Proseminar: „Moderne kryptische Verfahren“

Thema: Knobeln über das Telefon

Erläuterungen zu Commitment, zu Bit-Commitment, zu Secure Circuit Evaluation und zum Schutz vor allwissenden Orakeln.

Inhaltsverzeichnis:

1. Commitment (allgemein) 2
   1.1 Möglichkeiten in der Graphentheorie 2
2. Bit-Commitment (speziell) 4
   2.1 Münzwurf über das Telefon 5
3. Secure Circuit Evaluation 6
4. Das allwissende Orakel 8

Quellen:

BEUTELSBACHER / SCHWENK / WOLFENSTETTER *(Vieweg-Verlag, ISBN 3-528-36590-0)* — „Moderne Verfahren der Kryptographie“

das Internet *(nicht unbedingt noch aktuell)* — http://home.in.tum.de/~gerold/aktvol20002001/gliederung.html

(Vortrag vom 06.12.2001)

1. Commitment (allgemein)

Das Wort „Commitment“ bedeutet „Verpflichtung“ bzw. „Festlegung“.

Dies bedeutet (wie ein Beispiel zeigen soll):
Beispiel:

Die Stadt Berlin schreibt einen Auftrag für ein attraktives Bauprojekt aus. Viele Firmen sind am Erhalt des Zuschlages interessiert und wollen Angebote einreichen. Jedoch sehen alle Firmen folgende Probleme: *Keine andere Firma* soll im vornherein *das eigene Angebot* in Erfahrung bringen können, um so die Möglichkeit zu haben, dieses zu unterbieten. Weiterhin soll *keine Firma* in der Lage sein, ihr *Angebot,* noch das einer anderen Firma *zu ändern*.
Man entscheidet sich also, das Angebot in einen Safe (geeigneter Größe) zu schließen, und diesen dann an das zuständige Amt zu übergeben. Zu gegebenem Zeitpunkt wird dann der Schlüssel des Safes auch an das Amt geschickt, sodass dieses das Angebot offen legen kann.

Die entsprechende kryptographische Umsetzung besteht in einer *kollisionsfreien Einwegfunktion f* mit c=f(m):
Die *Einwegeigenschaft* garantiert, dass niemand von der Verschlüsselung c auf das Geheimnis m schließen kann.
Die *Kollisionsfreiheit* sorgt dafür, dass niemand (vor allem nicht A selber) ein m’ mit f(m) = f(m’) finden kann, und somit in der Lage wäre, das Commitment zu korrumpieren.

Der naheliegende Gedanke, ein Commitment einfach verschlüsselt zu hinterlegen, scheitert an der Möglichkeit, dass A bei der Offenlegung absichtlich einen falschen Schlüssel angeben könnte, um somit das Ergebnis der Entschlüsselung beeinflussen könnte.

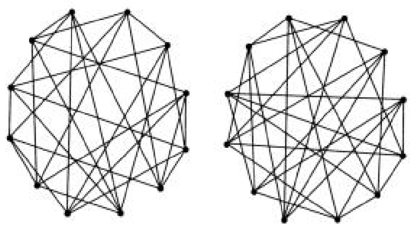
Kryptographisches Protokoll:

(1) Festlegen: A legt sich gegenüber B auf einen Datensatz m fest, indem es c = f(m) berechnet und c an B sendet.

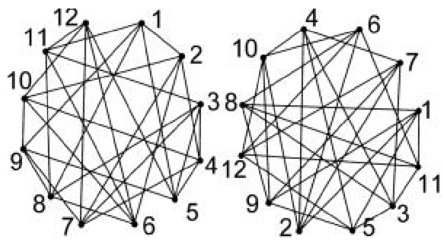
(2) Öffnen: Das Commitment wird geöffnet, indem A das Urbild m von c an B sendet, und B es entsprechend evaluiert.

