DIFFIE-HELLMAN, ELGAMAL UND DSS

AUSGEARBEITET VON DAVID GÜMBEL

VORTRAG IM RAHMEN VOM *SEMINAR KRYPTOLOGIE*

UNIVERSITÄT TÜBINGEN, 27.05.2002

INHALTSVERZEICHNIS

Einleitung.....1

Prinzipielle Probleme bei der sicheren Nachrichtenübermittlung.....1

Unsicherer Kanal.....1

Authentizität.....2

Theoretische Grundlagen.....3

Ein-Wege-Funktionen.....3

„Diskreter-Logarithmus“-Problem.....3

Diffie-Hellman Schlüsselaustausch.....4

Idee.....4

Verfahren.....4

Diffie-Hellman-Problem.....5

Historie.....5

Praxis.....6

ElGamal.....7

Idee.....7

Verfahren.....7

Digital Signature Algorithm (DSA).....8

Verfahren.....8

Praxis.....9

Performance.....10

Verschlüsselung – ElGamal und RSA.....10

Signatur – RSA und DSS.....11

Praxisbeispiel: GnuPG.....12

Einsatz der Verfahren.....12

„Dreigeteilte Schlüssel“.....12

Geschwindigkeit.....13

Adressen, Quellen, URLs, Informationen.....14

EINLEITUNG

Heute gängige asymmetrische Kryptosysteme basieren auf dem Problem, zu einer gegebenen Funktion f und einem Wert y=f(x) die Umkehrfunktion und damit x zu berechnen. Hierbei sind insbesondere diejenigen Funktionen von Interesse, deren Berechnung in geringer (Rechen)zeit erfolgen kann, während ihre Umkehrung nur mit Hilfe von zusätzlichen Informationen durchführbar wird.

Im Falle des am weitesten verbreiteten asymmetrischen Verschlüsselungsverfahren RSA wurde das Problem der Faktorisierung großer Zahlen als Fundament des Algorithmus herangezogen. Es sind jedoch natürlich auch Systeme denkbar, die auf anderen als diesem basieren. Ein weiteres – das „diskreter-Logarithmus-Problem“ - soll mit aus ihm gewonnenen Algorithmen im Folgenden vorgestellt werden. Hierbei werden die Verfahren in den Kontext der allgemeinen Problematik bei der Übermittlung geheimer Nachrichten eingeordnet und auch Einblick in tatsächliche Implementierungen gegeben.

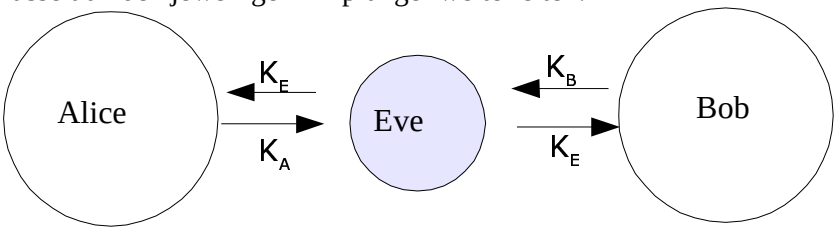
PRINZIPIELLE PROBLEME BEI DER SICHEREN NACHRICHTENÜBERMITTLUNG

UNSICHERER KANAL

Im Regelfall ist der Kanal, der für die Übermittlung von Nachrichten benutzt wird, nicht sicher, das heißt er kann von Dritten abgehört werden. Obwohl dies häufig unwahrscheinlich erscheint (z.B. bei privatem Mailverkehr) so ist es dennoch sinnvoll, von einem unsicheren Kanal auszugehen, und die Verschlüsselungsverfahren an dieser Problematik zu orientieren. Die „Sicherheit“ eines Kanals ist eine mathematisch nicht zu erfassende Größe, was Betrachtungen über ein Kryptosystem erschweren würde, wenn man von (relativ) sicheren Kanälen ausginge.

