b}$. Sobald man nun dem Unterzeichner die gefälschte Signatur vorlegt, um ihn dafür verantwortlich zu machen, erhält er eine f-Kollision, eben $(r_{i,b}^*, r_{i,b})$ mit demselben Bild $\ddot{o}_{i,b}$ unter f^V. Dies gilt als Fälschungsbeweis.

Umgekehrt folgt aus der Kollisionsfreiheit sofort, daß unter kryptographischen Annahmen nie ein Fälschungsbeweis auftritt. Dazu darf allerdings jetzt V nicht vom Unterzeichner gewählt werden, s. Kap. 2.2.3.

Innerhalb des Signatursystems bezeichnen wir den Schlüssel V der bündelnden Funktion als **Vorschlüssel**, weil er vor den eigentlichen öffentlichen und privaten Schlüsseln gewählt wird. Die Wahl von V hängt ab vom Parameter σ und einem zweiten Sicherheitsparameter k für die kryptographische Sicherheit (d.h. je größer k, desto sicherer ist die Kollisionsfreiheit, aber desto aufwendiger das System). Für σ erscheint ein Wert zwischen 20 und 100 sinnvoll. Die Funktionen müssen effizient zu berechnen sein.

Konstruktion bündelnder Funktionen: Wir konstruieren zunächst bündelnde Funktionen aus beliebigen kollisionsfreien Paaren von Permutationen (claw-free permutation pairs) [GoMR_88]. Bei Verwendung bestimmter kollisionsfreier Paare von Permutationen erhalten wir effiziente bündelnde Funktionen, deren Kollisionsfreiheit auf dem Faktorisierungsproblem beruht.

Hierzu sei zunächst die Definition **kollisionsfreier Paare von Permutationen** skizziert. Dies ist eine Familie von Algorithmen, zu der ein Schlüsselerzeugungsalgorithmus E gehört. Hat man damit einen Schlüssel V gewählt, so sind ${f_0}^V, {f_1}^V$ Permutationen auf demselben Definitionsbereich, etwa D^V. Außerdem ist es effizient möglich, ein Element aus D^V zufällig zu wählen. Kollisionsfreiheit bedeutet, daß es nicht effizient möglich ist, zwei Elemente x, y mit ${f_0}^V(x) = {f_1}^V(y)$ zu finden (obwohl solche existieren).

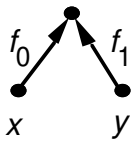

Bild 1: Kollision eines kollisionsfreien Paars von Permutationen (unter einer kryptographischen Annahme nicht zu finden).

Eine **bündelnde Funktion** *bünd* mit demselben Schlüssel V erhalten wir durch Iterierung von ${f_0}^V, {f_1}^V$: Ein Urbild r soll aus zwei Teilen $(string, x)$ bestehen; der erste ist ein Bitstring der Länge σ, etwa $string = (b_1,\ldots,b_\sigma)$, der zweite ein Element von D^V. Das Bild $\ddot{o} = bünd^V(r) = bünd^V((string, x))$ erhält man, indem

man die Bits aus *string* der Reihe nach verwendet, um auszuwählen, ob man als nächstes $f_0{}^V$ oder $f_1{}^V$ anwendet:

$$bünd^V(r) := f_{b_1}{}^V(f_{b_2}{}^V(\ldots(f_{b_\sigma}{}^V(x))\ldots)).$$

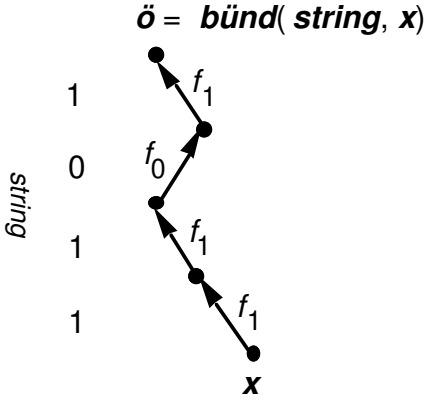

Bild 2: Beispiel zur bündelnden Funktion: $\sigma = 4$, *string* = 1011.

(Die Konstruktion ähnelt den Grundfunktionen in [GoMR_88] und den Hashfunktionen in [Damg_88], aber die Parameter sind jedesmal anders verteilt.)

Sicherheit (Skizze): Einerseits sieht man leicht, daß jedes Bild $\ddot{o}$ mindestens 2^σ Urbilder hat: Es gibt 2^σ Werte *string* der Länge σ. Da $f_0{}^V, f_1{}^V$ Permutationen sind, gibt es zu jedem davon ein x_{string} mit $bünd^V((string, x_{string})) = \ddot{o}$. Dies entspricht in Bild 2, daß von $\ddot{o}$ aus rückwärts ein vollständiger Binärbaum der Tiefe σ konstruiert werden kann.

Andererseits ist zu zeigen, daß man solche Kollisionen nicht effizient finden kann. Dazu nimmt man an, man könnte doch, und zeigt, daß man dann auch Kollisionen von f_0 und f_1 finden könnte (s. Bild 3): Eine Kollision bzgl. *bünd* bedeutet, daß man zwei Wege hat, die sich spätestens bei $\ddot{o}$ treffen. Die Strings von beiden müssen verschieden sein, weil $f_0{}^V, f_1{}^V$ Permutationen sind. Dort wo, von oben her gesehen, die Strings sich zum ersten Mal unterscheiden, befindet sich eine Kollision bzgl. des kollisionsfreien Paars.