1.1 Möglichkeiten in der Graphentheorie

Im ersten Augenblick, scheint die Möglichkeit, *Graphen zur Verschlüsselung* zu nutzen, nicht gerade nahe zu liegen.
Der Vorteil (in diesem Fall) liegt jedoch in der Tatsache, das es gerade für große Graphen *keinen effizienten Algorithmus* gibt, mit dem sich entscheiden lässt, *ob zwei Graphen isomorph* sind, d. h. ob der eine Graph lediglich eine Permutation des anderen ist.
Beispiel:



Wie die erste Abbildung zeigt, ist es schwer nachzuvollziehen, ob die beiden Graphen isomorph sind, wenn nicht *jede Möglichkeit durchprobiert* wird. Auch ein Computer hat nur die Möglichkeiten (Permutationen) durchzuprobieren.
Wie der Begriff Permutation schon vermuten lässt, müsste man also n! Möglichkeiten prüfen. Bekanntlich dauert gerade für großes n dies uneffizient lange.

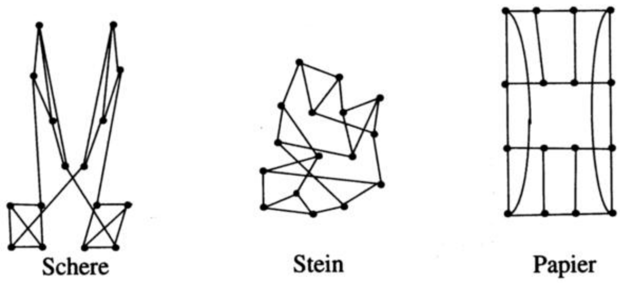


Ist jedoch die verwendete Permutation (*(1 3 5 6)(2 7)(4 12 9 8 10 11)*), wie in der zweiten Abbildung zu sehen, bekannt, kann man schnell überprüfen, dass Isomorphie vorliegt.

Beispiel:
Alice und Bob sind frisch geschieden und müssen sich nun einigen, wer den Hund bekommt. Da beide gern den Hund behalten würden, entscheiden sie sich, um ihn zu knobeln.
Da sie nun in unterschiedlichen Städten wohnen, beschließen sie, über das Telefon „Papier, Schere, Stein“ zu spielen. Dabei gilt:

Papier schlägt Stein
Stein schlägt Schere
Schere schlägt Papier

Da sie feststellen, dass es nicht praktikabel ist, dass beide gleichzeitig ihre Wahl nennen (sie verstehen nicht, was der andere sagt, wenn sie selber gerade sprechen), schlägt Bob vor, Alice solle doch ihre Entscheidung zuerst nennen, und er nennt ihr dann die seinige.
Alice bemerkt jedoch, dass dann Bob sich noch so entscheiden kann, dass er in jedem Fall gewinnen würde.
Deshalb schlägt sie vor, dass sie sich auf drei nicht isomorphe Graphen einigen, von denen jeder eine Möglichkeit repräsentiert:



(Die hier angegebenen Graphen sind für einen effizienten Schutz zu klein, für dieses Beispiel aber praktikabel.)

Alice permutiert nun einen der Graphen so, dass nicht mehr ersichtlich ist, von welchem der drei er abstammt. Die Permutation schickt sie nun per Fax an Bob, der nicht in annehmbarer Zeit ermitteln kann, die Permutation welches Graphen er erhalten hat.
Bob gibt daraufhin seine Wahl bekannt.
Nun kann Alice Bob ihre Entscheidung mitteilen und sendet ihm zusätzlich noch die Art der Permutation, so dass sich auch Bob von der Richtigkeit der Offenlegung überzeugen kann.

2. Bit-Commitment (speziell)

Wie in dem obigen Beispiel zu sehen war, kann es sein, dass die Information, auf die man sich festlegt, relativ kurz ist (im Beispiel hat man sich auf eins von dreien festgelegt).
Nun kann es passieren, dass der *Datensatz m gerade 1 Bit lang* ist, z. B. aus „ja“ oder „nein“ bzw. „0“ oder „1“ besteht.
Das Problem ist, dass es bei solch kleinen Mengen *nur sehr wenige Abbildungen* auf die Menge selber gibt, besonders wenn man eine Abbildung haben möchte, die kollisionsfrei ist. Eine Einwegfunktion gibt es zudem für eine zweielementige Menge nicht.

Jedoch sieht man auch am obigem Beispiel, dass man eine Abbildung wählen kann, die es ermöglicht, diese *Begrenzung zu umgehen*; man hat eine Festlegung durch einen *(großen) Graphen* repräsentiert, für den es bedeutend mehr *(genau n!) Abbildung* auf sich selbst gibt, so dass man eine *potentielle Korrumpierung* besser *ausschließen* kann.

Mathematische Grundlagen:

- <u>Quadratische Reste:</u>

  Eine Zahl $a \in Z_n^*$ heißt *Quadratischer Rest modulo n* (QR) wenn gilt:

  $\exists b \in Z_n^*: b^2 = b * b \equiv a \bmod n$

  (Beispiel: n=55, a=34, b={12, 43, 23, 32} → z. B. 23⊗23=529 mod 55=34)

  Andernfalls heißt a *Quadratische Nichtrest modulo n* (QNR).

  Aus $b^2 = b * b \equiv a \bmod n$ folgt, dass auch $(n-b)^2 \equiv a \bmod n$. Denn:

$$
\begin{aligned}
(n-b)^2 &= n^2 - 2nb + b^2 \\
&\equiv (n^2 - 2nb + b^2) \bmod n \\
&\equiv (n^2 \bmod n) + (-2nb \bmod n) + (b^2 \bmod n) \\
&\equiv 0 + 0 + (b^2 \bmod n) \\
&\equiv a \bmod n
\end{aligned}
$$

- <u>Quadratwurzeln und Faktorisierung:</u>

  Für „n ist Primzahl“ gibt es schnelle Verfahren, *Quadratwurzeln* zu berechnen.
  Ist n jedoch eine zusammen gesetzte Zahl (in dem für uns interessanten Fall z. B. n=pq), hängt die *Berechnung der Quadratwurzeln* eng mit der *Faktorisierung von n* zusammen.
  Es gilt: Das Berechnen von Quadratwurzeln modulo n ist genauso schwierig wie das Faktorisieren von n.
  Anzahl:
  Die *Anzahl* der *Quadratischen Reste modulo n* für n=p beträgt *(p-1)/2*

  $$(\pm 1)^2, (\pm 1)^2, (\pm 2)^2, \ldots, (\pm\left(\frac{p-1}{2}\right))^2$$

  Für *n=pq* ergibt sich analog die Anzahl *(p-1)/2 * (q-1)/2*, da jeder *QR modulo n QP modulo p* und *QR modulo q* ist (siehe Jacobisymbol).
  Für *n=pq* gilt weiterhin: Jedes Element aus Z_n^* hat entweder *keine oder genau 4 Quadratwurzeln* (2 durch p und 2 durch q).

- <u>Legendre- und Jacobisymbol:</u>

  Das *Legendresymbol (x|p)* ist für $x \in Z_p^*$ wie folgt definiert:

$$(x \mid p) := \begin{cases} 1 & \text{falls x QR modulo p} \\ -1 & \text{falls x QNR modulo p} \end{cases} := a^{\frac{p-1}{2}} \bmod p$$

Das *Jacobisymbol* $(x|n)$ gilt für $x \in Z_n^*$. Der relevante Fall ist $n=pq$. Dann gilt:

$(x \mid n) := (x \mid p)(x \mid q)$

Besitzt nun eine Zahl $x \in Z_n^*$ das *Jacobisymbol +1*, ist es nicht entscheidbar, ob x ein *QR* oder *QNR* ist, da $+1 = 1*1 = (-1)*(-1)$ gilt.
Zum Beispiel besitzt die *Zahl –1* (=n-1) das *Jacobisymbol +1* und ist ein *QNR modulo n.*

Eine bekannt Umsetzung dieses Problems stellt die Funktion $f(b,r) = y^b r^2 \bmod n$ dar.
Hierbei ist $n=pq$ das Produkt *zweier großer Primzahlen* und *y ein fixierter QNR modulo n* mit *Jacobisymbol +1*, *r* ist eine *beliebige, große Zufallszahl.*
Die Funktion ist *Einwegfunktion* und erfüllt für $b \in \{0,1\}$ die *Kollisionsfreiheit*:
f(0,r) ist QR und *f(1,r) ist QNR mit Jacobisymbol +1.* Weiterhin gilt *für alle Zufallszahlen r, s* $f(0,r) \neq f(1,s)$.

2.1 Münzwurf über das Telefon

Als Beispiel kann man zum Beispiel folgendes betrachten:
Alice und Bob haben noch einen Kanarienvogel, den einer der beiden nehmen muss – keiner will ihn aber so richtig haben.
Sie entscheiden sich, eine Münze zu werfen, und der Verlierer bekommt das Federvieh.
Schnell stellen sie wieder fest, dass das Telefon wiedereinmal nicht die Möglichkeit bietet, zu überprüfen, ob derjenige, der als zweites spricht (der, der die Münze wirft), die Wahrheit nennt.
Bob erinnert sich an das Bit-Commitment-Protokoll:
Alice und Bob einigen sich darauf, Kopf als „0“ und Zahl als „1“ zu betrachten.
Bob wählt nun eine Zufallszahl r und entscheidet sich für ein b (Kopf oder Zahl). Daraufhin schickt er Alice f(b,r). Diese wirft nun die Münze und nennt Bob das Resultat. Der gibt daraufhin b und r Alice, die nun Bobs Entscheidung b verifiziert, und die beiden haben einen Sieger und einen Verlierer.

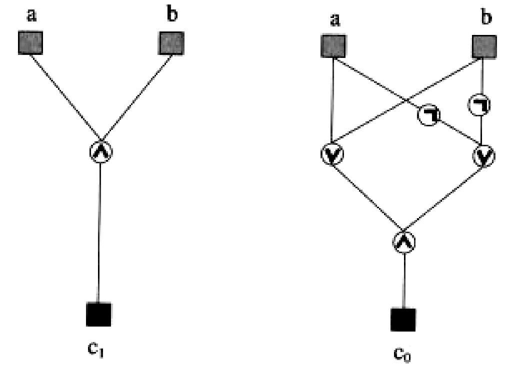
3. Secure Circuit Evaluation

Auch hier soll ein kleines Beispiel dienen, den Sachverhalt etwas näher bringen.
Beispiel:

Ein Steuerberater hat einen *ultimativen Algorithmus* zur Optimierung von Steuererklärungen entwickelt, und nun bietet er seinen Mandanten an, natürlich gegen entsprechendes Honorar, auch ihre Steuererklärung durchzusehen und zu verbessern.
Während das Vermögen des Steuerberaters wächst, fangen auch zwielichtige Gesellen an, sich für das Verfahren zu interessieren, und so kommt es, dass eines Tages die Mafia vor der Tür des Steuerberaters steht und die Herausgabe des Algorithmus fordert.
Der arme Kerl, der bis jetzt darauf bedacht war, *keinem* die Möglichkeit zu bieten, auch nun den kleinsten Teil seines Verfahrens in Erfahrung zu bringen, sieht sich in einer Zwangslage.
Da erinnert er sich an ein Protokoll von ABADI/FEIGENBAUM: die „*Secure Circuit Evaluation*" *(das sichere Auswerten von Funktionen).*

Dieses Protokoll soll folgenden Zweck erfüllen: Eine Person *A besitzt ein Algorithmus (Funktion f)*, den diese Person unbedingte *geheim halten* will. Eine zweite *Person B* will einen *Wert (x) durch A berechnen* lassen, B möchte jedoch nicht, dass der *Eingabewert*, noch das *Ergebnis* etwas an A verraten, die beiden Werte sollen also als *Geheimnisse* behandelt werden.
Es soll also weder die Funktion f, noch der x, noch der Wert f(x) irgendetwas von A oder B verraten.

Dies kann wie folgt bewerkstelligt werden:
Zuerst einmal muss *A* die *Funktion* in einen *BOOLEschen Schaltkreis* umwandeln, was für alle Funktionen mit *fester Länge des Argumentes* möglich ist (Beispiel: der hier gezeigt Schaltkreis berechnet die Summe zweier Bits a und b, das Ergebnis ist c_1c_0).



a	b	c_1	c_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Ein *BOOLEscher Schaltkreis* arbeitet mit den *logischen Grundfunktionen* „$\wedge$", „$\vee$" und „$\neg$". Da sich „$\vee$" auch durch „$\wedge$" darstellen lässt *($a\vee b = \neg(\neg a \wedge \neg b)$)*, braucht *A* also nur mit „*und*" und „*nicht*" rechnen können.
Jetzt muss noch gewährleistet sein, dass die Werte, die *B* berechnen lassen möchte, *verschlüsselt* werden. Auch hierfür bietet sich das bereits vorgestellte *Bit-Commitment* zusammen mit der *Quadratischen Resteannahme* an. *B* muss seinen *Wert*, damit man ihn im Schaltkreis überhaupt berechnen, *in Bitwerte* umwandeln: $x = b_n, \dots, b_2, b_1$.
Dazu wählt *B zwei Primzahlen p und q*, für die gilt: $p \equiv q \equiv 3 \bmod 4$. Damit weiß man auch, dass *–1 das Jacobisymbol +1* besitzt, aber ein *QNR modulo n* ist.
Zur *Verschlüsselung der Bits b* benutzt *B* nun zusätzlich eine *Zufallszahl k* und rechnet:

$$E(b) := (-1)^b k^2 \bmod n$$

Bits, die *0* sind erzeugen also einen *QR*, die, die *1* sind, *QNR mit Jacobisymbol +1*. Diese Operation kann leicht *rückgängig* gemacht werden, wenn man die *Faktorisierung* von *n* kennt *(D(c) = 1, falls c QNR, ansonsten 0)*.
Auf diese Weise erhält man sogar ein *asymmetrisches Verfahren*, sodass sich *von E nicht auf D* schließen lässt, jedoch weiterhin D(E(b)) = b gilt.
Die Person mit dem BOOLEschen Schaltkreis kennt nach der Verschlüsselung die Werte $E(b_1), \ldots, E(b_n)$, aber *nicht* $b_1, \ldots, b_n$ selber.
Dies wirft aber auch die Frage auf, wie *A die Bits invertieren*, also $\neg b$ berechnen kann ohne Kenntnis von b.
Dazu wählt *A* eine *Zufallszahl r* (was weniger der Negation, sondern eher der Verschleierung gegenüber B dient) und berechnet: $E(\neg b) := (-1)r^2E(b) \bmod n$.
Dies funktioniert, denn:

i) $b = 0$ (E(b) ist QR): $E(\neg b) := (-1)r^2(k^2) \bmod n = (-1)(rk)^2 \bmod n \Rightarrow QNR$

ii) $b = 1$ (E(b) ist QNR): $E(\neg b) := (-1)r^2((-1)k^2) \bmod n = (-1)^2(rk)^2 \bmod n = (rk)^2 \bmod n \Rightarrow QR$

Also kann A *Bits invertieren*, in dem es *E(b) mit –1 multipliziert*. Wie bereits angedeutet, dient r zur Verschleierung der Bitnegation gegenüber B, sodass B nicht nachverfolgen kann, welchen Weg durch den Schaltkreis seine Bits nehmen.
Ein wenig Hilfe benötigt A jedoch auch von B, denn bis jetzt ist kein Weg bekannt, wie A alleine $E(b \wedge b')$ aus $E(b)$ und $E(b')$ berechnen kann. Um aber nicht preiszugeben, welche Bits verknüpft werden, verschleiert A erneut die Bits wie folgt:
Zwei *zufällige Bits c, c'* und zwei *Zufallszahlen r und r'* werden folgendermaßen benutzt:
$E(d) = (-1)^c r^2 E(b) \bmod n$, $E(d') = (-1)^{c'} r'^2 E(b') \bmod n$ (Dabei gilt: $d = b$ falls $c = 0$, sonst $d = \neg b$; $d' = b'$ falls $c' = 0$, sonst $d' = \neg b'$).
Nachdem B *d* und *d'* erhalten hat, kann es schnell bestimmen, ob die Variablen 0 oder 1 entsprechen, da B die Faktorisierung von n kennt. Nun muss B d und d' in der Art und Weise, wie sie in der untenstehenden Tabelle angeben ist zurücksenden, denn B kann nur in Abhängigkeit *von c und c' auf* $b \wedge b'$ *schließen*.