Für symmetrische Verschlüsselungsverfahren muß vor dem eigentlichen Nachrichtentransfer eine Übereinstimmung zwischen Sender und Empfänger über den Schlüssel erfolgen. Dies muß über einen sicheren Kanal geschehen, da sonst der Schlüssel abgehört werden könnte und somit die Verschlüsselung nutzlos geworden wäre. Dies offenbart eine prinzipielle Schwäche von symmetrischen Verfahren: Wenn vor der eigentlichen Nachrichtenübermittlung ein sicherer Kanal für den Schlüsselaustausch erforderlich ist, warum kann dieser dann nicht gleich auch für die Nachrichtenübermittlung benutzt werden?

Asymmetrische Verfahren benötigen keinen sicheren Kanal für den Schlüsselaustausch. Allerdings kann hier ein Angreifer E (hier Eve genannt) über eine so genannte Middleperson-Attacke beim Schlüsselaustausch ebenfalls in Besitz der geheimen Nachricht gelangen. Hierzu fängt Eve die öffentlichen Schlüssel K_A bzw. K_B der beiden Kommunikationspartner Alice und Bob ab, und sendet Alice bzw. Bob ihren eigenen K_E. Alice und Bob verschlüsseln nun mit Eves Schlüssel – in der Annahme, es handele sich um K_A bzw K_B - und Eve kann die Nachricht mit K_A oder K_B verschlüsselt an den jeweiligen Empfänger weiterleiten.



Zeichnung 1: Middleperson-Attacke

Middleperson-Angriffe (auch „Man-in-the-middle“-Angriffe genannt) lassen sich für bekannte Protokolle wie SSH und SSL in der Praxis relativ leicht durchführen. Es existieren sogar Toolkits wie dsniff, welche diese Angriffe automatisieren.

AUTHENTIZITÄT

Sendet Alice Bob eine Nachricht, so passiert diese den unsicheren Nachrichtenkanal und kann – trotz Verschlüsselung – abgehört werden. Bei asymmetrischen Verfahren ist der öffentliche Schlüssel des Empfängers allgemein bekannt; jede(r) kann ihm also eine verschlüsselte Nachricht zukommen lassen. Der Empfänger steht daher vor dem Problem, sicherzustellen, daß die Nachricht wirklich von dem vorgegebenen Absender stammt. Hierfür existieren Signaturverfahren, die auf asymmetrischen Verschlüsselungsalgorithmen wie z.B. RSA aufbauen.

THEORETISCHE GRUNDLAGEN

EIN-WEGE-FUNKTIONEN

Wie bereits eingangs erwähnt, basieren die gängigen asymmetrischen Verfahren auf einer Funktion f, welche selbst leicht zu berechnen ist; nicht jedoch ihre Umkehrung. Zur Berechnung von f^{-1} wird eine zusätzliche Information benötigt. Funktionen dieses Typs heißen „ein-Wege"-Funktionen. Ein Beispiel wäre die Berechnung von $f(x) = x^e$ mod n im Falle der RSA-Verschlüsselung. Hier ist es sehr teuer, f^{-1} zu berechnen, wenn man nicht die Faktoren p-1 und q-1 von n (n = (p-1)*(q-1)) kennt. Für genügend große n ist der Rechenaufwand in existierenden Rechnern erheblich zu groß; die Umkehrung ist nur mit dem privaten Schlüssel, d.h. mit Kenntnis von n zu berechnen.

„DISKRETER-LOGARITHMUS"-PROBLEM

Ein weiteres mathematisches Problem, aus dem sich eine für die Kryptographie nützliche Funktion gewinnen läßt, ist das sogenannte „diskrete Logarithmus Problem":

Sei F_p^* die multiplikative Gruppe des Körpers F_p von Ganzzahlen modulo einer Primzahl p. Sei weiter g F_p^* ein festes Element (die Basis). Das diskrete Logarithmus Problem in F_p^* zur Basis g ist dann das Problem, zu einem gegebenen y F_p^* eine Ganzzahl x zu finden, so daß $y=g^x$. Falls ein solches x nicht exisitert, muß dies ebenfalls aus der Ausgabe erkennbar sein.

DIFFIE-HELLMAN SCHLÜSSELAUSTAUSCH

IDEE

Da asymmetrische Verfahren im Vergleich mit symmetrischen hohen Rechenaufwand erfordern, wird häufig bei der tatsächlichen Übermittlung von Nachrichten ein Mittelweg gewählt: Zunächst einigen sich die Kommunikationspartner Alice und Bob mittels Public-Key-Kryptographie (d.h. ihren öffentlichen Schlüsseln) auf einen Schlüssel für ein symmetrisches Verfahren und benutzen dieses dann. Dieses Vorgehen nennt sich „hybride Verschlüsselung“. Beispielsweise könnten sich Alice und Bob mit Hilfe ihrer RSA-Schlüssel gegenseitig auf ein Paßwort für den AES-Verschlüsselungsalgorithmus einigen und ihre geheimen Daten dann mittels AES austauschen.

VERFAHREN

Zur Vereinfachung sei im Folgenden angenommen, daß der Schlüssel (das Paßwort), auf das sich Alice und Bob einigen wollen, eine große Ganzzahl ist. Zwischen den Kommunikationspartnern existiert kein sicherer Kanal, weshalb alle Informationen, die die beiden sich zukommen lassen, auch der Angreiferin Eve zur Verfügung stehen.

Alice und Bob einigen sich nun zunächst auf eine Primzahl p und ein Basiselement g aus F_p^*. Beide Zahlen sind auch Eve bekannt. Im nächsten Schritt wählt Alice (geheim) eine zufällige Ganzzahl k(Alice) < p von etwa der selben Größenordnung wie p, berechnet den kleinsten positiven Rest von $g^{k(Alice)}$ modulo p, und sendet diesen an Bob. Analog hierzu sendet Bob $g^{k(Bob)}$ mod p an Alice. Der Schlüssel, auf den die beiden sich nun verständigt haben, ist die Zahl

$$g^{K(Alice)K(Bob)} \quad F_p^*$$

Bob kann den Schlüsel nun berechnen, indem er die Zahl $g^{k(Alice)}$, welche er von Alice erhalten hat, in die K(Bob)-te Potenz modulo p erhebt. Analog erhält Alice den Schlüssel durch Berechnung von $g^{k(Bob)K(Alice)}$. Dies ist deshalb möglich, weil in F_p^* gilt:

$$g^{K(Alice)K(Bob)} = (g^{K(Alice)})^{K(Bob)}) = g^{k(Bob)(K(Alice)}$$

DIFFIE-HELLMAN-PROBLEM

Das Problem, dem sich bei Anwendung des Diffie-Hellman- Schlüsselaustauschverfahrens die Angreiferin Eve gegenübersieht, lautet nun wie folgt:
Gegeben g, $g^{K(Alice)}$, $g^{K(Bob)}$ F_p^*, finde $g^{K(Alice)K(Bob)}$.

Dieses Problem wird auch das „Diffie-Hellman-Problem“ genannt. Es ist leicht zu erkennen, daß eine Lösung des diskreter-Logarithmus-Problem in F_p^* ebenfalls direkt zu einer Lösung des Diffie-Hellman-Problems führen würde. Über den Umkehrschluß ist allerdings nichts bekannt, d.h. es ist möglich (wenn auch unwahrscheinlich), daß jemand einen Weg finden könnte, das Diffie-Hellman-Problem ohne diskrete Logarithmen zu lösen. Einen formalen Beweis, der belegen würde, daß Diffie-Hellman und diskreter-Logarithmus äquivalent sind, gibt es (noch?) nicht. In der Praxis ist es wahrscheinlich sicher, anzunehmen, daß das Diffie-Hellman-Schlüsselaustauschverfahren sicher ist, solange das diskreter-Logarithmus-Problem keine real durchführbare Lösung besitzt.

HISTORIE

Im Allgemeinen läßt sich die „Sicherheit“ eines Kryptosystems nicht beweisen. Daher ist es für ein Verfahren umso wichtiger, daß es auf einem mathematisch gut erforschten Problem basiert, über das sich einigermaßen verläßliche Aussagen betreffend seiner Schwierigkeit und seiner Komplexität treffen lassen. Hierfür ist eine Analyse von Experten, möglichst über einen langen Zeitraum hinweg, unerläßlich.

