Aufgabe [Geburtstagsproblem bei Nicht-Gleich-Verteilung]

Ein Zufallsgenerator erzeugt Zufallszahlen aus der Menge $\{1,2,\dots,m\}$ mit dem Koinzidenzindex $IC(P) = \sum_{i=1}^{m} p_i^2$ wobei $P=(p_1,p_2,\dots,p_m)$ die W'Verteilung der Zahlen ist.
Man zieht n derartige Zufallszahlen. Berechne den Erwartungswert von X = Anzahl Dubletten und setze $E(X)=1$. Löse nach n auf!

Antwort: Die W'heit, dass 2 der Zufallszahlen identisch sind, ist $\sum_{i=1}^{m} (p_i)^2 = IC(P)$.

Daher ist $E(X) = \binom{n}{2} \cdot \sum_{i=1}^{m} p_i^2 = \binom{n}{2} \cdot IC(P) \stackrel{!}{=} 1$.

Also: $\frac{n(n-1)}{2} \cdot IC(P) = 1 \iff n(n-1) = \frac{2}{IC(P)}$

Approximativ: $(n-\frac{1}{2})^2 \approx \frac{2}{IC(P)}$

$$n \approx \frac{1}{2} + \sqrt{\frac{2}{IC(P)}}$$

Zahlenbeispiel: Bei $m=26$ und Gleichverteilung ergibt sich $n=7.7$. Bei $IC(P)=7\%$ wie bei deutscher Sprache ergibt sich $n=5.8$.

Zero Knowledge Protokolle

Es geht darum, dass ein Teilnehmer (hier oft „Prover", abgekürzt P, genannt) dem anderen Teilnehmer (dem „Verifier" V) seine Identität beweisen muss. Erst dann darf P irgendwelche Dienste bei V in Anspruch nehmen (Telebanking, Einloggen auf externem Rechner, etc.)

Dass P tatsächlich berechtigt ist, die Dienste bei V in Anspruch zu nehmen, erfordert, dass P zuvor bei V als „berechtigt" registriert wurde, indem bei V der öffentliche Schlüssel k von P eingetragen ist.

Der Prover P hat in einer Initialisierungsphase den öffentlichen Schlüssel k zusammen mit einem dazu gehörigen geheimen Schlüssel k' erzeugt.

Der Identitätsnachweis von P gegenüber V muss irgendwie so erfolgen, dass V davon überzeugt wird, dass P im Besitz des geheimen Schlüssels k' ist. Gleichzeitig soll aber k' auch nach Ablauf des Protokolls geheim bleiben (die sog. <u>Zero Knowledge</u> - Eigenschaft, d.h. weder V noch ein Abhörer der Kommunikation soll auf k' schließen können).