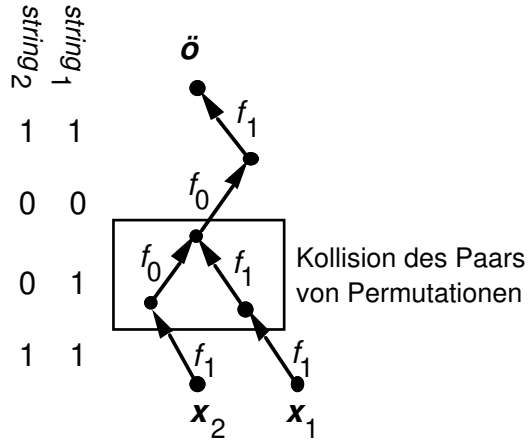

Bild 3: Beispiel, wie eine Kollision der bündelnden Funktion zu einer bzgl. f_0 und f_1 führen würde.

Spezialfall mit Faktorisierungsproblem: Die effizientesten bekannten kollisionsfreien Paare von Permutationen sind folgende aus [GoMR_88]: Schlüssel sind Zahlen $V = p \bullet q$ mit p, q prim und $p \equiv 3$, $q \equiv 7$ mod 8. Stellt man die Restklassen mod V symmetrisch dar, d.h. durch $\{-\frac{V-1}{2}, \ldots, +\frac{V-1}{2}\}$ und definiert darauf die übliche Betragsfunktion, so ist

$f_0{}^V(x) := |x^2|$
$f_1{}^V(x) := |4x^2|$

auf einer Teilmenge dieser Restklassen (den positiven mit Jacobisymbol +1). Es ist bewiesen, daß man jede Kollision verwenden könnte, um V ganz einfach zu faktorisieren.

Wendet man mehrfach nacheinander f_0 oder f_1 an, wie in den bündelnden Funktionen, braucht man von den Zwischenergebnissen nicht den Betrag zu bilden (denn die nächste Operation ist stets wieder eine Quadrierung, also für $\pm y$ gleich). Real würde man die Restklassen auch nicht symmetrisch darstellen, um normale Multiplikationsalgorithmen verwenden zu können. Man beachte auch, daß $4x^2 = (2x)^2$ ist, also außer einer Quadrierung nur eine Verdopplung, keine Multiplikation benötigt. Die Berechnung der bündelnden Funktion besteht also im wesentlichen aus σ modularen Quadrierungen, was effizienter ist als das Unterschreiben eines RSA-Blockes. Derjenige, der V gewählt hat, kann auch, wie bei RSA [QuCo_82], mod p und q einzeln rechnen und den chinesischen Restealgorithmus anwenden.

2.2.2 Effizienzverbesserungen

Der Hauptnachteil reiner Einmal-Signaturen ist die Länge des öffentlichen Schlüssels $\ddot{O}$, denn dieser muß außerhalb des Netzes zuverlässig verteilt werden (d.h. zumindest Unterzeichnende, Empfänger und ein Gericht müssen sich einig sein, was der gültige öffentliche Schlüssel ist). Bei reinen Einmal-Signaturen ist die Länge von $\ddot{O}$ proportional zur Anzahl der zu unterschreibenden Bits. Dieser Nachteil kann durch Verwendung von **Baumauthentikation** (tree authentication) vermieden werden, in Erweiterung von Verfahren aus [Merk_80, Merk_82].

Die Werte $bünd(r_{i,b})$ werden als Blätter eines Binärbaums verwendet (d.h. die Werte, die bei reinen Einmal-Signaturen der öffentliche Schlüssel wären). Der Wert jedes inneren Knotens wird als Hashwert seiner beiden Kinder berechnet, wozu eine Hashfunktion h benutzt wird. Nur der Wert der Wurzel wird als öffentlicher Schlüssel $\ddot{O}$ veröffentlicht (vgl. schon Bild 4). Als geheimer Schlüssel G genügen weiterhin die Werte $r_{i,b}$; zur Zeitersparnis wird man aber den ganzen Baum aufbewahren. Um später den Wert eines Blattes zu authentizieren, müssen die Werte aller seiner Ahnen und des anderen Kindes von jedem davon gezeigt werden. Wir benutzen kryptographisch kollisionsfreie Hashfunktionen (collision-free hash functions) [Damg_88] und setzen fest, daß Kollisionen bzgl. der Hashfunktion ebenfalls als Fälschungsbeweis zählen. Man kann sehen, daß jede Fälschung entweder zu einem Fälschungsbeweis in den ursprünglichen Einmal-Signaturen führt, oder zu einer Kollision bzgl. der Hashfunktion.

Die Hashfunktionen aus [Damg_88] ähneln den bündelnden Funktionen und benötigen etwa eine Quadrierung pro zu hashendem Bit. Speziell für die Baumauthentikation, wo die Länge der Nachrichten vorweg bekannt ist, können um den Faktor 2 effizientere Hashfunktionen konstruiert werden [Pfit_89], was im folgenden Bild nicht dargestellt ist. Auch können alle Werte $bünd(r_{i,b})$, die zu derselben Nachricht gehören, auf einmal gehasht werden (statt zunächst paarweise zusammengefaßt zu werden), da sie immer gemeinsam benötigt werden.

Wenn die zu unterschreibenden Nachrichten lang sind, können auch sie vor dem Unterschreiben **gehasht** werden. Wieder benutzen wir kryptographisch kollisionsfreie Hashfunktionen und zählen Kollisionen als Fälschungsbeweis; und man kann sehen, daß jede Fälschung entweder zu einem Fälschungsbeweis in den ursprünglichen Einmal-Signaturen führt, oder zu einer Kollision bzgl. der Hashfunktion.

In Bild 4 ist dargestellt, wie jetzt die Schlüssel gebildet werden, in Bild 5 das Unterschreiben einer Nachricht. Dabei sei L die (feste) Länge der Nachrichten nach dem Hashen. Die Vorschlüssel V wurden zur Vereinfachung weggelassen.

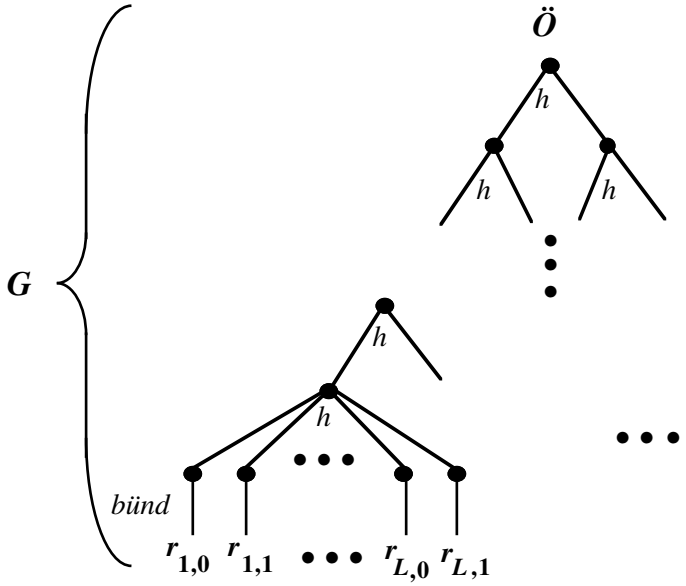

Bild 4: Schlüssel im System mit Baumauthentikation und Hashen

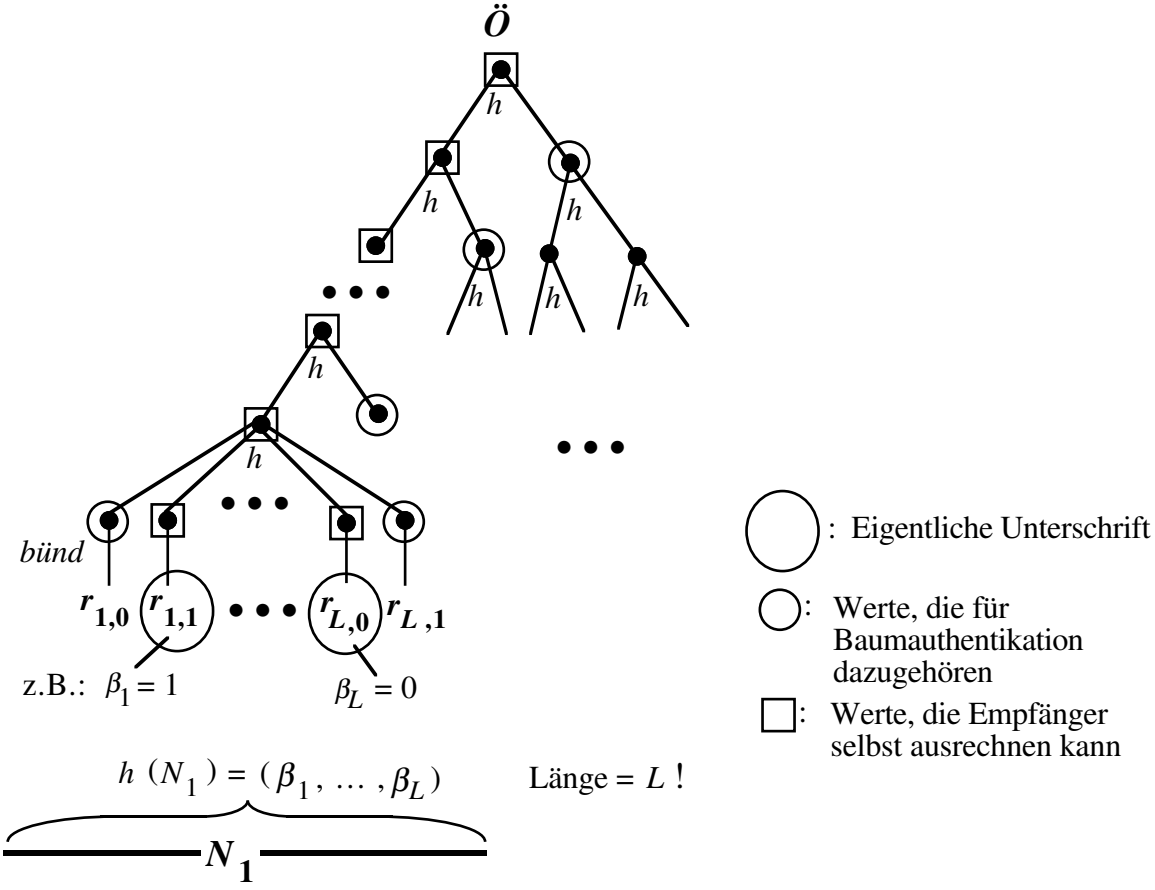

Bild 5: Unterschreiben einer Nachricht im System mit Baumauthentikation und Hashen

Weitere Effizienzverbesserungen sind: Nur Bits mit dem Wert 1 müssen unterschrieben werden, wenn ein Code benutzt wird, der alle Änderungen von 1 zu 0 erkennt [Merk1_82]. Daß bereits bei der Schlüsselwahl eine obere Schranke für die Anzahl damit zu leistender Signaturen festgelegt werden muß, kann vermieden werden, wenn man manche dieser Signaturen verwendet, um neue öffentliche Schlüssel zu authentizieren, ähnlich [Merk_88] (dies wurde allerdings bisher nicht bewiesen).

2.2.3 Vorschlüsselaustausch

Damit Kollisionen überzeugende Fälschungsbeweise sind, darf der Unterzeichner nicht fähig sein, selbst effizient Kollisionen zu berechnen. Kollisionsfreiheit, wie die meisten kryptographischen Eigenschaften, ist jedoch nur gegen Parteien garantiert, die nicht die Erzeugung der Vorschlüssel V für die bündelnden Funktionen und Hashfunktionen beobachten können. (Beispielsweise dürfen sie nicht sehen, aus welchen p und q ein $V = p \bullet q$ gebildet wird.) Daher darf nicht der Unterzeichner V wählen, sondern die Partei, die das Risiko beim Auftreten eines Fälschungsbeweises trägt.

Nur ein Risikoträger: In Anwendungen wie Zahlungssystemen trägt dieses Risiko im allgemeinen genau eine Person, nämlich der beabsichtigte Empfänger einer Signatur, dem Geld bezahlt oder versprochen wird (s. Kap.3). Daher kann diese allein V wählen. (Es ist dann nicht ausgeschlossen, daß der Empfänger einen Fälschungsbeweis zu empfangenen Signaturen erzeugen kann. Tut er dies tatsächlich, so schadet er aber nur sich selbst, denn die empfangenen Signaturen werden ungültig; z.B. erhält er das versprochene Geld nicht.)

Die Fähigkeit des Unterzeichners, Fälschungen zu beweisen, kann aber davon abhängen, daß V ein korrekter Vorschlüssel für die bündelnde Funktion ist. Daher muß der spätere Empfänger den Unterzeichner überzeugen, daß V korrekt ist, ohne jedoch dem Unterzeichner zu ermöglichen, Kollisionen zu bilden.

Im **allgemeinen Fall** kann dies durch eine übliche „cut-and-choose“ Technik [Rabi_78] erreicht werden: Der spätere Empfänger erzeugt $2 \bullet \sigma$ Vorschlüssel. Der Unterzeichner darf diese beliebig in zwei Hälften zerlegen. Für die eine Hälfte muß der spätere Empfänger die Korrektheit zeigen, die andere Hälfte wird in jeder Signatur verwendet. Eine Kollision für mindestens einen Vorschlüssel zählt als Fälschungsbeweis. Solange also mindestens ein verwendeter Vorschlüssel richtig ist, kann der Unterzeichner Fälschungen mit derselben Wahrscheinlichkeit beweisen wie bisher. Die Wahrscheinlichkeit, daß alle verwendeten Vorschlüssel falsch sind, während alle, deren Korrektheit gezeigt werden mußte, richtig waren, ist exponentiell gering. (Alternativ kann ein „zero-knowledge“-Beweis verwendet werden [GoMR_89, GoMW_86]. Innerhalb von diesem muß aber auch wieder die Korrektheit gewisser Schlüssel gesichert werden.)

In [Bleu_90, BlPW_90] wurde bewiesen, daß bei unserer **effizientesten Implementierung**, also dem auf dem Faktorisierungsproblem basierenden Spezialfall aus Kap. 2.2.1, der Unterzeichner jede ungerade Zahl als V akzeptieren darf. In diesem Fall wird also beim Schlüsselaustausch überhaupt kein interaktives Protokoll benötigt.

Mehrere Risikoträger: In Anwendungen wie Byzantinischen Übereinstimmungsprotokollen (Kap. 3.2), wo mehrere Personen dieses Risiko tragen, gibt es zwei Möglichkeiten: Entweder müssen sie versuchen, ein V gemeinsam zu wählen, unter Benutzung eines bestimmten Mehrparteienprotokolls [ChDG_88, PfWa_90]; diese Protokolle sind aber bisher kaum praktikabel. Oder jeder muß seinen eigenen Vorschlüssel V_i wählen. In diesem Fall besteht eine Gesamtsignatur aus einer Signatur bezüglich jedes Vorschlüssels, ein Fälschungsbeweis aus einer Kollision für jeden Vorschlüssel. Natürlich kann ein Empfänger dasselbe V_i für alle Unterzeichner verwenden.

Ein praktisch interessanter Fall mit zwei Vorschlüsseln pro Signatur ist der, daß an sich nur der Empfänger jeder Signatur das Risiko trägt, wenn dazu ein Fälschungsbeweis auftaucht, aber die Gerichte Empfehlungen bzgl. der Sicherheitsparameter geben wollen. Dann müssen die Gerichte sicher sein, daß jeder ihnen vorgelegte Fälschungsbeweis tatsächlich durch Brechen der kryptographischen Annahme zustandekam (also z.B. durch Faktorisieren einer großen Zahl, und nicht dadurch, daß ein Empfänger als Unfug die Faktoren seines Vorschlüssels bekanntgab.) Dazu könnte eine staatliche Stelle einen Vorschlüssel wählen, der in jeder Signatur mitverwendet wird. Damit aber niemand in diese eine Stelle volles Vertrauen setzen muß, hat jeder Empfänger weiterhin auch noch einen eigenen Vorschlüssel.

2.2.4 Resultierende Effizienz

Wir gehen hierzu von der auf dem Faktorisierungsproblem basierenden Implementierung aus, und zunächst von nur einem Risikoträger pro Signatur.

Wie schon erwähnt, ist der kritischste Punkt die Länge der Vorschlüssel und öffentlichen Schlüssel, da diese z.B. in Verzeichnissen gedruckt werden müssen. Diese sind genauso kurz wie sonst, nämlich eine Zahl $V = p \bullet q$, die schwer zu faktorisieren sein sollte, also z.B. der Länge $L = 800$ bit, und $\ddot{O}$ eine einzige Zahl modulo V.

Speziell für 1-bit-Nachrichten sind auch die Länge der Signaturen und der Aufwand beim Unterschreiben und Testen fast dieselben wie bei GMR, dem effizientesten beweisbar sicheren konventionellen Signatursystem. Der Aufwand wächst jedoch fast proportional mit der Länge der Nachricht, bis die Länge erreicht ist, auf die die Hashfunktion lange Nachrichten reduziert, was hier wieder L ist. (Speziell in Zahlungssystemen kann der Effizienzverlust allerdings sehr gering gehalten werden, siehe Kap. 3.1.3.)

Ein geheimer Schlüssel ist etwas kürzer als die Gesamtheit der damit zu leistenden Signaturen. Dies stört aber nicht weiter, da man empfangene und geleistete Signaturen sowieso aufbewahren muß (letztere, weil sie evtl. für Fälschungsbeweise gebraucht werden); und auch der private Schlüssel muß nur gespeichert (und in kleinen Stücken sequentiell abgearbeitet) werden.

Wir sehen jedoch keinen prinzipiellen Grund, warum jedes Fail-stop-Signatursystem auf Einmal-Signaturen beruhen müßte. Es müßte sich lohnen, nach einem System zu suchen, das auf irgendeiner Art von Falltür beruht, und in dem alle Signaturen dieselbe Länge und Unterschreiben und Testen ähnliche Laufzeiten hätten wie in konventionellen Signatursystemen.

Steigt die Anzahl der Risikoträger, so steigt auch der Aufwand beträchtlich; für Fälle, wo alle Teilnehmer am Risiko für jede Signatur beteiligt sind, wie bei Byzantinischer Übereinstimmung, sind Fail-stop-Signaturen z.Zt. wohl eher von theoretischem Interesse.

3 Anwendungen

3.1 Zahlungssysteme

Wichtige Anwendungen für Fail-stop-Signaturen sind Zahlungssysteme, die nun unbedingt sicher für die Kunden gestaltet werden können. (Gleiches gilt für alle anderen Anwendungen, wo eine größere Organisation viele Kunden hat und Signaturen nur von der Organisation an die Kunden und umgekehrt, nicht jedoch zwischen den Kunden ausgetauscht werden.)

3.1.1 Einsatz von Fail-stop-Signaturen allgemein

Wir gehen von einem einfachen digitalen Zahlungssystem aus, das einem nicht-digitalen mit Einzahlungen, Abhebungen und Überweisungen nachempfunden ist. Es soll so implementiert werden, daß die Kunden unbedingt sicher sind.

Dazu verwenden wir Fail-stop-Signaturen nur für die Signaturen der Kunden unter ihre Aufträge an die Bank (oder, allgemeiner, den Betreiber des Zahlungssystems), während die Bank für ihre Signaturen gegenüber den Kunden ein beweisbar sicheres konventionelles Signatursystem verwendet, etwa GMR.

Den Fail-stop-Signaturen der Kunden muß nur die Bank vertrauen, nicht andere Kunden. (Dazu müssen die Überweisungen so gestaltet werden, daß eine Gutschrift dem Empfänger mit einer Signatur der Bank bestätigt wird.) Die Bank ist also einziger Risikoträger im Sinne von Kap. 2.2.3 und kann daher die Vorschlüssel V für die bündelnden Funktionen und Hashfunktionen allein wählen. Nachdem sie diese veröffentlicht hat, kann jeder Kunde einen geheimen Schlüssel G wählen (d.h. Urbilder für die bündelnden Funktionen in den Einmal-Signaturen) und den zugehörigen öffentlichen Schlüssel $\ddot{O}$ veröffentlichen. Von

da an kann jeder, insbesondere ein Gericht, testen, was gültige Signaturen und gültige Fälschungsbeweise sind. Damit sind die Kunden unbedingt sicher vor Fälschungen ihrer Signaturen, und die Bank ist kryptographisch sicher, daß die Kunden keine Fälschungsbeweise erzeugen können, da sie V selbst gewählt hat. (Wie in Kap. 2.2.3 erwähnt, ist nicht ausgeschlossen, daß die Bank Fälschungsbeweise für V berechnen kann, aber sie würde sich nur selbst schaden, da damit die Überweisungsaufträge und Abhebungen ungültig würden, nicht jedoch die Gutschriften und Einzahlungen.)

Bezüglich der von der Bank erhaltenen Signaturen, also den erhaltenen Gutschriften und Einzahlungsbestätigungen, sind die Kunden ebenfalls unbedingt sicher (denn jede Signatur, die sie von der Bank erhalten, gilt), und die Bank ist kryptographisch sicher vor Fälschungen.

3.1.2 Vorteile

Die obige Risikoverteilung ist sinnvoll, denn die Bank kann gewöhnlich die Signatursysteme und Sicherheitsparameter wählen, d.h. sie kann ihre eigene Sicherheit so groß machen wie sie will; und die Bank müßte auch in einem Zahlungssystem, das nur konventionelle Signaturen verwendet, der kryptographischen Sicherheit vertrauen. Ein zweiter Vorteil ist, daß, wenn doch jemals eine Fälschung auftreten sollte, die Bank sich dessen sicher ist (denn eine gefälschte Banksignatur erkennt sie selbst, und zu einer gefälschten Kundensignatur muß ein Fälschungsbeweis vorgelegt werden); sie kann also entsprechende Maßnahmen ergreifen. Drittens ist ein solches Zahlungssystem nicht nur für die Kunden, sondern auch für die Bank vorteilhaft, denn die unbedingte Sicherheit für die Kunden kann ein gutes Werbeargument sein.

Die Protokolle können von bisherigen digitalen Zahlungssystemen übernommen werden, außer daß die Bank den Vorschlüssel V veröffentlichen muß. Verwendet man den auf dem Faktorisierungsproblem basierenden Spezialfall des Verstecksystems, wo kein interaktiver Test der Richtigkeit des Vorschlüssels nötig ist (s. Kap. 2.2.3), läßt sich diese Veröffentlichung aber leicht in bisherige Protokolle einbauen, weil auch dort die Bank zum selben Zeitpunkt schon etwas veröffentlichen muß, nämlich welches Signatursystem verwendet wird, den Sicherheitsparameter und ihren eigenen Schlüssel. Alle Teilnehmer testen dann, bevor sie ihren geheimen und öffentlichen Schlüssel wählen, ob V ungerade ist, so wie sie auch testen, ob ein sinnvoller Sicherheitsparameter gewählt wurde. Gleiches gilt für die Gerichte.

Die Effizienz eines Zahlungssystems muß unter dieser Einführung von Fail-stop-Signaturen nicht sehr leiden, denn einerseits konnte der effiziente Fall mit nur einem Risikoträger verwendet werden, andererseits sind die zu unterschreibenden Nachrichten in einem Zahlungssystem stark standardisiert und können daher sehr kurz codiert werden.

3.1.3 Protokolle mit 1-bit-Nachrichten

Man kann sogar erreichen, daß die Kunden nur 1-bit-Nachrichten unterschreiben, wenn das Protokoll, wie ein Kunde einen Auftrag gibt, etwas geändert wird:

- Zunächst teilt der Kunde der Bank mit, was sein i-ter Auftrag sein wird. Dies kann ohne Signatur erfolgen. Um Unfug vorzubeugen (d.h. daß jemand häufig solche Mitteilungen unter fremdem Namen macht, obwohl er dadurch keinen Gewinn hat), kann man eine konventionelle Signatur verlangen.
- Die Bank unterschreibt diesen Auftrag mit dem konventionellen Signatursystem, mitsamt dem Namen des Auftraggebers und der Nummer i.
- Dann unterschreibt der Kunde genau ein Bit mittels der i-ten Einmal-Signatur. Dies soll bedeuten, daß der i-te Auftrag, so wie von der Bank unterschrieben, korrekt ist. (Man muß also für die i-te Signatur nur ein einziges Urbild r_i wählen, nicht zwei wie bei den üblichen Einmal-Signaturen.)

Nach Erhalt dieser Signatur führt die Bank den Auftrag aus. Behauptet später der Kunde, die Bank habe seinen i-ten Auftrag nicht korrekt ausgeführt, so kann folgendermaßen entschieden werden: Die Bank muß zunächst als Beweis, daß sie überhaupt den i-ten Auftrag ausführen durfte, das unterschriebene Bit, also die i-te Einmal-Signatur dieses Kunden, vorlegen. Kann sie dies, und kann der Kunde keinen

Fälschungsbeweis dazu vorlegen, so muß er den von der Bank unterschriebenen *i*-ten Auftrag vorlegen. Dies gilt als die korrekte Form des Auftrags.

Wie oben sind die Kunden unbedingt, die Bank kryptographisch sicher: Der Kunde unterschreibt das Bit nur, wenn er eine Signatur der Bank unter seinen richtigen Auftrag hat. Seine eigene Signatur ist unbedingt sicher; und die erhaltene Signatur gilt unbedingt. Umgekehrt müßte der Kunde, um die Bank zu betrügen, entweder einen Fälschungsbeweis zu seiner Einmal-Signatur vorlegen oder die Signatur der Bank unter den Auftrag fälschen. Beides ist kryptographisch nicht möglich.