Das RSA-Kryptoverfahren ist bereits seit 1977 bekannt. Das zugrundeliegende Problem – die Faktorisierung großer Zahlen - ist seit langer Zeit Gegenstand mathematischer Untersuchungen. Daher kann das RSA-System als entsprechend sicher gelten. 1976 publizierten Diffie und Hellman in ihrem Paper "New Directions in Cryptography" das Diffie-Hellman-Verfahren. Auch hier kann von relativ großer Sicherheit ausgegangen werden, da auch hier bereits seit über 25 Jahren führende Mathematiker und Kryptologen versuchen, das Verfahren zu brechen.

Selbstverständlich besteht dennoch nach wie vor die Möglichkeit, daß beispielsweise ein Algorithmus zur effizienten Berechnung von diskreten Logarithmen entdeckt wird.

PRAXIS

Wegen seiner mathematischen Schlichtheit ist ein Diffie-Hellman-Schlüsselaustausch in Software leicht zu implementieren. Die kleinste Inkarnation paßt in zwei Zeilen Perl-Code.

Da Diffie-Hellman seit 1997 von keinerlei Patenten mehr geschützt wird (wie es aber z.B. bei RSA bis 2000 der Fall war), existieren zahlreiche Implementierungen. Zu nennen wären hier vor allem die freie kryptographische Bibliothek OpenSSL, diverse Kryptographie-Kits (Cryptlib, BeeCrypt,...) und die Java-Klassen des JCE. Zudem findet sich der Algorithmus in Verschlüsselungsprogrammen wie PGP und GnuPG und in Standards wie OpenPGP und S/MIME, welche insbesondere das sichere Austauschen von Nachrichten im Internet betreffen. Auch Software wie IPSec, das der sicheren Vernetzung per IP dient, benutzt Diffie-Hellman. Sogar ein Standard für die Integration von Diffie-Hellman in DNS-Einträge existiert in RFC 2539. Zusammen mit der DSS-Signatur und dem ElGamal-Kryptoverfahren, welche im Folgenden beide noch vorgestellt werden sollen, liegt DH standardmäßig GnuPG-Schlüsseln zugrunde.

Ein deutlicher Vorteil des Diffie-Hellman Schlüsselaustausches gegenüber RSA wird deutlich,wenn man folgendes Szenario betrachtet: Angenommen, Alice und Bob tauschen verschlüsselte Nachrichten über den unsicheren Kanal aus, und die Angreiferin Eve protokolliert allen Datenerkehr zwischen den beiden mit. Wenn Eve später irgendwann einmal - beispielsweise durch einen Einbruch – in den Besitz des privaten Schlüssels eines der beiden gelangt, kann sie die mitgelesene Kommunikation nachträglich entschlüsseln. Dies ist deshalb möglich, weil beide Empfänger für die Generierung von Signaturen ihren privaten Schlüssel aufbewahren müssen.

Bei Diffie-Hellman jedoch werden im Regelfall lediglich „Session-Keys“ generiert. Hierunter versteht man Schlüssel, die nur für einen einmaligen Nachrichtenaustausch verwendet werden sollen – eine Art „Wegwerf-Schlüssel“. Vernichten Alice und Bob nach Beendigung der Kommunikation den Schlüssel, auf den sie sich per Diffie-Hellman geeinigt haben, kann Eve die mitgelesene Kommunikation selbst dann nicht mehr entschlüsseln, wenn sie zu einem späteren Zeitpunkt in den Besitz z.B. der Rechner von Alice oder Bob kommt.

ELGAMAL

IDEE

Das diskrete-Logarithmus-Problem läßt sich auf vielfältige Arten in der Kryptographie einsetzen; Diffie-Hellman ist lediglich eine Möglichkeit von vielen. Es wäre beispielsweise wünschenswert, über die Möglichkeiten von Diffie-Hellman hinauszugehen und auf dem DLP ein asymmetrisches Verschlüsselungsverfahren aufzubauen. Ein solches Verfahren wäre beispielsweise der sog. ElGamal-Algorithmus.

