

Elektronisches Geld und elektronische Wahlen

(von Claudia Ernst)

Inhalt

1. Einleitung	1
2. Anonymität	2
3. MIX-Netze:	
3.1 Überblick.....	2
3.2 Idee und kryptologische Umsetzung	2
3.3 Weitere Möglichkeiten.....	3
3.4 Nachteile.....	3
3.5 Sicherheitsmaßnahmen.....	4
4. Blinde Signaturen:	
4.1 Ziele.....	4
4.2 Idee einer blinden Signatur.....	4
4.3 RSA-Konzept.....	4
5. Elektronisches Geld:	
5.1 Eigenschaften.....	5
5.2 Beispiele.....	5
5.3 Digitale Münzen / Online Protokoll.....	5
5.4 Digitale Schecks / Offline Protokoll.....	5
5.5 Protokoll von Chaum / Fiat / Naor.....	6
5.6 Restprobleme.....	7
6. Pseudonyme:	
6.1 Motivation / Idee.....	7
6.2 Umsetzung.....	7
7. Elektronische Wahlen	
7.1 Eigenschaften.....	7
7.2 Erste Ideen.....	8
7.3 Wählen mit Pseudonymen.....	8
7.4 Wählen mit blinden digitalen Signaturen..	9
7.5 Wählen mit zwei Wahlämtern.....	9
8. Nachwort.....	10
9. Quellen.....	10

1. Einleitung

Woran denkt man zuerst, wenn man etwas von elektronischem Geld und elektronischen Wahlen hört? Nun, wahrscheinlich zunächst an Online-Banking und an die Schwierigkeiten, die computerisierte Wahlen mit sich bringen. Vielleicht denkt man auch an kryptologisch sicheren Datentransfer, an PIN und TAN und andere Codes.

Doch das eigentliche Problem liegt nicht nur in der Sicherheit, sondern auch in der Wahrung der Anonymität von Systemteilnehmern.

Protokolle für elektronisches Geld müssen einiges leisten:

Sie müssen zunächst einmal die Anonymität z.B. eines Bankkunden gewährleisten. Wer will schon, dass die ganze Welt weiß, wieviel Euro man auf seinem Konto hat?

Zusätzlich muss der Datentransfer nach *Außen* sicher sein, damit kein Angreifer sich ins Netz einklinken und finanzielle Verhältnisse bzw. Geldtransfers anderer einsehen kann.

Außerdem muss der Datentransfer auch nach *Innen* sicher sein. Denn sonst gäbe es diverse Möglichkeiten für Bankkunden zu Betrügen.

Auch an Protokolle für elektronische Wahlen werden zahlreiche Ansprüche gestellt und auch hier steht die Sicherheit gegen Betrug und natürlich die Anonymität der Wähler im Vordergrund. Oder wollen Sie sich beim Voten über die Schulter schauen lassen?

Kommen wir also zunächst zur:

2. Anonymität

Die Anonymität von Systemteilnehmern ist eine relativ neue Aufgabe der Kryptografie. Es bedeutet die Teilnehmeridentitäten innerhalb eines Systems, sowie nach außen zu verbergen.

Hierbei gibt es drei Aspekte der Anonymität:

1. Die Senderanonymität

Der Sender einer Nachricht möchte seine Identität vor dem Empfänger der Nachricht verbergen.

Gibt es hierfür überhaupt eine praktische Anwendung?

Ja, es gibt sogar eine aus dem Alltagsleben: Ein anonymer Hinweis für die Polizei.

Wenn ein Zeuge einer Tat z.B. selbst polizeilich gesucht wird, so kann er per Telefon Hilfe holen, ohne seine Identität vor der Polizei preis geben zu müssen.

2. Die Empfängeranonymität

Hier möchte nun der Empfänger einer Nachricht seine Identität vor dem Sender verbergen.

Ein Alltagsbeispiel für Empfängeranonymität sind Chiffrennummern auf Kontaktanzeigen. Möchte man auf eine Zeitungsannonce antworten, so schreibt man den Brief nicht direkt an den Anzeigensteller, sondern an die Zeitung mit Verweis auf die entsprechende Chiffrenummer.

3. Anonymität der Kommunikationsbeziehung

Hierbei möchten zwei Systemteilnehmer miteinander kommunizieren, ohne dass andere davon wissen. Soll heißen: Ein Außenstehender oder auch andere Systemteilnehmer können nicht feststellen, dass die Nachricht, die Teilnehmer A gesendet hat und auch die ist, die Teilnehmer B bekommen hat.

Eine Möglichkeit, die Anonymität von Kommunikationsbeziehungen innerhalb eines Systems zu erreichen, bietet ein MIX.

3. MIX-Netze

3.1 Überblick

Ein MIX soll also die Kommunikationsbeziehungen zwischen Teilnehmern verbergen.

Wie könnte er das machen?

Einen MIX kann man sich vorstellen als eine große „Black Box“ durch die alle gesendeten Nachrichten durchfließen, ehe sie zum Empfänger gelangen.

Hierzu sammelt der MIX alle ausgehenden Nachrichten über einen bestimmten Zeitraum hinweg, mischt sie anschließend durch und versendet sie nach einer gewissen Zeit an die entsprechenden Empfänger weiter. Dadurch kann ein Angreifer durch Abfangen der vom MIX weitergeleiteten Nachrichten nicht feststellen, welche Nachricht von welchem Sender stammt.

Es ist offensichtlich, dass diese Anonymität nur bei möglichst vielen Nachrichten gewährleistet ist, denn wenn über eine gewisse Zeit nur eine Nachricht versendet und eine empfangen wird, so ist es wohl nicht schwer zu sehen, wer dort mit wem kommuniziert hat.

Wir kann man nun einen solchen MIX realisieren?

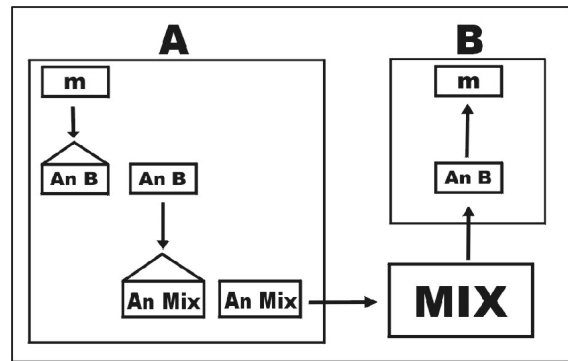
3.2 Idee und kryptologische Umsetzung

Wir wollen uns nun anhand eines Briefmodells die Idee eines MIX verdeutlichen.

Teilnehmer A möchte eine Mitteilung an Teilnehmer B senden, ohne dass ein anderer dies weiß oder gar die Mitteilung lesen kann.

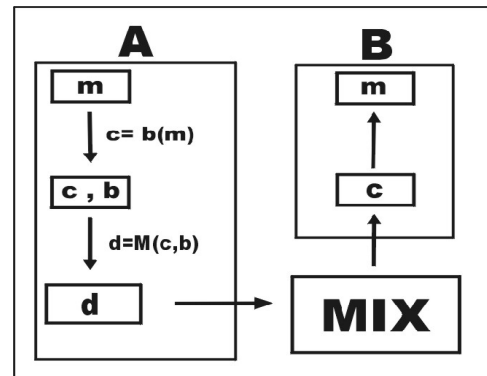
Hierfür steckt A seine Nachricht m in einen Umschlag und verschließt diesen gut. Auf den Umschlag schreibt A dann den Hinweis: „An B“. Anschließend steckt A diesen Umschlag in einen weiteren Umschlag, verschließt auch den und schreibt die Adresse „An den MIX“ darauf. Nun versendet A seinen Brief an den MIX.

Nachdem der MIX diesen bekommen hat, öffnet er den äußeren Umschlag und mischt den Brief mit anderen Briefen. Nach einiger Zeit versendet er den Brief an B weiter. Wenn B ihn empfängt, öffnet er den letzten Umschlag und kann nun die Nachricht m lesen.



Das gleiche Modell kann man auch mit einem Public-Key-System erreichen, wobei die Umschläge durch die entsprechende Codierung mit öffentlichen Schlüsseln und das Öffnen der Umschläge durch das Decodieren mit privaten Schlüsseln dargestellt werden.

Nebenstehend ist $b()$ der öffentliche Schlüssel von B und $M()$ der des MIX.



3.3 Weitere Möglichkeiten

Ein MIX kann jedoch noch mehr leisten, als nur die Kommunikationsbeziehungen zu verbergen.

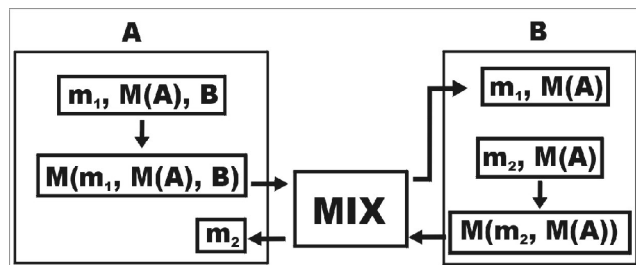
Wenn ein Sender in seiner Nachricht keinen Hinweis auf seine Identität hinterlässt, so ist das MIX-Protokoll auch senderanonym.

Das hat jedoch einen Nachteil: Der Empfänger hat keine Möglichkeit dem Sender zu antworten.

Doch auch hierfür gibt es eine Lösung: Die anonyme Rückadresse.

Dafür schreibt der Sender auf seine Nachricht auch noch seine eigene, mit dem öffentlichen Schlüssel des MIX verschlüsselte Adresse.

Diese kann dann der Empfänger zwar nicht entschlüsseln, er kann sie aber auf seinen Antwortbrief schreiben. Der MIX entschlüsselt dann die „anonyme Rückadresse“ und stellt die Nachricht entsprechend zu.



Auch Empfängeranonymität kann man mit einem MIX erreichen, indem alle Teilnehmer bei dem MIX eine Art Postfach einrichten und somit ihre Identität nicht preis geben müssen.

3.4 Nachteile des MIX

Ein Problem ist, dass die Teilnehmer dem MIX vertrauen müssen, da er als einzige Instanz alle Kommunikationen rekonstruieren kann. Abschwächen kann man dieses Problem, indem man eine MIX-Kaskade baut, d.h. man schaltet beliebig viele MIXe hintereinander. Arbeitet auch nur ein MIX in einer MIX-Kaskade korrekt, so sind die Kommunikationsbeziehungen anonym.

Ein weiteres Problem ist die Sicherheit, denn es gibt nur eine Maßnahme, nämlich das Public-Key-Verfahren. Ein möglicher Angriff auf den MIX könnte so ablaufen:

Der Angreifer fängt alle Nachrichten ab, die vom MIX ausgehen und in den MIX eingehen. Die Ausgegangenenen verschlüsselt er nun mit dem öffentlichen (also auch ihm bekannten) Schlüssel des MIX. Vergleicht er nun die verschlüsselt eingegangenen mit den selbst verschlüsselten ausgegangenen Nachrichten, so kann er sehen, wer mit wem kommuniziert hat. Damit wäre die Anonymität aufgehoben.

3.5 Sicherheitsmaßnahmen

Ein solcher Angriff wäre natürlich nur bei einem deterministischen Verschlüsselungsverfahren möglich. Ein probabilistisches Verfahren könnte man erreichen, indem der Sender an seine Nachricht eine Zufallszahl anhängt, die der MIX löscht, bevor er die Nachricht weiterversendet.

Auch die Nachrichtenlänge aller Teilnehmer muss gleich sein, da sonst ein Angriff durch Vergleich der Nachrichtenlängen möglich wäre. Zusätzlich sollte ein permanenter Netzbetrieb herrschen und es sollten möglichst alle Teilnehmer gleichzeitig gleich viele Nachrichten empfangen und versenden. Das könnte man durch den Versand so genannter Dummy-Nachrichten ermöglichen.

4. Blinde Signaturen

4.1 Ziele

Das Ziel einer blinden digitalen Signatur ist die anonyme Übermittlung von Signaturen. Eine direkte Anonymisierung ist hierbei nicht möglich, denn die Gültigkeit einer Signatur hängt von dem dazu verwendeten privaten Schlüssel ab. Möglich ist jedoch, dass ein Teilnehmer eine gültige Signatur einer zentralen Instanz erhält, die ihm ein bestimmtes Recht, z.B. den Zugang zu einem Dienst, einräumt. Es ist jedoch wichtig, dass weder der Dienstleister, noch die Zentrale anhand der Signatur die Teilnehmeridentität feststellen kann. Ein Beispiel hierfür ist elektronisches Geld (Zentrale = Bank, Dienstleister = Händler).

4.2 Idee einer blinden Signatur

Teilnehmer A erhält von der Zentrale eine gültige Signatur auf eine Nachricht m . Die Zentrale kennt m nicht und kann auch aus m nicht auf A schließen. D.h. falls die Zentrale später m vorgelegt bekommt, weiß sie nicht mehr, dass sie m für A signiert hat, da sie bei der Signierung m nicht sehen konnte.

Falls A nicht der einzige ist, der von der Zentrale eine Signatur erhält, so kann A seine Nachricht über einen MIX versenden, sodass kein nachvollziehbarer Zusammenhang mehr zwischen A und der Signatur bzw. der Nachricht besteht.

Erreichen kann man dies durch eine interaktive Signaturerstellung zwischen A und der Zentrale.

4.3 RSA-Konzept

Sei (n,e) der öffentliche Schlüssel der Zentrale und d der private.

A wählt zufällig eine Zahl r aus \mathbb{Z}_n^* , „blendet“ damit seine Nachricht m und erhält m' . Dazu rechnet er: $(r^e \cdot m) \bmod n = m'$. Es ist klar, dass m einem Redundanzschema entsprechen oder der Hashwert einer Klartextnachricht sein muss.

A sendet nun m' an die Zentrale, die m' signiert und sig' erhält. Sie kann aber m nicht lesen, da m mit r „geblindet“ wurde. Die Zentrale rechnet: $(m')^d \bmod n = sig'$. Die Zentrale schickt sig' an A zurück.

A kann die Signatur der Zentrale jetzt entblenden und erhält sig . Dazu rechnet A: $(r^{-1} \cdot sig') \bmod n = (r^{-1} \cdot (m')^d) \bmod n = (r^{-1} \cdot (r^e \cdot m)^d) \bmod n = (r^{-1} \cdot r^{e \cdot d} \cdot m^d) \bmod n = (r^{-1} \cdot r \cdot m^d) \bmod n = m^d \bmod n = sig$.

Damit hat A nun eine gültige Signatur von der Zentrale erhalten.

Verifizieren kann man diese durch den öffentlichen Schlüssel der Zentrale: $sig^e \bmod n = m^{e \cdot d} \bmod n = m$.

Falls r wirklich gleichverteilt ist, so ist es auch m' . Somit kann die Zentrale m wirklich nicht erkennen.

5. Elektronisches Geld

Im Folgenden suchen wir ein System zu Umsetzung von digitalem Geld mittels blinder Signaturen. Dazu müssen wir uns zunächst überlegen, welche Eigenschaften das Geld haben muss.

5.1 Eigenschaften von elektronischem Geld

- 1) Das Geld sollte fälschungssicher sein, d.h. es kann nur von einer autorisierten Bank erstellt werden.
- 2) Die Echtheit des Geldes sollte von jedem verifizierbar sein, z.B. durch bestimmte Sicherheitsmerkmale.
- 3) Das Geld sollte anonym sein und damit keine Informationen über den Besitzer preisgeben, weder für Außenstehende, noch für die Bank.

Dies ist mit Hilfe blinder digitaler Signaturen möglich:

- 1) Ein Datensatz kann nur von der Bank signiert werden und ist damit fälschungssicher.
- 2) Die Echtheit einer Banksignatur ist mit deren öffentlichen Schlüssel verifizierbar.
- 3) Der signierte Datensatz gibt keine Informationen über seinen Besitzer preis und ist damit anonym.

5.2 Beispiele

Die Anonymität von elektronischem Geld sollte bedingungslos sein. Das bedeutet es sollte keine Seriennummer oder Ähnliches tragen, womit man es zweifelsfrei identifizieren könnte. Damit scheiden Geldscheine als Beispiel schon mal aus.

Herkömmliche Münzen dagegen sind ein zutreffendes Beispiel. Auch digitale Schecks, bei denen die Schecknummer vom Besitzer selbst erstellt wurde, sind ein weiteres Beispiel für elektronisches Geld.

Wollen wir uns nun ein digitales Münzsystem ansehen.

5.3 Digitale Münzen/Online Protokoll

Eine Bank veröffentlicht einen öffentlichen Schlüssel (n,e), ein Redundanzschema und die Information, dass Datensätze, die mit (n,e) signiert sind, (z.B.) 10 Euro wert sind.

Der Kunde A möchte nun eine 10€ Münze von seinem Konto abheben.

Dazu authentifiziert er sich gegenüber der Bank und lässt sich dann einen Datensatz m blind von ihr mit dem „10€ - Privatschlüssel“ signieren. Nun hat A eine gültige 10€ Münze. Anschließend bucht die Bank das Geld von A's Konto ab.

Die Authentifizierung erfolgt nur, damit die Münze von dem richtigen Konto abgebucht wird. m muss jedoch das vorgegebene Redundanzschema erfüllen, sonst wird das Geld zwar von A's Konto abgebucht, aber A erhält keine gültige Münze.

Die Bank kann auch m später nicht wiedererkennen, da sie m nicht gesehen, sondern blind signiert hat.

Nun möchte A mit der Münze bei einem Händler B Ware bezahlen.

A gibt B also den Datensatz m. B lässt sich den Namen der Bank von A sagen und kann dann durch Verifizieren der Banksignatur mit (n,e) die Echtheit der Münze m feststellen. Daraufhin händigt B die Ware an A aus und lässt sich den Gegenwert von m von der Bank auf sein Konto gutschreiben.

Dieses System hat jedoch auch Nachteile.

B weiß nämlich nicht, ob er auch die „Originalmünze“ hat. A kann die Münze m, nachdem er sie einmal legal abgehoben hat, beliebig oft kopieren und ausgeben. Das bedeutet, dass B für jede Münze bei der Bank nachfragen muss, ob sie schon einmal eingezahlt wurde. Deswegen ist ein solches Münzsystem ein sogenanntes *Online-Protokoll*. Ein Online Münzsystem ist praktisch aber nicht durchführbar, da es eine zu hohe Netzlast bedeuten würde, wenn viele Verbindungen gleichzeitig zu allen Banken (und v.a. möglichst in Echtzeit) aufgebaut werden.

Auch die perfekte Anonymität der Münzen hat einen Nachteil: Das Geld aus Erpressungen und Geldwäsche lässt sich nicht zurückverfolgen und ist somit verloren. Um das zu verhindern gibt es *faire Münzsysteme*, bei denen die Anonymität in einem solchen Fall aufgehoben werden kann.

Außerdem bleibt auch noch das Problem des Wechselgeldes (denn meistens kostet etwas nicht 10 € sondern 9,99 €).

5.4 Digitale Schecks / Offline Protokoll

Es gibt auch *Offline-Protokolle*, wie z.B. digitale Schecks, bei denen die Netzlast nicht ganz so hoch ist.

Das funktioniert, indem der Händler B erst alle Schecks, die er von Kunden bekommt, einsammelt und dann gebündelt bei der Bank einreicht.

Das Problem hierbei ist, dass das Einreichen von gefälschten, also kopierten Schecks nicht zu verhindern ist. Allerdings sollte ein Betrug (und zwar sowohl einer vom Kunden, als auch einer vom Händler) von der Bank lückenlos aufgedeckt werden können.

Überlegen wir uns hierzu erst einmal die Anforderungen, die ein digitaler Scheck erfüllen muss.

1) Ein Scheck muss gedeckt sein.

Das ist einfacher zu erfüllen als bei Papierschecks, denn die Bank signiert nur Schecks die gedeckt sind.

2) Ein Scheck muss fälschungssicher sein.

D.h., wenn er zweimal verwendet wird, so wird der Betrug erkannt.

3) Der Zahlungsverkehr muss anonym bleiben.

Ja, warum das denn? Was hat denn die Bank davon, wenn sie weiß, wo der Kunde seine Kleider kauft?

Wenn man genau hinschaut, erfährt man mehr über den Kunden, als ihm lieb sein kann: Risikofreudigkeit, Pünktlichkeit, persönliche Vorlieben, Spenden, Alimente, Arbeitsverhältnisse, finanzielle Verhältnisse. Der gläserne Mensch sollte aber nicht das Ziel digitaler Schecks sein.

5.5 Protokoll von Chaum / Fiat / Naor

Dieses Protokoll versucht den Anforderungen, die an digitale Schecks gestellt werden, zu genügen. Hierbei ist von der Bank eine Hashfunktion h und eine Zahl N vorgegeben.

A möchte einen Scheck bei der Bank kaufen, um eine Ware von B damit zu bezahlen. A geht dafür folgendermaßen vor:

A stellt seine Person, d.h. seinen Namen, Kundennummer, Name der Bank, etc., als Zahl I dar. I könnte z.B. der Hashwert der Personenbeschreibung sein. Nun erzeugt A eine sehr lange Zufallszahl R , die Schecknummer. Sie sollte so lang sein, das sie von keinem anderen Kunden gewählt wird.

Anschließend erzeugt A drei gleichlange Zufallszahlenfolgen: (a_i) , (b_i) und (c_i) , z.B. à 40 Zahlen. Diese Zahlen müssen vom gleichen Datentyp wie I sein. Aus diesen Zahlen errechnet A nun die Folgen (x_i) , (y_i) : $x_i = h(a_i, b_i)$, $y_i = h(a_i + I, c_i)$, wobei h die von der Bank vorgegebene Hashfunktion ist.

Auf den Scheck schreibt A nun den Betrag, R , (x_i) und (y_i) .

A erzeugt insgesamt N (Zahl von der Bank) solcher Schecks, blendet die Hashwerte der Schecks und gibt diese der Bank. Die Bank wählt $N-1$ der Schecks aus, die A offenlegen muss, d.h. A muss der Bank diese Schecks sowie (a_i) , (b_i) und (c_i) geben und ihr den Blindfaktor verraten. Die Bank kann nun (x_i) und (y_i) prüfen. Ist alles in Ordnung, so unterschreibt die Bank den N -ten Scheck und bucht von A's Konto den Betrag ab.

Jetzt kann A den Händler B mit dem Scheck bezahlen. Dazu sendet A den Betrag, die Schecknummer R und den Hashwert des Schecks zusammen mit der Signatur der Bank (natürlich ohne Blindfaktor) an B. B prüft zunächst die Unterschrift der Bank mit deren öffentlichen Schlüssel. Dann sendet B eine 40-Bit-Zahlenfolge (z_i) an A. A muss für jedes Bit von z_i entweder a_i , b_i und y_i (falls $z_i = 1$) verraten oder $(a_i + I)$, c_i und x_i (falls $z_i = 0$). B kann sich nun die Zahlenfolgen (x_i) und (y_i) errechnen und kann somit den Hashwert des Schecks prüfen. Zuletzt händigt B die gewünschte Ware an A aus, gibt alle Daten, die er von A bekommen hat der Bank weiter, die diese speichert, und lässt sich den Betrag auf sein Konto gutschreiben.

Die Vorteile dieses Protokolls sind groß:

1) Die Anonymität von A wird gewährleistet, denn die Schecknummer hat die Bank bei der blinden Signatur nicht gesehen. Auch I kann die Bank nicht erkennen, denn für kein $i \in \{1, \dots, 40\}$ kennt sie a_i und $a_i + I$.

2) Ein Betrug von A wird aufgedeckt, denn die Möglichkeit bei der Signatur der Bank schon zu betrügen, liegt bei $1/N$. Zusätzlich prüft die Bank, ob die Schecknummer schon gespeichert wurde. Falls ja, kann die Bank A's Personenbeschreibung I herausfinden, denn die Zahlenfolgen (z_i) zweier Händler sind nur mit einer Wahrscheinlichkeit von $1/(2^{40})$ gleich. D.h. es gibt meist mindestens ein i , für das die Bank dann a_i und $a_i + I$ kennt und damit auch I herausbekommt.

Zusatzinfo: Falls doch mal zufällig die Schecknummern zweier Kunden gleich sind, so ergeben sich sehr wahrscheinlich verschiedene und/oder unsinnige Personenbeschreibungen I.

3) Auch ein Betrug von B wird schnell aufgedeckt, denn gleiche Datenfolgen a_i , b_i und y_i bzw. $(a_i + I)$, c_i und x_i sind noch seltener als $1/(2^{40})$. Schecks erfinden kann B auch nicht, denn er kann die Signatur der Bank nicht fälschen.

5.6 Restprobleme

Das Protokoll ist sicher und anonym, aber nur solange ausschließlich A, B und die Bank beteiligt sind. Fängt ein Angreifer A's Scheck ab und gibt ihn aus, so wird A später als Betrüger hingestellt. Tut er dies bei B und zahlt den Scheck bei der Bank ein, so sieht es aus, als hätte B betrogen. Deshalb muss der gesamte Datentransfer kryptologisch sicher sein.

Leider ist auch bei diesem Protokoll durch den mehrstufigen Dialog zwischen A, B und der Bank die Netzbelastung sehr hoch.

Außerdem sind die Einnahmen von B nicht anonym (was aber bei Händlern schon immer der Fall war) und das Problem des Wechselgeldes besteht auch weiterhin.

6. Pseudonyme

6.1 Motivation / Idee

Wir wissen bereits, dass bei digitalen Signaturen die direkte Anonymisierung nicht möglich ist, da der öffentliche Schlüssel (meist mit einem sogenannten Zertifikat) an den Namen der Besitzers gekoppelt ist.

Wie wäre es aber, wenn sich der Systemteilnehmer einen Alias-Namen ausdenken und sich diesen zusammen mit seinem öffentlichen Schlüssel zertifizieren lassen würde? Dies könnte in der Tat mit Hilfe blinder Signaturen durch eine zentrale Zertifizierungsinstanz ermöglicht werden.

Definition: Ein **Pseudonym** ist ein Zertifikat mit einem Alias-Namen für einen öffentlichen Schlüssel.

6.2 Umsetzung

Teilnehmer A möchte ein Pseudonym zusammen mit einem öffentlichen und einem privaten Schlüssel haben. Dazu identifiziert er sich zunächst gegenüber der Zertifizierungsinstanz, um zu zeigen, dass er zugangsberechtigt ist. Dann wählt sich A einen Alias-Namen \tilde{A} und die zwei Schlüssel und lässt sich \tilde{A} zusammen mit dem öffentlichen Schlüssel von der Zertifizierungsinstanz blind signieren.

Damit hat A nun den Nachweis über einen echten Schlüssel (sein Pseudonym), ohne dass jemand (auch nicht die Instanz selbst) weiß, dass der Schlüssel A gehört.

A kann jetzt eine Nachricht mit seinem privaten Schlüssel signieren und diese zusammen mit dem Pseudonym an B schicken. B kann die Signatur mit dem öffentlichen Schlüssel von \tilde{A} verifizieren, er kann aber nicht auf die Identität von A schließen.

Ein wichtiges Einsatzgebiet von pseudonymisierten öffentlichen Schlüsseln sind elektronische Wahlen.

7. Elektronische Wahlen

Zunächst wollen wir uns überlegen, welche Eigenschaften digitale Wahlen haben sollten.

7.1 Eigenschaften

- 1) Es sollten nur Wahlberechtigte abstimmen können, um die Wahl nicht zu verfälschen.
- 2) Jeder Wähler sollte höchstens eine Stimme haben.
- 3) Niemand sollte das Votum eines anderen duplizieren oder ändern können.

- 4) Die Wahl sollte anonym sein.
- 5) Niemand darf erkennen können, was andere gewählt haben.
- 6) Die Wahl muss auf Korrektheit von jedem überprüfbar sein.

Bis auf Bedingung 6) werden alle Eigenschaften von „herkömmlichen“ Wahlen erfüllt.

7.2 Erste Ideen

Wie könnte man eine solche elektronische Wahl nun durchführen? Schauen wir uns ein erstes Abstimmungsprotokoll an:

Ein zentrales Wahlamt hat einen privaten und einen öffentlichen Schlüssel, und gibt den öffentlichen bekannt. Ein Wähler chiffriert nun sein Votum mit dem öffentlichen Schlüssel des Wahlamts und sendet es an das Amt. Das Wahlamt kann nun das Votum dechiffrieren, die Stimmen auszählen und das Ergebnis veröffentlichen.

Dieses Protokoll klingt einfach, birgt aber viele Probleme in sich:

Es kann jeder abstimmen, sogar mehrfach, und man kann dies nicht prüfen. Die Eigenschaft 3) ist hier unerheblich, da man mit dem duplizieren des eigenen Votums schon genug Schaden anrichten kann. Zwar ist die Wahl anonym, also 4) und 5) erfüllt, aber sie ist keinesfalls korrekt, womit auch 6) entfällt.

Versuchen wir es mal mit Abstimmungsprotokoll 2:

Ein Wähler signiert sein Votum erst einmal mit seinem privaten Schlüssel, bevor er es mit dem öffentlichen Schlüssel des Wahlamtes chiffriert. Dann sendet er das Votum an das Amt. Dieses dechiffriert das Votum, prüft die Signatur, zählt die Stimmen aus und veröffentlicht das Ergebnis.

Die Vorteile dieses Protokolls sind schon größer: Es können nur Wahlberechtigte wählen und diese auch nur einmal (Signaturprüfung durch das Wahlamt). Die Eigenschaft 3) ist hier auch erfüllt, allerdings ist nun die Anonymität, also 4) und 5), nicht mehr gegeben. Das wäre so, als würde ein Wahlbeobachter dem Wähler in der Wahlkabine über die Schulter schauen. Außerdem hat nur das Wahlamt die 6. Eigenschaft.

Wie kann man es aber besser machen?

7.3 Wählen mit Pseudonymen

In diesem Protokoll ist ein Wahlamt vorgesehen, dass Pseudonyme für die Wähler signieren soll, womit diese dann abstimmen können.

Schritt 1: Ein Wähler authentifiziert sich zunächst beim Wahlamt [Eig. 1]. Dann wählt er sich einen privaten und einen öffentlichen Schlüssel sowie ein Pseudonym, wobei er einen, von dem Amt vorgegebenen Signaturalgorithmus beachten muss. Anschließend lässt er sich den öffentl. Schlüssel zusammen mit dem Pseudonym blind vom Wahlamt signieren. Damit hat er einen Schlüssel mit Zertifikat, aus dem keiner (auch nicht das Amt) auf ihn schließen kann [Eig. 4, 5].

Schritt 2: Nun beginnt die Wahl. Jede Wahloption entspricht hier einer natürlichen Zahl. Der Wähler signiert eine Zahl mit seinem privaten Schlüssel [Eig. 3] und sendet die Signatur zusammen mit dem Zertifikat an das Amt oder zunächst an einen MIX, der eine Art Wahlurne ist.

Schritt 3: Das Wahlamt dechiffriert alle Signaturen mit den öffentl. Schlüsseln [Eig. 2], zählt die korrekten Stimmen aus und veröffentlicht das Ergebnis, sowie alle abgegebenen Stimmen zusammen mit den entsprechenden Pseudonymen.

Vorteile:

Eigenschaft 1) - 5) sind hiermit also gegeben und auch Eigenschaft 6) ist erfüllt, denn jeder Wähler kann prüfen, ob seine eigene Stimme gezählt wurde, ob alle Pseudonyme nur einmal vorkommen und ob die Auszählung korrekt ist.

Nachteile:

Zum einen müssen die Wähler dem Wahlamt vertrauen, denn es kann einem Wähler auch mehrere Pseudonyme signieren, womit die Wahl verfälscht werden würde.

Zum anderen kann ein Wähler durch die Offenlegung seines Pseudonyms beweisen, was er gewählt hat. Das bedeutet, dass ein Angreifer einen Wähler bezahlen kann, etwas Bestimmtes zu wählen. Also sind Stimmenkauf und Erpressung möglich.

Wir müssen also das Votum irgendwie vom Wähler trennen.

7.4 Wählen mit blinden digitalen Signaturen

Sei A ein Wähler, der für eine bestimmte Partei votieren möchte. Dazu geht A folgendermaßen vor:

Schritt 1: A erstellt N Stimmenmengen. Die Zahl N ist vom Wahlamt vorgegeben. Jede Stimmenmenge enthält genau eine Stimme für jede Wahloption (also z.B. Parteien). Nun erzeugt A zufällig N verschiedene Identifikationsnummern I_1, \dots, I_N [Eig. 4, 5], die so lang sein sollten, dass kein anderer Wähler diese hat. I_1 schreibt A nun auf alle Stimmen in der ersten Menge, I_2 auf alle Stimmen in der zweiten Menge, usw.

Schritt 2: A macht diese Nachrichten (Stimmenmengen) jetzt unkenntlich und schickt sie an das Amt, das sich $N-1$ davon aussucht, die A offenlegen muss. Das Amt prüft nun, ob die Stimmenmengen die Eigenschaften aus Schritt 1 auch wirklich erfüllen und ob A wahlberechtigt ist [Eig. 1].

Schritt 3: Das Amt signiert nun blind die N -te Nachricht [Eig.4], sendet sie an A zurück und speichert A's Namen, damit A nicht noch einmal wählen kann [Eig. 2].

Schritt 4: A entblendet die Nachricht und hat nun für jede Partei eine gültige Stimme.

Schritt 5: A kann sich eine Stimme aussuchen und chiffriert diese mit dem öffentlichen Schlüssel des Amtes [Eig.3(ändern), 4, 5].

Schritt 6: A schickt sein Votum an einen MIX (Wahlurne) und von da aus zum Wahlamt

Schritt 7: Das Wahlamt dechiffriert die Stimmen, prüft die eigene Signatur, sucht nach doppelten Identifikationsnummern [Eig. 3(duplizieren)], speichert die Identifikationsnummern, zählt die Stimmen aus und veröffentlicht sie zusammen mit den entsprechenden Identifikationsnummern.

Vorteile:

Die Eigenschaften 1) - 5) sind auch hier wieder erfüllt. Ebenso die Eigenschaft 6), denn jeder Wähler kann prüfen, ob seine Stimme mitgezählt wurde, ob alle Identifikationsnummern nur einmal vorkommen und ob die Auszählung richtig ist. Außerdem beträgt die Wahrscheinlichkeit, dass man sich mehrere Stimmen für eine Partei von dem Amt signieren lassen kann, nur $1/N$. A kann nun auch nicht mehr beweisen, welche Stimme seine ist, denn die Identifikationsnummern sind zufällig und unabhängig.

Nachteile:

Erstens müssen die Wähler dem Wahlamt vertrauen, da es eigene gültige Stimmen abgeben kann und zweitens kann ein Wähler, der einen Fehler in der Wahl entdeckt, diesen nicht mehr beweisen.

7.5 Wählen mit zwei Wahlämtern

In diesem Protokoll ist ein Registrieramt und ein Auszählungsamt vorgesehen, die jeweils nicht alleine betrügen können.

Wieder möchte unser Wähler A wählen gehen.

Schritt 1: A schickt eine Nachricht mit der Bitte um eine Registriernummer an das Registrieramt.

Schritt 2: Das Registrieramt prüft, ob A wahlberechtigt ist [Eig. 1], schickt ihm dann eine zufällige Registriernummer R, schreibt diese auf eine Liste und speichert A's Namen [Eig. 2].

Schritt 3: Das Registrieramt schickt die Liste mit den Registriernummern an das Auszählungsamt.

Schritt 4: A wählt sich eine Identifikationsnummer I und sendet I, R und sein Votum [Eig.4] verschlüsselt an das Auszählungsamt [Eig.3(ändern), 5].

Schritt 5: Das Auszählungsamt überprüft R [Eig. 3(duplizieren)] und speichert I bei der von A gewählten Partei.

Schritt 6: Das Auszählungsamt veröffentlicht das Ergebnis und die parteispezifischen Listen mit den Identifikationsnummern [Eig. 6].

Vorteile:

Alle Eigenschaften von digitalen Wahlen sind erfüllt. Außerdem überwacht das Registrieramt das Auszählungsamt, welches keine falschen Stimmen dazugeben kann, da das Registrieramt die Liste mit den Registriernummern hat. Das Auszählungsamt kann auch keine Stimmen verändern, da jeder seine Identifikationsnummer und sein Votum kennt.

Nachteile:

Das Registrieramt kann unberechtigte Wähler zulassen, was aber durch den Aushang einer Wahlberechtigtenliste verhindert werden kann.

Das größte Problem bei diesem Protokoll ist aber, das bei dem Zusammenschluss der Ämter die Anonymität aufgehoben wäre, da das Registrieramt die Registriernummer mit dem Namen und das Auszählungsamt die Registriernummer mit dem Votum hat.

8. Nachwort

Es gibt noch viel mehr Protokolle für elektronisches Geld und elektronische Wahlen, die noch ausgeklügelter sind, als die hier vorgestellten. Doch wir haben gesehen, dass jedes Protokoll seine Schwachstellen hat und jede Interaktion zwischen zwei Systemteilnehmern irgendwie angreifbar ist.

Beim Geld sind wir auf das Problem des Wechselgeldes, der Netzlast und des kryptologisch sicheren Datentransfers gestoßen, bei den Wahlen auf die zahlreichen Möglichkeiten (v.a. der Wahlämter) zu betrügen.

Als Fazit lässt sich also sagen, dass es trotz der vielen Ideen, die in den Protokollen stecken, bis heute keine hundertprozentige Sicherheit besteht, weder für elektronisches Geld, noch für elektronische Wahlen!

9. Quellen

- 1) Beutelsbacher,
Neumann, Schwarzpaul: Kryptografie in Theorie und Praxis, Vieweg 2005
- 2) Schneier: Angewandte Kryptographie, Addison Wesley 1996
- 3) Wobst: Abenteuer Kryptologie Addison Wesley 1997