Fachhochschule Rhein-Sieg

Fachbereich Angewandte Informatik / Kommunikationstechnik
Lehrveranstaltung Informationssicherheit
Sommersemester 1998

Asymmetrische Verschlüsselungsverfahren

Thomas Hungenberg

15. Mai 1998

Inhalt:

1. Allgemeines

1.0 Was ist asymmetrische Verschlüsselung?

Die herkömmliche Kryptographie basiert auf **symmetrischer** Verschlüsselung. Das bedeutet, daß zum Verschlüsseln und späteren Entschlüsseln der **gleiche** Schlüssel verwendet wird. Aus diesem Grund muß der Schlüssel **geheim** gehalten werden (*secret-key*). Wollen nun zwei Personen Daten verschlüsselt übermitteln, so müssen sie sich auf einen geheimen Schlüssel einigen, ohne daß ihn jemand anderes zu Gesicht bekommt. Wenn sie sich an zwei verschiedenen geographischen Orten befinden, müssen sie dazu einem Kurier oder einem Kommunikationsmedium **vertrauen**, um die Offenlegung des geheimen Schlüssels zu verhindern. Jeder, der den Schlüssel auf dem Transportweg abfangen oder (unbemerkt) mithören kann, kann anschließend alle mit diesem Schlüssel verschlüsselten Nachrichten lesen und ändern oder Nachrichten fälschen und unterschreiben.
Whitfield Diffie und Martin Hellmann schlugen deshalb 1976 das Konzept der **asymmetrischen Kryptographie** vor [1, 2]. In ihrem Konzept hat jeder Beteiligte ein Schlüssel**paar**: Einen **öffentlichen** Schlüssel (*public-key*), der verteilt wird, und einen **privaten** (*private-key*), der beim Besitzer geheim gehalten wird. Die Notwendigkeit eines gemeinsamen Geheimnisses zwischen den Beteiligten ist damit verschwunden. Es werden nur öffentliche Schlüssel ausgetauscht. Private Schlüssel werden nie übertragen oder geteilt. Daher ist es nicht länger notwendig, einem Kommunikationskanal hinsichtlich Abhörsicherheit vertrauen zu müssen. Die einzige Anforderung ist, daß der **Zuordnung** zwischen öffentlichem Schlüssel und dessen Besitzer getraut werden kann.
Jeder, der eine Nachricht chiffriert übertragen will, verschlüsselt sie mit dem öffentlichen Schlüssel des Empfängers. Das Chiffrat kann dann nur noch mit dem privaten Schlüssel des Empfängers, der sich ausschließlich in der Hand des Empfängers befindet, wieder entschlüsselt werden. Kein Zuhörer auf dem Übertragungsweg kann die Nachricht entschlüsseln. Es ist jedoch klar, daß niemand den privaten Schlüssel aus dem zugehörigen öffentlichen Schlüssel gewinnen können darf.
Die asymmetrische Kryptographie kann weiterhin nicht nur zur Verschlüsselung, sondern auch zur **Authentifikation** (*digitale Signaturen*) verwendet werden.
In den folgenden Kapiteln werden einige der bekanntesten asymmetrischen Verschlüsselungsverfahren beschrieben.

1.1 Vor- und Nachteile gegenüber symmetrischen Verfahren

Der primäre Vorteil von asymmetrischer Kryptographie ist die erhöhte Sicherheit und Bequemlichkeit: Private Schlüssel müssen nie übertragen oder jemandem gezeigt werden.
Ein weiterer wichtiger Vorteil ist die Möglichkeit der **Authentifikation** durch **elektronische Unterschriften** (*digitale Signaturen*). Eine Authentifikation mit symmetrischer Kryptographie erfordert das Teilen eines Geheimnisses.

