Zukunftssichere digitale Signaturen
FutureSign

Johannes Buchmann
Fachbereich Informatik
Technische Universität Darmstadt
Hochschulstr. 10, D-64289 Darmstadt
Email: buchmann@cdc.informatik.tu-darmstadt.de

Erik Dahmen
Fachbereich Informatik
Technische Universität Darmstadt
Hochschulstr. 10, 64289 Darmstadt
Email: dahmen@cdc.informatik.tu-darmstadt.de

27. Juni 2008

Zusammenfassung

Elektronische Signaturen sind eine Schlüsseltechnologie für die Absicherung von IT-Systemen der Gegenwart und Zukunft. Sichere elektronische Signaturen sind eine wichtige Voraussetzung für die Erschließung des Potenzials, das das Internet der Zukunft bietet. Die wenigen verfügbaren Signaturverfahren sind aber in ihrer Sicherheit bedroht, z.B. durch Quantencomputer. Dies ist ein hohes Risiko. FutureSign löst dieses Problem und garantiert langfristig die Existenz sicherer elektronischer Signaturverfahren. Dies eröffnet große Marktchancen.

1 Stand der Forschung/Technik

Elektronische Signaturen spielen eine zentrale Rolle beim Schutz des Internets der Gegenwart und Zukunft mit seinen vernetzten, mobilen IT-Systeme. Elektronische Signaturen garantieren zum Beispiel die Authentizität von Software-Updates, Personen und Computern. Sichere elektronische Signaturen sind eine wichtige Voraussetzung für die Entfaltung des Potenzials des Internets der Zukunft.

Die Sicherheit der wenigen praktischen Signaturverfahren z.B. RSA, DSA und ECDSA beruht auf der Unlösbarkeit von Berechnungsproblemen aus der Zahlentheorie: der Faktorisierung von Produkten zweier großer Primzahlen und der Berechnung diskreter Logarithmen. Die Quantencomputer-Algorithmen von Shor [20] machen diese Berechnungsprobleme leicht lösbar und diese Signaturverfahren unsicher, sobald genügend große Quantencomputer gebaut werden können. Physiker arbeiten mit Hochdruck an Quantencomputern. In Kanada gibt es bereits das Unternehmen D-Wave [6], das Quantencomputer für kommerzielle Anwendungen anbieten will. Diese Bedrohung wird ernst genommen. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) hat die Arbeitsgruppe von Prof. J. Buchmann mit einer Studie beauftragt, Quantencomputerresistente Signaturverfahren zu identifizieren. Andere Länder interessieren sich ebenfalls sehr dafür. Quantencomputer sind nicht die einzige ernstzunehmende Bedrohung für die gegenwärtigen Signaturverfahren. Auch andere innovative Ideen sind jederzeit möglich, die die wenigen bekannten Signaturverfahren unsicher machen. In der Vergangenheit ist es immer wieder zu unerwarteten Durchbrüchen gekommen. Die Sicherheit der gängigen Signaturverfahren steht also auf tönernen Füßen. Angesichts ihrer Bedeutung (siehe Kapitel 3) ist das ein sehr hohes Risiko. Praktisch verwendbare, langfristig sichere Alternativen werden dringend gebraucht.

Viele alternative Signaturverfahren sind vorgeschlagen worden, zum Beispiel NTRU, S-Flash und Quartz. Sie haben sich aber als unsicher erwiesen oder ihre Sicherheit ist mindestens zweifelhaft.

1979 stellte Merkle [15] unter Verwendung des Einmal-Signaturverfahrens von Lamport [12] ein sehr einfaches Signaturverfahren mit minimalen Sicherheitsvoraussetzungen vor. Benötigt wird nur irgendeine kollisionsresistente kryptographische Hashfunktion. Diese Voraussetzung ist minimal, weil jedes Signaturverfahren eine Hashfunktion benutzt. Es ist sogar so, dass jedes sichere Signaturverfahren selbst als eine kollisionsresistente Hashfunktion betrachtet werden kann. Signaturverfahren machen nämlich aus großen Dokumenten kleine Strings (die Signatur). Sie sind also Hashfunktionen. Könnte ein Angreifer eine Kollision einer solchen Signatur-Hashfunktion konstruieren, dann wären zwei verschiedene Dokumente mit der gleichen Signatur gefunden. Damit wäre das Signaturverfahren unsicher. Das zeigt, dass sichere elektronische Signaturverfahren tatsächlich kollisionsresistente Hashfunktionen sind. Im Gegensatz zu den anderen Signaturverfahren verwendet das Merkle-Verfahren aber keinerlei anderen Sicherheitsannahmen, wie zum Beispiel die Schwierigkeit von zahlentheoretischen Berechnungsproblemen. Damit sind die Sicherheitsanforderungen minimal: Das Merkle-Verfahren ist solange sicher, wie es überhaupt sichere elektronische Signaturverfahren gibt.

Kryptographische Hashfunktionen gibt es in großer Zahl. Jede neue Hashfunktion führt zu einer neuen Merkle-Variante, deren Sicherheit von allen anderen Merkle-Varianten unabhängig ist, solange die Hashfunktion von den in den anderen Merkle-Varianten verwendeten Hashfunktionen unabhängig ist. Das Merkle-Verfahren löst also grundsätzlich das Problem, langfristig sichere

Signaturverfahren zu finden. Wird eine Merkle-Variante unsicher, kann die verwendete Hashfunktion gegen eine sichere ausgetauscht werden, und schon ist das Signaturverfahren wieder sicher.