VERFAHREN

Da es sich bei ElGamal um ein asymmetrisches Verschlüsselungsverfahren handelt, müssen für jeden Benutzer A zwei Schlüssel generiert werden: der öffentliche und der private. Dies geschieht wie folgt:

Zunächst wählt A einen sehr großen endlichen Körper F_p und ein Element g F_p^* (g kann, muß aber kein Erzeuger sein). Nun wählt A eine zufällige Ganzzahl a im Intervall $0 < a < p-1$. a ist der geheime private Schlüssel, g^a der öffentliche.

Um eine geheime Nachricht P an A zu schicken, nimmt man eine Zufallszahl k und sendet das Paar

$$(g^k, Pg^{ak})$$

an A.

Genau wie bei Diffie-Hellman kann g^{ak} berechnet werden, ohne Kenntnis von a zu haben, und zwar einfach durch das Erheben von g^k in die a-te Potenz. Der Empfänger A, welcher a kennt, kann nun P berechnen, indem er das erste Element des Paares in die a-te Potenz erhebt und das zweite Element durch das Ergebnis teilt. Die Information, die an A über den unsicheren Kanal übermittelt wird, enthält sozusagen die Nachricht P in einer „Verkleidung“ g^{ak}, zusammen mit einem „Hinweis“ g^k, mittels dem die Verkleidung entfernt werden kann.

Auch hier ist deutlich sichtbar, daß eine Lösung des DLP sofort einen Bruch von ElGamal bedeuten würde, denn dann ließe sich der geheime Schlüssel a direkt aus dem öffentlichen g^a berechnen. Theoretisch wäre es außerdem denkbar, daß sich mittels Kenntnis von g^k und g^a g^{ak} ermitteln ließe. Bisher ist lediglich bewiesen, daß ElGamal und das Diffie-Hellman-Problem äquivalent sind.

DIGITAL SIGNATURE ALGORITHM (DSA)

VERFAHREN

Bei DSA handelt es sich um einen offiziellen US-Standard namens DSS (Digital Signature Standard), der von der amerikanischen Behörde „National Institute of Standards and Technology“ (NIST) vorgeschlagen wurde. Das Verfahren basiert auf auf dem diskreter-Logarithmus-Problem in einem Primkörper F_P. Um in der Lage zu sein, digitale Signaturen nach dem DSS-Standard durchzuführen, geht jeder Benutzer wie folgt vor:

1. er wählt eine Primzahl q von etwa 160 Bit Länge
2. er wählt eine zweite Primzahl p = 1 mod q von etwa 500 Bit Länge. Die vom Standard vorgegebene Länge liegt bei einem Vielfachen von 64 Bits und zwischen 512 und 1024 Bit
3. er wählt einen Erzeuger g der zyklischen Untergruppe von F_p^*. Dies geschieht durch Berechnung von $g_0^{(p-1)/q}$ mod p für ein zufälliges g_0; falls diese Zahl ungleich eins ist, ist g_0 ein Erzeuger
4. er wählt eine zufällige Ganzzahl x im Intervall $0 < x < q$ als privaten Schlüssel, und legt den öffentlichen Schlüssel mit $y = g^x$ mod p fest.

Um nun tatsächlich eine Nachricht zu signieren, berechnet er zunächst einen Hashwert $0 < H < q$ der zu signierenden Nachricht. Dann wählt er eine Zufallszahl k im gleichen Intervall, berechnet g^k mod p, und setzt $r = (g^k \bmod q) \bmod p$. Schließlich findet er noch eine Zahl s mit sk = H + xr mod q. Die Signatur ist nun das Paar (r,s) von Ganzzahlen modulo q.

Die Verifikation von Nachrichten funktioniert wie folgt:

1. Berechne $u = s^{-1}H \bmod q$
2. Berechne $t = s^{-1}r \bmod q$
3. Überprüfe: $g^u y^t \bmod q = r$
   Falls diese Gleichung erfüllt ist, ist die Signatur gültig. Zu beachten ist, daß nur jemand mit Kenntnis von x (dem privaten Schlüssel) in der Lage ist, ein gültiges s zu berechnen.