Dadurch kann ein Sender eine zuvor von ihm authentifizierte Nachricht abstreiten, indem er behauptet, daß das geteilte Geheimnis durch einen der Beteiligten kompromittiert wurde.
Ein Nachteil der asymmetrischen Verschlüsselung ist die **Geschwindigkeit**. Es gibt beliebte symmetrische Verschlüsselungsverfahren, die deutlich schneller als jedes derzeit verfügbare asymmetrische Verfahren sind.
Die asymmetrische Verschlüsselung läßt sich mit symmetrischer kombinieren, um das Beste aus beiden Welten zu nutzen: Die Sicherheitsvorteile der asymmetrischen und die Geschwindigkeitsvorteile der symmetrischen Verschlüsselung. Dabei wird eine Nachricht mit einem symmetrischen Verfahren verschlüsselt und der zur Entschlüsselung nötige Schlüssel des symmetrischen Verfahrens mit einem asymmetrischen Verfahren verschlüsselt und zusammen mit der verschlüsselten Nachricht übertragen. Diese *Hybridtechnik* wird **digitaler Umschlag** genannt.

1.2 *Einwegfunktionen* und ihre Bedeutung für die Kryptographie

Eine *Einwegfunktion* ist eine mathematische Funktion, die (vorwärts) deutlich leichter zu berechnen ist, als die zugehörige Umkehrfunktion (rückwärts). Ein Rechner braucht beispielsweise nur einige Sekunden, um die Funktion für einen Wert zu berechnen, für die Umkehrung braucht er jedoch möglicherweise Monate oder sogar Jahre.
Eine *Einwegfunktion mit Hintertür* ist eine Einwegfunktion, bei der die Umkehrfunktion mittels einer **zusätzlichen Information** leicht berechnet werden kann, ohne diese Zusatzinformation jedoch nur sehr schwer.

Asymmetrische Kryptosysteme basieren auf einer (angenommenen) Einwegfunktion mit Hintertür. Der öffentliche Schlüssel enthält die Information, welche Einwegfunktion verwendet wird und der private Schlüssel enthält die Hintertür. Jeder, der diese Hintertür kennt, kann die Funktion leicht in beide Richtungen berechnen. Derjenige, dem diese Zusatzinformation fehlt, kann die Funktion jedoch nur vorwärts berechnen. Die Vorwärtsrichtung wird für die Verschlüsselung und Unterschriftenprüfung, die Rückwärtsrichtung für das Entschlüsseln und Unterschreiben benötigt.
Je größer die Eingabedaten der Einwegfunktion (und damit der Schlüssel) gewählt werden, desto größer ist auch der Unterschied in der Rechenzeit für die Hin- und Rückrichtung ohne Kenntnis der Hintertür. Für eine sichere Verschlüsselung oder eine elektronische Unterschrift, die über Jahre gültig sein soll, muß also eine Einwegfunktion mit Hintertür verwendet werden, deren Eingabedaten groß genug sind, um einen Angreifer ohne Kenntnis der Hintertür einige Jahre mit der Berechnung der Umkehrfunktion zu beschäftigen.
Alle praktisch verwendbaren asymmetrischen Kryptosysteme basieren auf **angenommenen Einwegfunktionen**, d.h. Funktionen, von denen man glaubt, daß es Einwegfunktionen sind, dieses jedoch bisher nicht bewiesen wurde.

2. RSA

2.0 Was ist RSA?

RSA ist ein asymmetrisches Kryptosystem, welches sich zum Verschlüsseln und zur Authentifikation (*digitale Signatur*) eignet. Es wurde 1977 von Ron **R**ivest, Adi **S**hamir und Leonard **A**dleman entwickelt [3] und nach ihnen benannt. *"RSA ist heutzutage das meistgenutzte "Public-Key"-Kryptosystem und wird oft als de facto Standard bezeichnet"* [4]. Es wird beispielsweise beim Verschlüsselungsprogramm "*Pretty Good Privacy*" (PGP) [5, 6, 7] von Phil Zimmermann eingesetzt.
RSA kann sowohl zur Verschlüsselung, als auch zur Authentifikation eingesetzt werden. Es wird als sicher angesehen, wenn genügend lange Schlüssel benutzt werden. Heute (1998) bedeutet dies, daß eine Schlüssellänge von 512 Bit als **unsicher**, 1024 Bit als **hinreichend sicher** für die meisten Anwendungen und 2048 Bit als **sehr sicher** angesehen wird. Für weitere Informationen zu den empfohlenen Schlüssellängen für RSA siehe [5], [6] und [8].
RSA ist derzeit der wichtigste "Public-Key"-Algorithmus. Er ist in den USA patentiert (US Patent 4,405,829, 1983), dieses Patent läuft am 20. September 2000 aus. Überall sonst ist RSA frei.

2.1 Wie funktioniert RSA genau?

RSA funktioniert folgendermaßen:

- Nehme zwei große Primzahlen *p* und *q*. (In der Praxis wählt man nur genügend große Pseudoprimzahlen. Das sind natürliche Zahlen, die die leicht und schnell überprüfbaren Primzahlenkriterien erfüllen.)
- Bilde deren Produkt $n = p*q$, welches *Modul* genannt wird.
- Wähle eine Zahl *e*, die kleiner als *n* und teilerfremd (*relativ prim*) zu *p-1* und *q-1* ist.
- Finde eine Zahl *d*, so daß (*e*d*)-1 durch (*p-1*)*(*q-1*) teilbar ist. D.h., (*e*d*) mod (*p-1*)*(*q-1*) = 1. Die Werte *e* und *d* werden *öffentlicher* und *privater Exponent* genannt.
- Der *öffentliche Schlüssel* ist das Paar (*n*, *e*), der *private Schlüssel* das Paar (*n*, *d*).
- Die Faktoren *p* und *q* werden sicher vernichtet oder zusammen mit dem privaten Schlüssel aufbewahrt.

Es wird angenommen, daß es schwierig ist, den privaten Exponenten *d* aus dem öffentlichen Schlüssel (*n*, *e*) zu berechnen. Wenn es jemand schafft, *n* in *p* und *q* zu faktorisieren, so kann er natürlich *d* berechnen. Deshalb ist die Sicherheit von RSA mindestens an die Annahme geknüpft, Faktorisierung sei schwierig. Ein leichter Faktorisierungsalgorithmus würde RSA "brechen".

Um eine Nachricht (beziehungsweise den Schlüssel eines symmetrischen Verfahrens, mit dem eine Nachricht verschlüsselt wurde) *M* mit RSA zu verschlüsseln, schreiben wir *M* als eine Zahl und berechnen:

$C = M^e \bmod n$,

wobei (*n*, *e*) der **öffentliche** Schlüssel des **Empfängers** und *C* die verschlüsselte Nachricht ist.

Um die Nachricht wieder zu entschlüsseln, berechnen wir:

$M = C^d \bmod n$,

wobei (*n*, *d*) der **private** Schlüssel des **Empfängers** ist.

Beispiel:

***p* = 5**, ***q* = 11**, $n = p*q = 55$, $(p-1)*(q-1) = 40$.
e muß kleiner als 55 und teilerfremd zu 40 sein. Wir wählen ***e* = 7**.
$(e*d) \bmod (p-1)*(q-1)$ soll gleich 1 sein. Wir setzen ***d* = 23**,
da $23*7 = 161 = 1 \bmod 40$.

Verschlüsselung:

Angenommen, die zu verschlüsselnde Nachricht *M* sei 2.
Dann ist $C = M^e \bmod n = 2^7 \bmod 55 = 18$.

Entschlüsselung:

$M = C^d \bmod n = 18^{23} \bmod 55 = 18^{(1+2+4+16)} \bmod 55$
$= (18^1 * 18^2 * 18^4 * 18^{16}) \bmod 55 = (18 * 49 * 36 * 26) \bmod 55 = 2$.

2.2 Wie kann RSA gebrochen werden?

Es gibt einige mögliche Interpretationen, was es heißt, RSA zu "brechen".

Den meisten Schaden richtet sicherlich eine **Offenlegung des privaten Schlüssels** an. Dies erlaubt dem Angreifer, **alle** Nachrichten zu lesen und Unterschriften zu fälschen.
Es drängt sich daher der Versuch auf, das öffentliche *Modul n* in seine Primfaktoren *p* und *q* zu zerlegen, denn aus *p*, *q* und dem *öffentlichen Exponenten e* läßt sich leicht der *private Exponent d* und somit der private Schlüssel errechnen. Aber genau auf diesem schwierigen Problem der Faktorisierung von *n* in seine **genau zwei** Primfaktoren beruht die Sicherheit von RSA.

Ein anderer Weg, RSA zu brechen, besteht darin, einen Algorithmus zur Berechnung der *e*-ten modularen Wurzel zu finden. Da $c = M^e$ ist, ist die *e*-te modulare Wurzel aus *c* bezüglich *n* gleich der Nachricht *M*. Dieser Angriff gestatt es dem Angreifer, Nachrichten zu entschlüsseln und Unterschriften zu fälschen, ohne den privaten Schlüssel kennen zu müssen. Es ist bisher nicht bekannt, ob dieser Ansatz äquivalent zur Faktorisierung ist oder nicht. Es sind bis heute aber keine generellen Algorithmen bekannt, die RSA in dieser Weise gefährlich werden könnten.

Man sollte beachten, daß der RSA Algorithmus **sehr anfällig** gegen ***gewählte Klartext-Angriffe*** (*Chosen-Plaintext Attack*) ist. Dabei ist der Angreifer in der Lage, jeden **beliebigen** Klartext mit dem privaten Schlüssel des Opfers verschlüsseln bzw. signieren zu lassen und die Ergebnisse wieder einzusehen. Durch einen Vergleich dieser Ergebnisse läßt sich auf den verwendeten privaten Schlüssel rückschließen.
Genauso steht es mit ***gewählten Chiffretext-Angriffen*** (*Chosen-Ciphertext Attack*), bei denen der Angreifer beliebige Chiffretexte erstellen und diese mit dem privaten Schlüssel des Opfer entschlüsseln lassen kann.
Bei der Benutzung des RSA Algorithmus sollte also sehr darauf geachtet werden, daß es einem Angreifer **auf keinen Fall** gelingen kann, **beliebige** Texte ver- bzw. entschlüsseln zu lassen und wieder einsehen zu können.

Außer diesen Angriffsmethoden, die RSA so brechen, daß der Angreifer **alle** Nachrichten lesen und Unterschriften eines bestimmten Schlüssels fälschen kann, gibt es weitere Methoden, die eine **einzelne** Nachricht aufdecken können, jedoch keine weiteren desselben Schlüsselpaares.

Der einfachste Angriff auf Einzelnachrichten ist das ***Erraten eines Klartextes***. Ein Angreifer errät zu einem Chiffrat den Klartext und verschlüsselt ihn mit dem öffentlichen Schlüssel des Empfängers. Stimmt seine Verschlüsselung mit dem Chiffrat überein, so hat er vollkommen richtig geraten.
Um sich gegen diese Art von Angriff zu schützen, reicht es, jeweils einige zufällige Bits im Klartext unterzubringen.

Eine andere Angriffsmöglichkeit auf eine Einzelnachricht ist dann gegeben, wenn der Sender einen Text an mehrere Empfänger verschlüsselt, die alle den **gleichen kleinen öffentlichen Exponenten** haben. Weiß der Angreifer dies, so kann er die Nachricht über den *Chinesischen Restesatz* gewinnen [9]. Glücklicherweise genügt es, die Nachricht in nur wenigen Füllbits an die verschiedenen Empfänger zu verändern, um sich gegen diesen Angriff zu schützen.

Natürlich gibt es auch Angriffe, die nicht auf RSA selbst, sondern auf eine unsichere oder fehlerhafte Implementation von RSA zielen. Dies zählt jedoch nicht als ein "Brechen" von RSA, da kein Schwachpunkt des Algorithmus selbst ausgenutzt wird. Wenn ein Benutzer beispielsweise seinen privaten Schlüssel unsicher abspeichert, so kann ihn ein Angreifer auffinden. Es kann nicht deutlich genug gesagt werden, daß ein sicheres RSA-Kryptosystem eine sichere Implementation voraussetzt. Mathematische Sicherheit, wie die Wahl eines genügend großen Schlüssels, ist zwar wichtig, aber nicht hinreichend.

Der RSA Algorithmus wird als sicher angesehen, wenn er richtig benutzt wird, aber man muß bei der Benutzung auch sehr vorsichtig sein, um sich gegen die genannten Attacken zu schützen.

3. Diffie-Hellman

3.0 Was ist Diffie-Hellman?

Das **Diffie-Hellman-Exponential-Schlüsselaustausch-Protokoll** (*Diffie-Hellman Key Agreement Protocol*) ist eine unglaublich einfache Methode, die es zwei Kommunikationspartnern erlaubt, kryptographische Schlüssel auszutauschen. Dabei ist es einem Unbefugten, der den Transfer mitliest, nicht möglich, den übermittelten Schlüssel zu bestimmen.
Beide Teilnehmer starten jeweils mit einem privaten Schlüssel. Durch den Austausch von Informationen, die mit diesen Schlüsseln kodiert werden, ergibt sich ein neuer Schlüssel, der dann für alle weiteren Chiffriervorgänge benutzt wird.
Die Sicherheit von Diffie-Hellman beruht auf dem Problem der Berechnung von diskreten Logarithmen, von dem angenommen wird, daß es hinsichtlich der benötigten Rechenzeit mit der Faktorisierung von großen Zahlen vergleichbar ist.
Das US Patent für Diffie-Hellman (US Patent 4,200,700, 1980) ist am 29. April 1997 ausgelaufen.

3.1 Wie funktioniert Diffie-Hellman?

Das *Diffie-Hellman-Exponential-Schlüsselaustausch-Protokoll* hat zwei Systemparameter *p* und *g*, die öffentlich sind und von allen Benutzern eines Systems genutzt werden können. Der Parameter *p* ist eine Primzahl und der Parameter *g* (üblicherweise *Generator* genannt) ist eine natürliche Zahl, die in der Lage ist, jede beliebige Zahl zwischen 1 und *p*-1 zu generieren, wenn sie nur oft genug mit sich selber multipliziert und modulo der Primzahl *p*-1 genommen wird.

Angenommen, zwei Kommunikationspartner A und B möchten sich auf einen gemeinsamen **geheimen** Schlüssel einigen und dafür das Diffie-Hellman-Protokoll benutzen.
Dazu gehen sie folgendermaßen vor:

- A und B erzeugen jeder eine **private** Zufallszahl *Pa* bzw. *Pb*, die kleiner als *p* ist und die den **privaten Schlüssel** darstellt.
- A und B leiten jeder von ihren privaten Schlüsseln unter Benutzung der Parameter *p* und *g* ihre öffentlichen Schlüssel ab, indem A rechnet: $Oa = g^{Pa} \bmod p$ und B rechnet: $Ob = g^{Pb} \bmod p$.
- A und B tauschen ihre öffentlichen Schlüssel aus.

- Nun berechnet A: $Kab = (Ob)^{Pa} \bmod p = (g^{Pb})^{Pa} \bmod p$
  und B: $Kba = (Oa)^{Pb} \bmod p = (g^{Pa})^{Pb} \bmod p$
  Da $Kab = (g^{Pb})^{Pa} = (g^{Pa})^{Pb} = Kba = K$ gilt, haben A und B nun einen gemeinsamen **geheimen Schlüssel** K.

Die Sicherheit des Diffie-Hellman-Protokolls beruht auf der Schwierigkeit, diskrete Logarithmen schnell und effizient zu berechnen. Es wird angenommen, daß es in einem angemessenen Zeitraum rechentechnisch nahezu unmöglich ist, den geheimen Schlüssel $K = g^{Pa*Pb} \bmod p$ zu berechnen, wenn nur die öffentlichen Schlüssel $g^{Pa} \bmod p$ und $g^{Pb} \bmod p$ gegeben sind und die Primzahl p groß genug ist.

3.2 Wie kann Diffie-Hellman gebrochen werden?

Der *Diffie-Hellman-Schlüsselaustausch* ist anfällig gegen einen ***Man-In-The-Middle - Angriff*** (*Middleperson-Attack*). Dabei "hängt" sich ein Angreifer C zwischen die beiden Kommunikationspartner A und B, die einen Schlüssel austauschen möchten.
Wenn A seinen öffentlichen Schlüssel an B schickt, fängt C diesen ab und schickt stattdessen seinen eigenen öffentlichen Schlüssel an B weiter. Wenn B seinen öffentlichen Schlüssel an A schickt, fängt C diesen ebenfalls ab und schickt auch A seinen eigenen öffentlichen Schlüssel.
Das bedeutet, daß nun A und C einen gemeinsamen geheimen Schlüssel haben und C und B einen anderen.

Nach diesem Austausch fängt C nun jede verschlüsselte Nachricht zwischen A an B ab und liest und modifiziert sie eventuell, bevor er sie mit dem passenden Schlüssel wieder verschlüsselt und an den eigentlichen Empfänger weitersendet.
Diese Verwundbarkeit des Protokolls begründet sich darauf, daß der Diffie-Hellman-Schlüsselaustausch die Kommunikationspartner nicht gegenseitig authentifiziert. Mögliche Abhilfe würde die Benutzung von digitalen Signaturen oder anderen Protokoll-Variationen schaffen.

4. Weitere asymmetrische Verfahren

4.0 Elliptische Kurven

Elliptische Kurven (*elliptic curves*)sind mathematische Konstruktionen aus der Zahlentheorie und der algebraischen Geometrie, die in den letzten Jahren zunehmend Bedeutung für die Kryptographie gefunden haben. Ihre Berechnung ist zwar sehr langsam, mit modernen Rechnern jedoch inzwischen machbar.
Kryptosysteme auf der Basis von Elliptischen Kurven (*elliptic curve cryptosystems*) [10, 11, 12] sind vergleichbar mit "Public-Key"-Kryptosystemen wie RSA oder ElGamal, bei denen die modulare Potenzierung durch eine Additionsoperation in einer Elliptischen Kurve ersetzt wurde.
Sie werden als sehr sicher angesehen, sind jedoch bisher noch nicht den gleichen Überprüfungen unterzogen worden, wie zum Beispiel RSA. Ein sehr informativer Text ist in [13] zu finden.

4.1 ElGamal

Das *ElGamal-Kryptosystem* [14] ist ein asymmetrisches Verfahren, welches auf dem *Diskreten Logarithmus-Problem* basiert. Es enthält sowohl Verschlüsselungs-, als auch Unterschriftsalgorithmen für digitale Signaturen. Der Verschlüsselungsalgorithmus ist prinzipiell mit dem Diffie-Hellman Schlüsselaustausch identisch.
Eine Analyse auf der Basis der besten verfügbaren Faktorisierungsalgorithmen hat gezeigt, daß RSA und ElGamal bei gleicher Schlüssellänge ähnliche Sicherheit bieten. Der Hauptnachteil von ElGamal ist die Forderung nach Zufallszahlen und die etwas langsamere Geschwindigkeit (speziell beim digitalen Signieren). Ein anderer möglicher Nachteil von ElGamal ist, daß das Chiffrat doppelt so groß, wie die Klartext-Nachricht ist. Dies ist jedoch in Hybridsystemen, wo nur die geheimen Schlüssel eines symmetrischen Verfahrens asymmetrisch verschlüsselt werden, vernachlässigbar.

4.2 LUC

LUC [15] ist ein "Public-Key"-Kryptosystem, welches von einer Gruppe von Forschern in Australien und Neuseeland entwickelt wurde und *Lucas*-Funktionen anstelle der Potenzierung benutzt.
LUCDIF ist das Analogon zu Diffie-Hellman, *LUCRSA* zu RSA, *LUCELG PK* zur ElGamal "Public-Key" Verschlüsselung, *LUCELG DS* zur digitalen Signatur von ElGamal und *LUCDSA* zum US Digital Signature Standard.
Eine Veröffentlichung von Bleichenbacher et al. [16] zeigt jedoch, daß viele der mutmaßlichen Sicherheitsvorteile von LUC gegenüber Kryptosystemen, die auf modularer Potenzierung basieren, entweder nicht vorhanden oder nicht so bedeutend sind, wie behauptet wurde.

LUC Encryption Technology Ltd. (*LUCENT*) hat Patente für die kryptographische Verwendung von *Lucas*-Funktionen in den USA und Neuseeland erhalten.

4.3 DSA / DSS

Der *Digital Signature Algorithm* (DSA) wurde vom *National Institute of Standards and Technology* (NIST) im Rahmen des *Digital Signature Standard* (DSS) veröffentlicht, der selbst Teil des US-Regierungsprojekts *Capstone* ist. DSS wurde vom NIST in Zusammenarbeit mit der *National Security Agency* (NSA) ausgewählt, um einen elektronischen Unterschriftenstandard für die US-Behörden zu erhalten. Dieser Standard wurde am 19. Mai 1994 veröffentlicht.
DSA basiert auf dem *Diskreten Logarithmus-Problem* und benutzt die Kryptosysteme von Schnorr [17] und ElGamal. Der Algorithmus ist generell als sicher anzusehen, wenn die verwendete Schlüssellänge hinreichend ist. DSS wurde ursprünglich vom NIST mit festen 512 Bit Schlüsseln vorgeschlagen. Nach heftiger Kritik, die sich speziell auf die langfristige Wirkung von Unterschriften bezog, hat das NIST die Schlüssellänge auf 1024 Bit erhöht. DSA eignet sich **nur zur Authentifikation**. Eine detaillierte Beschreibung findet sich in [18] und [19].

DSA wurde seit seiner Ankündigung von der Computerindustrie kritisiert. Die Kritik behandelte besonders folgende zentrale Punkte:

- Es ist kein Schlüsselaustausch möglich
- Das zugrundeliegende Kryptosystem ist zu jung und nicht ausreichend erforscht, um von der Sicherheit überzeugt sein zu können
- Die Überprüfung von Unterschriften ist zu umständlich
- Die Existenz eines zweiten Unterschriftenverfahrens behindert die Hard- und Softwarehersteller, da die meisten bereits RSA lizensiert haben
- Die Entwicklung von DSA war zu geheimniskrämerisch und überraschend (der Einfluß der NSA war zu groß)

Weitere Kritikpunkte betreffen die Veränderungen, die das NIST am Originalvorschlag vorgenommen hat. Eine genaue Aufstellung der Kritikpunkte ist in [19] und eine ausführliche Antwort darauf in [20] zu finden.

5. Quellen und Literaturhinweise

- [1] W. Diffie and M.E. Hellman. *New Directions in Cryptography*. IEEE Transactions on Information Theory IT-22, 1976.
- [2] W. Diffie: *The First Ten Years of Public-Key Cryptography*. Proceedings of the IEEE 76, 1988.
- [3] R.L. Rivest, A. Shamir and L.M. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM 21, 1978.
- [4] *RSA Laboratories Frequently Asked Questions About Today's Cryptography* (Cryptography FAQs): http://www.rsa.com/rsalabs/newfaq/
- [5] S. Garfinkel. *PGP: Pretty Good Privacy*, 1994.
- [6] *PGP-FAQ*. http://www.pgp.net/pgpnet/pgp-faq/ , deutschsprachige Fassung: http://www.iks-jena.de/mitarb/lutz/security/pgpfaq.html
- [7] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.
- [8] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 1994.
- [9] J. Hastad. *Solving simultaneous modular equations of low degree.* SIAM Journal of Computing 17, 1988.
- [10] V.S. Miller. *Use of elliptic curves in cryptography*. Advances in Cryptology - Crypto '85, S. 417-426, Springer-Verlag, 1986.
- [11] N. Koblitz. *Elliptic curve cryptosystems*. Mathematics of Computation 48, S. 203-209, 1987.
- [12] IEEE P1363 Draft Standard on *Elliptic Curve Cryptosystems and Related Methods.*
- [13] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, 1994.
- [14] T. ElGamal. *A public-key cryptosystem and a signature scheme based on discrete logarithms*. IEEE Transactions on Information Theory 31, S. 469-472, 1985.
- [15] P. Smith and C. Skinner. *A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms.* Advances in Cryptology - Asiacrypt '94, S. 357-364, Springer-Verlag, 1995.
- [16] D. Bleichenbacher, W. Bosma, and A. Lenstra. *Some remarks on Lucas-based cryptosystems.* Advances in Cryptology Crypto '95, S. 386-396, Springer-Verlag, 1995.
- [17] C.P. Schnorr. *Efficient identification and signatures for smart cards.* Advances in Cryptology - Crypto '89, S. 239-251, Springer-Verlag, 1990.
- [18] National Institute of Standards and Technology (NIST). FIPS Publication 186: *Digital Signature Standard (DSS)*, May 1994.
- [19] National Institute of Standards and Technology (NIST). *The Digital Signature Standard, proposal and discussion*. Communications of the ACM 35, July 1992.
- [20] M.E. Smid and D.K. Branstad. *Response to comments on the NIST proposed Digital Signature Standard*. Advances in Cryptology - Crypto '92, S. 76-87, Springer-Verlag, 1993.