Aufgabe: Beim Zero Knowledge-Protokoll mit Graph Isomorphie lautet

(Commitment, Challenge, Response) = (H, b, τ)

wobei $\tau(G_b) = H$

Gib einen Simulator an (also einen probabilistischen Algorithmus, der ebensolche Tripel (H, b, τ) erzeugt, und der ohne das Geheimnis π auskommt).

Antwort:

Simulator(Eingabe ist (G_0, G_1)):

Rate zufällig $b \in \{0,1\}$ und $\sigma \in S_n$.

if $b=0$ then output$(\sigma(G_0), 0, \sigma)$

else output$(\sigma(G_1), 1, \sigma)$

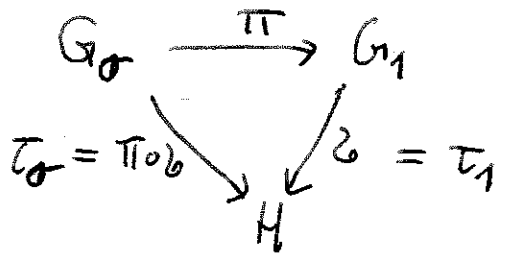
Aufgabe: Nochmals zum ZK-Protokoll mit Graph-Isomorphie. Man zeige:

der Prover kennt den geheimen Schlüssel π so dass $\pi(G_0) = G_1$

$\Leftrightarrow$

der Prover kann beide Challenges $b=0$ und $b=1$ korrekt beantworten.

Antwort: Nochmals die Skizze:



($\Rightarrow$) Das ist die Durchführbarkeit des Protokolles.

Bei $b=0$ liefert der Prover $\tau_0 = \pi \circ b$

Bei $b=1$ liefert der Prover $\tau_1 = b$

($\Leftarrow$) Wenn der Prover bei $b=0$ (bzw $b=1$) die korrekten Antworten τ_0 bzw τ_1 liefern kann, (also $\tau_0(G_0) = H$, $\tau_1(G_1) = H$), dann weiß der Prover auch π, denn $\pi = \tau_0 \circ \tau_1^{-1}$.

Aufgabe: Paul hat große Primzahlen p, q bestimmt, berechnet $n = p \cdot q$ und verwendet n als öffentlichen und p, q als geheimen Schlüssel.

Viktor schlägt folgende Methode vor, um sich davon zu überzeugen, dass Paul die Faktorisierung von n kennt:

Paul | Viktor

Wählt $x < n$ zufällig

Berechnet $y = x^2 \bmod n$

y →

berechnet eine Quadratwurzel w von y (unter Verwendung der Kenntnis von p, q)

w → überprüft, dass $w^2 \equiv y \pmod n$

Sollte sich Paul auf dieses „Zero knowledge"-Protokoll einlassen?

Antwort: Nein, denn wie bei dem Beweis von

Faktorisieren von n $\leq_{\text{prob-eff}}$ Quadratwurzelziehen modulo n

erfährt Viktor mit W'keit $\frac{1}{2}$ eine Quadratwurzel $w \not\equiv \pm x \pmod n$. Dann kann Viktor n faktorisieren, denn $w^2 - x^2 = (w-x)(w+x) \equiv 0 \pmod n$

Mittels $ggT(w-x, n)$ bzw. $ggT(w+x, n)$ ergibt sich ein Faktor von n.

Interessanterweise hat dieses Protokoll die Zero Knowledge - Eigenschaft:

Simulator (Eingabe ist n):
Wähle $w < n$ zufällig
Output $(w^2 \bmod n, w)$

Die Zero Knowledge - Eigenschaft ist offensichtlich nur für Außenstehende (die nur y und w erfahren, aber x nicht kennen so wie Viktor) konzipiert.

Aufgabe: Letztes Mal wurde in abstrakter Weise ein Zero Knowledge - Protokoll definiert, basierend auf einer homomorphen Einwegfunktion f, wobei im Definitionsbereich von f die Operation $\circ$ und im Wertebereich die Operation $\bullet$ zur Verfügung stand. Man gebe die Details dieses ZK-Protokolls konkret an, wenn f die modulare Exponentiation ist.