Daß die Beziehung $g^u y^t \bmod q = r$ tatsächlich die Gültigkeit einer Signatur angibt, sieht man wie durch simples Einsetzen, denn es gilt: $\mathbf{g^u y^t} = g^u g^{xt} = g^{(H + xr)/s} = g^k = \mathbf{r}$

PRAXIS

Das DSA-System kommt beispielsweise in der Form von DSS in PGP- bzw. GnuPG-Schlüsseln für die Signatur zum Einsatz. Die hierbei verwendete Schlüssellänge von 1024 Bit liegt an der oberen Grenze des vom Standard Zugelassenen und wird vom BSI (Bundesamt für Sicherheit in der Informationstechnik) für nach derzeitigem Stand der Forschung sicher bis zum Jahre 2003 angesehen. Da es sich bei DSS um einen US-Standard handelt, ist seine Verwendung allgemein weit verbreitet; so ist es auch Bestandteil der Standards OpenPGP und S/MIME. DSA-Signaturen kommen zudem bei SSH zum Einsatz; das Verfahren ist auch in vielen kryptographischen Bibliotheken wie z.B. OpenSSL zu finden.

PERFORMANCE

VERSCHLÜSSELUNG – ELGAMAL UND RSA

Beim direkten Vergleich der Performance von RSA und ElGamal fällt auf, daß das Verschlüsseln mit RSA deutlich schneller vonstatten geht. Bei der Entschlüsselung liegt jedoch ElGamal bei der Geschwindigkeit wieder vorne. Da Daten im Normalfall deutlich öfter ent- als verschlüsselt werden, gilt ElGamal als das schnellere Verfahren. Allerdings wird für die Sicherung größerer Datenmengen hybride Verschlüsselung angewandt, d.h. die Daten werden mit einem symmetrischen Verfahren (wie z.B. AES) – welches deutlich schneller ist als die bekannten asymmetrischen – gesichert, und nur der Schlüssel (das Paßwort) hierfür wird mit ElGamal bzw. RSA

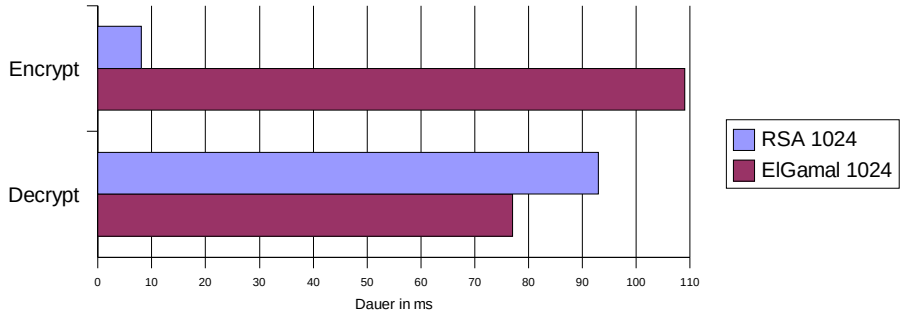


Abbildung 1: Performance RSA/ElGamal auf Sparc II

verschlüsselt. Die Performanceunterschiede sind also in den Anwendungen normalerweise nicht von Belang. In kryptographischer Hardware mit geringer Rechenkapazität (Smartcards) oder in Umgebungen mit hohem Datendurchsatz gilt dies allerdings nicht.

SIGNATUR – RSA UND DSS

Das Signieren von Daten mit DSS ist deutlich schneller als mit RSA; allerdings werden Signaturen in der Praxis öfter überprüft als generiert. Daher gilt RSA als das schnellere Verfahren. Auch hier ist jedoch zu beachten, daß niemals die kompletten Daten, sondern stets nur ihr Hashwert signiert werden; die Geschwindigkeitsdifferenzen haben also auch hier nur geringe Relevanz, solange die Algorithmen nicht auf kryptographischer Hardware oder in Umgebungen mit hohen Datendurchsätzen zum Einsatz kommen.