(Ein etwas sinnloser Angriff gegen dieses System und seine Abwehr seien noch erwähnt: Der Kunde könnte behaupten, die Bank weigere sich, den *i*-ten Auftrag auszuführen. Die Bank dagegen würde sagen, der Kunde habe ihr bereits einen anderen Auftrag als *i*-ten mitgeteilt (z.B. eine Abhebung eines niedrigeren Betrags) und sie habe ihm schon ihre Signatur dazu gegeben; nun könne sie den geänderten Auftrag nicht auch noch unterschreiben. Kommt dies selten vor, verwendet man statt dessen einfach den (*i*+1)-ten Auftrag; kommt es häufiger vor, so sollte, weil die Anzahl zu leistender Signaturen beschränkt ist, das Gericht festlegen, daß der *i*-te Auftrag in der Version des Kunden ausgeführt wird, und speichern, daß, wenn der Kunde später doch die Signatur der Bank unter die andere Version vorlegt, diese nicht mehr gilt.)

Ein spezielles Zahlungssystem, in dem Fail-stop-Signaturen ganz ohne Effizienzverlust eingesetzt werden können, weil die Kunden sowieso nur 1-bit-Nachrichten unterschreiben, ist das aus [ChFN_90]. In diesem System sind Zahlungsbeziehungen zwischen den Kunden (nicht jedoch die Kontoführungen der einzelnen Kunden, vgl. z.B. [PWP_90, BüPf_89]) unbedingt unbeobachtbar, und es wurde explizit bedauert, daß die Kunden nicht auch unbedingt sicher gegen Betrug seien. Mit Fail-stop-Signaturen sind sie es.

3.2 Byzantinische Übereinstimmung

Eine weitere interessante Anwendung von Fail-stop-Signaturen ist Byzantine Übereinstimmung [PeSL_80], d.h. Protokolle zum Erreichen zuverlässiger Verteilung, wo diese physisch nicht gegeben ist. Hier konnten Protokolle mit neuen Sicherheitseigenschaften entwickelt werden. Hierzu, und für eine darauf aufbauende Anwendung auf unbeobachtbare Kommunikation, sei auf [PfWa1_91, WaPf_89] verwiesen.

4 Zusammenfassung

Mit den Fail-stop-Signaturen wurde eine neue Art digitaler Signaturen eingeführt, die sicherer als die bisherigen sind, und ihre Vorteile diskutiert (Kap.1).

Mit den Versteckssystemen wurden konkrete Fail-stop-Signatursysteme konstruiert (Kap. 2). Ihre wichtigste Grundlagen waren

- Einmal-Signaturen mit bündelnden Funktionen, d.h. kollisionsfreien Funktionen mit einer garantierten Mindestgröße für Urbildmengen,
- baumförmige Authentikation mit kollisionsfreien Hashfunktionen und
- die Vorschrift, daß in allen Fällen eine gefundene Kollision als Fälschungsbeweis gilt.

Zunächst wurde die Konstruktion aus beliebigen kollisionsfreien Paaren von Permutationen gezeigt, dann ein auf der Schwierigkeit des Faktorisierens beruhender Spezialfall. Das so erhaltene System ist praktikabel, aber im allgemeinen Fall nicht sonderlich effizient (vgl. Kap. 2.2.4). Es ist kein theoretischer Grund zu sehen, weshalb es nicht noch wesentlich effizientere Fail-stop-Signaturen geben sollte.

Als Anwendung, für die sich Fail-stop-Signaturen besonders eignen, wurden vor allem digitale Zahlungssysteme behandelt. Hier zeigte sich einerseits, daß besonders günstige Sicherheitseigenschaften erreicht werden können (Kap. 3.1.1-2), andererseits, daß hierzu nur minimale oder gar keine Effizienzverluste in Kauf zu nehmen sind (Kap. 3.1.3).

Dank: Wir danken *Andreas Pfitzmann* für einige gute Ideen, *Gerrit Bleumer*, *Manfred Böttger*, *David Chaum*, *Dirk Fox* und *Jörg Lukat* für hilfreiche Diskussionen und der *DFG* für finanzielle Unterstützung.

Literatur

BeMi_88 Mihir Bellare, Silvio Micali: How to sign given any trapdoor function; 20th Symposium on Theory of Computing (STOC) 1988, ACM, New York 1988, 32-42.

Bleu_90 Gerrit Bleumer: Vertrauenswürdige Schlüssel für ein Signatursystem, dessen Brechen beweisbar ist; Studienarbeit, Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe 1990.

BlPW_90 Gerrit Bleumer, Birgit Pfitzmann, Michael Waidner: A Remark on a Signature Scheme where Forgery can be Proved; Eurocrypt '90 – Abstracts, Århus, Denmark, May 1990, 403–407.

BrDe_86 Ernest F. Brickell, John M. DeLaurentis: An Attack on a Signature Scheme Proposed by Okamoto and Shiraishi; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 28-32.

BüPf_89 Holger Bürk, Andreas Pfitzmann: Digital Payment Systems Enabling Security and Unobservability; Computers & Security 8/5 (1989) 399-416.

ChDG_88 David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty Computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

ChFN_90 David Chaum, A. Fiat, M. Naor: Untraceable Electronic Cash; Crypto '88, LNCS 403, Springer Verlag, Berlin 1990, 319-327.

ChRo_90 David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures; Abstracts of Crypto '90, Santa Barbara 1990, 209-217.

Damg_88 Ivan Bjerre Damgård: Collision free hash functions and public key signature schemes; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 203-216.

DaPr_85 Donald W. Davies, Wyn L. Price: Digital Signatures - An Update; Proc. 7th international conference on computer communication (ICCC) Sydney 1984, "The New World of the Information Society", J. M. Bennett, T. Pearcey (eds.); Elsevier Science Publishers B. V. (North-Holland), 1985, 843-847.

Denn_84 Dorothy E. Denning: Digital Signatures with RSA and Other Public-Key Cryptosystems; Communications of the ACM 27/4 (1984) 388-392.

DiHe_76 Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644-654.

EAKM_86 Dennis Estes, Leonard M. Adleman, Kireeti Kompella, Kevin S. McCurley, Gary L. Miller: Breaking the Ong-Schnorr-Shamir Signature Scheme for Quadratic Number Fields; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 3-13.

EkHV_88 Jan Ekberg, Siegfried Herda, Jorma Virtamo: TeleTrusT - Technical concepts and basic Mechanisms; Research into Networks and Distributed Applications; R. Speth (ed.) Elsevier Science Publishers B. V., Brussels and Luxembourg, 1988, 523-533.

Elga_85 Taher ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms; IEEE Transactions on Information Theory 31/4 (1985) 469-472.

GiMS_74 E. N. Gilbert, F. J. Mac Williams, N. J. A. Sloane: Codes which detect deception; The Bell System Technical Journal 53/3 (1974) 405-424.

Gira_88 Marc Girault: Hash-functions using modulo-N operations; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 217-226.

GoMW_86 Oded Goldreich, Silvio Micali, Avi Wigderson: Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design; 27th FOCS, IEEE Computer Society 1986, 174-187.

GoMR_88 Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

GoMR_89 Shafi Goldwasser, Silvio Micali, Charles Rackoff: The Knowledge Complexity of Interactive Proof Systems; SIAM J. Comput. 18/1 (1989) 186-207.

JoCh_86 Wiebren de Jonge, David Chaum: Attacks on Some RSA Signatures; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 18-27.

Merk_80 Ralph C. Merkle: Protocols for Public Key Cryptosystems; Proceedings of the 1980 Symposium on Security and Privacy, April 14-16, 1980 Oakland, California, 122-134.

Merk_82 Ralph C. Merkle: Protocols for Public Key Cryptosystems; AAAS Selected Symposium 69, Secure Communications and Asymmetric Cryptosystems; G. Simmons (ed.); Westview Press, Boulder 1982, 73-104.

Merk1_82 Ralph Charles Merkle: Secrecy, authentication, and public key systems; UMI Research Press 1982.

Merk_88 Ralph C. Merkle: A digital signature based on a conventional encryption function; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 369-378.

NaYu_89 Moni Naor, Moti Yung: Universal One–way Hash Functions and their Cryptographic Applications; 21st Symposium on Theory of Computing (STOC) 1989, ACM, New York 1989, 33-43.

Odly_84 Andrew M. Odlyzko: Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir's Fast Signature Scheme; IEEE Transactions on Information Theory 30/4 (1984) 594-601.

PeSL_80 Marshall Pease, Robert Shostak, Leslie Lamport: Reaching Agreement in the Presence of Faults; Journal of the ACM 27/2 (1980) 228-234.

Pfit_89 Birgit Pfitzmann: Für den Unterzeichner sichere digitale Signaturen und ihre Anwendung; Diplomarbeit, Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe 1989.

PfWa_90 Birgit Pfitzmann, Michael Waidner: Formal Aspects of Fail-stop Signatures; Interner Bericht 22/90 der Fakultät für Informatik, Universität Karlsruhe, Dezember 1990.

PfWa1_91 Birgit Pfitzmann, Michael Waidner: Unbedingte Unbeobachtbarkeit mit kryptographischer Robustheit; diese Tagung.

PWP_90 Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen; Datenschutz und Datensicherung DuD14/5-6 (1990) 243-253, 305-315.

QuCo_82 Jean-Jaques Quisquater, C. Couvreur: Fast Decipherment Algorithm for RSA Public-Key Cryptosystem; Electronics Letters 18/21 (1982) 905-907.

Rabi_78 Michael O. Rabin: Digitalized Signatures; Foundations of Secure Computation, ed. by R.A. DeMillo, D.P. Dobkin, A.K. Jones, R.J. Lipton; Academic Press, N.Y. 1978, 155-166.

RSA_78 Ronald L. Rivest, Adi Shamir, Leonard Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; Communications of the ACM 21/2 (1978) 120-126, nachgedruckt: 26/1 (1983) 96-99.

WaPf_89 Michael Waidner, Birgit Pfitzmann: Unconditional Sender and Recipient Untraceability in spite of Active Attacks – Some Remarks; Interner Bericht 5/89 der Fakultät für Informatik, Universität Karlsruhe, März 1989.