Wir beschreiben jetzt das ursprüngliche Merkle-Signaturverfahren (siehe Abbildung 1) um daran unsere Verbesserungen und geplanten Weiterentwicklungen illustrieren zu können. Initial werden festgelegt: eine Hashfunktion, ein Einmal-Signaturverfahren und die Höhe des sogenannten Merkle-Baumes $h \geq 2$. Die Höhe h bestimmt die Anzahl der Signaturen, die mit dem öffentlichen Merkle-Schlüssel verifizierbar sind, nämlich 2^h viele. Der Signierer erzeugt 2^h Einmal-Signatur-Schlüsselpaare (Signier- und Verifikationsschlüssel) und konstruiert einen (binären) Hashbaum. Seine Blätter sind die Hashwerte der Verifikationsschlüssel. Jeder innere Knoten ist der Hashwert der Konkatenation seiner beiden Kinder. Der öffentliche Schlüssel des Merkle-Verfahrens ist die Wurzel des Hashbaums. Der geheime Schlüssel ist die Folge der Einmal-Signierschlüssel. Beim Signieren werden die Signierschlüssel nacheinander benutzt. Die Signatur eines Dokuments d, die mit dem i-ten Signierschlüssel erzeugt wurde, besteht aus der Einmal-Signatur des Dokuments, dem i-ten Verifikationsschlüssel, dem Index i und einem Authentisierungspfad, der es dem Verifizierer erlaubt, die Gültigkeit des i-ten Verifikationsschlüssels auf die Gültigkeit des öffentlichen Schlüssels zurückzuführen. Der Authentisierungspfad für den i-ten Verifikationsschlüssel besteht aus den Geschwistern aller Knoten auf dem Pfad vom i-ten Blatt des Merkle-Baumes zur Wurzel.

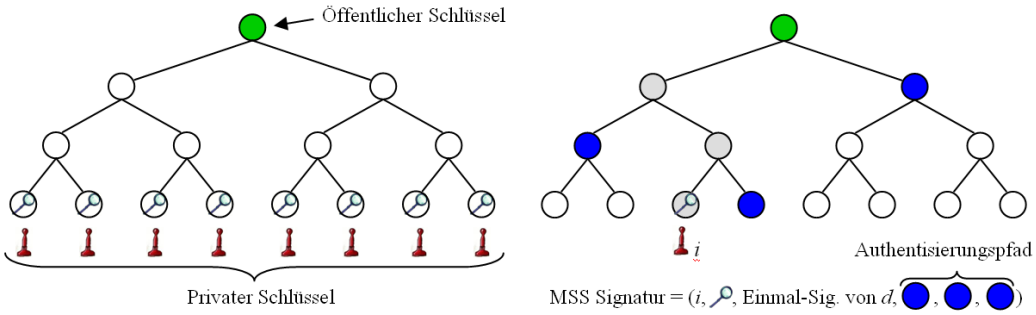

Abbildung 1: Schlüsselerzeugung (links) und Signaturerzeugung (rechts) des Merkle-Verfahrens.

Leider ist das Original-Merkle-Verfahren für die Praxis viel zu ineffizient. Im einzelnen hat das Verfahren folgende Nachteile:

- Die Signaturen sind zu groß, weil die Einmal-Signaturen sehr groß sind.
- Der geheime Schlüssel ist viel zu groß. Er besteht ja aus allen 2^h Signierschlüsseln.
- Wenn die Anzahl der verifizierbaren Signaturen groß ist, wird die Schlüsselerzeugungszeit sehr lang, weil der gesamte riesige Hashbaum vorberechnet werden muss.
- Die Berechnung des Authentisierungspfads dauert zu lang, wenn der gesamte Hashbaum immer wieder neu berechnet werden muss. Wird der Baum gespeichert, benötigt die Berechnung viel zu viel Speicherplatz.

Aus diesen Gründen war das Merkle-Verfahren nie ein ernst zu nehmender Konkurrent für die etablierten Signaturverfahren. Das hat sich auch nicht geändert, als im Laufe der Zeit eine Reihe

von technischen Verbesserungen gefunden wurden. Die Verbesserungen aus [21, 22, 11] ermöglichen eine effizientere Authentisierungspfadberechnung. In [7] wird eine Verallgemeinerung von Merkles Einmal-Signaturverfahren [15] vorgestellt. Dieses so genannte Winternitz-Verfahren ermöglicht einen Kompromiss zwischen der Zeit für die Generierung der Einmal-Signatur und der Signaturgröße.

2 Idee

Unsere Idee ist es, das Merkle-Verfahren zu einem für alle Anwendungsbereiche sicheren und effizienten Signaturverfahren weiterzuentwickeln, und damit das bedeutende Problem zu lösen, sichere Signaturverfahren für IT-Systeme der Zukunft bereitzustellen. Unsere algorithmischen Fortschritte zeigen, dass das möglich ist. Unser Plan ist es, das Verfahren in die Praxis zu bringen und für die deutsche IT-Sicherheitswirtschaft nutzbar zu machen.

Im Folgenden erläutern wir kurz, was wir erreicht haben. Dann stellen wir die Ideen zur Weiterentwicklung und Nutzung dar.