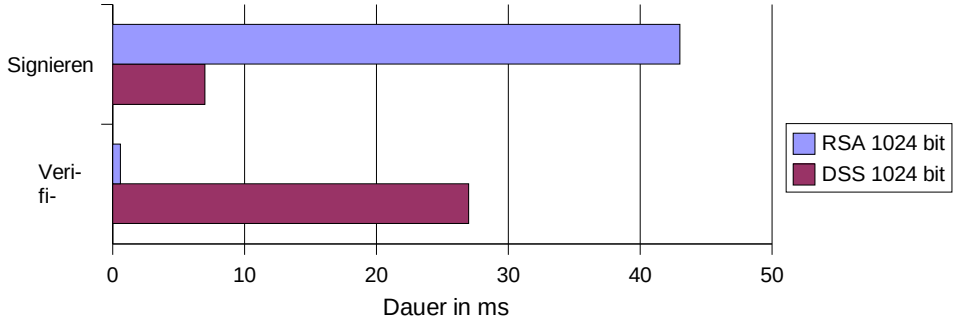


Abbildung 2: Performancevergleich RSA/DSS

PRAXISBEISPIEL: GNUPG

EINSATZ DER VERFAHREN

In der mittlerweile zum Quasi-Standard avancierten Open-Source-Implementierung von PGP bzw. dem OpenPGP-Standard - dem „Gnu Privacy Guard“, kurz GnuPG oder GPG – kommen alle drei vorgestellten Verfahren zum Einsatz. GPG-Schlüssel bestehen aus einem DSS-Schlüssel der Länge 1024 Bit für die Durchführung von digitalen Signaturen und einem ElGamal-Schlüssel für die Verschlüsselung. Der Schlüsselaustausch geht per Diffie-Hellman vonstatten.

Bei GPG kommt hybride Verschlüsselung zum Einsatz; das heißt die Daten werden mit einem der unterstützen symmetrische Algorithmen verschüsselt, und lediglich das Paßwort hierfür wird mit dem asymmetrischen Verfahren gesichert. Derzeit unterstützt GPG die symmetrischen Algorithmen 3DES, CAST5, BLOWFISH, AES, AES192, AES256, und TWOFISH. Als Public-Key-Systeme stehen RSA (Verschlüsselung und Signatur), DSA (Signatur) und ElGamal (Verschlüsselung) zur Verfügung.

„DREIGETEILTE SCHLÜSSEL“

Die Dreiteilung der Schlüssel – d.h. die Verwendung je eines anderen Kryptosystems für Signatur, Verschlüsselung, und Schlüsselaustausch - bringt den Vorteil mit sich, daß der Bruch eines Verfahrens nicht automatisch den Bruch der anderen beiden bedeutet; bei RSA wäre dies beispielsweise der Fall. Die Verwendung von Diffie-Hellman für den Schlüsselaustausch bringt – wie bereits erläutert – zusätzliche Sicherheit, die das RSA-Verfahren alleine nicht bieten könnte.

Alle drei verwendeten Algorithmen basieren jedoch auf dem DLP, weshalb eine effiziente Lösung desselben automatisch die Hinfälligkeit aller drei Verfahren bedeuten würde. Zudem vergrößert ElGamal-Verschlüsselung die Nachrichtenlänge; dies ist allerdings dank hybrider Verschlüsselung ein zu vernachlässigendes Problem.

GESCHWINDIGKEIT

Um ein Gefühl für die Geschwindigkeit der Verschlüsselung auf aktueller Hardware und mit realistischen Datenmengen zu bekommen, seien hier die Egebnisse einer Zeitmessung auf einem PIII-1.1GHz bei der Verschlüsselung von 50MB Zufallsdaten angegeben.

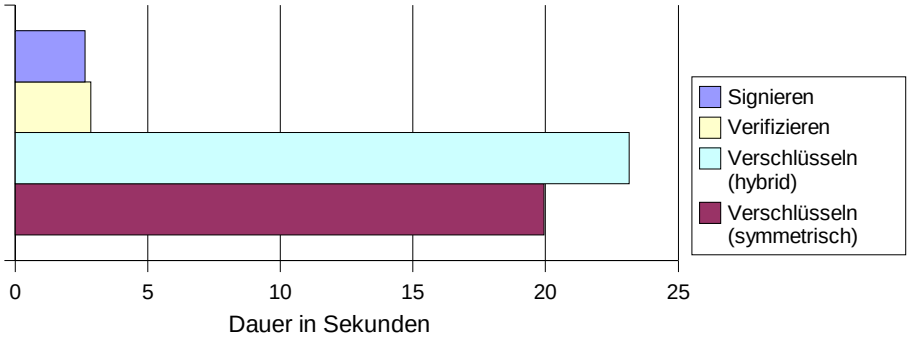


Abbildung 3: Performance Verschlüsselung/Signierung mit GPG auf PIII-1.1GHz

Die Dauer der symmetrischen AES-Verschlüsselung unterscheidet sich also nicht gravierend von der der asymmetrischen, welche aber einen deutlichen Sicherheitszuwachs mit sich bringt. Auch der Geschwindigkeitsunterschied zwischen Signieren und Verifizieren bei DSS schrumpft in der Realität auf ein geringes Maß zusammen, da ja lediglich ein Hashwert signiert bzw. überprüft wird. Ein großer Teil der CPU-Zeit dürfte für Aufgaben verwendet werden, die mit der eigentlichen Verschlüsselung nicht direkt zu zun haben – Speicherallozierung, Ein/Ausgabe, Benutzerinteraktion etc. Ein Teil der Zeitdifferenz zwischen den beiden Verschlüsselungen liegt auch darin begründet, daß bei der Public-Key-Variante erst ein Schlüsselring eingelesen und der richtige Schlüssel gefunden werden muß, was zusätzlich Ressourcen in Anspruch nimmt.

ADRESSEN, QUELLEN, URLS, INFORMATIONEN

Name	*Adresse*	*Hinweis/Beschreibung*
RFC 2539	http://www.faqs.org/rfcs/rfc2539.html	RFC, welche die Einbindung von Diffie-Hellman in DNS-Records beschreibt
Diffie-Hellman in two lines of Perl	http://crypto.stanford.edu/~dabo/abstracts/DDH.html	DH-Implementierung in zwei Zeilen Perl-Code
An Overview of Cryptography	http://www.garykessler.net/library/crypto.html	Ausführliche Übersicht über Kryptographie
PGP DH vs. RSA FAQ	http://www.scramdisk.clara.net/pgpfaq.html	Ausführliche FAQ, welche sich insbesondee mit PGP und RSA bzw. DH beschäftigt.
Public Key Encryption Algorithms	http://www.cs.adfa.oz.au/teaching/studinfo/csc/lectures/less17.html	Beschreibt RSA und ElGamal-Verfahren und gibt Auskunft über sinnvolle Schlüssellängen und derzeitigen Stand der Faktorisierung.
DSS	http://www.itl.nist.gov/fipspubs/fip186.htm	DSS-Standard
Algorithmen	http://www.digital-law.net/knupfer/algorithmen1.htm	Kurze Übersicht über Verschlüsselungs-, Signatur- und Hashalgorithmen mit Links.
PGP Arbeitsweise	http://www.bdpajournal.org/Old-Journals/2001-02/persprivacy.htm	Wie arbeitet PGP?
GnuPG Homepage	http://www.gnupg.org/	Homepage des GnuPG-Projektes mit FAQs, Anleitungen, Quellcode, usw.
RFC 2440	http://www.gnupg.org/rfc2440.html	OpenPGP-Standard-RFC

Name	*Adresse*	*Hinweis/Beschreibung*
S/MIME and Open-PGP	http://www.imc.org/smime-pgpmime.html	Vergleich/Übersicht OpenPGP und S/MIME
OpenSSL	http://www.openssl.org	Open-Source-Bibliothek mit kryptographischen Verfahren
Dsniff	http://www.monkey.org/~dugsong/dsniff/	Toolkit für die automatische Durchführung z.B. von Middleperson-Angriffen auf SSL und SSH
RSA FAQ	http://www.rsasecurity.com/rsalabs/faq/index.html	RSA Kryptographie FAQ.