Antwort: Die Aufgabenstellung schlägt vor,

$$f(x) = a^x \bmod n$$

zu setzen, wobei n Primzahl und a Primitivwurzel modulo n ist. Der Definitionsbereich von f ist $\{1,2,\ldots,n-1\}$ (oder gleichwertig: $\{0,1,\ldots,n-2\}$). Der Wertebereich ist $\mathbb{Z}_n^* = \{1,2,\ldots,n-1\}$.

Die Operation $\circ$ auf dem Definitionsbereich ist die Addition modulo $(n-1)$. Die Operation $\bullet$ auf dem Wertebereich ist die Multiplikation modulo n. Begründung:

$$f(x \circ y) = f((x+y) \bmod (n-1)) = a^{(x+y) \bmod (n-1)} \bmod n$$

$$= a^x \cdot a^y \bmod n = (a^x \bmod n) \cdot (a^y \bmod n) \bmod n$$

$$= f(x) \bullet f(y).$$

In der Initialisierungsphase wählt der Prover die Primzahl n und die Primitivwurzel a sowie $x \in \{0,\ldots,n-2\}$ zufällig, berechnet $y = a^x \bmod n$.

Der öffentliche Schlüssel ist (n, a, y); der geheime Schlüssel ist x.

Das ZK-Protokoll ist wie folgt:

Prover	Verifier
Wählt $u \in \{0,\dots,n-2\}$ zufällig	
berechnet $v = a^u \bmod n$	
$\xrightarrow{v}$	
	wählt $b \in \{0,1\}$ zufällig
$\xleftarrow{b}$	
$w := \begin{cases} u, & b=0 \\ \underbrace{u \circ x}_{=(u+x) \bmod (n-1)}, & b=1 \end{cases}$	
$\xrightarrow{w}$	
	überprüft: $\underbrace{f(w)}_{a^w \bmod n} = v$, falls $b=0$
	bzw. $\underbrace{f(w)}_{a^w \bmod n} = \underbrace{v \bullet y}_{v \cdot y \bmod n}$, falls $b=1$

Die Eigenschaften dieses Protokolls (Durchführbarkeit, zero knowledge-Eigenschaft) wurden schon in allgemeinerer Form für $(f,o,\bullet)$ diskutiert.

Eine Erweiterung des letzten Protokolls nennt sich Schnorr Identifikations-Protokoll (nicht im Buch) und funktioniert (in der Basisversion) wie folgt: Wieder hat der Prover einen öffentlichen Schlüssel (n, a, y) und den geheimen Schlüssel x mit $y = a^x \bmod n$.

P	V
Wählt $u \in \{0, \ldots, n-2\}$ zufällig	
$v := a^u \bmod n$	
$\xrightarrow{v}$	Wählt $b \in \{1, \ldots, n-1\}$ zufällig
$\xleftarrow{b}$	
$w := (u + b \cdot x) \bmod (n-1)$	
$\xrightarrow{w}$	Überprüft: $a^w \bmod n = v \cdot y^b \bmod n$

Der Zero Knowledge Simulator:

Wähle b und w zufällig

Berechne $v = a^w \cdot (y^b)^{-1} \bmod n$

Output (v, b, w)

Das Protokoll braucht nicht mehrfach wiederholt zu werden, da der große Wertebereich von b genügend Sicherheit garantiert.

Bem: Dieses Protokoll kann in ein 1-Runden Zero knowledge Protokoll verwandelt werden:

Der Prover schickt (v, b, w) an V wobei b ein Zeitstempel ist (Datum, aktuelle Uhrzeit)

Der Verifier überprüft nicht nur, dass

$$a^w \bmod n = v \cdot y^b \bmod n$$

sondern auch, dass die Uhrzeit b korrekt ist.

Problem: Der falsche Prover $\tilde{P}$ könnte einen Zeitpunkt b in naher Zukunft auswählen, sich zu diesem b wie beim Simulator passende w und v mit $a^w \equiv v \cdot y^b \pmod n$ ausrechnen, dann den Zeitpunkt b abwarten und zu diesem Zeitpunkt seine Nachricht (v, b, w) an den Verifier schicken.

Um dies zu verhindern fügen wir der Challange b noch den Hashwert $h(v)$ hinzu. Auf diese Weise muss der Prover zuerst v festlegen und kann nicht so vorgehen wie beim Simulator beschrieben. Die Nachricht an V besteht also aus:

$$(v, (b, h(v)), w).$$

V verifiziert: ☐ die Uhrzeit b, ☐ $h(v) = v$

☐ $a^w \equiv v \cdot y^b \pmod n$

Aufgabe: Man führe das abstrakte ZK-Protokoll das bzgl. $(f, o, \bullet)$ angegeben wurde, konkret durch mit der Quadratfunktion modulo einer Zahl $n = p \cdot q$. (Das Ergebnis ist das sog. Fiat-Shamir ZK-Protokoll).

Antwort: Initial wählt der Prover 2 große Primzahlen p, q, berechnet $n = p \cdot q$ und wählt x zufällig $\in \mathbb{Z}_n^*$; berechnet $y = x^2 \bmod n$.

Öffentlicher Schlüssel: n, y

Geheimer Schlüssel: x, p, q

(wobei p, q nicht weiter benötigt werden)

Die o und die $\bullet$-Operation sind jedesmal die Multiplikation modulo n.

Hier ist das Protokoll:

P	V

Wählt $u \in \mathbb{Z}_n^*$ zufällig.

Berechnet $v = u^2 \bmod n$

v →

← Wählt $b \in \{0,1\}$ zufällig

b ←

$$z := \begin{cases} u & , b=0 \\ u \cdot x & , b=1 \end{cases}$$

(kurz: $z = u \cdot x^b$)

z →

Überprüft:

$z^2 \equiv v \pmod n$, $b=0$

$z^2 \equiv v \cdot y \pmod n$, $b=1$

(kurz: $z^2 \equiv v \cdot y^b$)

Bemerkungen zu „Zero Knowledge"

In der Literatur oft noch verschärftere Definition: Es muss für jeden (auch sich nicht ans Protokoll haltenden) Verifier $\tilde{V}$ einen Simulator $S = S(\tilde{V})$ geben, der Protokolle statistisch exakt „nachempfinden" kann (genannt „perfect zero knowledge").

Des Weiteren gibt es abgeschwächte Definitionen von zero knowledge:

1.) „statistical zero knowledge": Die W'keits-Verteilung der vom Simulator ausgegebenen Protokolle muss nicht exakt, aber bis auch ein kleines $\varepsilon > 0$, mit der W'keitsverteilung beim echten Protokoll übereinstimmen.

2.) „computational zero knowledge": die beiden W'keitsverteilungen können durchaus voneinander abweichen, sind aber durch einen polynomialen Algorithmus nicht voneinander unterscheidbar.

In der Sprache der Statistik:

Die Hypothese H_0, dass die beiden W'keits-Verteilungen (von Simulator und von richtigem Protokoll) identisch sind, lässt sich durch keinen statistischen Test, sofern dieser in Polynomialzeit arbeitet, widerlegen.

3.) Es gibt Protokolle (und entsprechende Definitionen) wie „witness hiding protocol", die einen ähnlichen Charakter und ähnliche Anwendungsszenarien haben wie Zero Knowledge Protokolle, die aber anderen Definitionen unterliegen.

Weitere Bemerkungen: Perfekte Zero knowledge-Protokolle gelingen für Situationen, in denen das Geheimnis des Provers auf Graph Isomorphie bzw. bestimmten Einwegfunktionen wie

$x \longmapsto x^2 \bmod n$ (wobei $n = p \cdot q$) beruhen.

Graph Isomorphie liegt in NP, ist aber wohl nicht NP-vollständig.
Man kann sich fragen, ob man Zero Knowledge-Protokolle für alle Probleme in NP, insbesondere die NP-vollständigen, konstruieren kann.