Gesendeter Wert	**Entspricht E(b∧b') falls**
$E(d \wedge d')$	c = 0 und c' = 0
$E(d \wedge \neg d')$	c = 0 und c' = 1
$E(\neg d \wedge d')$	c = 1 und c' = 0
$E(\neg d \wedge \neg d')$	c = 1 und c' = 1

B kann nach Berechnung aller Werte nun mit Hilfe der *Ausgabewerte* $E(a_1), \ldots, E(a_m)$ den *Funktionswert* $f(x) = a_m \ldots a_1 = D(E(a_m)) \ldots D(E(a_1))$ erhalten.

Abschließendes Beispiel:

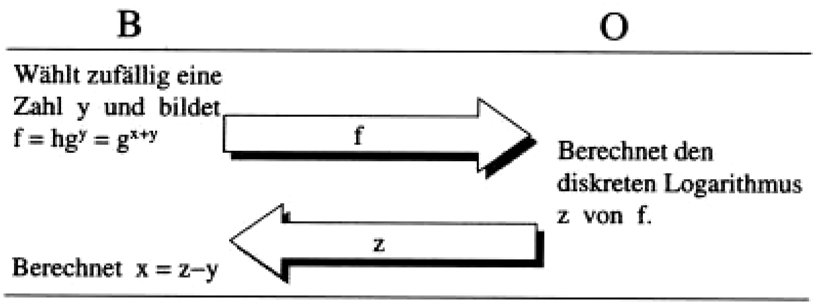
Eine Firma hat ein Verschlüsselungsverfahren entwickelt, dass besonders große Sicherheit gerade beim Transport über das Internet bietet. Es ist natürlich auch nicht daran interessiert, dass der Algorithmus bekannt wird, da dies für sie eine wichtige Einnahmequelle ist.
Kunden, die diese Methode nutzen möchten, wenden sich an die Firma, um ihre Daten vor Ort chiffrieren zu lassen, und die verschlüsselten Daten dann an eine andere Niederlassung zu schicken.
Jedoch sind viele Kunden annähernd paranoid, was die Sicherheit und Geheimhaltung ihrer Daten angeht, sodass sich auch hier das Protokoll der *„Secure Circuit Evaluation“* anbietet.

4. Das allwissende Orakel

Für die Lösung eines Problems, wie zum Beispiel die Berechnung des Diskreten Logarithmus, kann es nötig sein, dass man die Hilfe eines Hochleistungsrechners mit unbegrenzter Leistungsfähigkeit, einem sogenanntem „Orakel“ braucht.
Jedoch möchte man nicht unbedingt, dass dieses Orakel die privaten Eingaben bzw. das Ergebnis erfährt.
Da für das Orakel Faktorisierung und ähnlich Aufgaben kein Problem darstellen, fallen übliche Verschlüsselungsverfahren, die ja gerade auf einer uneffizient langen Entschlüsselung bei fehlenden Informationen aufsetzen, aus.
Für bestimmte Probleme gibt es durchaus Lösungen:

Beispiel 1:

Bob möchte den *diskreten Logarithmus x* eines Elementes $h=g^x$ aus der *Gruppe G* berechnen. Dazu will er ein *Orakel* benutzen, jedoch den Wert *h geheim halten.*
Da ihm die *Basis g* bekannt ist, wählt er ein *zufälliges y* und rechnet: $f=h*g^y=g^{x+y}$.
Den Wert f schickt er an das Orakel, das nun als *Ergebnis z (mit* $z = x + y$*)* liefert.
Da das *Orakel* aber nicht weiß, wie groß y ist, kann es auch *nicht auf x schließen.*
Bob hingegen kennt y und kann nun einfach $x=z–y$ rechnen.



Beispiel 2:

Eine Frau möchte sich in der Praxis ihres Hausarztes wiegen lassen, da sie ihrer eigenen Waage nach Jahren der Benutzung nicht mehr traut.
Jedoch hat sie Angst ihr wahres Gewicht so zu offenbaren. Deshalb nimmt sie ein paar Gewicht mit, und lässt sich mit diesen wiegen.
Danach zieht sie vom Ergebnis die Masse der Gewichte ab und erhält so ihr eigenes.