Wir haben das Merkle-Signaturverfahren praktikabel gemacht:

Signieren beschleunigt. Der mit Abstand aufwändigste Teil der Signaturerzeugung ist das Berechnen der Authentisierungspfade. In [3] beschreiben wir einen Algorithmus für das Berechnen der Authentisierungspfade. Dieser Algorithmus hat eine signifikant bessere Laufzeit als der beste, derzeit bekannte Algorithmus von Szydlo [21]. Der Algorithmus von Szydlo ist bei manchen Authentisierungspfaden sehr schnell, bei anderen aber sehr langsam. Szydlos Algorithmus berechnet immer etwa gleich viele Knoten des Hashbaumes. Szydlos Algorithmus berücksichtigt aber nicht, dass die Berechnung von Blättern des Baumes sehr viel teurer, als die Berechnung von inneren Knoten ist. Unser Algorithmus balanciert die Anzahl der Blätter aus, die für die Authentisierungspfade berechnet werden müssen. Dies reduziert die Laufzeit der Authentisierungspfadberechnung signifikant. Für einen Merkle-Baum der Höhe $h = 20$ ist unser Algorithmus bis zu 15% schneller als der von Szydlo.

Privaten Schlüssel klein gemacht. Im ursprünglichen Merkle-Verfahren besteht der private Schlüssel aus allen 2^h Einmal-Signierschlüsseln. In der Praxis kann eine solche Menge an Schlüsseln nicht gespeichert werden. In [1] schlagen wir vor, die Einmal-Signierschlüssel mit einem Pseudo-Zufallszahlengenerator zu erzeugen und als privaten Schlüssel den Seed des Pseudo-Zufallszahlengenerators zu verwenden. Die Länge dieses Seeds ist die Ausgabelänge der verwendeten Hashfunktion, z.B. 256 Bits bei SHA-256. Dieses Verfahren reduziert den Speicherplatz für den privaten Schlüssel drastisch. Gleichzeitig steigt die Rechenzeit nur unwesentlich, weil die verwendeten Pseudo-Zufallszahlengeneratoren sehr effizient sind.

Berechnung des öffentlichen Schlüssels beschleunigt. Bei der Berechnung des öffentlichen Schlüssels muss der gesamte Merkle Baum aufgebaut werden. Dies erfordert die Berechnung von $2^{h+1} - 1$ Knoten, wobei h die Höhe des Baumes ist. Für Bäume, deren Höhe größer als 20 ist, ist das unpraktikabel. Für viele Anwendungen ist es aber notwendig, mehr als $2^{20} \approx 1.000.000$ Signaturen mit einem Schlüsselpaar erzeugen zu können. Ein Beispiel für solche Anwendungen sind Server von Homebanking-Anwendungen. Sie müssen sich bei jeder neuen Homebanking-Aktivität authentisieren und dazu eine Signatur erstellen. In [1] stellen wir das Tree-Chaining-Verfahren vor. Dieses Verfahren zerlegt den großen Merkle-Baum in viele kleinere Teilbäume, die unabhängig voneinander nach und nach berechnet werden (siehe Abbildung 2). Zerlegt man einen Merkle Baum der Höhe 40 in Teilbäume der Höhe 20, werden für das Berechnen des öffent-

lichen Schlüssels nur noch $2 \cdot (2^{21} - 1)$ statt $2^{41} - 1$ Knoten benötigt. Benutzt man Teilbäume der Höhe 10, benötigt man nur $4 \cdot (2^{11} - 1)$ Knoten. Das Tree-Chaining-Verfahren kann also die Zeit, die für das Erzeugen des öffentlichen Schlüssels benötigt wird, beliebig verkürzen. Dadurch ist es nun möglich, effizient mehr als 2^{20} Signaturen mit einem Schlüsselpaar zu erzeugen.

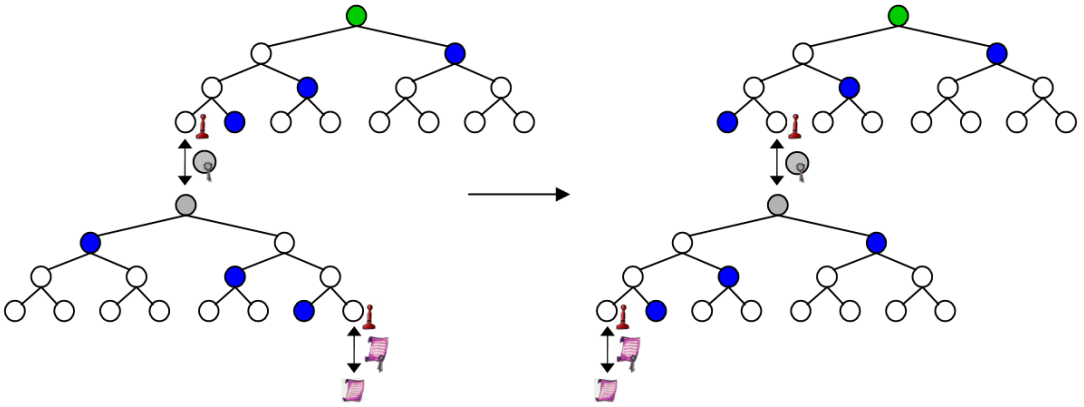

Abbildung 2: Tree-Chaining.

Signieren weiter beschleunigt und Signaturen kürzer gemacht. Das Tree-Chaining-Verfahren hat einen Nachteil: Neben der Einmal-Signatur der Nachricht sind in Signaturen, die mit dem Tree-Chaining-Verfahren erzeugt werden, auch die Einmal-Signaturen der Wurzeln der Teilbäume enthalten. Dadurch steigt die Größe der Signaturen. Dieses Problem beheben wir mit dem in [2] beschriebenen Verfahren der verteilten Signaturberechnung. Die Idee ist, die Berechnung der Einmal-Signaturen der Teilbaum-Wurzeln gleichmäßig auf viele Signierschritte zu verteilen. Dadurch wird die Signaturerzeugung zunächst extrem effizient. Um diesen Zeitvorteil in einen Platzvorteil umzuwandeln, benutzen wir das in [7] beschriebene Winternitz-Einmal-Signaturverfahren. Dieses Einmal-Signaturverfahren ermöglicht einen Kompromiss zwischen der Zeit für das Signieren und der Signaturgröße. Die Kombination der verteilten Signaturberechnung und des Winternitz-Einmal-Signaturverfahrens ermöglicht es, die Größe der Einmal-Signaturen der Teilbaum-Wurzeln zu reduzieren, und sie trozdem effizient zu berechnen.

Die Sicherheit bewiesen. Mit den in [5] vorgestellten Argumenten kann die Sicherheit von FutureSign bewiesen werden. Diese Dissertation wurde von Prof. J. Buchmann betreut. Es stellt sich heraus, dass das FutureSign im allgemeinsten Sicherheitsmodell für digitale Signaturen (Angriffe mit adaptiv gewählten Nachrichten) sicher ist, falls die verwendete Hashfunktion und der verwendete Pseudo-Zufallszahlengenerator kryptographisch sicher sind. Der Pseudozufallszahlengenerator kann seinerseits aus einer kryptographischen Hashfunktion konstruiert werden. Damit reduziert sich die Sicherheit auf die Sicherheitsforderungen für die Hashfunktion. Dies ist auch Voraussetzung für die Sicherheit jedes anderen Signaturverfahrens. Bei den anderen Signaturverfahren benötigt man aber weitere Sicherheitsvoraussetzungen, z.B. die Schwierigkeit zahlentheoretischer Berechnungsprobleme. Außerdem kann die Sicherheit vieler anderer Signaturverfahren nicht bewiesen werden. Mit FutureSign liegt also ein Signaturverfahren mit minimalen Sicherheitsvoraussetzungen vor, dessen Sicherheit bewiesen werden kann. Der Sicher-

heitsbeweis zeigt sogar, dass die Sicherheit von FutureSign fast die Sicherheit der verwendeten Hashfunktion ist. Beim Beweis geht also kaum Sicherheit verloren. Das trägt sehr zur hohen Effizienz von FutureSign bei.

In Java implementiert. Wir haben FutureSign in unserer Java-Bibliothek FlexiProvider [8] implementiert. Der FlexiProvider ist eine von der Arbeitsgruppe von Prof. J. Buchmann an der TU Darmstadt entwickelte Kryptographie-Bibliothek. Sie implementiert die innovativen kryptographischen Verfahren, die in der Arbeitsgruppe entwickelt werden und stellt sie der internationalen Öffentlichkeit zur Verfügung. Tabelle 1 zeigt Laufzeiten und Platzbedarf von FutureSign bei der Verwendung von SHA1 als Hashfunktion auf einem Pentium Dual-Core 1.83GHz. Verglichen wird FutureSign mit den gängigen Signaturverfahren RSA und ECDSA. Dabei wurden die Parameter so gewählt, dass alle Verfahren nach den Voraussagen von Lenstra [13] bis zum Jahre 2018 sicher sind. Wir zeigen Werte für verschiedene Signaturanzahlen pro öffentlichem Schlüssel und zwei unterschiedliche Parametersätze, die Speicherplatz und Rechenzeit unterschiedlich balancieren. Der Vergleich zeigt, dass FutureSign tatsächlich mit RSA und ECDSA konkurrieren kann. Die Signaturgröße wird in Zukunft noch verringert werden. Ideen dazu beschreiben wir unten. Die Ergebnisse belegen zum einen die Praktikabilität und zum anderen die enorme Flexibilität von FutureSign. Die Flexibilität besteht darin, dass unser Verfahren an den jeweiligen Bedarf angepasst werden kann. Wenn Speicherplatz knapp ist, kann der Speicherbedarf auf Kosten der Laufzeit verringert werden ohne die Laufzeit zu sehr zu steigern. Wenn dagegen genug Speicher verfügbar ist, aber Laufzeit ein Engpass ist, können auch dafür Parameter entsprechend gewählt werden.

Tabelle 1: Timings und Größen von FutureSign im Vergleich mit RSA und ECDSA bei gleichem Sicherheitsniveau.

Verfahren	Öffentlicher Schlüssel	Privater Schlüssel	Signatur	Schlüssel-erzeugung	Signieren	Verifizieren
FutureSign	20 Bytes	2.800 Bytes	1.460 Bytes	0,6 min	12,6 ms	16,5 ms
2^{20} Sig.	20 Bytes	3.480 Bytes	1.960 Bytes	0,1 min	7,8 ms	1,5 ms
FutureSign	20 Bytes	3.960 Bytes	1.740 Bytes	7,3 min	17,2 ms	5,7 ms
2^{30} Sig.	20 Bytes	4.480 Bytes	2.160 Bytes	2,3 min	10,7 ms	1,5 ms
FutureSign	20 Bytes	6.100 Bytes	2.380 Bytes	15,6 min	12,6 ms	10,4 ms
2^{40} Sig.	20 Bytes	7.140 Bytes	3.060 Bytes	3,1 min	7,8 ms	2,4 ms
RSA-1536	162 Bytes	634 Bytes	128 Bytes	1,4 sec	24,8 ms	1,5 ms
ECDSA-160	68 Bytes	56 Bytes	46 Bytes	37,7 ms	18,1 ms	21,8 ms

Auf Microcontroller implementiert. Zusammen mit der Ruhr Universität Bochum haben wir FutureSign auf einem 8-Bit Atmel ATmega128 Mircocontroller implementiert [19]. Für diese Implementierung verwenden wir eine AES basierte Hashfunktion. Tabelle 2 zeigt die Laufzeiten und den Platzbedarf dieser Implementierung ebenfalls für verschiedene Signaturanzahlen

pro öffentlichem Schlüssel und zwei unterschiedliche Parametersätze. Wir beschränken uns auf maximal 2^{16} Signaturen pro Schlüsselpaar, da die Lebensdauer des Speichers auf dem Microcontroller nur ungefähr 2^{16} Schreib-/Lese-Zugriffe beträgt. Außerdem vergleicht diese Tabelle unsere Implementierung mit Referenzimplementierungen heute verwendeter Signaturverfahren. Die Timings belegen, dass FutureSign RSA weit überlegen und mit ECDSA vergleichbar ist. Ein weiterer Vorteil von Future Sign ist die geringe Code-Größe, ein wichtiger Aspekt auf resourcenarmen Geräten.

Tabelle 2: Timings und Größen von FutureSign im Vergleich mit Referenzimplementierungen von RSA und ECDSA auf einem 8-Bit Atmel ATmega128 Mircocontroller.

Verfahren	Öffentlicher Schlüssel	Privater Schlüssel	Signatur	Code-Größe	Signieren	Verifizieren
FutureSign	16 Bytes	1.472 Bytes	2.350 Bytes	6.600 Bytes	1.072 ms	85 ms
2^{16} Sig.	16 Bytes	1.472 Bytes	1.330 Bytes	6.600 Bytes	1.665 ms	127 ms
FutureSign	16 Bytes	876 Bytes	2.290 Bytes	6.600 Bytes	598 ms	82 ms
2^{10} Sig.	16 Bytes	876 Bytes	1.234 Bytes	6.600 Bytes	946 ms	124 ms
RSA-1024	131 Bytes	128 Bytes	128 Bytes	7.400 Bytes	5.495 ms	215 ms
RSA-2048	259 Bytes	256 Bytes	256 Bytes	10.600 Bytes	41.630 ms	970 ms
ECDSA-160	40 Bytes	21 Bytes	40 Bytes	43.200 Bytes	423 ms	423 ms
ECDSA-160	40 Bytes	21 Bytes	40 Bytes	17.900 Bytes	1.001 ms	1218 ms

In S/MIME und TLS integriert. Um die praktische Einsetzbarkeit von FutureSign zu belegen, haben wir ein FutureSign-S/MIME plug-in für MS Outlook geschrieben, das es ermöglicht E-Mails digital mit FutureSign zu signieren. Der Schüler Elias Most hat im Rahmen eines Jugend-Forscht-Projektes FutureSign in das TLS-Protokol integriert. Damit ist es möglich die Authentizität von Internet-Seiten mit Hilfe von FutureSign zu prüfen. S/Mime und TLS sind zentrale Anwendungen elektronischer Signaturen. Diese Implementierungen sind also wichtige Praxistests. Experimente zeigen, dass zwischen FutureSign-S/MIME bzw. FutureSign-TLS und den auf RSA und ECDSA beruhenden Varianten kein Unterschied festzustellen ist.

Wir beschreiben nun unsere weiteren Entwicklungsziele.

Signaturgröße weiter reduzieren. Die FutureSign-Signaturen sind im Vergleich zu herkömmlichen Signaturen noch relativ groß. Das liegt daran, dass die verwendeten Einmal-Signaturverfahren für die Signatur eines n-Bit Hashwertes etwa n^2 Bits benötigen. Das Winternitz-Verfahren [7] reduziert diese Größe zwar, aber der Effekt ist noch nicht ausreichend. In jüngerer Zeit haben Lyubashevsky und Micciancio [14] ein Einmal-Signaturverfahren vorgeschlagen, das Signaturen der Länge $O(n \log n)$ produziert. Diese Signaturen haben also etwa die gleiche Größe wie die zu signierenden Hashwerte. Das Verfahren von Lyubashevsky und Micciancio ist aber noch nicht praktikabel, weil die Parameter für eine sichere Implementierung viel zu groß sind. Unsere

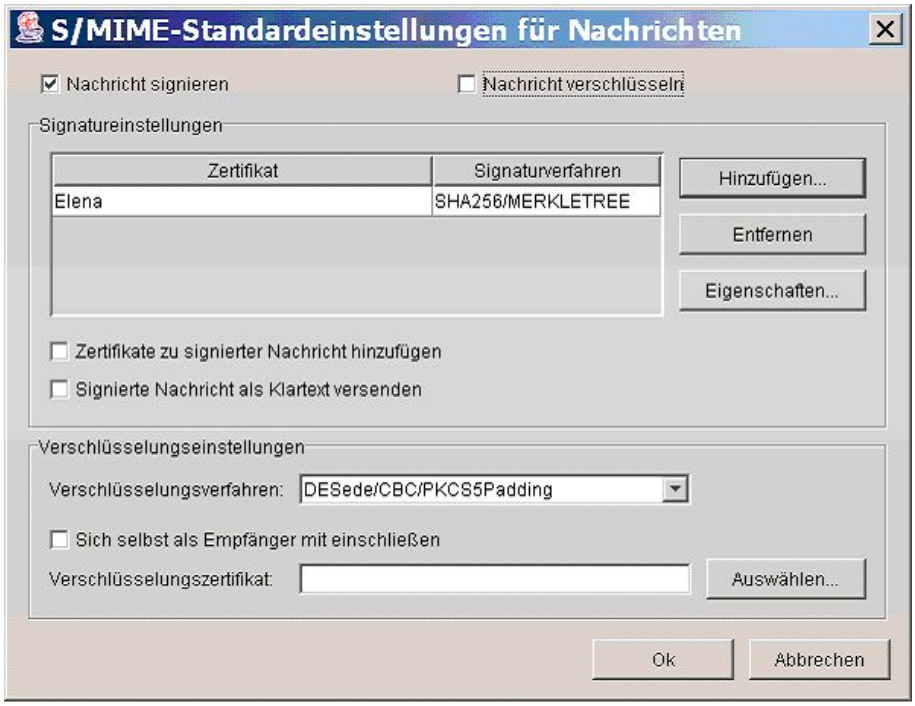

Abbildung 3: FutureSign-S/MIME für MS Outlook

Idee ist es, Ansätze aus dem NTRU-Verfahren [16] zu übertragen, um eine praktikable Variante des Verfahrens von Lyubashevsky und Micciancio zu erhalten. Für diese Ideen läuft bereits ein Patent-Verfahren [17].

Für Spezialanwendungen optimieren. Wir werden FutureSign-Varianten für verschiedene Szenarien experimentell entwickeln. Dies ist möglich, weil FutureSign mit sehr vielen Parametern konfigurierbar ist. Solche Szenarien sind zum Beispiel:

- Compute Server, die sich häufig authentisieren müssen und darum Signaturen produzieren, die mit dem selben öffentlichen Schlüssel verifizierbar sein müssen.
- Eingebettete Systeme, zum Beispiel in Autos. Hier kommt es besonderes darauf an, die FutureSign-Codegröße zu begrenzen.
- RFIDs und Sensoren. Hier gibt es nur geringen Speicherplatz und eine geringe Kommunikationsbandbreite. Die optimierten FutureSign-Varianten mit geringer Signaturgröße können hier zum Einsatz kommen.

Wir wollen für diese Anwendungen adäquate Zertifizierungsinfrastrukturen entwickeln.

Standardisieren. Bis jetzt ist FutureSign oder einer seiner Vorgänger in keinem der Standards für elektronische Signaturen enthalten. Relevante Standards sind PKCS [18] und FIPS

[9]. Die Standardisierung ist aber extrem wichtig, da nur so die Kompatibilität von FutureSign-Implementierungen mit Anwendungen, die FutureSign einsetzen, sichergestellt ist. Der angestrebte Standard soll FutureSign genau spezifizieren, aber gleichzeitig die Verwendung beliebiger Hash-Funktionen ermöglichen. Auf diese Weise kommt ein Signaturverfahren in die Standards, das für eine sehr lange Zeit sicher ist.

Integration in Server- und Clientanwendungen. Wir planen plug-ins für weit verbreitete Anwendungssoftware, wie z.B. Mail-Clients, Browser, Web-Server zu entwickeln, um den Einsatz von FutureSign zu ermöglichen.

Langfristige Sicherheit. Manche elektronische Signaturen müssen sehr lange gültig sein oder mindestens regelmäßig erneuert werden, zum Beispiel bei Langzeitarchivierung und bei der Verwendung von digitalen Signaturen für lange gültige Verträge. Hier ist FutureSign besonders geeignet. Die heute üblichen Ansätze erfordern Mehrfachsignaturen oder Übersignaturen. Dazu müssen hinreichend viele Signaturverfahren vorhanden sein. FutureSign beinhaltet sehr viele Signaturverfahren. Ersetzt man die Hashfunktion und den Pseudozufallszahlengenerator, bekommt man auf einfache Weise eine neue Instanz von FutureSign und damit ein neues elektronisches Signaturverfahren. Wir werden auf FutureSign basierende Langzeit-Signaturen und Archivierungssysteme entwickeln.

3 Nutzen

Der Nutzen von FutureSign besteht darin, dass dieses Verfahren die Existenz von sicheren elektronischen Signaturverfahren für die Zukunft sichert und damit die Erschließung des Potenzials des Internets der Zukunft ermöglicht. Gleichzeitig beendet FutureSign die Abhängigkeit von den wenigen praktikablen elektronischen Signaturverfahren, die ein großes Risiko darstellt. Die Anzahl der Anwendungen von digitalen Signaturen nimmt nämlich dramatisch zu:

Software-Updates: Elektronische Signaturen beweisen die Authentizität von Software-Updates. Microsoft (und die anderen Betriebssystemhersteller) stellen im Internet ihren Nutzern (mehr als eine Milliarde!) solche Software-Updates zur Verfügung. Diese schließen zum Beispiel Sicherheitslücken. Sehr viele andere Anwendungen verwenden Software-Uploads und -Updates über das Internet: Antivirensoftware, Handys, PDAs, Steuerungen von Motorelektronik, Fahrdynamik-Regelung (ESP) und Antiblockiersystem (ABS) in Fahrzeugen. Die moderne Welt ist ohne authentische Software-Uploads und -Updates nicht funktionsfähig. Beim Authentizitätsnachweis gibt es zu digitalen Signaturen keine praktikable Alternative.

Internet-Kommunikation: Die Protokolle SSL/TLS sind die de facto Standards für die Absicherung von e-Commerce, e-Banking und vielen anderen Internet-Anwendungen. SSL/TLS verwendet digitale Signaturen zur Authentisierung von Webservern und Benutzern. Der E-Mail-Standart S/MIME verwendet digitale Signaturen zur Authentisierung von E-Mails. Virtuelle private Netzwerke (VPN) verwenden zunehmend digitale Signaturen für die Authentisierung ihrer Benutzer.

Identity Management: Deutschland und viele andere Länder geben elektronische Reisepässe aus. Solche Reisepässe verwenden digitale Signaturen zur Authentisierung ihrer Inhaber bei Grenzübertritt. Viele Länder haben elektronische Identitätskarten eingeführt, zum Beispiel Belgien, Finnland, Estland und Italien. Solche Identitätskarten verwenden elektronische Signaturen bei der Authentisierung der Bürger z.B. in eGovernment-Anwendungen. Deutschland hat den elektronischen Personalausweis für 2009 angekündigt. Die digitale Patientenkarte, die Arztkarte, die Jobkarte und die Chipkarte für Sozialleistungen verwenden ebenfalls digitale Signaturen.

Die Beispiele zeigen: Elektronische Signaturen werden heute in allen Lebensbereichen angewendet. Die Welt funktioniert nur, wenn es sichere elektronische Signaturverfahren gibt. Aber die Anzahl der praktikablen und sicheren elektronischen Signaturverfahren ist sehr gering. Im Wesentlichen sind dies drei Verfahren: RSA, DSA, ECDSA. Deren Sicherheit ist stark in Frage gestellt. Sie beruhen alle drei auf der Schwierigkeit verwandter mathematischer Berechnungsprobleme. Diese Schwierigkeit ist aber keineswegs garantiert. Im Gegenteil, sie ist zum Beispiel durch die Möglichkeit stark bedroht, dass es in den nächsten Jahren Quantencomputer gibt. Die Abhängigkeit der gesamten Informations- und Kommunikationstechnologie von wenigen elektronischen Signaturverfahren ist ein ernstes Problem, das dringend gelöst werden muss.

FutureSign löst dieses Problem und trägt dazu bei, dass das Internet der Zukunft seine großen Möglichkeiten entfalten kann. FutureSign ist nämlich nicht nur ein einziges elektronisches Si-

gnaturverfahren. FutureSign beinhaltet viele elektronische Signaturverfahren: Jede neue Hashfunktion, die man in FutureSign verwendet, liefert ein neues Signaturverfahren. Diese enorme Flexibilität hebt die Abhängigkeit von den wenigen jetzt verfügbaren Signaturverfahren auf und stellt die Sicherheit, jedenfalls was Signaturverfahren angeht, auf stabile Füße. FutureSign ist aber noch in anderer Weise enorm flexibel. Die verschiedenen Parameter erlauben es, Future Sign genau an die verschiedenen Anwendungen für digitale Signaturen anzupassen: Server, eingebettete Systeme, RFIDs, usw. Das konnten wir mit unseren Experimenten zeigen.

Die öffentliche Beachtung für FutureSign ist groß. Die Autoren dieses Vorschlags wurden unter anderem von Hitachi, Fujitsu, Siemens, Infineon, IBM, der Academica Sinica Taiwan und der Chinesischen Akademie der Wissenschaften zu Vorträgen über FutureSign eingeladen. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) nimmt die Bedrohung durch Quantencomputer sehr ernst und interessiert sich sehr für FutureSign. Die Arbeitsgruppe von Prof. J. Buchmann wurde beauftragt, in einer Studie die Quantencomputer-Resistenz verschiedener Kryptoverfahren zu prüfen und Empfehlungen auszusprechen [4]. Als Signaturverfahren wurde von uns FutureSign vorgeschlagen. Das BSI stimmte diesem Vorschlag zu. In einem Nachfolgeprojekt des BSI, an dem wir auch beteiligt sind, wird FutureSign jetzt auf Chipkarten, FPGAs und in Software implementiert.

All das zeigt den großen Nutzen von FutureSign.

4 Marktchancen

FutureSign hat aus unserer Sicht große Marktchancen. Dies belegt die Entwicklung der Unternehmen RSA Security und Certicom. RSA Security enstand nach der Erfindung des RSA-Verfahrens. Heute ist RSA eine Division von EMC und macht einen Jahresumstz von 250 Millionen US $. Certicom, spezialisiert auf Elliptische-Kurven-Kryptographie, erwirtschaftete 2007 mit seinen 350 Patenten und seinen Toolkits fast 15 Millionen US $ und mit Services fast 7 Millionen US $. Das ist eine Steigerung von 25% gegenüber dem Vorjahr. Certicom begann mit der Vermarktung der Elliptische-Kurven-Kryptographie unmittelbar nach deren Erfindung. Wir halten eine entsprechende Entwicklung auch auf der Basis des Merkle-Verfahrens für möglich, wenn der günstige Zeitpunkt schnell genutzt wird.

Im Einzelnen sehen wir folgende Verwertungsmöglichkeiten für FutureSign:

Spezialanwendungen: Solange FutureSign noch nicht standardisiert ist, lässt es sich schon für Anwendungen in geschlossenen Benutzergruppen einsetzen. Denkbar ist zum Beispiel der Authentizitätsnachweis für Codeuploads und -Updates in Automobilunternehmen und Anwendungen wie der LKW-Maut. Unser Verfahren ist auch für den Hochsicherheitsbereich interessant, wo langfristige Sicherheit eine große Rolle spielt. Mögliche Produkte sind Toolkits für den Aufbau von entsprechenden Zertifizierungsinfrastrukturen und für Clientanwendungen, besonders für eingebettete Systeme und Chipkarten, USB-Token usw. Dabei entstehen auch neue, patentfähige Technologien.

Lizenzierungen: Wir erwarten, dass FutureSign mittelfristig weltweit auf großes Interesse stoßen wird. Die bei der Weiterentwicklung von FutureSign erworbenen Schutzrechte sollen vermarktet werden. Dabei handelt es sich um Spezialalgorithmen für verschiedene Anwendungsszenarien.

Globale Anwendungen: Sobald FutureSign standardisiert ist, wird es großen Bedarf geben, das Verfahren in großen Anwendungen, zum Beispiel im Banking-Bereich, im Identity Management und für Code-Uploads und -Updates über das Internet einzusetzen. Dann haben die Toolkits und Lizenzen einen großen Marktwert.

Services: Die Integration von FutureSign in Anwendungen erfordert sachkundigen Service. Es kann also im Zusammenhang mit FutureSign ein nicht unerhebliches Projektgeschäft geben.

5 Literaturverzeichnis

[1] Johannes Buchmann, Carlos Coronado, Erik Dahmen, Martin Döring und Elena Klintsevich, CMSS - An Improved Merkle Signature Scheme. Progress in Cryptology - INDOCRYPT 2006, LNCS 4392, Springer, 2006, pp. 349-363.

[2] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyki Okeya und Camille Vuillaume, Merkle Signatures with Virtually Unlimited Signature Capacity. Applied Cryptography and Network Security - ACNS'07, LNCS 4521, Springer, 2007, pp. 31-45.

[3] Johannes Buchmann, Erik Dahmen und Michael Schneider, Merkle tree traversal revisited. Eingereicht zur PQ Crypto 2008.

[4] Quantencomputerresistente Kryptoverfahren, Studie erstellt von der Arbeitsgruppe Prof. J. Buchmann im Auftrag des Bundesamt für Sicherheit in der Informationstechnik (BSI), Auftrag-Nr. 29758/2006, 2007.

[5] Carlos Coronado, Provably Secure and Practical Signature Schemes. Dissertation betreut von Johannes Buchmann, Technische Universität Darmstadt, 2005. Erhältlich unter `http://elib.tu-darmstadt.de/diss/000642/`.

[6] D-Wave - The Quantum Computing Company.
`http://www.dwavesys.com/`.

[7] C. Dods, N. P. Smart und M. Stam, Hash Based Digital Signature Schemes. Cryptography and Coding, LNCS 3796, Springer, 2005, pp. 96–115.

[8] FlexiProvider, an open source Java Cryptographic Service Provider.
`http://www.flexiprovider.de/`.

[9] Federal Information Processing Standards Publications.
`http://csrc.nist.gov/publications/PubsFIPS.html`

[10] Lov K. Grover. A fast quantum mechanical algorithm for database search. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. ACM Press, 1996, pp 212–219.

[11] Markus Jakobsson, Tom Leighton, Silvio Micali und Michael Szydlo, Fractal Merkle tree representation and traversal. Cryptographer's Track at RSA Conference - CT-RSA'03, LNCS 2612, Springer, 2003, pp. 314–326.

[12] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[13] Arjen K. Lenstra. Key lengths. Contribution to The Handbook of Information Security, 2004. Erhältlich unter `http://cm.bell-labs.com/who/akl/key_lengths.pdf`.

[14] V. Lyubashevsky, D. Micciancio, Asymptotically efficient lattice-based digital signatures. Theory of Cryptography Conference - TCC. LNCS 4948, Springer, 2008, pp. 37–54.

[15] Ralph Merkle. A certified digital signature. Advances in Cryptology – CRYPTO '89, LNCS 1462, Springer, 1989, pp. 218–238.

[16] Jeffrey Hoffstein, Jill Pipher und Joseph H. Silverman, NTRU: A Ring-Based Public Key Cryptosystem. Algorithmic Number Theory - ANTS, LNCS 1423, Springer, 1998, pp. 267–288.

[17] Lin Trap OTS, Patent, amtliches Aktenzeichen DE 10 2008 015 865.8, 2008.

[18] Public-Key Cryptography Standards. `http://www.rsa.com/rsalabs/node.asp?id=2124`

[19] Sebastian Rohde, Thomas Eisenbarth, Erik Dahmen, Johannes Buchmann, and Christof Paar, Fast Hash-Based Signatures on Constrained Devices. Eighth Smart Card Research and Advanced Application Conference - CARDIS 2008, erscheint noch.

[20] Peter W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring. 35th Annual Symposium on Foundations of Computer Science, pp. 124-134, IEEE Comput. Soc. Press, 1994.

[21] Michael Szydlo, Merkle tree traversal in log space and time. Preprint, erhältlich unter `http://www.szydlo.com/`, 2003.

[22] Michael Szydlo, Merkle Tree Traversal in Log Space and Time. Advances in Cryptology – EUROCRYPT 2004, LNCS 3027, Springer, 2004, pp 541–554.