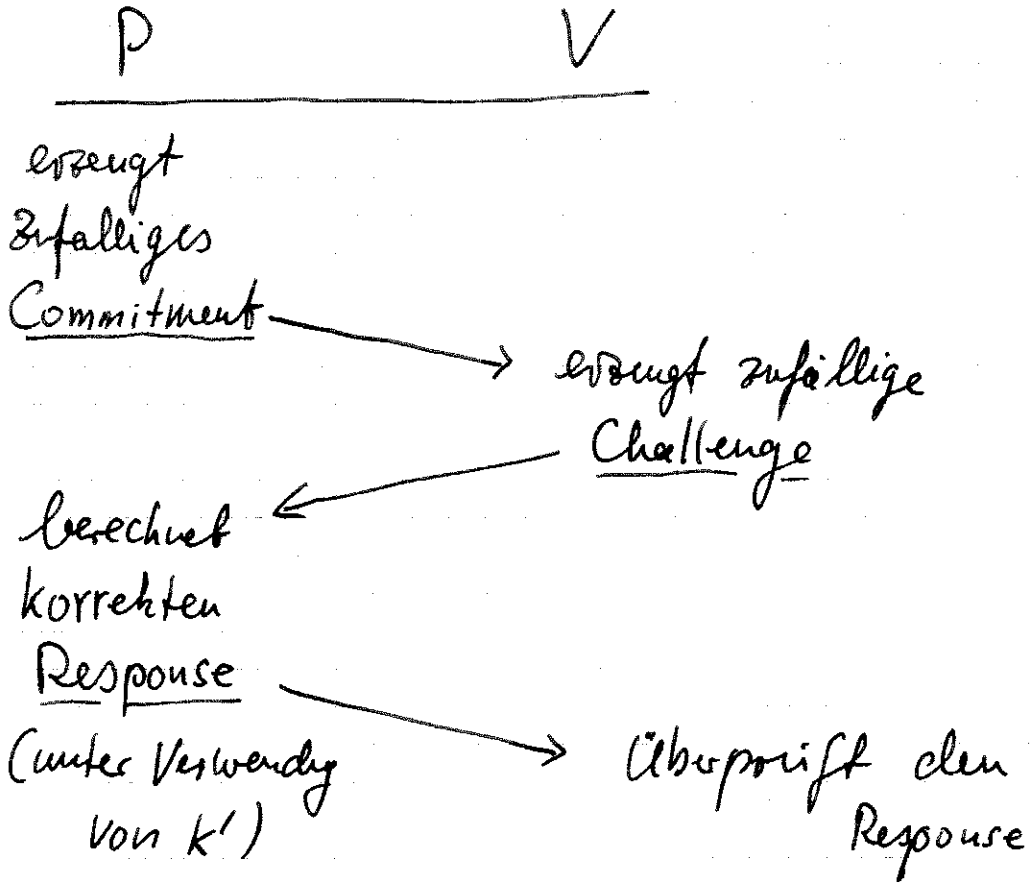
Versucht ein Nicht-Berechtigter $\tilde{P}$ („crooked P") (der k' nicht kennt) sich als P gegenüber V auszugeben – selbst, wenn $\tilde{P}$ zuvor die Kommunikation zwischen P und V abgehört hat — so soll es $\tilde{P}$ nicht gelingen, V davon zu überzeugen, er sei P.

Eine erste Idee: V erzeugt eine Zufallsnachricht m und schickt sie P. Sodann unterschreibt P diese Nachricht: $\hat{m} = D(k', m)$

und schickt $\hat{m}$ zurück an V. Dann kann V überprüfen, dass die Unterschrift korrekt ist: $m \stackrel{?}{=} E(k, \hat{m})$. Aus der Sicht von V wäre diese Vorgehensweise zufriedenstellend.

Nicht jedoch aus der Sicht von P!
P kann nicht wissen, ob V die zu unterschreibende Nachricht m in irgendeiner Weise speziell gewählt hat. Z.B. könnte m eine verblendete Nachricht sein – im Sinne einer blinden Unterschrift, so dass P also ein versteckes Dokument unterschreibt, das er gar nicht kennt.

Zero Knowledge-Protokolle müssen so gestaltet sein, dass es der Prover P ist, der mit einer Zufallsnachricht an V beginnt.
Diese erste Nachricht heißt auch „Commitment".



Zero Knowledge Protokolle laufen nach dem Schema: Commitment – Challenge – Response.

Zero Knowledge mit Hilfe einer Einwegfunktion f:

In der Initialisierungsphase wählt P seinen geheimen Schlüssel als Zufallszahl x.
Dann berechnet er $\tilde{x} = f(x)$ und gibt $\tilde{x}$ als öffentlichen Schlüssel bekannt.

Nun das eigentliche Protokoll:

Prover P	Verifier V

Wählt weitere Zufallszahl y, berechnet $\tilde{y} = f(y)$

$\tilde{y}$ — Commitment →

Wählt Zufallsbit $b \in \{0,1\}$

← b — Challenge

Berechnet:

$$z = \begin{cases} y, & b=0 \\ x \circ y, & b=1 \end{cases}$$

z — Response →

Überprüfung:
Falls $b=0$, so muss $f(z) = \tilde{y}$ gelten.
Falls $b=1$, so muss $f(z) = \tilde{x} \bullet \tilde{y}$ gelten.

Die Einwegfunktion f ist eine homomorphe Einwegfunktion: für gewisse Operationen $\circ$ und $\bullet$ muss $f(x \circ y) = f(x) \bullet f(y)$ gelten.

Analyse:

- Durchführbarkeit: Der korrekte Prover P kann alle Schritte des Protokolls ausführen, so dass die Überprüfung von V am Ende positiv ausgeht.
- Der betrügerische Prover $\tilde{P}$ hat 2 Möglichkeiten:

  $\tilde{P}$ hofft darauf, dass die Challenge $b=0$ ist.
  In diesem Fall führt er die Schritte genau so aus wie im Protokoll vorgesehen: Wählt y, berechnet $\tilde{y}=f(y)$, schickt $\tilde{y}$ an V. Bei der Challenge $b=0$ schickt er $z=y$ und die Überprüfung gelingt. Sollte die Challenge jedoch $b=1$ sein, so kann $\tilde{P}$ kein geeignetes z berechnen, so dass die Überprüfung gelingt.

2. Möglichkeit: $\tilde{P}$ hofft darauf, dass die Challenge $b=1$ ist.

Abweichend vom obigen Protokoll wählt $\tilde{P}$ dann eine Zufallszahl z und bestimmt $\tilde{y}$ so dass $\tilde{x} \bullet \tilde{y} = f(z)$

(mittels: $\tilde{y} := (\tilde{x})^{-1} \bullet f(z)$). In der ersten Runde schickt $\tilde{P}$ dann $\tilde{y}$ an V.
Die Challenge $b=1$ kann $\tilde{P}$ zufriedenstellend mit z beantworten.
Allerdings kann $\tilde{P}$ dann die Challenge $b=0$ nicht beantworten, da er kein y kennt mit $f(y) = \tilde{y}$. (Das ist die Einwegeigenschaft von f.)

Fazit: Ein falscher Prover $\tilde{P}$ scheitert mit Wahrscheinlichkeit $\frac{1}{2}$. Wenn man dieses 3-Runden-Protokoll also t-mal wiederholt, so kann $\tilde{P}$ die Verifikation (jedesmal) nur noch mit W'keit $(\frac{1}{2})^t$ gelingen.

Aus Sicht des Verifiers, der nicht weiß, ob er mit P oder mit $\tilde{P}$ kommuniziert:

Wenn der Prover jedesmal die Verifikation besteht, so muss V davon ausgehen, dass der Prover tatsächlich jedesmal beide Challenges hätte bestehen können.
D.h. zu dem Commitment $\tilde{y}$ kennt der Prover sowohl ein y mit $f(y) = \tilde{y}$ als auch ein z mit $f(z) = \tilde{x} \bullet \tilde{y}$. Mit der Homomorphie-Eigenschaft von f rechnet man dann nach: $f(z) = \tilde{x} \bullet \tilde{y} = f(x) \bullet f(y) = f(x \circ y)$, somit: $z = x \circ y$ bzw. $x = z \circ y^{-1}$.

Das heißt, mit den Informationen y und z des Provers kann dieser grundsätzlich auch das Geheimnis x berechnen. Also muss der Prover mit dem V kommuniziert, der <u>richtige</u> Prover P sein!

Diese Betrachtung hat sich vor allem am Verifier orientiert, dass dieser mit der Überprüfung zufrieden sein kann und (höchstens mit einer Fehlerquote $(\frac{1}{2})^t$) die Identität von P nachgewiesen ist.

Aus Sicht des Provers P geht es darum, dass sein geheimer Schlüssel x sicher sein soll, und auch während des Protokolls keinerlei Information über x hinaus „leckt".

Hierzu dient die Definition der Zero Knowledge-Eigenschaft des Protokolls: Zwischen Prover und Verifier findet ein Informationsaustausch statt:

$$(\text{Commitment}, \text{Challenge}, \text{Response}) = (\tilde{y}, b, z)$$

wobei $f(z) = \begin{cases} \tilde{y}, & \text{falls } b=0 \\ \tilde{x} \circ \tilde{y}, & \text{falls } b=1 \end{cases}$

Da sowohl P als auch V probabilistische Algorithmen sind, ist (Com, Cha, Res) eine (vektorwertige) Zufallsvariable, die einer bestimmten Wahrscheinlichkeits-Verteilung unterliegt.

Man sagt, das Protokoll habe die Zero-Knowledge-Eigenschaft, wenn es einen effizienten probabilistischen Algorithmus S gibt (den Simulator), der die Geheiminformation x nicht zur Verfügung hat, der als Ausgabe ebensolche Tripel (Com, Cha, Res) liefert, die identisch verteilt sind wie die im echten Protokoll zwischen P und V auftretenden Tripel.

Tatsächlich besitzt das obige Protokoll die Zero Knowledge-Eigenschaft. Beim echten Protokoll werden zunächst Com und Cha

zufällig und unabhängig gewählt.
Die anschließende Berechnung der korrekten Res. erfordert das Geheimnis x.

Es geht aber auch anders: Wählt man die 3 Bestandteile des Protokolls in anderer Reihenfolge, so kann man Tripel (Com, Cha, Res) mit exakt derselben W'keitsverteilung erzeugen, ohne das Geheimnis verwenden zu müssen:

Simulator S:

Wähle zufällig $b \in \{0,1\}$

if $b=0$ then wähle zufällig y,
berechne $\tilde{y} = f(y)$
output $(\tilde{y}, 0, y)$

else wähle zufällig z,
berechne $\tilde{y} := (\tilde{x})^{-1} \cdot f(z)$
output $(\tilde{y}, 1, z)$

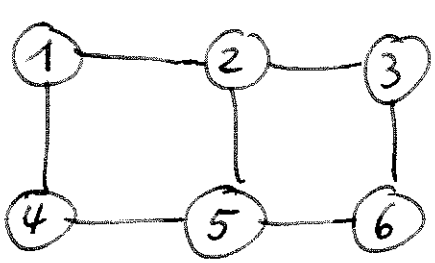
Das historisch erste Zero Knowledge Protokoll beruht auf Graph Isomorphie:

$$GI = \{ (G_1, G_2) \mid G_1 \text{ und } G_2 \text{ sind isomorph; das heißt, wenn } G_1 = (V_1, E_1), G_2 = (V_2, E_2) \text{ und } V_1 = V_2 = \{1, 2, \ldots, n\}, \text{ so gibt es } \pi \in S_n \text{ so dass für alle } x, y \in V \text{ gilt:}$$
$$\{x, y\} \in E_1 \iff \{\pi(x), \pi(y)\} \in E_2 \}$$

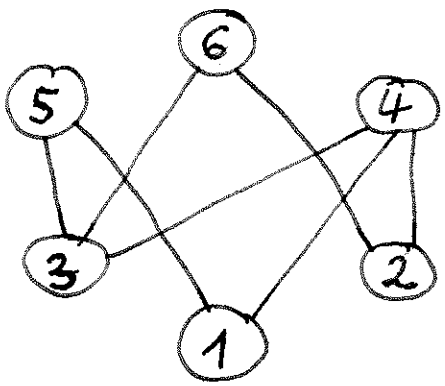
Wenn G_2 auf die beschriebene Weise aus G_1 hervorgeht, schreiben wir auch: $G_2 = \pi(G_1)$.

Beispiel:

G_1:



G_2:



$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 3 & 6 & 1 & 4 & 2 \end{pmatrix}$$

Außer für spezielle Graphen (z.B. planare Graphen) sind keine effizienten Algorithmen bekannt, um festzustellen, ob 2 Graphen isomorph sind (bzw. um einen solchen Isomorphismus π zu berechnen).

In der Initialisierungsphase wählt P einen großen zufälligen Graphen $G_0 = (\{1,2,\dots,n\}, E_0)$ und eine Zufallspermutation $\pi \in S_n$ und bestimmt $G_1 = \pi(G_0)$.

→ Öffentlicher Schlüssel von P: (G_0, G_1)

→ Geheimer Schlüssel von P: π

Protokoll:

P	V

Wählt eine weitere Zufallspermutation σ

$H := \sigma(G_1)$

H →

Wählt Zufallsbit $b \in \{0,1\}$

← b

$$\tau := \begin{cases} \sigma, & b=1 \\ \pi \circ \sigma, & b=0 \end{cases}$$

$\xrightarrow{\tilde{\tau}}$ überprüft, ob $\tau(G_b) = H$

Skizze: