PGP & Internetwahlen

Philipp Hügelmeyer

14. Mai 2001

Inhaltsverzeichnis

1 Einleitung

In diesem Dokument werden zwei praktische Anwendungen von Verschlüsselungsverfahren vorgestellt und genauer beleuchtet. Konkret wird Verschlüsselung von E-Mails und Wählen über ein Netzwerk vorgestellt.

2 Verschlüsselungsverfahren und andere Algorithmen

Um Verschlüsselung von E-Mails genauer betrachten zu können, braucht man ein paar Algorithmen als Grundlage.

2.1 DES - Data Encyption Standard

Der Data Encryption Standard wurde 1974 von IBM entwickelt und 1977 zum Standard ernannt. Grob gesagt ist DES ein Blockchiffre der mit symmetrischer Verschlüsselung Blöcke zu je 64 Bit mit einem 56 Bit Schlüssel verarbeitet. Jeder 64 Bit Block wird in 16 Stufen verschlüsselt und in umgekehrter Reihenfolge wieder entschlüsselt.

Ein Vorteil von DES liegt in der Performence. Hier wird ein recht guter Datendurchsatz erreicht. Weiterhin ist dieser Algorithmus sehr alt und standardisiert. Die besten und schnellsten Ergebnisse werden allerdings erst erzielt, wenn er direkt in der Hardware implementiert worden ist.

Kommen wir aber jetzt zum eigentlichen Aufbau:

Vor dem eigentlichen Verschlüsseln wird der Klartext als erstes in 64 Bit Blöcke unterteilt. Jeder dieser Blöcke wird einer Eingangspermutation unterzogen. Permutation heißt, daß die Bits um bestimmt Stellen, also nach einem bestimmten Schlüssel, verschoben werden. Ihre Anordnung wird also verändert. Danach wird der 64 Bit Block in einen rechten und einen linken Block geteilt. Im rechten befinden sich alle ungerade und im linken alle geraden Bit-Positionen. Zunächst wird nur mit dem rechten Block gearbeitet. Dieser wird zunächst in einer Expansion von 32 auf 48 Bit erweitert. Dazu wird hinter jedem 4. Bit 2 Bit eingeschoben die aus dem letzten Bit des Vorblocks und aus dem ersten Bit des Nachblocks kopiert werden.

Als Nächstes muß erst der richtige Schlüssel generiert werden. Dazu wird als erstes der Schlüssel permutiert. Anschließend wird der Schlüssel in 2 Blöcke geteilt. Als nächstes wird der eigentliche Rundenschlüssel in einer Schlüsselauswahl aus den beiden Schlüsselblöcken berechnet. Der Ausgewählte Schlüssel besteht aus 48 Bit. Jetzt wird der 48Bit Datenblock mit

dem 48Bit Schlüsselblock per XOR verknüpft. Nachdem das geschehen ist kommt der Datenblock in die 8 S-Boxen. Dazu werden werden die 48 Bit in 8 x 6Bit Folgen geteilt. Das erste und das letzte der 6 Bits dienen zur Auswahl der Reihe (0-3). die restlichen 4 Bit zur Auswahl der Spalte. Die S-Boxen kann man sich wie eine Tabelle mit 4 Zeilen und 16 Spalten vorstellen, gefüllt mit Hexadezimalen Werten. Hat man die entsprechende Zelle ausgewählt erhält man einen hex-Wert. Dieser umgerechnet in einen Binärwert ergibt wieder 4 Bit. Das mal die 8 S-Boxen erhält man als Ausgangsdatenblock wieder 32 Bit. Dieser Block wird wieder einer Permutation unterzogen. Danach wird dieser per XOR mit dem am Anfang linken Datenblock verknüpft. Das Ergebnis dieser Verknüpfung wird in den rechten Block kopiert, nachdem dessen ursprünglicher Inhalt in den linken Block verschoben wurde. Das Ganze wird dann 16 mal wiederholt. Nach der 16ten Runde wird der rechte und der linke Datenblock durch eine Ausgangspermutation wieder zusammengeführt und ausgegeben. Die Ausgangspermutation ist invers zur Eingangspermutation. Das trägt zwar nichts zur Sicherheit bei, ist aber notwendig, das dies der Standard verlangt.

Eins ist allerdings noch wichtig: damit in jeder einzelnen der 16 Runden ein anderer Schlüssel verwendet wird, wird der Schlüssel rotiert. Dazu werden die Bits der 2 Schlüsselböcke jeweils um ein bis 2 Bit nach links verschoben. Dies geschieht so, daß nach der 16 Runde die Ausgangsposition wieder hergestellt ist. Hiermit ist gewährleistet, daß ohne weitere Schritte gleich der nächste 64 Bit Datenblock geladen und weiter verschlüsselt werden kann.

Die Entschlüsselung läuft analog wie die Verschlüsselung ab, nur daß hier die Schlüsselrotation nach rechts verläuft.

2.2 IDEA Verschlüsselung

Der International Data Encryption Algorithm (IDEA) stammt aus dem Jahre 1990 und ist ebenfalls, wie der DES, ein 64 Bit blockorientierter Chiffre. Er wurde an der Schweizer Bundesanstalt für Technologie entwickelt. Es wird ein Schlüssel von 128 Bit verwendet, wodurch sich ein größerer Schlüsselraum von 2128 = 3 x 1038 verschiedenen Schlüssel ergibt.

Ein 64 Bit Ausgangstextblock wird in 4 Teilblöcke (Xi) zu jeweils 16 Bit aufgeteilt. Diese Teilblöcke durchlaufen 8 Iterationen in denen 3 einfache Grundoperationen ausgeführt werden. Diese Grundoperationen sind, in Kombination mit den 52 Teilschlüsseln (Z1-Z52), XOR, Additions-und Subtraktionsoperationen. Hier finden keine Bit-Vertauschungen wie in den S-Boxen von DES statt. Nach einer Ausgangsumwandlung erhält man den

chiffrierten Text. Die Entschlüsselung verläuft identisch, jedoch mit umgekehrter Reihenfolge der Teilschlüssel.

Die Vorteile gegenüber dem DES sind die größere Schlüssellänge, welche eine höhere Sicherheit und eine kürzere Rechenzeit gewährleistet, da weniger Iterationen durchgeführt werden. Zudem gibt es keine Permutationsoperationen.

2.3 Diffie - Hellmann

Der Diffie Hellmann Algorithmus ist der erste Public-Key Algorithmus der für die Schlüsselvereinbarung über einen unsicheren Kanal verwendet wird. Er wurde 1976 entwickelt. Bis dahin sind ausschließlich die symmetrischen Verfahren verwendet worden. Somit stellte das Verfahren eine Revolution in der Kryptographie dar. Da Diffie-Hellmann keinerlei Authentifizierung der Kommunikationspartner beinhaltet, ist dieser Algorithmus anfällig für den M̈an-in-the-middle Angriff: Das DH Verfahren ist auch für mehr als 2 Partner geeignet, das heißt eine Gruppe kann sich auf einen gemeinsamen Schlüssel einigen.

Dieffie Hellmann beruht auf der Berechnung eines gemeinsamen Geheimnisses, das für einen nicht beteiligten nicht nachvollziehbar´ ist, es sei denn er kann den diskreten Logarithmus in einem finiten Feld berechnen, was äußerst schwierig ist.

Das Verfahren zur Schlüsselerzeugung läuft folgendermaßen ab:

Beide Kommunikationsteilnehmer (A und B) überlegen sich eine Zahl s und eine große Primzahl p, für die zusätzlichgelten soll:

(p-1)/2 ist ebenfalls eine Primzahl.

Weiter überlegen beide für sich zwei große, ganze Zahlen aA und aB, die den geheimen Teil des Schlüssels darstellen. Für die Schlüsselübergabe schickt A an B bA und B an A bB. Mit diesen Zahlen können beide ein gemeinsames k berechnen. Durch Einsetzen in die Gleichungen ergibt sich für k = kA = kB. Dies ist der gemeinsame geheime Schlüssel. Selbst wenn nun der Angreifer s, p sowie bA und bB kennt, kann er daraus wegen der aufwendigen Berechnung des diskreten Logarithmus innerhalb angemessener Zeit nicht auf k schließen, da er aA und aB nicht kennt.

2.4 Der RSA Algorithmus

RSA wurde von Rivest, Shamir und Adleman 1977 als ein public key Verfahren entwickelt. Dieser Algorithmus ermöglicht sowohl Verschlüsselungen als auch digitale Signaturen. Die Sicherheit dieses Algorithmus beruht auf der

Schwierigkeit der Faktorisierung großer Zahlen. Das ist besonders schwierig wenn diese große Zahl das Produkt zweier ungefähr gleich großer Primzahlen ist. Große Zahlen sind in diesem Fall mehrere hundert dezimal Stellen lang.

Um einen Schlüssel zu erzeugen wählt man 2 zufällige, große Primzahlen p und q Nun berechnet man das Produkt $n = pxq$. Als nächstes wählt man den öffentlichen Exponenten e, sodaß der größte gemeinsame Teiler von e und $(p-1)x(q-1)$ gleich 1 ist. Um eine schnelle Verschlüsselung zu gewährleisten wird e meist niedrig gewählt. der geheime Exponent berechnet sich dann :

$d = e - 1(mod[(p-1)(q-1)])$

Jetzt ist der öffentliche Schlüssel n und e, der geheime Schlüssel ist d.

Die Verschlüsselung der Nachricht M erfolgt dann durch:

$C = M^e mod n$

Die Entschlüsselung wird dann durch:

$M = C^d mod n$ berechnet

Die Erstellung einer Signatur verläuft genau umgekehrt:

$S = H^d mod n$ wobei H der Hashwert der Nachricht ist. Die Verifikation erfolgt durch überprüfen ob $H = S^e mod n$ gleich dem Hashwert ist.

Übliche Längen für den Modulus sind 512, 1024 und 2048 Bit, wobei 512 Bit nicht wirklich empfohlen werden kann.

2.5 One Way Hashfunktionen

Meistens ist es nicht praktikabel ein gesamtes Dokument zum Zwecke der Signatur zu verschlüsseln. An diesem Punkt können Hashfunktionen eingesetzt werden. Die Idee ist es, nicht das gesamte Dokument, sondern eine volumenmäßig kleinere Abbildung dieses Dokuments zu verschlüsseln bzw. zu signieren.

Bei der Verwendung in der TLS Spezifikation werden die Hashfunktionen zum Zwecke der Signatur eingesetzt. Dabei sind 2 wichtige Voraussetzungen zu erfüllen:

- es soll sehr unwahrscheinlich sein, daß zwei Datenmengen zufälligdie gleiche Prüfsumme ergeben. Dies wird umso besser erfüllt, je größer der Umfang der Prüfsumme ist. 512 Bit Prüfsummen sind in der Regel gut ausreichend.

- Es soll sehr schwierig sein, zu einer vorgegebenen Prüfsumme einen passenden Text zu finden , selbst dann wenn der Algorithmus bekannt ist. Hier kann natürlich die Notwendigkeit abgeleitet werden, daß eine

sehr kleine Änderung im Dokument eine komplett andere Prüfsumme ergibt.

Der Ablauf der Signatur und Verifiaktion ist einfach: Server A bildet den Hashwert des zu sendenden Dokumentes und verschlüsselt diesen mit seinem privat key. Dieser wird jetzt zusammen mit dem Dokument zum Server B gesendet. Dieser entschlüsselt den Hashwert mit dem public Key des Servers A und vergleicht das Ergebnis mit dem zuvor selbst gebildeten Dokumentenhashwert. Ergibt dies eine Übereinstimmung ist das Dokument echt, d.h es stammt vom Server A da nur dieser seinen private key besitzt womit der Hashwert verschlüsslt wurde.

Das Verfahren ist relativ sicher, trotzdem gibt es mögliche Sicherheitsrisiken:

- Herstellen eines zweiten Dokumentes welches den gleichen Hashwert wie das Original besitzt. Dies ist aber sehr unwahrscheinlich, da es mehrere 100.000 Jahre dauern würde bis ein entsprechend zweites Dokument gefunden wäre.
- Die sogenannte Birthday Attack. Hier werden 2 Dokumente erzeugt die den gleichen Hashwert haben. Diese Methode ist etwas kurios, denn hier müßte man jemanden dazu bewegen ein Dokument A zu unterschreiben um auch einen gültige unterschrift zu B zu haben

Um Hashfunktionen sicher gegenüber der Geburtstagsattacke zu machen, sollte man mit 160 Bit Hashwerten arbeiten.

Als Beispiel Algorithmen wäre zu nennen:

- MD5 , arbeite mit 128 Bit
- SHA , arbeitet mit 160 Bit.

Der MD5-Fingerprint wurde von Ron Rivest am MIT entwickelt und 1991 veröffentlicht. Die Buchstaben MD stehen für Message Digest. Dieser Algorithmus ist eingehend getestet worden und es ist bislang keine Kritik daran aufgekommen. Diese Funktion berechnet aus einer beliebig langen Nachricht ein 128-Bit-Hashwert. Mittlerweile erscheint der Wert vielen als zu kurz, da es zu leicht ist, eine Nachricht zu finden, die zu einem bestimmten Hashwert paßt. Somit könnte man also die Unterschrift einer bestimmten Person unter ein anderes Dokument setzen. Momentan wird MD5 noch von PGP unterstützt.

Der Secure Hash Algorithmünd ist im Gegensatz zu MD5 160 Bit lang. Auch an diesem Algorithmus sind bislang keine Zweifel aufgekommen. SHA ist ab Version 5.x in PGP integriert. Man kann es dort anstatt von MD5 optional nutzen. Der Algorithmus ist von der NSA entwickelt worden.

2.6 Radix-64-Format

Da in vielen Email-Systemen nur der Versand von ASCII-Text möglich ist, können Binärdaten nicht versandt werden. Deswegen kann PGP die verschlüsselten Daten im Radix- 64-Format darstellen. Dieses Format verwendet ausschließlich druckbare ASCII-Zeichen. Radix-64 ist also ein Art Transport-Verpackung; die Schutz vor Verstümmelung auf dem Transportweg bietet. Die Datei wird zwar um ca. 33% vergrößert, was aber wegen der vorherigen Komprimierung vernachlässigbar ist.

3 PGP

3.1 Was ist PGP ?

PGP ist ein benutzerfreundliches und leistungsfähiges Ver- und Entschlüsselungsprogramm für beliebige Daten, das den Austausch von elektronischen Nachrichten ohne Verzicht auf Privatsphäre gestattet. Richtig eingesetzt, schließt es die Möglichkeit, per Email versandte Daten ohne Berechtigung zu entschlüsseln, weitgehend aus.

PGP steht für Pretty Good Privacy(zu deutsch etwa ziemlich gute Privatsphäre), wobei Pretty Good auch Kurzform von Phil's Pretty Good Software ist. Es wurde ursprünglich von Philip R. Zimmermann in den USA entwickelt und basiert auf einem sogenannten Hybridverfahren, welches asymmetrische und symmetrische Verschlüsselungsverfahren koppelt.

Seit Juni 1991 ist PGP für den privaten Gebrauch freeware in den USA und hat sich seitdem zu einer Art de-facto-Standard für Email entwickelt.

Das Programm versendet die verschlüsselten Daten nicht selbst, ist jedoch mittlerweile in sehr viele Email-Programme komfortabel einbindbar. Zudem ist PGP schnell und auch portabel, so z.B. auf Commodore Amiga, Atari ST, Mac, OS/2, MS-DOS und Windows (auch NT), Unix, VAX/VMS, Acorn RISC OS (Archimedes), CompuServe WinCIM, CSNav etc.

3.2 Geschichte von PGP

Philip R. Zimmermann entwickelte PGP Anfang der Neuniger Jahre, nachdem im US-Seant eine Beschlussvorlage einging, daß in jede Verschlüsselungssoftware oder Hardware eine Hintertür eingebaut werden sollte, so das der Staat mitlesen kann. . Seit 1991 sind verschiedene Versionen verfügbar. Die ersten Versionen waren illegal (in den USA) benutzbar, da sie US-Patentrechte für Public-Key-Cryptography verletzten. Darüber hinaus gelten für alle US-Versionen US-Exporteinschränkungen; die Programme dürfen nicht ausgeführt werden.

Erst seit 1994 dürfen US-Bürger PGP für den privaten Gebrauch frei benutzen, während bei einem kommerziellen Einsatz eine Lizenz von der Firma ViaCrypt erworben werden muß. Das gilt insbesondere auch für die aktuelle Version 2.6.

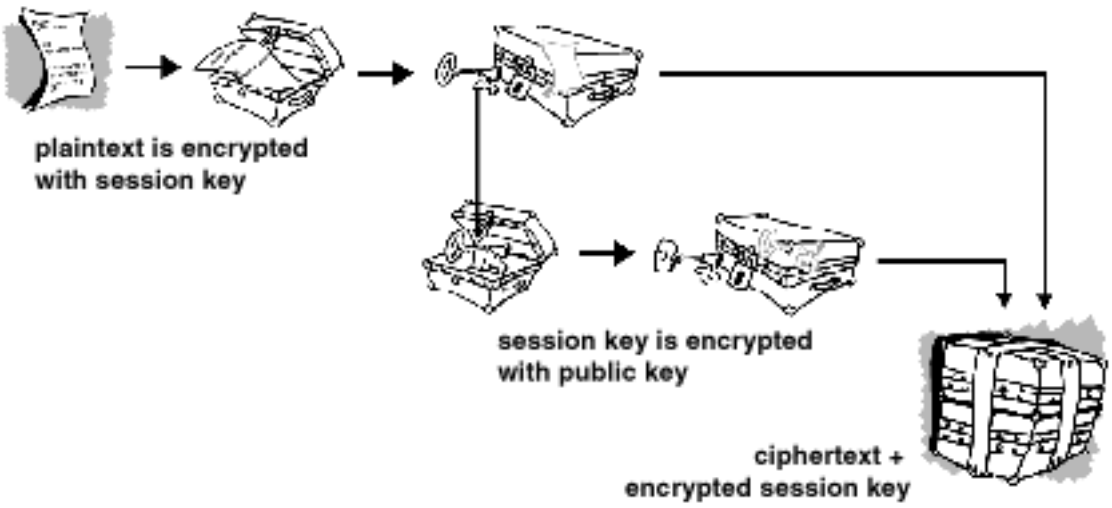
Außerhalb der USA soll eine in Europa entwickelte Version 2.6i verwendet werden, die nicht den Lizenzproblemen und Exportrichtlinien der USA beim RSA-Verfahren unterliegt. Damit diese Version in Europa entwickelt werden konnte, hat man den Code gedruckt und als Buch verkauft. Der Buchstabe 'i' steht für *international*. Diese Version darf aber nicht in den USA benutzt werden. Zum Glück sind beide Versionen kompatibel. Ältere Versionen von PGP sind jedoch nicht mehr kompatibel zur Version 2.6.

Es gibt aber gegebenenfalls noch eine weitere Hürde: Die USA und beispielsweise auch Deutschland möchten aus Sicherheitserwägungen dem Normalbürger das Verwenden der Verschlüsselungstechnik untersagen. Es gibt bereits Staaten, die das getan haben (Frankreich, Iran, Irak).

3.3 Verschlüsseln

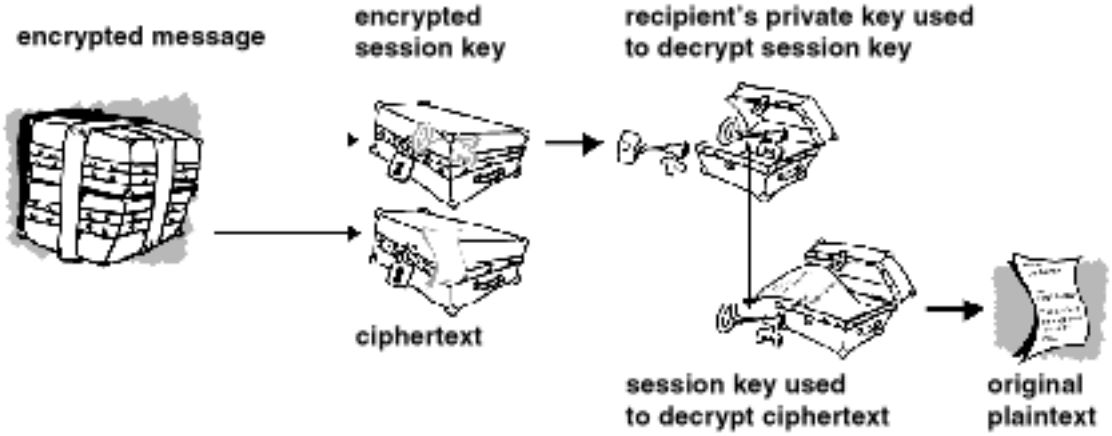
Bei PGP handelt sich um ein *hybrides Cryptosystem*, es werden also symmetrische und asymmetrische Verschlüsselungsverfahren kombiniert. PGP geht beim Versenden einer Nachricht folgendermassen vor:

1. Der zu versendende Text wird komprimiert.
2. PGP generiert einen Session Key, der einmalig verwendet wird
3. Die Nachricht wird mit dem Session-Key verschlüsselt
4. Der Session-Key wird mit dem öffentlichen Schlüssels des Empfängers verschlüsselt.
5. Die verschlüsselte Nachricht und der Session-Key werden zusammen verschickt.



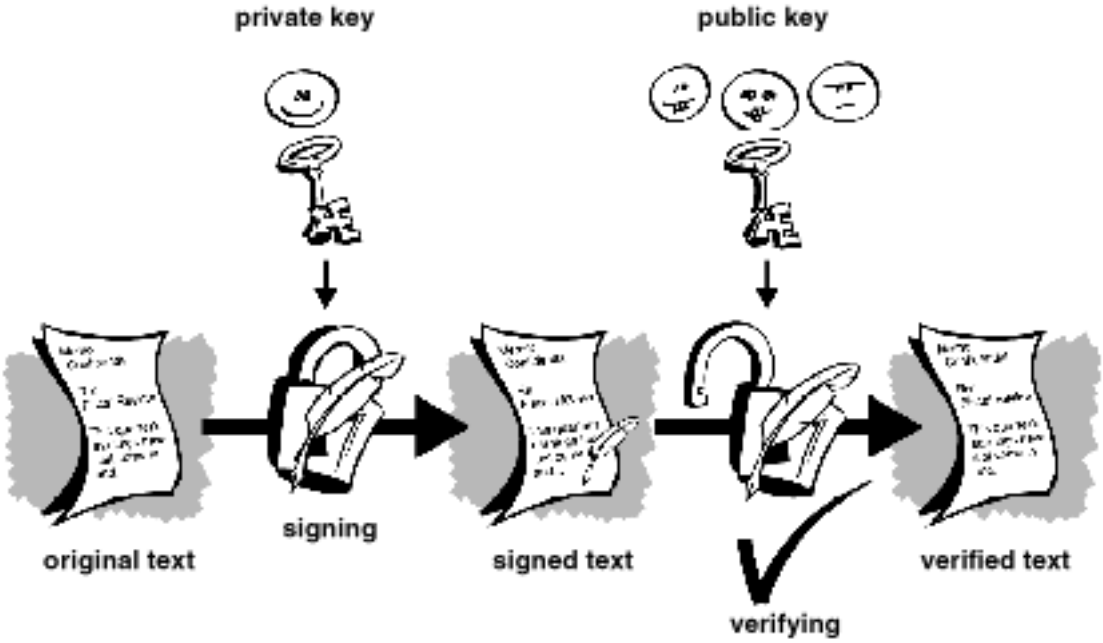
Das Entschlüsseln der erhaltenen Nachricht funktioniert genau umgekehrt:

1. Der Empfänger der Nachricht entschlüsselt mit seinem privaten Schlüssel den Session Key
2. Mit dem Session Key kann die Nachricht entschlüsselt werden.
3. Die Nachricht wird entpackt und kann vom Empfänger gelesen werden.



3.4 Signieren

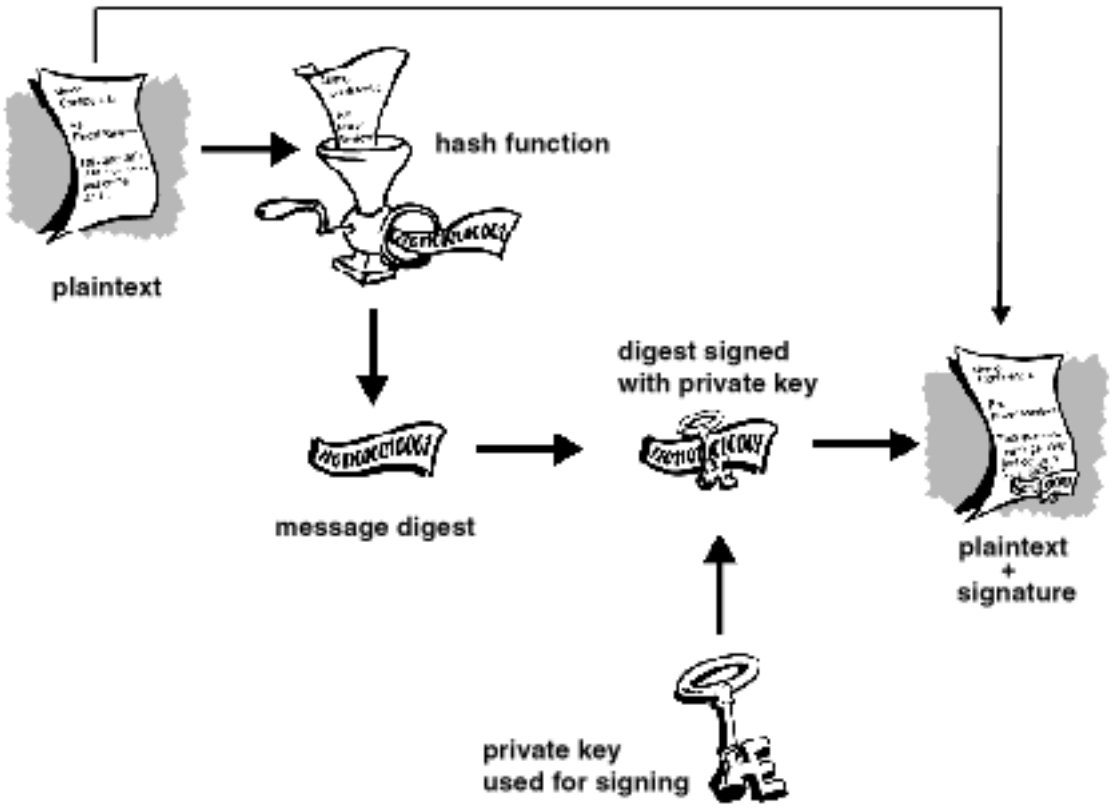
Mit der eben gezeigten Methode kann ich nun eine Nachricht verschlüsseln. Es ist aber noch nicht gewährleistet, das die Nachricht wirklich von dem angegebenen Empfänger kommt.



Oft ist es gar nicht so wichtig, das eine Nachricht unbekannt ist, ich will nur wissen, ob eine Nachricht unverändert ist und der Absender wirklich der angegebene ist. Dazu wird folgendermaßen vorgegangen:

1. Über den zu signierenden Text wird ein Hashwert errechnet.
2. Der Hashwert wird mit dem privaten Schlüssel des Versenders verschlüsselt.
3. Der Text wird verschickt.
4. Der Empfänger entschlüsselt den Hash-Wert mit dem öffentlichen Schlüssel des Versenders
5. Der Empfänger berechnet den Hashwert des Textes und vergleicht ihn mit dem mitverschickten.

Stimmen die Hashwerte überein, ist der Text unverändert und der Absender korrekt.



3.5 Zertifikate

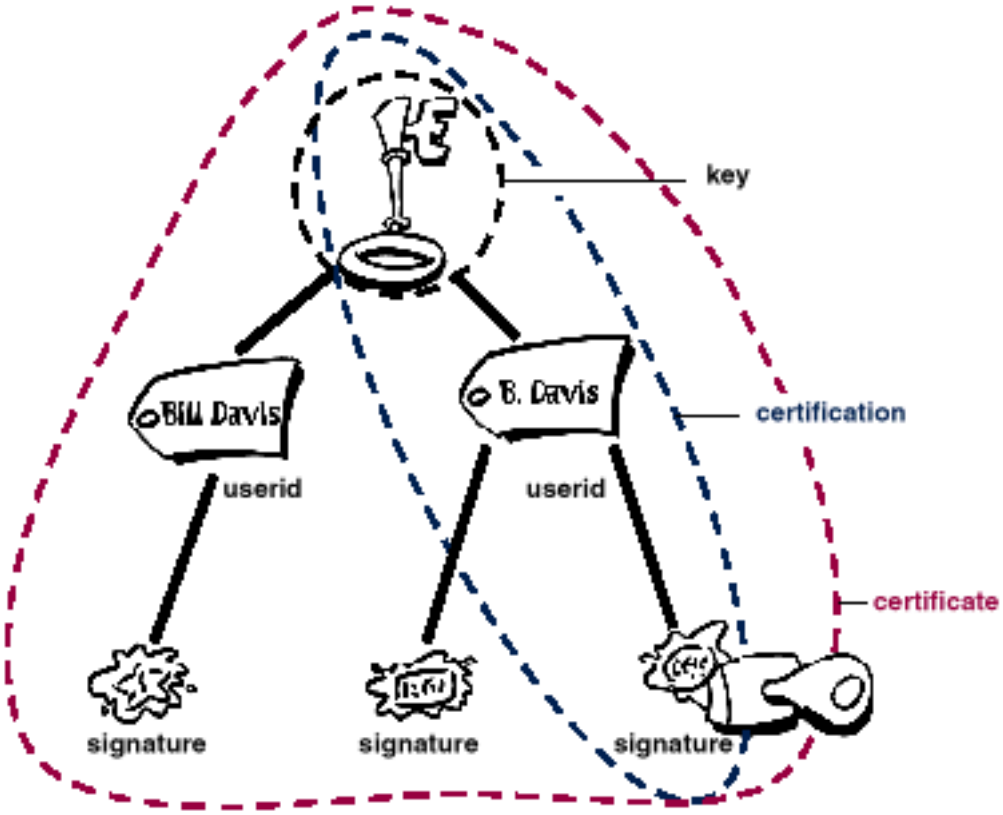
Jeder kann einen PGP-Schlüssel selber generieren. Wer nun wissen will, ob dieser Schlüssel auch wirklich der angegebenen Person gehört, muß dies überprüfen. Das ist aber schwierig, wenn man diese Person gar nicht kennt.

PGP ermöglicht neben dem X.509, das zentral von einer CA (*Certificate Agency*)verwaltet wird, auch noch ein eigenes Zertifikatformat. PGP Zertifikate bestehen aus folgende Features:

- Der PGP-Versionsnummer
- Der Public-Key des Zertifikatinhabers
- Informationen über den Benutzer (z.B. Namen, ID, etc.)
- Die digitale Signatur des Eigentümers
- Die Gültigkeit des Zertifikats

- Der bevorzugte symmetrische Verschlüsselungsalgorithmus

Zertifikate können nicht nur die eigene Signatur enthalten, sondern auch die Signatur von anderen Benutzer, die bestätigen, daß sie diesem Schlüssel



vertrauen.

3.6 Web of trust

Bei jedem Public Key Verfahren ist die Authentizität (Echtheit) der Public Keys ein Problem. Ein kleines Beispiel: Bob erhalte von Carol eine elektronisch unterschriebene Mail. Um die Unterschrift zu prüfen, braucht Bob Carols public key.

Eine Mail an pgp-public-keys@keys.ch.pgp.net mit dem Subject GET carol@error-42.de liefert ihm ihre public key. Aber ist das wirklich ihr public key? Mallory könnte einen Key mit dem Namen Carol erzeugt und auf dem Keyserver deponiert haben.

Bob muss nun die Echtheit dieses public keys überprüfen. Dazu hat Bob verschiedene Möglichkeiten, z.B:

1. Bob sucht im offiziellen Telefonbuch nach Carols Telefonnummer, ruft sie an und läßt sich von ihr ihre Schlüsseldaten geben.
2. Bob geht bei Carol vorbei, läßt sich ihren Ausweis zeigen und die Schlüsseldaten geben.
3. Bob schaut, ob er jemanden kennt, der ihre Identität elektronisch bestätigt.

Die ersten beiden Möglichkeiten sind aufwendig und zum Teil undurchführbar (wenn der Key z.B. einer Person weit weg gehört). Die Dritte ist dagegen relativ einfach.

Jeder Schlüssel ist in genau einer Vertrauenstufe. Hat ein Schlüssel einen anderen Schlüssel signiert, wird anhand der Vertrauensstufe bestimmt, ob der signierte Schlüssel als gültig anerkannt wird. Es gibt drei Stufen des Vertrauens, die man Schlüsseln von anderen Leuten entgegenbringen kann:

- *Vollständiges* Vertrauen
- *Teilweises* Vertrauen
- Mißtrauen

Um das ganze etwas verwirrender zu machen, gibt es auch noch 3 Stufen der Gültigkeit:

- gültig
- Teilweise gültig
- Ungültig

Als Beispiel: Bobs Schlüsselring enthält Alices Schlüssel. Bob hat Alices Schlüssel für gültig befunden und dies dadurch bestätigt, daß er Alices Schlüssel unterschreibt.Bob weiß, daß Alice sehr gründlich vorgeht, wenn sie die Schlüssel anderer Leute unterschreibt. Deswegen stattet Bob Alices Schlüssel mit vollständigem Vertrauen aus. Wenn Alice jetzt einen anderen Schlüssel signiert, ist dieser in Bobs Schlüsselring gültig.

PGP verlangt einen komplett oder zwei teilweise vertrauenswürdige Schlüssel, um einen Schlüssel als gültig zu akzeptieren.

Aus diesen verschiedenen Schlüsseln baut sich nun ein Netzwerk auf das sogenannte *Web-of-Trust*. Da es in diesem Netz nicht zwingend Instanzen geben muß, die eine große Menge von Schlüssel unterschreiben, kann der Weg von einem Schlüssel zum nächsten so weit sein, das man keine Möglichkeit findet, einen Schlüssel zu validieren.

4 Internetwahlen

In diesem Abschnitt soll, beschrieben werden, wie sichere Abstimmungen über das Internet durchgeführt werden können. Zuerst müssen die Bedingungen aufgeschrieben werden, die für so eine Wahl gelten sollen, dann werden Verfahren darauf überprüft, ob sie alle Anforderungen erfüllen.

4.1 Sichere Wahlen

Bei Wahlen über das Internet sollen die gleichen Dinge gewährleistet werden wie bei einer normalen Wahl:

1. Nur autorisiere Wähler können wählen
2. Niemand kann mehr als einmahl wählen
3. Niemand kann sagen, wofür ein anderer gestimmt hat
4. Niemand kann die Wahl eines anderen duplizieren
5. Niemand kann unentdeckt die Wahl eines anderen ändern
6. Jeder Wähler kann feststellen , daß seine Stimme am Ende gezählt wurde.
7. Jeder kann sehen, wer gewählt hat und wer nicht. (Dieses braucht man nicht bei jeder Wahl).

Grundvoraussetzung ist ein (Sender-) anonymer Kommunikationskanal. Durch diesen kann die Anonymität der Wähler unter kryptographischen Annahmen sicher gestellt werden. Ein (Sender-) anonymer Kommunikationskanal 'verschleiert', welcher Sender mit dem Empfänger kommuniziert und gewährleistet daher Anonymität des Sender gegenüber dem Empfänger. Eine Zurückverfolgung der Nachricht durch den Empfänger zum Absender wird durch diesen speziellen Kanal verhindert.

4.2 Blinding Verfahren

Im Allgemeinen ist es eine gute Eigenschaft beim Signieren, daß der Unterschreibende weiß, was er unterschreibt. Dies ist aber nicht immer gewollt, bei einer Wahl zum Beispiel will Bob nicht, das jemand weiß, wofür er stimmt.

4.3 Komplettes Blinding

Am einfachsten ist natürlich ein komplettes Blindingverfahren:

1. Alice nimmt ihr zu verschlüsselndes Dokument und multipliziert dazu einen zu fälligen Wert. Dieser Wert heißt *Blinding Faktor*
2. Alice sendet das verschlüsselte Dokument an Bob.
3. Bob signiert das geblindete Dokument.
4. Alice teilt durch den Blinding Faktor und erhält das Original-Dokument, das vom Bob signiert ist.

In diesem Verfahren weiß Bob nicht, was in dem Dokument steht, er sagt nur, das er es zu einem bestimmten Zeitpunkt zur Kenntnis genommen hat. Bei einem guten Blindungfaktor hat Bob keine Ahnung was er unterschreibt, jedoch kann ihm,m da er gültig signiert, jederzeit nachgewiesen werden, daß er das Dokument unterschrieben hat.

In vielen Szenerien ist dieses Verfahren nicht besonders sinnvoll.

4.4 Blinde Signaturen

Bob will Alice eine Nachricht schicken, die ungefähr bekannt seien darf. Nehmen wir an Bob sei ein Agent und will diplomatschen Schutz unter seinem Decknamen, aber nicht das Alice (Agency's Large Intelligent Computer Engine) diesen kennt. dann schickt er ihr verschieden Dokumente, die bis auf den Decknamen identisch sind nach folgendem Protokoll:

1. Bob bereitet n Dokumente vor, jedes mit einem anderen Decknamen, die besagen, daß Bob diplomatische Immunität genießt.
2. Bob verschlüsselt jedes dieser Dokumente mit einem anderen Blinding Faktor.
3. Bob schickt die n verschlüsselten Dokumente an Alice.
4. Alice wählt $n-1$ Dokumente zufällig aus und fragt Bob nach den Blindingfaktoren für diese Dokumente.
5. Bob schickt Alice die verlangten Faktoren.
6. Alice öffnet damit $n-1$ Dokumente und stellt sicher, daß alle korrekt sind.

7. Alice signiert das übriggebliebene Dokument und schickt dieses zurück zu Bob.

8. Bob entfernt den Blinding Faktor von diesem Dokument und erfährt, daß sein neuer Deckname "RMS" ist.

Die einzige Möglichkeit zum mogeln wäre für Bob, daß er für jedes Dokument zwei Blindingfaktoren findet, die es einmal in das von Alice erwartete und ein anderes Mal in ein Dokument verwandelt, was Alice nicht signieren würde. Die Chancen solche zwei Dokumente und Bildingfaktoern zu finden ist sehr klein und so wohl nur theoretisch gegeben.

4.5 Einfaches Wahlprotokoll I

1. W verschlüsselt seinen Stimmzettel mit dem Public Key von CTF

2. W sendet den Stimmzettel zu CTF

3. CTF entschlüsselt mit seinem Private Key den Stimmzettel, zählt alle eingelangten Stimmen und veröffentlicht das Ergebnis

Probleme:

- Verletzung von Punkt 1, 2, 4, 6

- Punkt 5 ist für böswilligen W nicht sinnvoll, da ohnedies jede Person beliebig oft wählen kann

4.6 Einfaches Wahlprotokoll II

1. W unterzeichnet Stimmzettel mit seinem privaten Schlüssel

2. W verschlüsselt Stimmzettel mit Public Key [3] von CTF

3. W sendet Stimmzettel zu CTF

4. CTF entschlüsselt mit seinem Private Key [3] den Stimmzettel und vergleicht die Unterschrift mit Hilfe von Ws Public Key [3]

5. CTF zählt und veröffentlicht Ergebnis(se)

Bemerkung:

- Verletzung von Punkt 3, 4, 5, 6

- CTF kann Verbindung zwischen Stimmzettel und der Identität von W herstellen
- Nur autorisierte Personen können wählen
- Niemand kann mehr als zweimal wählen.

4.7 Wahl mit Blinder Signatur

Wahlverfahren:

1. Generierung von 10 NE
2. NE:..... Beinhaltet alle möglichen gültigen Stimmmöglichkeiten + IN
3. Blinden aller NE mit verschiedenen Blinding Faktoren
4. Verschlüsseln der NE mit Public Key von CTF
5. Senden aller NE zu CTF
6. Auswahl von 9 NE von CTF (Cut-and-choose)
7. Unterzeichner fordert die zugehörigen Blindingfaktoren von W an
8. Unterzeichner öffnet und überprüft die 9 NE auf Korrektheit
9. CTF unterzeichnet die zehnte NE und sendet dieses zu W zurück
10. W wählt Stimmzettel aus
11. W verschlüsselt Stimmzettel mit Public Key von CTF
12. W sendet Stimmzettel zu CTF
13. CTF überprüft Signatur, zählt und veröffentlicht Ergebnis(se) mit IN

Bemerkung:

- Böswilliger W kann das System nicht betrügen, da Blind Signature für Eindeutigkeit der Wahl sorgt
- Durch Verbindungsaufnahme von W mit CTF (wird in Datenbank vermerkt) ist zweimaliges Wählen ausgeschlossen

- Erzeugung eigener Stimmzettel nicht möglich, da CTFs Private Key [3] nicht zugänglich
- CTF kann große Anzahl von gültigen Stimmen generieren = Beeinflussung des Gesamtergebnisses

4.8 Wahl mit zwei zentralen Einrichtungen

Wie der Name bereits verrät, werden hier zwei Institutionen benötigt. Diese bezeichnen sich als:

CLA (Central Legitimation Agency)

Bedeutet soviel wie Zentrale Legitimations-Agentur und dient dazu, einer wahlberechtigten Person eine Gültigkeitsnummer (VN) zuzuordnen. Dazu fordern die Wähler mit ihren Namen bei der CLA eine VN an. Die CLA registriert in einer Datenbank die Namen und zugeordneten Nummern. Jedem Wähler wird seine zugewiesene VN bekanntgegeben, anhand welcher er sich nun beim CTF anmelden kann, sobald die CLA alle VNs dem CTF mitgeteilt hat (die CTF erhält nur die VNs, nicht jedoch die zugehörigen Namen)

CTF (Central Tabulating Facility)

Die Zentrale Zähl-Einrichtung prüft anhand eines Vergleiches mit der Liste der VNs (erhalten von der CLA), ob die Stimmabgabe eines Wählers gültig ist. Jeder Wähler muß dazu seine VN mit der Stimmabgabe mitliefern. Dadurch wird gewährleistet, daß ein Wähler nur einmal wählen kann bzw. ein nicht wahlberechtigter Teilnehmer von der Zählung ausgeschlossen wird. Zusätzlich wird noch eine IN an das CTF von jedem Wähler mit der Stimmabgabe gesendet, wodurch eine Überprüfung der richtigen Stimmzählung durch den Wähler gegeben ist. Graphische Veranschaulichung

Wahlverfahren:

1. W fordert Validation Number (VN) bei CLA an
2. CLA sendet (zufällige) VN zurück
3. CLA schickt VNs an CTF
4. W generiert (zufällige) Identification Number IN
5. W sendet IN + VN + S (verschlüsselt mit CTFs Public Key [3]) an CTF

6. CTF entschlüsselt mit seinem Private Key [3] den Stimmzettel und prüft auf vorhandene VN

7. Veröffentlichung von IN + S

Bemerkung:

- W kann sich durch Vergleich seiner IN vergewissern, daß sein Stimmzettel richtig gezählt wurde
- CTF kann 'Wahlurne' nicht antasten, da CTF von CLA überwacht wird
- CLA muß alle W ohne zugehörige VN veröffentlichen = nur eine VN pro W möglich
- Sowohl die Zufallszahlen VN als auch IN müssen (entsprechend der Anzahl von wählenden Personen) entsprechend groß sein, um Kollisionen auszuschließen.
- Wenn CLA und CTF kooperieren ist die Anonymität verletzt

4.9 Wahl mit einer zentralen Einrichtungen

Das Wahlverfahren ist dem Wahlverfahren von 'Wahl mit zwei zentralen Einrichtungen' (siehe oben) gleich. Der einzige Unterschied liegt darin, das hier die beiden Organisationen zusammengefaßt sind (CLA + CTF = CTF). Um nun zu einer VN zu kommen, welche jedoch gegenüber dem CTF anonym (Feststellung des Wählers aufgrund der VN soll nicht möglich sein) ist, bedient man sich des ANDOS-Verfahrens.

1. Die CFT veröffentlicht eine Liste mit allen Wahlbeteiligten.

2. Jeder Wähler teilt vor der Wal mit, ob er teilnehmen will.

3. Die CFT veröffentlicht eine Liste aller Wahlbeteiligten.

4. Jeder Wähler erhält über ein ANDOS-Protokoll eine Identifikationsnummer I

5. Jeder Wähler generiert ein Schlüsselpaar k, d aus öffentlichem und privatem Schlüssel. Mit dem Votum v sendet er folgende Nachricht an die CFT:

   $I, E_k(I, v)$

6. Die CFT bestätigt den Empfang, indem sie $E_k(I, v)$ veröffentlicht.
7. Jeder Wähler sendet: I, d
8. Die CFT dechiffriert die Stimmabgabe. Nach beendeter Wahl veröffentlicht sie die Abstimmungsergebnisse, und zu jedem möglichen Votum die Liste aller Werte $E_k(I, v)$, die dieses Votum enthalten.
9. Stellt ein Wähler fest, daß seine Stimme nicht korrekt berücksichtigt wurden protestiert er dagegen mit folgender Nachricht an die CFT:
   $I, E_k(I, v), d$

4.9.1 ANDOS-Verfahren

ANDOS steht für All or Nothing Disclosure Of Secrets. Dieses Protokoll erlaubt mehreren Teilnehmern (mindestens zwei werden für das Funktionieren des Protokolls benötigt) individuell ein Geheimnis oder auch mehrere von einem Anbieter zu erstehen. Das ANDOS-Protokoll weist dabei zwei wesentliche Merkmale auf:

- Der 'Verkäufer' kennt nicht den Artikel, welcher ein 'Käufer' erwirbt
- Der 'Käufer' erwirbt nur den gekauften Artikel und wird über die Verbleibenden im unklaren gelassen.

Für das Erlangen einer VN kann dieses Verfahren ideal zum Einsatz kommen, da einerseits die CTF keine Verbindung zum Wähler anhand der VN herstellen kann, andererseits der Wähler nur eine VN vom CTF erstehen kann.
Bemerkung:

So sicher dieses Verfahren auch ist, besteht für die CTF trotz allem ein Weg zu betrügen. Aufgrund der Verwaltung aller VNs besteht die Möglichkeit, nicht berechtigten Wähler eine VN oder mehrere VNs zu geben.

4.10 Wahl ohne zentrale Einrichtung

Dieses Verfahren kommt ohne jegliche zentrale Einrichtung aus. D.h., die Wähler kontrollieren sich gegenseitig, wodurch es nicht möglich ist, zu betrügen bzw. öfters als einmal zu wählen. So perfekt dieses Wahlverfahren in Bezug auf Sicherheit, Anonymität, etc. ist, kann es für reale Wahlen nicht zum Einsatz gebracht werden.

Nehmen wir einfach mal an, Alice,Bob,Carol und Dave wollen eine Wahl durchführen:

1. Jeder Wähler trifft seine Wahl und geht wie folgt vor:

   (a) Er fügt an sein Votum eine Zufallsfolge an.

   (b) Er verschlüsselt das Ergebnis aus Schritt a) mit Daves öffentlichem Schlüssel

   (c) Er verschlüsselt das Ergebnis aus Schritt b) mit Carols öffentlichen Schlüssel

   (d) Er verschlüsselt das Ergebnis aus Schritt c) mit Bobs öffentlichen Schlüssel

   (e) Er verschlüsselt das Ergebnis aus Schritt d) mit Alice öffentlichen Schlüssel

   (f) Er wiederholt die Schritte b)-e), wobei er jedesmal eine Zufallsfolge dazu addiert und sich diese merkt.

   Mit E als Verschlüsselungsfunktion, R_i als Zufallsfolge und V als Votum sieht seine Nachricht wie folgt aus:

   $E_A(R_5, E_B(R_4, E_C(R_3, E_D(R_2, E_A(E_B(E_C(E_D(V, R_1)))))))))$

2. Nun wird die Wahl an Alice geschickt.

3. Alice entschlüsselt mit ihrem privaten Schlüssel und überprüft, ob ihre Stimme dabei ist. Sie gibt die Stimmen an Bob weiter ... (Alle entschlüsseln nun 2 mal).

4. Am Ende erhält Dave das Ergebnis, reicht es an die anderen weiter, jeder kann auszählen und überprüfen, ob seine Stimme dabei ist.

5 Schlußfolgerung

Da das Mißtrauen unter den Menschen vorhanden und darüber hinaus in vielen Bereichen durchaus gerechtfertigt ist, trifft in der Computerwelt diese Tatsache noch verstärkt zu. Durch die für den Menschen nicht überschaubare 'Flut' von Informationen, welche durch elektronische Medien an den Einzelnen herangetragen wird, sowie die Nicht-Nachvollziehbarkeit der Datenwege an sich, ist es naheliegend, daß elektronische Wahlen nur dort zum Einsatz kommen, wo die Daten in gewissem Rahmen überschaubar sind, sowie der sicherheitsrelevante Aspekt hinsichtlich Anonymität, Persönlichkeit, etc. nicht so sehr im Vordergrund steht. Ein typisches Anwendungsgebiet wäre z.B. Betriebsratswahlen in einer Firma.

Den Sinn der Verschlüsselung von E-Mails brauche ich an dieser Stelle nicht noch einmal zu bestätigen, hierzu bietet PGP ein praktisches Werkzeug. Einzig die Probleme, die bei der Schlüsselzertifizierung auftreten können und die mangelnde Integration in (viele) E-Mail-Clients verhindert wohl eine weitere Verbreitung von PGP.

Literatur

[1] http://www.pgpi.org - Internationale Pretty-Good-Privacy Homepage

[2] John Wiley & Sons Inc.;Bruce Schneier APPLIED CRYPTOGRAPHY, SECOND EDITION (Protocols, algorithms and source code in C)

[3] Arto Salomaa; VERIFYING AND RECASTING SECRET BALLOTS IN COMPUTER NETWORKS, Abstract

[4] William Stallings; Ph.D. NETWORK AND INTERNETWORK, SECURITY, PRINCIPLES AND PRACTICE

[5] Reinhard Wobst: Abenteuer Kryptologie, Kapitel 7.1

[6] Klaus Schmeh: Safer Net, Kapitel 15

[7] Das GNU-Handbuch zum Schutz der Privatsphäre http://www.gnupg.org/gph/de/manual/