Zum Beispiel: Der Prover konstruiert sich einen großen Graphen G mit einem ihm bekannten Hamiltonkreis H. Öffentlich: G; Geheim: H.
Lässt sich damit ein Zero Knowledge Protokoll angeben?

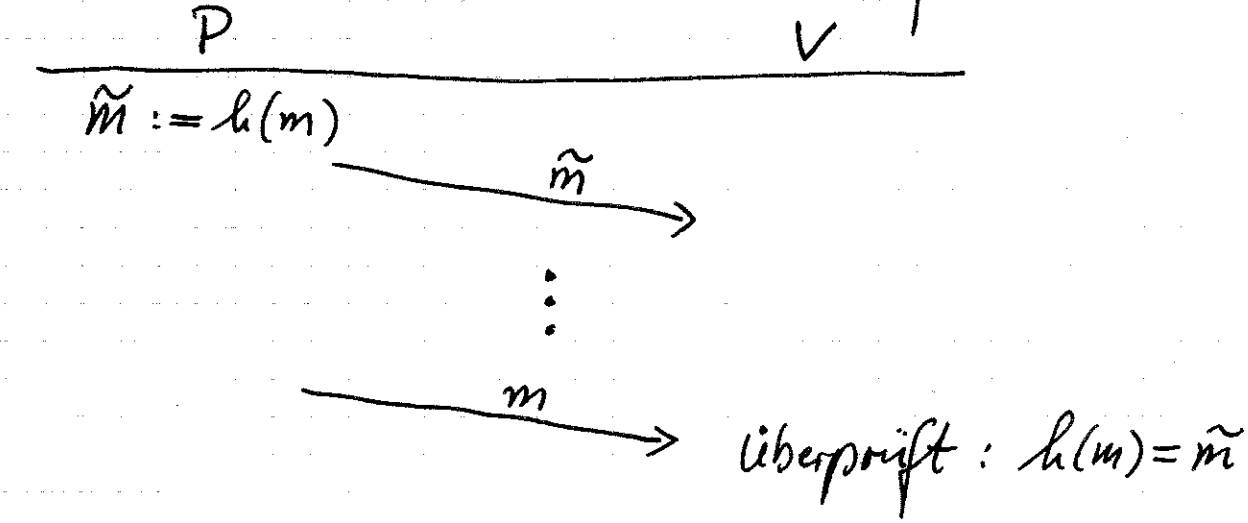
Antwort: Ja, aber nur im Sinne von computational zero knowledge (und zwar geht das im Prinzip für alle NP-vollständigen Probleme).

Ferner lässt sich zeigen:
Unter der Annahme, dass „die Polynomialzeit-Hierarchie nicht kollabiert" (eine stärkere Annahme als „$P \neq NP$") folgt, dass kein NP-vollständiges Problem ein perfektes Zero Knowledge-Protokoll besitzen kann.

Die erste Nachricht bei einem Zero Knowledge-Protokoll heißt Commitment.
Dahinter steckt allgemeiner das Konzept eines Commitment Schemes (oder -Protokolls).
P schickt eine verschlüsselte Nachricht $\tilde{m}$ an V. Zu einem späteren Zeitpunkt schickt er den betreffenden Schlüssel an V, so dass sich V vom Inhalt m überzeugen kann.
Wichtig ist hierbei, dass V nach der ersten Nachricht die (zwar verschlüsselte) Nachricht $\tilde{m}$ „in der Hand" hat und somit P diese nicht mehr ändern kann. Die Methode der Ver- und Entschlüsselung von m muss allerdings so sein, dass es für P keine andere Möglichkeit gibt, als eben m beim Entschlüsseln zu offenbaren. Es darf nicht möglich sein, durch Senden eines „falschen" Schlüssels eine andere Nachricht m' zu entschlüsseln.

Es gibt verschiedene Möglichkeiten, solch ein Commitment Scheme zu realisieren.

1.) Mit einer kollisionsresistenten Hashfunktion h:



2.) Mittels Modulare Exponentiation / Diskreter Logarithmus: