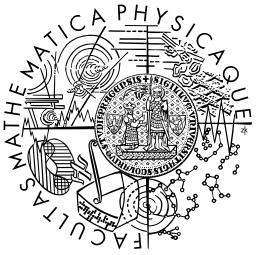


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

**BAKALÁŘSKÁ PRÁCE**

Tomáš Krňák

# Moderní aplikace zero-knowledge protokolů

Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: Mgr. Pavel Hubáček, Ph.D.
Studijní program: Matematika (B1101)
Studijní obor: MMIT (1103R041)

Praha 2019

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Děkuji svému vedoucímu Mgr. Pavlu Hubáčkovi, Ph.D. za jeho trpělivost, rady a odbočky ke kryptografickým zajímavostem. Děkuji své přítelkyni Míše Volfové za podporu a pomoc s korekturou při psaní této práce.

Název práce: Moderní aplikace zero-knowledge protokolů

Autor: Tomáš Krňák

ústav: Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: Mgr. Pavel Hubáček, Ph.D., Informatický ústav Univerzity Karlovy

Abstrakt: zk-SNARK je kryptografický protokol, který umožňuje libovolný výpočet přeměnit na krátký efektivně ověřitelný argument jeho správnosti. Navíc tento protokol umožnuje dokazovateli přesně rozhodnout, která část vstupů výpočtu bude veřejná a která zůstane skrytá. Cílem této práce je seznámit čtenáře s vlastnostmi, konstrukcí a aplikacemi moderních zk-SNARKů. V konstrukční části popisujeme konstrukci založenou na lineárním PCP a Paillierově kryptosystému. V aplikační části vysvětlujeme princip anonymní kryptoměny Zcash a přicházíme s zcela novým uplatněním zk-SNARKs v reputačních systémech.

Klíčová slova: zk-SNARK zero-knowledge proofs zerocash blockchain

Title: Modern applications of zero-knowledge protocols

Author: Tomáš Krňák

institute: Computer Science Institute of Charles University

Supervisor: Mgr. Pavel Hubáček, Ph.D., Computer Science Institute of Charles University

Abstract: zk-SNARK is a cryptographic protocol, which enables transformation of an arbitrary computation into short effectively verifiable argument of correctness of this computation. Further more, it enables a prover to decide exactly, which inputs of the computation will be public and which inputs will stay private. The goal of this work is to present features, construction and applications of modern zk-SNARKs. In a construction part of this work we describe construction based on linear PCP and Paillier cryptosystem. In an application part we explain principles of anonymous cryptocurrency Zcash and we describe a completely new application of zk-SNARKs in networks of trust.

Keywords: zk-SNARK zero-knowledge proofs zerocash blockchain

# Obsah

# Seznam použitého značení a zkratek

| | |
|---|---|
| PPT | pravděpodobnostní polynomiální Turingův stroj |
| DPT | deterministický polynomiální Turingův stroj |
| QAP | Quadratic Arithmetic Program |
| PQAP | Polynomial Quadratic Arithmetic Program |
| PCP | Linear Probabilisticly Checkable Proof |
| LPCP | Linear Probabilisticly Checkable Proof |
| SNARG | Succinct Non-interarctive ARGument |
| SNARK | Succinct Non-interarctive ARgument of Knowledge |
| zk-SNARK | zero-knowledge SNARK |
| HVPZK | Honest-Verifier Perfect Zero-Knowledge |
| dvSNARK | designated-verifier SNARK |
| pvSNARK | publicly-verifiable SNARK |
| $\circ$ | Hadamardův součin, čili násobení vektorů po složkách |
| $V(G)$ | vrcholy hrafu $G$ |
| $E(G)$ | hrany grafu $G$ |
| $x^\ell$ | $\ell$-složkový řádkový vektor $(x,\ldots,x)$ |
| $\widehat{x}$ | $\widehat{x} = \mathtt{Enc}_{\mathtt{pk}}(x)$ |
| $\lvert x\rvert$ | délka $x$ |
| $\mathbb{P}$ | množina všech prvočísel |
| $[n]$ | je množina čícel $\{1,2,\ldots,n\}$ |
| $A^B$ | stroj $A$, který používa stroj $B$ jako podprogram |
| $x\|\|y$ | zřetězení slov $x$ a $y$ |

# Úvod

zk-SNARK je kryptografický protokol, který umožnuje k nějakému algoritmu a vstupu vytvořit krátký efektivně ověřitelný argument, že algotritmus na danném vstupu doběhl s daným výstupem. Díky dlouholetému výzkumu v oblasti ověřitelného výpočtu se od roku 1990, kdy se objevují první čistě teoretické konstrukce PCP (Probabilisticly Checkable Proof), dostáváme do roku 2014, kdy dochází k prvnímu reálnému použití zk-SNARK v oblasti anonymních kryptoměn [1].

Klíčovou vlastností zk-SNARK je, že umožňují vstupy ověřitelného výpočtu rozdělit na veřejné (například dotaz na databázi) a soukromé (databáze). Díky zk-SNARK může server k odpovědi na klientův dotaz navíc přidat kryptografickou záruku (argument) toho, že se jedná o skutečná data z databáze. Zároveň má server kryptografickou záruku, že se klient z jeho odpovědi a argumentu nedozví o databázi nic navíc.

Síla zk-SNARKů spočívá v jejich univerzalitě. Lze pomocí nich ověřit doopravdy libovolný výpočet určený například aritmetickým obvodem, nebo Turingovým strojem. Daní za tuto vlastnost jsou jejich silné předpoklady a nutnost jejich inicializace důvěryhodnou autoritou.

## Struktura práce a použité zdroje

Práce je rozdělena do dvou část: Konstrukce a Aplikace zk-SNARK.

V kapitole 1.1 nadefinuje aritmetické obvody a vysvětlíme, proč jsou dobrým výpočetním modelem. Dále formálně zadefinujeme SNARG a jeho vlastnosi pomocí pojmů z dokazatelné bezpečnosti. Tyto vlastnosti zesilujeme, čímž dosahujeme doopravdy robustní definice zk-SNARK. Definici zk-SNARKu přebíráme z [2].

V kapitolách 1.2, 1.3 a 1.4 popisujeme základní kameny konstrukce a diskutujeme nutné kryptografické předpoklady použité k důkazům formální bezpečnosti výsledného zk-SNARKu. Konstrukci LPCP přebíráme z [3].

V kapitolách 1.5 a 1.6 se věnujeme skládání kryptografických schémat z předchozích kapitol a důkazům, jejich bezpečnosti. Výslednou konstrukci přebíráme z [2] spolu s nástinem důkazu, který se snažíme více rozvést.

V kapitolách 2.1, 2.2 a 2.3 rozebereme anonymitu bitcoinových transakcí a protokol Zerocash [1], který zaručuje absolutní anonymitu transakcí pomocí zk-SNARK. Čerpáme přitom především z článků o kryptoměnách Bitcoin [4] , Monero [5] a Zcash [1].

V poslední kapitole Síť důvěry 2.4 budujeme model sítě nepřímé důvěry. Pro tento model poté navrhujeme nové schéma elektronického podpisu založené na zk-SNARK, které zobecňuje ring signatures [6].

# 1. Konstrukce zk-SNARK

*'For the Snark's a peculiar creature, that won't*
*Be caught in a commonplace way.*
*Do all that you know, and try all that you don't:*
*Not a chance must be wasted to-day!*

— Lewis Carol, *The Hunting of the Snark*

Nejprve uvedeme dva scénáře jako motivaci:

**Scénář 1:** Alice udělala nějaký složitý dlouhý výpočet a nyní chce jeho výsledek sdílet s Bobem (Uvažme např. program, pomocí kterého George Gonthier formálně dokázal [7] větu o čtyřech barvách, nebo libovolnou optimalizační úlohu.). Bob by ale chtěl nějakou záruku, že Alice výpočet provedla správně. Co kdyby ho Alice mohla přesvědčit o správnosti svého výsledku i bez toho, aby Bob musel celý její výpočet opakovat?

**Scénář 2:** Alice zná řešení nějakého problému a chce o tom přesvědčit Boba. Může to udělat i bez toho, aby Bobovi prozradila cokoliv o svém řešení?

Odpověď na obě otázky je kladná. Existují metody, jak záznam průběhu nějakého výpočtu přeměnit na snadně ověřitelný „argument" jeho správnosti. Nepoužíváme slovo „důkaz", protože tento argument bude jen pravděpodobnostní zárukou založenou na nějakém kryptografickém předpokladu. To znamená, že musíme předpokládat, že dokazovatel je výpočetně polynomiálně omezený [8]. Tím se SNARKy liší např. od interactive proofs.

Rovněž existují metody, jak argumentovat pro správnost nějakého výpočtu bez vyzrazení jeho průběhu.

Nakonec chceme omezit interakci mezi jednotlivými stranami na minimum a délka argumentu by měla být mnohem menší, než délka samotného výpočtu. Kryptografické schéma splňující všechny tyto vlastnosti nazýváme zk-SNARK (zero-knowledge Succint Non-iteractive ARguments of Knowledge). Následující tabulka je neformálním shrnutím.

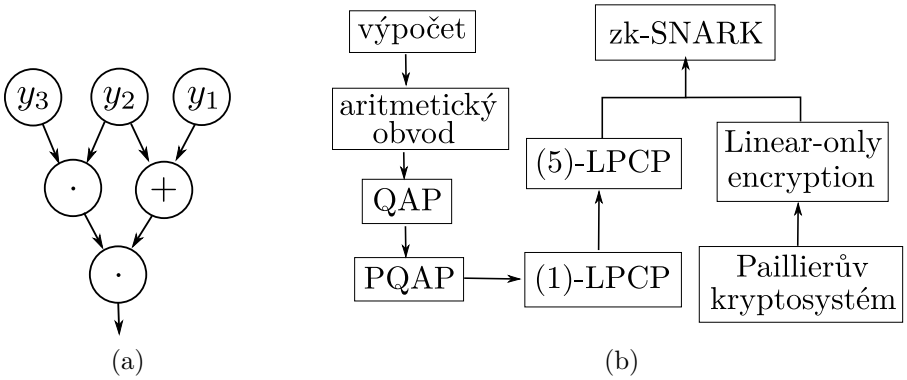
| | |
|---|---|
| zero-knowledge | průběh výpočtu zůstává utajen |
| Succint | výsledný argument je krátký a snadno ověřitelný |
| Non-iteractive | k ověření argumentu není potřeba interakce s jeho autorem |
| ARguments | výsledný důkaz je jen pravděpodobnostní záruka |
| of Knowledge | pokud někdo vytvořil argument, pak musí znát opovídající průběh výpočtu |

Na obrázku 1.1a je mapa kontrukce, kterou popisujeme v této části.

## 1.1 Definice

Ještě předtím, než zadefinujeme zk-SNARK musíme zvolit nějaký výpočetní model. Vystačíme si s aritmetickými obvody.

**Definice 1.** ***Aritmetický obvod*** *nad tělesem $\mathbb{F}$ s proměnnými $y_1, \ldots, y_\ell$ je orientovaný acyklický graf $G$ s vrcholy označenými následujícím způsobem. Vrcholy s*




Obrázek 1.1: (a) aritmetický obvod (b) mapa konsrukce zk-SNARK

*vstupním stupněm 0 jsou označeny jedním z $y_i$ nebo prvkem tělesa $\mathbb{F}$. Ostatní vrcholy jsou označeny $+$ nebo $\cdot$. Jeden vrchol je navíc označen jako výstupní. Navíc požadujeme, aby vstupní stupeň každého vrcholu byl právě buď 0 nebo 2.*

*Vrcholy označené $+$ (resp. $\cdot$) nazýváme součtová (resp. součinová) hradla.*

Každý aritmetický obvod nám přirozeně dává funkci $C : \mathbb{F}^n \to \mathbb{F}$. Navíc nám dává, narozdíl od například polynomiální reprezentace funkce, možnost popsat, jak tuto funkci efektivně spočítat.

Rozdělíme proměnné obvodu $C$ na dvě skupiny s různými významy. $x_1, \ldots, x_n$ budou reprezentovat instanci nějakého problému a $w_1, \ldots, w_m$ řešení tohoto problému. Nyní můžeme zadefinovat relaci $\mathcal{R}_C := \{(x,w) \in \mathbb{F}^n \times \mathbb{F}^m \mid C(x,w) = 1\}$ a jazyk $\mathcal{L}_C := \{x \in \mathbb{F}^n \mid \exists w \in \mathbb{F}^m : C(x,w) = 1\}$. $(x,w) \in \mathcal{R}$ znamená, že $w$ je řešením problému $x$. $x \in \mathcal{L}_C$ znamená, že existuje nějaké řešení problému $x$.

Pro některé aplikace bychom chtěli obvodem $C(x',w')$ zachytit skutečnost, že $F(x) = y$ pro nějaká veřejná $(x,y)$. Existuje přirozená redukce tohoto problému na splnitelnost obvodu. Ať $\tilde{C}$ je obvod, který počítá $F$. Pak z $C(x' = (x,y), w' = ()) := 1 + y - \tilde{C}(x)$ je obvod, který vrací 1, právě když $\tilde{C}(x) = y$. Instance $x'$ je dvojice $(x,y)$ a řešení $w'$ je prázdné. Všimněme si, že

$$x' = (x,y) \in \mathcal{L}_C \leftrightarrow \tilde{C}(x) = y$$

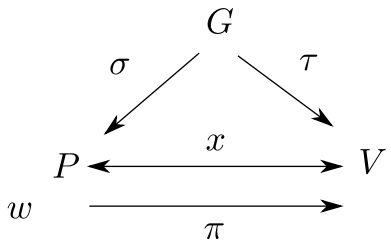
.

Pomocí aritmetického obvodu můžeme rovněž reprezentovat skutečnost, že nějaký stroj $M$ vrátí 1 na vstupu $x$ po nejvíce $T$ krocích. Stačí reprezentovat přechodovou funkci $M$ v jednom kroce pomocí aritmetického obvodu $\tilde{C}$ a následně tento obvod $T$ krát iterovat. Podobně jako v předchozím odstavci lze pro stroj $\tilde{M}$, který počítá funkci $F$, sestrojit stroj $M$, který vrátí 1 na $(x,y)$, právě když $F(x) = y$. Efektivním převodem programů z jazyce C na aritmetický obvod s nízkým počtem hradel je popsán v [9].

K definicím bezpečnosti budeme používat zanedbatelné funkce. To jsou funkce, které s rostoucím vstupem klesají k nule rychleji, než libovolný polynom.

**Definice 2.** *Funkce $f : \mathbb{N} \to \mathbb{R}$ je* zanedbatelná*, pokud pro každý polynom $p(x) \in \mathbb{R}[x]$*

$$\lim_{n\to\infty} f(n)p(n) = 0$$




Obrázek 1.2: protokol SNARG

**Značení.** *Výskyt symbolu* $\mathrm{negl}(\lambda)$ *v predikátu* $P(\mathrm{negl}(\lambda))$ *překládáme jako: Existuje zanedbatelná funkce* $f$ *taková, že* $\forall\lambda \in \mathbb{N} : P(f(\lambda))$.

K zápisu bezpečnostních experimentů budeme používat tuto notaci.

**Značení.** *Pro množinu* $X$ $x \leftarrow X$ *znamená, že* $x$ *je zvoleno náhodně z rovnoměrného rozdělení na* $X$.

*Mějme pravděpodobnostní algoritmus* $A$. $A(x)$ *je distribuce výstupů* $A$ *na vstupu* $x$. $y \leftarrow A(x)$ *znamená, že* $y$ *bylo zvoleno náhodně z distribuce* $A(x)$.

**Definice 3.** *Mějme aritmetický obvod* $C : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}$. *Trojice algoritmů* $(G, P, V)$ *je* SNARG (Succinct Non-interactive ARGument) *pro relaci* $\mathcal{R}_C$, *pokud má následující syntax a splňuje tyto podmínky:*

| *algoritmus* | *vstupy* | *popis vstupu* | *výstupy* | *popis výstupů* |
|---|---|---|---|---|
| *G (generátor)* | $1^\lambda$ | *bezp. parametr* | $\sigma$<br>$\tau$ | *dokazovací klíč*<br>*ověřovací klíč* |
| *P (dokazovatel)* | $\sigma$<br>$x$<br>$w$ | *dokazovací klíč*<br>*instance*<br>*svědek* | $\pi$ | *argument* |
| *V (ověřovatel)* | $\tau$<br>$x$<br>$\pi$ | *ověřovací klíč*<br>*instance*<br>*argument* | $b$ | $b = 1 \leftrightarrow V$ *přijmul* $\pi$ |

**Completeness:** *Pro každé* $(x,w) \in \mathcal{R}_C$

$$\Pr\left[V(\tau,x,\pi) = 1 \;\middle|\; \begin{array}{c}(\sigma,\tau) \leftarrow G(1^\lambda)\\ \pi \leftarrow P(\sigma,x,w)\end{array}\right] = 1.$$

**Soundness:** *Lze nadefinovat více způsoby. Rozlišujeme adaptive a non-adaptive soundness a podle toho také výsledný SNARG.*

- ***non-adaptive*** *Pro každého PPT dokazovatele* $P^*$ *a každé* $x$ *pro které* $\nexists w : (x,w) \in \mathcal{R}_C$,

$$\Pr\left[V(\tau,x,\pi) = 1 \;\middle|\; \begin{array}{c}(\sigma,\tau) \leftarrow G(1^\lambda)\\ \pi \leftarrow P(\sigma,x)\end{array}\right] \leq \mathrm{negl}(\lambda).$$

- ***adaptive*** *Pro každého PPT dokazovatele* $P^*$

$$\Pr\left[\begin{array}{c}V(\tau,x,\pi) = 1\\ \wedge\\ \nexists w : (x,w) \in \mathcal{R}\end{array} \;\middle|\; \begin{array}{c}(\sigma,\tau) \leftarrow G(1^\lambda)\\ (x,\pi) \leftarrow P(\sigma)\end{array}\right] \leq \mathrm{negl}(\lambda).$$

**Efektivita:** *Pro každý dostatečně velký bezpečnostní pararmetr $\lambda \in \mathbb{N}$ a každou instanci $x$*

- *generátor $G$ skončí v čase $poly(\lambda + \log|C|)$*
- *dokazovatel $P$ skončí v čase $poly(\lambda + |C|)$*
- *ověřovatel $V$ skončí v čase $poly(\lambda + |x| + \log|C|)$ a*
- *poctivě vygenerovaný důkaz má délku $poly(\lambda + \log|C|)$.*

Adaptive soundness je silnější podmínka než non-adaptive verze, protože si $P^*$ dokonce může zvolit, pro jaké $x$ se bude snažit podvádět.

Definici soundness zesílíme ještě více na tzv. knowledge soundness. V některých aplikacích je totiž třeba dokázat, nejenže $x \in \mathcal{L}_C$, ale i že Alice doopravdy „zná“ $w$ takové, že $(x,w) \in \mathcal{R}_C$. Mějme například $\mathcal{R} = \{(x,w)|g^w = x\}$, kde $g$ je generátor nějaké grupy $\mathbb{G}$, ve které je diskrétní logaritmus těžký, $x \in \mathbb{G}$ a $w \in \mathbb{Z}_{|\mathbb{G}|}$. Pro tuto relaci $\mathcal{L} := \{x|\exists w : (x,w) \in \mathcal{R}\}$. Pak Alice a Bob oba vědí, že $\forall x \in \mathbb{G} : x \in \mathcal{L}$ a proto je zbytečné aby to Alice Bobovi dokazovala. Vědět, že řešení existuje a „znát“ řešení je rozdíl. Jak ale formálně nadefinovat, že někdo něco „zná“?

Elegantní přístup popsaný v [10] spočívá v tom, že ke každému $P^*$ umíte sestavit jiný PPT extraktor $E^{P^*}$, který používá $P^*$ jako černou skřínku a který, až na zanedbatelný počet výjimek, umí $w$ dopočítat. Jednodušeji: pokud $P^*$ zná $w$, tak ho $E^{P^*}$ dostane (vyextrahuje) ven.

**Definice 4.** *Trojice algoritmů $(G, P, V)$ je* SNARK (Succinct Non-interactive ARgument of Knowledge) *pro relaci $\mathcal{R}_C$, pokud se jedná o SNARG pro $\mathcal{R}_C$, kde a adaptive soundness je nahrazena silnější podmínkou:*

- ***adaptive proof of knowledge**: Pro každého PPT dokazovatele $P^*$ existuje PPT extraktor $E$ takový, že pro každý dodatečný vstup $z \in \{0,1\}^{poly(\lambda)}$*

$$\Pr\left[\begin{array}{c|c} V(\tau,x,\pi) = 1 & (\sigma,\tau) \leftarrow G(1^\lambda) \\ \wedge & (x,\pi) \leftarrow P^*(z,\sigma) \\ (x,w) \notin \mathcal{R} & w \leftarrow E^{P^*}(z,\sigma) \end{array}\right] \leq \text{negl}(\lambda).$$

Nakonec chceme, aby se Bob nic nedozvěděl o $w$. Knowledge soundness říká, že někdo něco ví. Jak ale nadefinovat, že někdo něco neví? Uvědomme si přitom, že nelze jen znegovat knowledge soundness. Nejdříve uvedeme příklad modelu, ve kterém lze zero-knowledge nadefinovat jednodušeji. Definice zero-knowledge pro SNARG potom bude z velké části analogická, ale zato mnohem robustnější a komplikovanější.

Interaktivní důkazový systém $(P,V)$ je model, ve kterém se spolu dva Turigovy stroje komunikují pomocí sdílené pásky, přičemž cílem $P$ je opět přesvědčit $V$ o tom, že daná instance $x$ náleží do jazyku $\mathcal{L}$. Nadefinujeme
$\text{View}\,[P(x,w) \leftrightarrow V(x)]$ jako záznam komunikace mezi $P$ a $V$ na vstupu $(x,w) \in \mathcal{R}$. $\text{View}\,[P(x,w) \leftrightarrow V(x)]$ je distribuce, protože $P$ a $V$ jsou pravděpodobnostní stroje. Interaktivní důkazový protokol $(P,V)$ je perfektní zero-knowledge, pokud pro každý PPT $V^*$ existuje PPT simulátor $S$ t.ž.

$$\forall(x,w) \in \mathcal{R}\ \forall z \in \{0,1\}^* : S(x,z) = \text{View}\,[P(x,w) \leftrightarrow V^*(x,z)]$$

Jinými slovy $S$ dokáže simulovat komunikaci mezi $P$ a $V$ i bez znalosti $w$. To ale znamená, že $\text{View}\,[P(x,w) \leftrightarrow V(x)]$ nezávisí na $w$, a proto se o něm stroj $V$ nedozví v průběhu protokolu nic nového. Navíc $V$ se o $w$ nic nového nedozví, ani pokud protokol bude opakovat a ukládat si jeho průběhy na vedlejší pásku $z$.

Tuto definici dále zeslabujeme. V mnoha případech nám stačí, když budou distribuce $S(x)$ a $\text{View}\,[P(x,w) \leftrightarrow V(x)]$ jen statisticky či výpočetně nerozlišitelné. Pro zk-SNARG budeme požadovat pouze výpočetní nerozlišitelnost.

**Definice 5.** *Dvě posloupnosti náhodných veličin $(X_\lambda)_{\lambda\in\mathbb{N}}$ a $(Y_\lambda)_{\lambda\in\mathbb{N}}$ jsou výpočetně nerozlišitelné, pokud pro každý PPT rozlišovač $D$*

$$\left| \Pr_{x\leftarrow X_\lambda}[D(x,1^\lambda)=1] - \Pr_{x\leftarrow Y_\lambda}[D(x,1^\lambda)=1] \right| < \text{negl}(\lambda).$$

Pro větší robustnost definice 5 a přiblížení se modelem realitě navíc algoritmu $D$ povolíme zvolit si $x$ a $w$, pro které se bude snažit rozlišit důkazy $\pi$ a $\pi'$.

**Definice 6.** *Trojice algoritmů $(G,P,V)$ je* perfect zero-knowledge *SNARK pro relaci $\mathcal{R}_C$ pokud je to SNARK pro relaci $\mathcal{R}_C$ a navíc existuje PPT simulátor $S = (S_1,S_2)$ splňující následující:*

- *Výstupem $S_2$ je buď $\pi'$, nebo $\perp$.*
- *Pro každou instanci $x$ a každý dotatečný vstup $z \in \{0,1\}^{poly(\lambda)}$*

$$\Pr\left[S_2(x,\sigma,\tau,\texttt{trap},z) = \perp \;\middle|\; (\sigma,\tau,\texttt{trap}) \leftarrow S_1(1^\lambda)\right] < \text{negl}(\lambda)$$

- *Pro každé $\lambda \in \mathbb{N}$, každou instanci $x$ a každý dotatečný vstup $z \in \{0,1\}^{poly(\lambda)}$ označme $\tilde{S}_2$ podmíněnou distribuci*

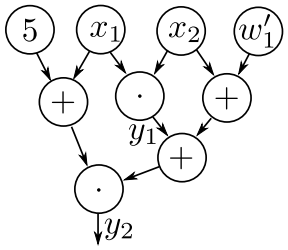
$$\Pr\left(S_2(x,\sigma,\tau,\texttt{trap},z) = \pi' \middle| \begin{matrix}(\sigma,\tau,\texttt{trap}) \leftarrow S_1(1^\lambda)\\ S_2(x,\sigma,\tau,\texttt{trap},z) \neq \perp\end{matrix}\right).$$

*Pak pro každý algoritmus $D$ a každý dodatečný vstup $z \in \{0,1\}^{poly(\lambda)}$*

$$\begin{aligned}&\Pr\left[\begin{matrix}(x,w)\in\mathcal{R}_C\\ D(\pi,\sigma,\tau,z)=1\end{matrix}\;\middle|\;\begin{matrix}(\sigma,\tau)\leftarrow G(1^\lambda)\\ (x,w)\leftarrow D(\sigma,\tau,z)\\ \pi\leftarrow P(\tau,x,w)\end{matrix}\right]\\ =&\Pr\left[\begin{matrix}(x,w)\in\mathcal{R}_C\\ D(\pi',\sigma,\tau,z)=1\end{matrix}\;\middle|\;\begin{matrix}(\sigma,\tau,\texttt{trap})\leftarrow S_1(1^\lambda)\\ (x,w)\leftarrow D(\sigma,\tau,z)\\ \pi'\leftarrow S_2(x,\sigma,\tau,\texttt{trap},z)\end{matrix}\right].\end{aligned}$$

Tato definice si zasloží podrobnnější rozbor. Nejprve si všimněme, že hodnoty $(\pi,\sigma,\tau)$ jsou přepis komunikace během SNARG protokolu podobně, jako u interaktivního důkazového systému $\text{View}\,[P(x,w) \leftrightarrow V(x)]$. V první pravděpodobnostní závorce $D$ dostává přepis protokolu simulovaný dvojicí $(G,P)$. V druhé pravděpodobnostní závorce $D$ dostane přepis protokolu simulovaný pomocí $S$.

Důležitý rozdíl mezi $P$ a $S_2$ je, že $S_2$ nedostává $w$ jako vstup. $S$ dokáže vygenerovat nový validní důkaz pro instanci $x$ i bez znalosti svědka $w$ díky tomu, že může vygenerovat parametry SNARGu. Při jejich generování si totiž zapamatuje nějakou informaci `trap`, která mu později pomůže falzifikovat důkaz $\pi'$.




Obrázek 1.3: Aritmetický obvod 2

Přímo z definice zk-SNARGu tedy vyplývá, že ten, kdo vygeneroval parametry $\sigma$ a $\tau$, může falzifikovat důkazy.

$S$ může v zanedbatelné počtu případů selhat a vrátit symbol $\perp$. Díky tomuto zeslabení požadavků na $S$ můžeme u výsledného zk-SNARKu požadovat perfect zero-knowledge. To znamená, že distribuce přepisu průběhu normálního protokolu a simulace jsou identické a tím pádem naprosto nerozlišitelné i výpočetně neomezeným algoritmem.

Nakonec dělíme SNARKy na dvě kategorie. Dedicated-Verifier SNARK (dvSNARK) předpokládá jen jednoho ověřovatele, zatímco Publicly-Verifiable (pvSNARK) umožnuje k dané instanci $x$ vytvořit jeden důkaz, který si může ověřit více ověřovatelů. To formálně znamená, že zatímco pvSNARK bude splňovat všechny vlastnosti i potom, co dokazatel $P$ získá přístup k ověřovacímu klíči $\tau$, u dvSNARKu bychom takovou změnou ztratili soundness.

## 1.2 Kvadratické aritmetické programy

V této kapitole uvedeme jiný způsob, jak lze vyjádřit skutečnost $x \in \mathcal{L}_C$. Mějme aritmetický obvod $C$ jako na obrázku 1.3. Výstupu každého jeho součinového hradla přiřadíme jednu z proměnných $\boldsymbol{y} = (y_1,y_2)$. Pro instanci $\boldsymbol{x} = (x_1,x_2)$ a svědka $\boldsymbol{w'} = (w'_1)$ definujeme vektor

$$\boldsymbol{w} := (1|\boldsymbol{x}|\boldsymbol{w'}|\boldsymbol{y}) = (1,x_1,x_2,w'_1,y_1,y_2).$$

Hodnoty $\boldsymbol{y}$ jsou jednoznačně určené hodnotami $\boldsymbol{x}$ a $\boldsymbol{w'}$. Pro náš obvod platí rovnice

$$y_1 = x_1 \cdot x_2 \quad \text{a} \quad y_2 = (5 + x_1) \cdot (y_1 + x_2 + w'_1).$$

Tyto dvě rovnice můžeme zapsat také takto:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ w'_1 \\ y_1 \\ y_2 \end{pmatrix} \circ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ w'_1 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ w'_1 \\ y_1 \\ y_2 \end{pmatrix}$$

kde $\boldsymbol{u} \circ \boldsymbol{v}$ značí násobení vektorů po složkách. Tím jsme maticově zachytili skutečnost, že $y_1$,$y_2$ odpovádají hodnotám součinových hradel $C$ na vstupu $(\boldsymbol{x},\boldsymbol{w'})$. Tento postup lze zobecnit.

Ať $C$ má $n > 0$ součinových hradel a pro každé součinové hradlo $h \in C$ jsou vektory $\boldsymbol{a}_h$,$\boldsymbol{b}_h$,$\boldsymbol{c}_h \in \mathbb{F}^{1\times m}$ takové, že rovnice $\boldsymbol{a}_h\boldsymbol{w} \cdot \boldsymbol{b}_h\boldsymbol{w} = \boldsymbol{c}_h\boldsymbol{w}$ odpovídá rovnici výpočtu hradla $h$. Definujeme matice $A$, $B$ a $C$ typu $m \times n$ vztahy

$$A := \begin{pmatrix} \boldsymbol{a}_{h_1} \\ \hline \vdots \\ \hline \boldsymbol{a}_{h_n} \end{pmatrix}, B := \begin{pmatrix} \boldsymbol{b}_{h_1} \\ \hline \vdots \\ \hline \boldsymbol{b}_{h_n} \end{pmatrix} \text{ a } C := \begin{pmatrix} \boldsymbol{c}_{h_1} \\ \hline \vdots \\ \hline \boldsymbol{c}_{h_n} \end{pmatrix}.$$

Tím dostáváme, že $\boldsymbol{y}$ odpovídá hodnotám hradel obvodu $C$ na vstupu $(\boldsymbol{x},\boldsymbol{w}')$, právě když $A\boldsymbol{w} \circ B\boldsymbol{w} = C\boldsymbol{w}$. Když do těchto matic přidáme ještě jeden řádek pro rovnici výstupu obvodu $y_n = 1$, dostáváme vztah

$$\boldsymbol{x} \in \mathcal{L}_C \leftrightarrow \exists \boldsymbol{w} : (w_0,w_1,\ldots,w_k) = (1,x_1,\ldots,x_k) \wedge A\boldsymbol{w} \circ B\boldsymbol{w} = C\boldsymbol{w}.$$

Jednoduchost tohoto zápisu motivuje následující definici.

**Definice 7.** *Mějme celá čísla $m$, $n$, $k$ tž. $n - 1 \geq k$.* Kvadratický aritmetický program (QAP) *nad tělesem $\mathbb{F}$ dimenze $(m,n,k)$ je trojice matic $(A, B, C)$ typu $m \times n$ nad $\mathbb{F}$.*

*Řekneme, že QAP $(A, B, C)$ přijímá $\boldsymbol{x} \in \mathbb{F}^k$, právě když existuje vektor $\boldsymbol{w} \in \mathbb{F}^n$, pro který platí, že $(w_0,w_1,\ldots,w_k) = (1,x_1,\ldots,x_k)$ a zároveň $A\boldsymbol{w} \circ B\boldsymbol{w} = C\boldsymbol{w}$.*

Navíc jsme právě ukázali, že pro každý obvod $C(\boldsymbol{x},\boldsymbol{w}')$, který definuje jazyk $\mathcal{L}_C$, umíme zkonstruovat kvadratický aritmetický program $P_C = (A,B,C)$ takový, že $P_C$ přijme $\boldsymbol{x}$ právě když $\boldsymbol{x} \in \mathcal{L}_C$.

Kvadratický aritmetický program je tedy maticově zapsaná soustava kvadratických rovnic s parametrem $x$, jejíž řešením je $w$. To se nám bude hodit ze dvou důvodů. Nejprve v kapitole 1.3 ukážeme, že tuto soustavu dokážeme redukovat na dvě kvadratické rovnice a následně v kapitole 1.4 ukážeme, že existují šifrování, která umožnují homomorfně ověřovat nuly kvadratických polynomů.

Převodu QAP na dvě kvadratické rovnice předchází ještě jeden krok, ve kterém matematicky nepěknou operaci násobení vektorů po složkách převedeme na násobení polynomů. Tím vznikne Polynomiální QAP.

**Definice 8.** *Mějme celá čísla $m$, $n$, $k$ tž. $n - 1 \geq k$.* Polynomiální kvadratický aritmetický program (PQAP) *nad tělesem $\mathbb{F}$ dimenze $(m,n,k)$ je čtveřice $(\boldsymbol{A}(x),\boldsymbol{B}(x),\boldsymbol{C}(x), Z(x))$, ve které $\boldsymbol{A}(x)$,$\boldsymbol{B}(x)$ a $\boldsymbol{C}(x)$ jsou řádkové vektory polynomů stupně nejvýše $m$ z $(\mathbb{F}[x])^n$ a $Z(x)$ je polynom stupně nejvýše $m$ z $F[x]$.*

*Řekneme, že PQAP $(\boldsymbol{A}(x),\boldsymbol{B}(x),\boldsymbol{C}(x), Z(x))$ přijímá $\boldsymbol{x} \in \mathbb{F}^k$, právě když existuje vektor $\boldsymbol{w} \in \mathbb{F}^n$, pro který platí, že $(w_0,w_1,\ldots,w_k) = (1,x_1,\ldots,x_k)$ a zároveň*

$$Z(x) | \boldsymbol{A}(x)\boldsymbol{w} \cdot \boldsymbol{B}(x)\boldsymbol{w} - \boldsymbol{C}(x)\boldsymbol{w}.$$

Na závěr této kapitoly ukážeme, že každý QAP lze převést na PQAP, který definuje stejný jazyk.

**Konstrukce 1.** *(PQAP z QAPu) Mějme QAP* $(\tilde{A},\tilde{B},\tilde{C})$ *dimenze* $(m,n,k)$. *Zvolíme podmnožinu* $S \subset \mathbb{F}$ *velikosti* $m$ *a označíme její prvky* $s_1,\ldots,s_m$. *Definujeme polynom* $Z_S(x) := \prod_{s\in S}(x-s)$. *Dále najdeme pomocí interpolace polynomy*

$$\begin{aligned}\boldsymbol{A}(x) &= (A_1(x),\ldots,A_n(x))\ ,\\ \boldsymbol{B}(x) &= (B_1(x),\ldots,B_n(x))\ a\\ \boldsymbol{C}(x) &= (C_1(x),\ldots,C_n(x))\end{aligned}$$

*takové, že*

$$\forall i \in [m]\ \forall j \in [n] : A_j(s_i) = \tilde{A}_{ij} \wedge B_j(s_i) = \tilde{B}_{ij} \wedge C_j(s_i) = \tilde{C}_{ij}.$$

*Tím získáme PQAP* $(\boldsymbol{A}(x),\boldsymbol{B}(x),\boldsymbol{C}(x), Z_S(x))$ *dimenze* $(m,n,k)$.

**Věta 1.** *Mějme PQAP* $(\boldsymbol{A}(x),\boldsymbol{B}(x),\boldsymbol{C}(x), Z_S(x))$ *získaný konstrukcí 1 z QAPu* $(\tilde{A},\tilde{B},\tilde{C})$. *Pak pro každé* $\boldsymbol{x} \in \mathbb{F}^k$, $(\tilde{A},\tilde{B},\tilde{C})$ *přijímá* $\boldsymbol{x}$ *právě když* $(\boldsymbol{A}(x),\boldsymbol{B}(x),\boldsymbol{C}(x), Z_S(x))$ *přijímá* $\boldsymbol{x}$.

*Důkaz.* Mějme $\boldsymbol{w} \in \mathbb{F}^n$. $(\tilde{A},\tilde{B},\tilde{C})$ přijímá $\boldsymbol{w}$, právě když

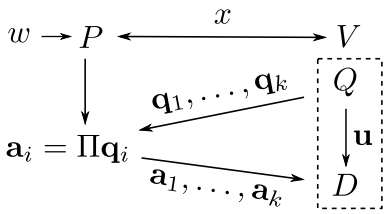
$$\begin{aligned}&\tilde{A}\boldsymbol{w} \circ \tilde{B}\boldsymbol{w} = \tilde{C}\boldsymbol{w}\\ \iff &\forall i \in [m] : \langle \tilde{A}_i,\boldsymbol{w}\rangle \cdot \langle \tilde{B}_i,\boldsymbol{w}\rangle = \langle \tilde{C}_i,\boldsymbol{w}\rangle\\ \iff &\forall i \in [m] : \sum_{j=1}^{n} \tilde{A}_{ij}\boldsymbol{w}_j \cdot \sum_{j=1}^{n} \tilde{B}_{ij}\boldsymbol{w}_j = \sum_{j=1}^{n} \tilde{C}_{ij}\boldsymbol{w}_j\\ \iff &\forall i \in [m] : \sum_{j=1}^{n} A_j(s_i)\boldsymbol{w}_j \cdot \sum_{j=1}^{n} B_j(s_i)_{ij}\boldsymbol{w}_j = \sum_{j=1}^{n} C_j(s_i)\boldsymbol{w}_j\\ \iff &\forall i \in [m] : \langle \boldsymbol{A}(s_i),\boldsymbol{w}\rangle \cdot \langle \boldsymbol{B}(s_i),\boldsymbol{w}\rangle = \langle \boldsymbol{C}(s_i),\boldsymbol{w}\rangle\\ \iff &\forall i \in [m] : (\langle \boldsymbol{A}(x),\boldsymbol{w}\rangle \cdot \langle \boldsymbol{B}(x),\boldsymbol{w}\rangle - \langle \boldsymbol{C}(x),\boldsymbol{w}\rangle)(s_i) = 0\\ \iff &\forall i \in [m] : (x-s_i) | \langle \boldsymbol{A}(x),\boldsymbol{w}\rangle \cdot \langle \boldsymbol{B}(x),\boldsymbol{w}\rangle - \langle \boldsymbol{C}(x),\boldsymbol{w}\rangle \qquad (1.1)\\ \iff &Z_S(x) | \langle \boldsymbol{A}(x),\boldsymbol{w}\rangle \cdot \langle \boldsymbol{B}(x),\boldsymbol{w}\rangle - \langle \boldsymbol{C}(x),\boldsymbol{w}\rangle\end{aligned}$$

Poslední ekvivalence platí, protože polynomy $(x-s_i)$ jsou ireducibilní a tedy po dvou nesoudělné.

□

## 1.3 LPCP

V předchozí kapitole jsme předvedli, jak lze problém splnění aritmetického obvodu převést na problém nalezení řešení soustavy kvadratických rovnic. V této kapitole popíšeme konstrukci LPCP (Linear Probabilisticly Checkable Proof) z [3], což je protokol, který Alici umožní vytvořit pro Boba krátký argument toho, že zná řešení této soustavy rovnic. LPCP je základním kamenem naší konstrukce zk-SNARKu. Sám o sobě je LPCP v praktických aplikacích ale nepoužitelný, protože jeho model je příliš silný.




Obrázek 1.4: protokol LPCP

V modelu LPCP si Alice (v roli dokazovatele) a Bob (v roli ověřovatele) vymění dvě zprávy v podobě vektorů nad nějakým konečným tělesem. Bob vygeneruje $k$ dotazů a pošle je naráz Alici. Alice mu poté pošle $k$ odpovědí. Prvním předpokladem je, že Alice generuje odpovědi pomocí nějaké lineární funkce. Druhý předpoklad je, že Alice používá k tvorbě všech odpovědí stejnou lineární funkci.

Oba tyto přepoklady zaručíme v následujích kapitolách tím, že dotazy „zabalíme“ do jednoho dlouhého dotazu a ten poté zašifrujeme šifrou s homomorními vlastnostmi. Prozatím jsou tyto dva předpoklady prostě jen vlastnosti modelu.

Následující definice LPCP je zobecněním definice LPCP z [2]. Zatímco v definici LPCP [2] je odpověď pouze jeden prvek tělesa, v naší definici je odpověd $m$-prvkový vektor nad tímto tělesem. Toto zobecnění nám umožní najednou zadefinovat LPCP i LIP (Linear Interactive Proof) z [2], což jsou dva velmi podobné modely protokolů. Pro délku dotazů $m = 1$ dostáváme přesně LPCP z [2]. Pro obecnou délku odpovědí dostáváme LIP z [2] s tou výjimkou, že dokazovatel smí používat pouze lineární zobrazení, nikoliv affiní. To nám nevadí, protože v kapitole 1.4 ukážeme, že dokážeme zaručit, že potenciální útočník může používat jen lineární operace, nikoliv affiní.

**Definice 9.** *Mějme $\mathcal{R}$ binární relaci, $\mathbb{F}$ konečné těleso, PPT $P_{LPCP}$ generátor důkazů a PPT $V_{LPCP}$ ověřovací algoritmus. Dvojici $(P_{LPCP}, V_{LPCP})$ nazveme k-dotazové lineární PCP (LPCP) pro $\mathcal{R}$ nad $\mathbb{F}$ s knowledge error $\varepsilon$, délkou dotazů $n$ a délkou odpovědí $m$, pokud splňuje následující:*

**Syntax:** *Mějme $(x,w) \in \mathcal{R}$. Výstupem $P_{LPCP}(x,w)$ je matice $\Pi \in \mathbb{F}^{m\times n}$ . Ověřovatel $V_{LPCP}$ se skládá ze dvou algoritmů: Pravděpodobnostní $Q_{LPCP}$ vygeneruje dotazy $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k \in \mathbb{F}^n$ a stavovou informaci $\boldsymbol{u}$. Deterministický $D_{LPCP}(x,\boldsymbol{u},\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)$ pak přijme či zamítne na základě odpovědí $\boldsymbol{a}_i = \Pi\boldsymbol{q}_i$ a stavové informace $\boldsymbol{u}$.*

**Completeness:** $\forall(x,w) \in \mathcal{R}$ :

$$\Pr\left[ D_{LPCP}(x,\boldsymbol{u},\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k) = 1 \;\middle|\; \begin{array}{r} (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k, \boldsymbol{u}) \leftarrow Q_{LPCP} \\ \Pi \leftarrow P_{LPCP}(x,w) \\ (\boldsymbol{a}_1 | \ldots | \boldsymbol{a}_k) = \Pi(\boldsymbol{q}_1 | \ldots | \boldsymbol{q}_k) \end{array} \right] = 1$$

**Knowledge error $\varepsilon$:** *Existuje PPT knowledge extraktor $E_{LPCP}$ takový, že pro každou instanci $x$ a každou matici $\Pi \in \mathbb{F}^{m\times n}$, pokud $\Pr[V^{\Pi}_{LPCP}(x,\boldsymbol{u}) = 1] > \varepsilon$, pak $E^{\Pi}_{LPCP}(x)$ vrátí takové $w$, že $(x,w) \in \mathcal{R}$.*

Nakonec zadefinuje zero-knowledge pro LPCP. Omezíme se na Honest-Verifier zero-knowledge. To znamená, že schéma je zero-knowledge, pokud ověřovatel dodržuje protokol. To jinými slovy znamená, že pokud se ověřovatel chová podle

protokolu, pak přepis komunikace mezi ověřovatelem a dokazovatelem neobsahuje žádnou informaci o $w$.

**Definice 10.** *Řekneme, že dvojice algoritmů ($P_{LPCP}$,$V_{LPCP}$) je HVPZK-LPCP (Honest Verifier Perfect Zero-Knowledge) se simulation error $\varepsilon_s$, pokud je to LPCP a navíc existuje PPT simulátor $S$ splňující následující:*

- *Výstupem $S$ je buď $(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_k)$, nebo $\perp$.*
- *Pro danou instanci $x$ a dotatečný vstup $z \in \{0,1\}^{polylog(|\mathbb{F}|)}$*

$$\Pr\left[S(x,\boldsymbol{u},z) = \perp \;\middle|\; (\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k,\boldsymbol{u}) \leftarrow Q_{LPCP}\right] \leq \varepsilon_s$$

- *Označme $\tilde{S}(x,\boldsymbol{u},z)$ podmíněnou distribuci*

$$\Pr\left(S(x,\boldsymbol{u},z) = (\boldsymbol{a}_1,\ldots,\boldsymbol{a}_k) \middle| \begin{array}{c}(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k,\boldsymbol{u}) \leftarrow Q_{LPCP} \\ S(x,\boldsymbol{u},z) \neq \perp\end{array}\right).$$

*Pak pro každý algoritmus $D$ a každý dodatečný vstup $z \in \{0,1\}^{polylog(|\mathbb{F}|)}$*

$$\begin{aligned}&\Pr\left[\begin{array}{c}(x,w) \in \mathcal{R}_C \\ D(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_k,\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k,\boldsymbol{u},z) = 1\end{array} \middle| \begin{array}{r}(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k) \leftarrow Q_{LPCP} \\ (x,w) \leftarrow D(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k,\boldsymbol{u},z) \\ \Pi \leftarrow P_{LPCP}(x,w) \\ \forall i \in [k] : \boldsymbol{a}_i = \Pi\boldsymbol{q}_i\end{array}\right] \\ =&\Pr\left[\begin{array}{c}(x,w) \in \mathcal{R}_C \\ D(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_k,\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k,\boldsymbol{u},z) = 1\end{array} \middle| \begin{array}{r}(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k) \leftarrow Q_{LPCP} \\ (x,w) \leftarrow D(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k,\boldsymbol{u},z) \\ (\boldsymbol{a}_1,\ldots,\boldsymbol{a}_k) \leftarrow \tilde{S}(x,\boldsymbol{u},z)\end{array}\right]\end{aligned}$$

Nyní se můžeme pustit do samotné konstrukce LPCP. Využijeme přitom konstrukci z předchozí kapitoly, která nám umožňuje převést QAP na PQAP. Pro přehlednost je konstrukce 2 odsazena na další stránku.

**Značení.** *Pro $x^\ell$ je $\ell$-složkový vektor $(x,\ldots,x)$.*

Dokážeme, že takto zkonstruované LPCP má doopravdy požadované vlastnosti.

**Věta 2.** *Dvojice algoritmů ($P_{LPCP}$,$V_{LPCP}$) získaná konstrukcí 2 je 5ti-dotazové LPCP s knowledge error $\frac{2m}{|\mathbb{F}|}$.*

*Důkaz.* Completeness tohoto LPCP vyplývá přímo z konstrukce. Stačí si uvědomit, že pro čestného dokazovatele lze rovnici 1.2 přepsat jako

$$A(\tau)B(\tau) - C(\tau) = H(\tau)Z_S(\tau) \iff H(\tau) = \frac{A(x)B(x) - C(x)}{Z_S(x)},$$

což platí pro každé $\tau$ přímo z definice $H$. Druhá rovnice 1.3 podobně.

**Konstrukce 2.** *Mějme aritmetický obvod $C(x,w)$ nad tělesem $\mathbb{F}$ s $m$ násobicími hradly. Označíme $k = |x|$ a $n = 1 + |x| + |w| + m$. Zkonstruujeme 5ti-dotazové LPCP nad $\mathbb{F}$ s délkou odpovědí 1.*
*Nejprve obvod $C$ převedeme na kvadratický aritmetický program $(\tilde{A},\tilde{B},\tilde{C})$ dimenze $(m \times n)$ a ten následně pomocí konstrukce 1 převedeme na PQAP $(\boldsymbol{A}(x),\boldsymbol{B}(x),\boldsymbol{C}(x), Z_S(x))$. Algoritmy $P_{LPCP}$, $Q_{LPCP}$ a $D_{LPCP}$ fungují následovně:*

**Dokazovatel** $P_{\text{LPCP}}$ nejprve zvolí prvky $\delta_A$, $\delta_B$ a $\delta_C$ nezávisle z rovnoměrného rozdělení na $\mathbb{F}$. Dále pro dané $(x,w) \in \mathcal{R}_C$ spočte $\boldsymbol{w}$ takové, že $(\tilde{A},\tilde{B},\tilde{C})$ přijímá $\boldsymbol{w}$. Položí polynomy

$$\begin{aligned} A(x) &:= \langle \boldsymbol{A}(x),\boldsymbol{w}\rangle + \delta_A Z_S(x), \\ B(x) &:= \langle \boldsymbol{B}(x),\boldsymbol{w}\rangle + \delta_B Z_S(x), \\ C(x) &:= \langle \boldsymbol{C}(x),\boldsymbol{w}\rangle + \delta_C Z_S(x) \text{ a} \\ H(x) &:= \frac{A(x)B(x) - C(x)}{Z_S(x)}. \end{aligned}$$

$Z_S(x)$ skutečně dělí $A(x)B(x) - C(x)$, protože

$$\begin{gathered} A(x)B(x) - C(x) = \langle \boldsymbol{A}(x),\boldsymbol{w}\rangle \cdot \langle \boldsymbol{B}(x),\boldsymbol{w}\rangle - \langle \boldsymbol{C}(x),\boldsymbol{w}\rangle + Z_S(x)D(x), \\ \text{kde } D(x) = \delta_A\delta_B + \delta_A\langle \boldsymbol{B}(x),\boldsymbol{w}\rangle) + \delta_B\langle \boldsymbol{A}(x),\boldsymbol{w}\rangle) + \delta_C \end{gathered}$$

Dokazovatelova lineární funkce $\boldsymbol{\pi}$, pomocí které odpovídá na dotazy, je řádkový vektor $(\delta_A|\delta_B|\delta_C|\boldsymbol{w}|\boldsymbol{h})$, kde $\boldsymbol{h}$ jsou koeficienty polynomu $H(x)$.
**Dotazovatel** $Q_{\text{LPCP}}$ nejprve zvolí $\tau$ z rovnoměrného rozdělení na $\mathbb{F}$. Dále odešle pět dotazů

$$\begin{aligned} \boldsymbol{q_1} &:= (Z_S(\tau)|0|0|\boldsymbol{A}(\tau)|0^{m+1})) && (\langle \boldsymbol{\pi},\boldsymbol{q_1}\rangle = A(\tau)), \\ \boldsymbol{q_2} &:= (0|Z_S(\tau)|0|\boldsymbol{B}(\tau)|0^{m+1}) && (\langle \boldsymbol{\pi},\boldsymbol{q_2}\rangle = B(\tau)), \\ \boldsymbol{q_3} &:= (0|0|Z_S(\tau)|\boldsymbol{C}(\tau)|0^{m+1}) && (\langle \boldsymbol{\pi},\boldsymbol{q_3}\rangle = C(\tau)), \\ \boldsymbol{q_4} &:= (0|0|0|0^n|(1,\tau,\tau^2,\ldots,\tau^m)) && (\langle \boldsymbol{\pi},\boldsymbol{q_4}\rangle = H(\tau)), \\ \boldsymbol{q_5} &:= (0|0|0|(1,\tau,\tau^2,\ldots,\tau^k)|0^{n-k-1}|0^{m+1}) && (\langle \boldsymbol{\pi},\boldsymbol{q_5}\rangle = \langle (1|\boldsymbol{x}),(1,\tau,\ldots,\tau^k)\rangle) \end{aligned}$$

a uchová si stavovou informaci $\boldsymbol{u} = (1,\tau,\ldots,\tau^k,Z_S(\tau))$.
**Ověřovatel** $D_{\text{LPCP}}$ na vstupu $(\boldsymbol{x},a_1,\ldots,a_5,\boldsymbol{u})$ vrátí 1, právě když platí obě tyto rovnosti:

$$a_1a_2 - a_3 = a_4u_{k+2} \tag{1.2}$$

$$a_5 = u_1 + \langle \boldsymbol{x},(u_2,\ldots,u_{k+1})\rangle. \tag{1.3}$$

Jinak vrátí 0.

Konstrukce 1.5: LPCP

Mějme nyní vektor $\boldsymbol{\pi}^* = (\delta_A^*|\delta_B^*|\delta_C^*|\boldsymbol{w}^*|\boldsymbol{h}^*)$ takový, že $\Pr[V_{\text{LPCP}}^{\boldsymbol{\pi}} = 1] > \frac{2m}{|\mathbb{F}|}$ a uvažujme rovnosti 1.2 a 1.3 jako polynomiální rovnice ve formální proměnné $z$ místo $\tau$, tedy

$$\begin{aligned}
0 =& a_1 a_2 - a_3 - a_4 u_{k+2} \\
=& \langle \boldsymbol{\pi}^*, \boldsymbol{q_1} \rangle \langle \boldsymbol{\pi}^*, \boldsymbol{q_2} \rangle - \langle \boldsymbol{\pi}^*, \boldsymbol{q_3} \rangle - Z_S(z) \langle \boldsymbol{\pi}^*, \boldsymbol{q_4} \rangle \\
=& (\langle \boldsymbol{w}^*, A(z) \rangle + \delta_A Z_S(z))(\langle \boldsymbol{w}^*, B(z) \rangle + \delta_B Z_S(z)) \\
& - (\langle \boldsymbol{w}^*, C(z) \rangle + \delta_C Z_S(z)) - Z_S(z) H^*(z) \quad \text{a} \\
0 =& a_5 - u_1 - \langle \boldsymbol{x}, (u_1, \ldots, u_{k+1}) \rangle \\
=& \langle \boldsymbol{w}^*, (z, \ldots, z^k) \rangle - 1 - \langle \boldsymbol{x}, (z, \ldots, z^k) \rangle.
\end{aligned}$$

Oba dva tyto polynomy mají stupeň menší roven $2m$ a zároveň má dle předpokladu každý z nich více, než $2m$ kořenů. Musí se tedy jednat o nulové polynomy. Tím pádem musí platit rovnost po koeficientech, čili $\boldsymbol{w}_0^* = 1$ a $(\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_k^*) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$.

Dále pro každé $s \in S$ musí platit

$$\langle \boldsymbol{w}^*, \boldsymbol{A}(s) \rangle \langle \boldsymbol{w}^*, \boldsymbol{B}(s) \rangle - \langle \boldsymbol{w}^*, \boldsymbol{C}(s) \rangle = 0$$

To je ale přesně ekvivalence 1.1. Proto musí platit, že PQAP $(\boldsymbol{A}(x), \boldsymbol{B}(x), \boldsymbol{C}(x), Z_S(x))$ přijímá $\boldsymbol{x}$ a $\boldsymbol{w}^*$ je svědek této skutečnosti. Extraktor pomocí dotazů ve tvaru $\boldsymbol{e}_i \in \mathbb{F}^n$, kde $\boldsymbol{e}_i$ je i-tý vektor kononické báze $\mathbb{F}^n$, může získat libovolnou složku $\boldsymbol{w}^*$ včetně složek svědka $w$.

□

**Věta 3.** *Dvojice algoritmů $(P_{LPCP}, V_{LPCP})$ získaná konstrukcí 2 je HVZK-LPCP se simulation error $\frac{m}{|\mathbb{F}|}$.*

*Důkaz.* Zkonstruujeme simulátor $S$. $S$ dostane $x$ a $\boldsymbol{u}$, ze kterého získá $\tau \in \mathbb{F}$. Pokud $Z_S(\tau) = 0$, pak $S$ vrátí $\perp$ a skončí. To pro náhodně zvolené $\tau$ stane s pravděpodobností nejvýše $\frac{m}{|\mathbb{F}|}$. V případě, že $Z_S(\tau) \neq 0$, nejprve dokážeme, že distribuce standartně vygenerovaných odpovědí nezávisí na $w$ a následně ji nasimujeme.

$\delta_A, \delta_B, \delta_C$ byly zvoleny nezávisle z rovnoměrného rozdělení na $\mathbb{F}$. Díky nenulovosti $Z_S(\tau)$ jsou i $\delta_A Z_S(\tau), \delta_B Z_S(\tau), \delta_C Z_S(\tau)$ nezávislé z rovnoměrného rozdělení na $\mathbb{F}$. Zároveň je každý z $\delta_A Z_S(\tau), \delta_B Z_S(\tau), \delta_C Z_S(\tau)$ sčítancem postupně v odpovědích $a_1, a_2, a_3$ a všechny ostatní sčítance už jsou deterministické. Proto jsou $a_1, a_2, a_3$ také z nezávislé z rovnoměrného rozdělení na $\mathbb{F}$. $a_4 = \frac{a_1 a_2 - a_3}{Z_S(\tau)}$ má tedy rovněž rovnoměrné rozdělení na $\mathbb{F}$. Proto $a_1, \ldots, a_4$ nezávisí na $w$. $a_5 = 1 + \langle \boldsymbol{x}, (\tau, \ldots, \tau^k) \rangle$ závisí jen na veřejné části $\boldsymbol{w}$, nikoli na samotném svědkovi $w$.

Simulátor $S$ proto může bez ohledu na dotazy $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_5$ zvolit $a_1, \ldots, a_3$ z rovnoměrného rozdělení na $\mathbb{F}$ a následně dopočíst

$$a_4 = \frac{a_1 a_2 - a_3}{Z_S(\tau)} \quad \text{a} \quad a_5 = 1 + \langle \boldsymbol{x}, (\tau, \ldots, \tau^k) \rangle.$$

Tyto odpovědi budou vždy přijaty $D_{\text{LPCP}}$ a navíc mají stejnou distribuci, jako čestně vygenerované odpovědi.

□

## 1.4 Linear-only encryption

V této kapitole budeme studovat šifrování a kódování s homomorfními vlastnostmi. Jedno takové šifrování popíšeme a ukážeme, na jakých předpokladech je založena jeho bezpečnost. Díky linear-only encryption zaručíme první z předpokladů modelu LPCP, totiž že dokazovatel používá k tvorbě odpovědí pouze lineární funkci. Ačkoliv se budeme zabývat především šifrováním, které je snadnější na pochopení, na konci této kapitoly vysvětlíme, proč se v praktických aplikacích používá kódování na eliptických křivkách.

Kódování a šifrování není totéž. Zatímco kód je jakékoliv schéma, pomocí kterého lze reprezentovat nějakou informaci, na šifrování máme více požadavků. Chceme, aby se znalostí nějakého tajemství bylo možné ze zašifrované zprávy opět spočítat původní text, a naopak, aby tato operace byla výpočetně složitá bez znalosti tohoto tajemství.

Řekneme, že asymetrická šifra je sémanticky bezpečná, pokud žádný polynomiální útočník není schopen bez znalosti soukromého klíče přiřadit dané ciphertexty k původním plaintextům, které si může zvolit. Sémantickou bezpečnost definujeme ji pomocí následující bezpečnostní hry:

**Definice 11.** *Asymetrické šifrovací schéma* (Gen, Enc, Dec) *je* sémanticky bezpečné*, pokud pro každého PPT útočníka $A$*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{r}(\mathtt{sk},\mathtt{pk}) \leftarrow \mathtt{Gen}(1^\lambda)\\ (a_0,a_1) \leftarrow A(\mathtt{pk})\\ b \leftarrow \{0,1\}\\ b' \leftarrow A(\mathtt{pk}, \widehat{a_b})\end{array}\right] \leq 1/2 + \mathrm{negl}(\lambda).$$

Ať výchozí množina plaintextů je grupa $(\mathcal{P},+)$ a množina ciphertextů je grupa $(\mathcal{C},\cdot)$. Pro konstrukci SNARKu bude důležité, aby šifrovací operace $E : \mathcal{P} \to \mathcal{C}$ byla homomorfismus. To znamená, že

$$\forall a,b \in \mathcal{P} : E(a) \cdot E(b) = E(a+b).$$

Chceme, aby protivník mohl z ciphertextů daných zpráv spočíst ciphertext libovolné jejich lineární kombinace. Na druhou stranu bychom chtěli, aby protivník nemohl vypočíst nic víc. Tedy například, aby nemohl homomorfně spočíst součin dvou zpráv. Schématu s takovou vlastností říkáme linear-only encryption.

Využijeme stejné úvahy jako při definici knowledge soundness SNARKu. Pokud protivník se znalostí nějakých ciphertextů vytvořil nový ciphertext, pak existuje extrahovací algoritmus, který z něho nějak „vytáhne“ koeficienty příslušně lineární kombinace.

**Značení.** *Pro plaintext $m$ položíme $\widehat{m} := \mathtt{Enc}_{\mathtt{pk}}(m)$ a obráceně pro ciphertext $\widehat{m}$ položíme $m := \mathtt{Dec}_{\mathtt{sk}}(\widehat{m})$.*

**Definice 12.** *Asymetrické šifrovací schéma* (Gen, Enc, Dec) *má* linear-only *vlastnost, pokud pro každého PPT protivníka $A$ existuje PPT extraktor $E^A$ takový, že pro každý dodatečný vstup $z \in \{0,1\}^{poly(\lambda)}$ a každý generátor plaintextů $M$*

$$\Pr\left[\begin{array}{c}\exists i \in [k] :\\ \mathtt{ImVer}_{\mathtt{sk}}(\widehat{b}_i) = 1\\ \wedge\\ b_i \neq c_i\end{array} \;\middle|\; \begin{array}{r}(\mathtt{sk},\mathtt{pk}) \leftarrow \mathtt{Gen}(1^\lambda)\\ (a_1,\ldots,a_m) \leftarrow M(\mathtt{pk})\\ (\widehat{b}_1,\ldots,\widehat{b}_k) \leftarrow A(\mathtt{pk},\widehat{a}_1,\ldots,\widehat{a}_m,z)\\ \Pi \leftarrow E^A(\mathtt{pk},\widehat{a}_1,\ldots,\widehat{a}_m,z)\\ (c_1,\ldots,c_k) = \Pi(a_1,\ldots,a_m)^\top\end{array}\right] \leq \mathrm{negl}(\lambda)$$

*kde* $\Pi : \mathbb{F}^m \to \mathbb{F}^k$ *je linearní zobrazení.*

**Definice 13.** Linear-only encryption *schéma je pětice algoritmů* (Gen, Enc, Dec, Add, ImVer) *s následující syntaxí a vlastnostmi:*

- *Pro daný bezpečnostní parametr* $\lambda$*, PPT* $\texttt{Gen}(1^\lambda)$ *vrátí soukromý klíč* sk *a veřejný klíč* pk*. Veřejný klíč obsahuje popis tělesa* $\mathbb{F}$*, které reprezentuje prostor plaintextů.*
- *PPT* Enc *je randomizované šifrování a DPT* Dec *deterministické dešifrování. Pro každý plaintext* $m$ *platí*

$$\Pr\left[\texttt{Dec}_{\texttt{sk}}(\texttt{Enc}_{\texttt{pk}}(m)) = m \;\middle|\; (\texttt{pk},\texttt{sk}) \leftarrow \texttt{Gen}(1^\lambda)\right] = 1$$

- *DPT* $\texttt{Add}(\texttt{pk},\alpha_1,\ldots,\alpha_m,c_1,\ldots,c_m)$ *umožňuje homomorfně počítat lineární kombinace ciphertextů. Konkrétně pro veřejný klíč* pk *a* $p_1,\ldots,p_m,\alpha_1,\ldots,\alpha_m \in \mathbb{F}$ *platí:*

$$\texttt{Add}(\texttt{pk},\texttt{Enc}_{\texttt{pk}}(p_1),\ldots,\texttt{Enc}_{\texttt{pk}}(p_m),\alpha_1,\ldots,\alpha_m) = \texttt{Enc}_{\texttt{pk}}\Big(\sum_{i=1}^{m} \alpha_i p_i\Big).$$

- *Pro ciphertext* $c$*, DPT* $\texttt{ImVer}_{\texttt{sk}}(c)$ *testuje, jestli* $c$ *leží v obrazu funkce* $Enc_{\texttt{pk}}$*.*
- (Gen, Enc, Dec) *je sémanticky bezpečné.*
- (Gen, Enc, Dec) *má linear-only vlastnost.*

Schématem, ze kterého zkonstruujeme linear-only encryption, je **Paillierův kryptosystém**, definovaný následovně:

**Konstrukce 3.** Paillierův kryptosystém *je asymetrická šifra tvořená algoritmy* (Gen, Enc, Dec)*, které fungují takto:*

**Generování klíčů:** $\texttt{Gen}(1^\lambda)$ *zvolí* $N = p \cdot q$ *součin dvou prvočísel* $p,q \leftarrow [2^\lambda,2^{\lambda+1}] \cap \mathbb{P}$*. Soukromý klíč* $\texttt{sk} = (N,d)$ *pro* $d$ *takové, že*

$$d \equiv 1 \ (mod\, N),$$
$$d \equiv 0 \ (mod\, (p-1)(q-1)).$$

*Takové* $d$ *lze nalézt pomocí čínské věty při znalosti* $p$ *a* $q$*. Veřejný klíč je samotné* $\texttt{pk} = N$*.*

**Šifrování:** *Zprávy reprezentujeme pomocí prvků* $\mathbb{Z}_N$*. Při šifrování zprávy* $m$ $\texttt{Enc}_{\texttt{pk}}(m)$ *nejprve zvolí* $r$ *z rovnoměrného rozdělení na* $Z^*_{N^2}$ *náhodně. Ciphertext* $c$ *pak spočte jako*

$$c = (1+N)^m \cdot r^N \ (mod\, N^2)$$

*.*

**Dešifrování:** $\texttt{Dec}_{\texttt{sk}}(m)$ *nejprve spočte* $t := c^d \ (mod\, N^2)$ *a následně dopočte zprávu* $m = \frac{t-1}{N}$*.*

**Pozorování 4.** *Dešifrovací funkce* `Dec` *Paillierova kryptosystému funguje korektně.*

*Důkaz.*

$$\begin{aligned} c^d &\equiv (1+N)^{dm} \cdot r^{dN} \pmod{N^2} \\ &\equiv (1+N)^{dm} && \text{používáme } d \equiv 0 \pmod{(p-1)(q-1)} \\ &\equiv 1 + d \cdot m \cdot N \\ &\equiv 1 + m \cdot N && \text{používáme } d \equiv 1 \pmod{N} \end{aligned}$$

Proto $m = ((c^d \bmod N^2) - 1)/N$.

□

Pallierův kryptosystém má homomorfní vlastnosti. Zatímco u RSA, Rabinova a ElGamalova schématu se jedná o homomorfismus multiplikativní, Pallierův kryptosystém převádí součin na součet, čili pro zprávy $m_1$ a $m_2$ platí

$$\mathtt{Enc}_{\mathtt{pk}}(m_1) \cdot \mathtt{Enc}_{\mathtt{pk}}(m_2) = \mathtt{Enc}_{\mathtt{pk}}(m_1 + m_2),$$

protože

$$(1+N)^{m_1}\, r_1^N \cdot (1+N)^{m_2}\, r_2^N \equiv (1+N)^{m_1+m_2} \cdot (r_1 r_2)^N \pmod{N^2}.$$

Sémantickou bezpečnost lze dokázat na základě předpokladu, že je pro každého PPT protivníka bez přístupu k prvočíselnému rozkladu $N$ je těžké pro náhodně volená $y \in \mathbb{Z}_{N^2}$ určit, zda k nim existuje $x \in \mathbb{Z}_{N^2}$ t.ž. $y \equiv x^N \mod N$.

**Předpoklad 1.** Nth-Residuosity Indistinguishability Assumption(NRIA)*:*
*Pro každého PPT protivníka A platí:*

$$\Pr\left[ b = b' \;\middle|\; \begin{array}{r} p,q \leftarrow \mathbb{P} \cap [2^\lambda, 2^{\lambda+1}];\ N \leftarrow pq \\ b \leftarrow \{0,1\};\ x \leftarrow \mathbb{Z}_{N^2} \\ y = \begin{cases} x & b = 0 \\ x^N \bmod N^2 & b = 1 \end{cases} \\ b' \leftarrow A(y,N) \end{array} \right] < \frac{1}{2} + \mathrm{negl}(\lambda).$$

**Pozorování 5.** *Pokud platí NRIA, pak je Paillierův kryptosystém sémanticky bezpečný.*

*Důkaz.* Důkaz se provede pomocí standartní redukce mezi bezpečnostními experimenty. Na základě předpokladu existence útočníka $A$ na sémantickou bezpečnost Paillierova kryptosystému zkonstuujeme útočníka $S$, který dokáže rozlišovat N-té mocniny v $\mathbb{Z}_{N^2}$ od náhodných prvků $Z_{N^2}$, tj. falzifikuje NRIA.

| $A$ | | $S$ | | Orákulum |
|---|---|---|---|---|
| | $\xleftarrow{\mathtt{pk}=N}$ | $z \leftarrow \{0,1\}$ | | $b \leftarrow \{0,1\}$ |
| | $\xrightarrow{m_1,m_2}$ | | | $x \leftarrow \mathbb{Z}_{N^2}$ |
| | $\xleftarrow{c}$ | $c = ((1+N)^{m_z} y)$ | $\xleftarrow{y,N}$ | $y = \begin{cases} x & b = 0 \\ x^N \bmod N^2 & b = 1 \end{cases}$ |
| | $\xrightarrow{z'}$ | vrať $b' = \begin{cases} 1 & z = z' \\ 0 & z \neq z' \end{cases}$ | | |

Povšimněme si, že pokud $b = 0$, pak $A$ dostává náhodné číslo a tudíž nemůže být úspěšnější, než $\frac{1}{2}$. V opačném případě je úspěšnost $S$ stejná, jako úspěšnost $A$. V součtu je tedy úspěšnost $S$ právě $\frac{1}{2}$ úspěšnosti $A$.

□

Paillierův kryptosystém nemá linear-only vlastnost. Útočník totiž může vytvořit nový ciphertext náhodným výběrem z $\mathbb{Z}_{N^2}$, přičemž pravděpodobnost, že takto vybraný ciphertext bude validní, je

$$\frac{|\mathbb{Z}^*_{N^2}|}{|\mathbb{Z}_{N^2}|} = \frac{\varphi(N^2)}{N^2} = \frac{(p-1)(q-1)}{pq}$$

. Znemožníme útočníkovi tvořit nové ciphertexty tím, že Paillierův kryptosystém upravíme. Budeme přitom potřebovat safe primes.

**Definice 14.** *Řekneme, že prvočíslo $p \in \mathbb{P}$ je safe prime, pokud je tvaru $p = 2q+1$, kde $q \in \mathbb{P}$.*

**Konstrukce 4.** (Linear-only encryption z Paillirova kryptosystému)
*Setavíme linear-only encryption schéma* (`Gen`, `Enc`, `Dec`, `Add`, `ImVer`) *z Paillierova kryptosystému* (`Gen′`, `Enc′` ,`Dec′`) *takto:*

- $\texttt{Gen}(1^\lambda)$ *zvolí* $N = pq$ *jako součin dvou safe primes v intervalu* $[2^\lambda,2^{\lambda+1}]$ *a dopočte* $d$ *stejně, jako* `Gen′`. *Dále zvolí* $\alpha$ *z rovnoměrného rozdělení na* $\mathbb{Z}_N$ *náhodně. Veřejný klíč* $\texttt{pk} = N$. *Soukromý klíč* $\texttt{sk} = (N,d,\alpha)$
- $Enc(\texttt{sk},m)$ *zašifruje* $m$ *jako* $c' = \texttt{Enc}'(N,m)$ *a vrátí ciphetext*

$$c = (c',(c')^\alpha \bmod N^2).$$

- $\texttt{ImVer}(\texttt{sk},(a,b))$ *vrátí 1, právě když* $a^\alpha \equiv b \mod \mathbb{N}^2$.
- $\texttt{Dec}(\texttt{sk},m) = \texttt{Dec}'((N,d),m)$.
- $\texttt{Add}(\texttt{pk},(a_1,b_1),\ldots,(a_n,b_n),k_1,\ldots,k_n) = (\prod_{i=1}^n a_i^{k_i}, \prod_{i=1}^n b_i^{k_i})$.

Při konstrukci 4 jsme se dopustili drobného odklonu od syntaxe linear-only encryption. K vytvoření nového cihertextu je totiž potřeba znát nikoli `pk`, ale `sk`. To je na první pohled zvláštní. Linear encyption totiž nebudeme používat jako šifru, ale jako mechanizmus, pomocí kterého lze vynutit, že dokazovatel dělá na prvcích dotazu z modelu LPCP jen linérní operace. Tedy ověřovatel zašifruje dotazy a pošle je dokazovateli. Ten z nich lineárním zobrazením spočte nějaké odpovědi. Nakonec si dokazovatel odšifruje tyto odpovědi a zkotroluje jejich validitu.

Díky tomu, že protivník nedokáže tvořit nové ciphertexty, se můžeme ve všech důkazech omezit z affiních zobrazení na lineární.

Vysvětlíme předpoklad, ze kterého vyplyne linear-only vlastnost upraveného Paillierova kryptosystému. Mějme grupu $(\mathbb{G},\cdot)$ řádu $q$, ve které je těžké najít diskrétní logaritmus. Pro $\alpha \in \mathbb{Z}_q$ definujeme $\alpha$-pár jako každou dvojici $(a,b) \in \mathbb{G}^2$, která splňuje $b = a^\alpha$. Nyní uvažme hru, ve které protivník dostane nějaké $\alpha$-páry $(a_1,b_1),\ldots,(a_n,b_n)$ a má za úkol vytvořit nějaký nový $\alpha$-pár. Triviálně může

zvolit nějaké $k$ a vrátit $(a_1^k,b_1^k)$. Tento nápad lze zobecnit. Protivník zvolí libovolné exponenty $k_1,\ldots,k_n \in \mathbb{Z}_q$ a vrátí $(\prod_{i=1}^n a_i^{k_i}, \prod_{i=1}^n b_i^{k_i})$. S vysokou pravděpodobností tímto způsobem vytvoří nový $\alpha$-pár. Následující předpoklad říka, že žádná další strategie není.

**Předpoklad 2.** Knowledge of exponent assumption (KEA)*: Mějme posloupnost distribucí grup $(\mathcal{G}_\lambda)_{\lambda\in\mathbb{N}}$, kde řád $\forall\mathbb{G} \in \mathcal{G}_\lambda$ je prvočíslo ležící v intervalu $[2^\lambda,2^{\lambda+1}]$ a ve kterých je problém hledání diskrétního logaritmu těžký. Pak pro každý pravděpodobnostní generátor $S$ prvků grupy, každý dodatečný vstup $z \in \{0,1\}^{poly(\lambda)}$ a každého PPT útočníka $A$ existuje PPT knowledge extraktor $E$ takový, že:*

$$\Pr\left[\begin{array}{c|r} f' = f^\alpha & \mathbb{G} \leftarrow \mathcal{G}_\lambda \\ \wedge & g_1,\ldots,g_n \leftarrow S(\mathbb{G}) \\ f \neq \prod_{i=1}^n g_i^{k_i} & \alpha \leftarrow \mathbb{Z}_{ord(\mathbb{G})} \\ & f,f' \leftarrow A(\mathbb{G},g_1,g_1^\alpha,\ldots,g_n,g_n^\alpha,z) \\ & k_1,\ldots,k_n \leftarrow E^A(\mathbb{G},g_1,g_1^\alpha,\ldots,g_n,g_n^\alpha,z) \end{array}\right] < \mathrm{negl}(\lambda).$$

I přesto, že hledání diskrétního logaritmu v $\mathbb{Z}_{N^2}^*$ je těžké díky tomu, že $p$ a $q$ jsou safeprimes, tento předpoklad nemůžeme použít přímo pro $g_i = c_i$, protože grupa $\mathbb{Z}_{N^2}^*$ nemá prvočíselný řád. Přes veškeré snahy se nám nepodařilo, tento problém vyřešit. Na druhou stranu nám nepřipadá nerozumné předpoklad zesílit tím, že se vzdáme požadavku na prvočíselnost řádu grup. Podobný Knowledge of Exponent Assumption pro RSA je diskutován zde [11].

Podobný problém je, že množina ciphertextů Paillierova kryptosystému netvoří těleso, ale jen okruh. I přesto stejně jako v [12] se domníváme, že toto v reálných aplikacích není problém, protože narazit na neinvertibilní prvek v $\mathbb{Z}_N$ je stejně těžké, jako faktorizovat $N$.

Výsledná konstrukce SNARKU, kterou provedeme v sekci 1.6, bude založena na tom, že se LPCP dotazy zašifrují pomocí linear-only encryption, dokazovatel z nich homomorfně spočítá odpovědi a tyto odpovědi následně ověřovatel odšifruje soukromým klíčem a ověří si jejich validitu. Podobnou techniku nelze použít při konstrukci publickly-verifiable SNARK, protože u nich všechny parametry musí být veřejné. Můžeme však zkonstruovat a následně použít kódování na eliptických křivkách s několika speciálními vlastnostmi. Kromě homomorfního počítaní lineárních kombinací plaintextů totiž budeme schopni homomorfně testovat nuly kvadratických polynomů. Díky tomu, že práce LPCP ověřovatele spočívá právě v ověření nul kvadratických polynomů, nebudeme potřebovat prvky výsledného argumentu dekódovat.

Následující definici 15 přebíráme z [13], kde je rovněž uvedena konstrukce takového párování.

**Definice 15.** *Mějme grupy $\mathbb{G}$ a $\mathbb{G}_T$. Zobrazení $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ je* bilineární párování *na $(\mathbb{G},\mathbb{G}_T)$, pokud splňuje následující:*

**Bilinearita:** $\forall P,Q,R \in \mathbb{G} : \bigwedge \begin{array}{l} e(P+Q,R) = e(P,R) + e(Q,R) \\ e(P,Q+R) = e(P,Q) + e(P,R) \end{array}$

**Nedegenerovanost:** $\forall P \in \mathbb{G} : e(P,P) \neq 1$.

**Efektivita:** *$e$ je efektivně spočitatelné.*

Zvolíme si nějaký generátor $G$ grupy $\mathbb{G}$ takový, že $e(G,G)$ má prvočíselný řád $q$. Kódujeme opět jako dvojice $\texttt{Enc}(m) := (mG,mF)$, kde $F = \alpha G$ pro nějaké $\alpha$ z rovnoměrného rozdělení na $\mathbb{Z}_q$, které je součástí soukromého klíče. Validitu ciphertextu $(P, Q)$ (chceme, aby $Q = \alpha P$), lze ověřit jako rovnost $e(F,P) = e(G,Q)$. Tento způsob šifrování nám opět zaručí liner-only vlastnost tohoto kódování.

Toto kódování nám neumožnuje z kódových textů získat původní zprávy. I přesto existuje způsob, jak se o zakódovaných prvcích něco dozvědět. Konkrétně můžeme testovat, zda se jedná o nulu libovolného polynomu stupně nejvýše 2. Například: Alice tvrdí, že zná $x$ takové, že $x^2 - 5x + 2 \equiv 0 \mod q$, ale nechce ho prozradit. Alice zakóduje $x$ jako $P = xG$ a pošle $P$ Bobovi. Víme, že $e(aP,bQ) = e(P,Q)^{ab}$. Proto si Bob může ověřit Aliciino tvrzení spočtením

$$e(P,P)e(-5G,P)e(2G,G) = e(G,G)^{x^2}e(G,G)^{-5x}e(G,G)^2 = e(G,G)^{x^2-5x+2} \stackrel{?}{=} 1.$$

Totéž lze udělat pro polynomy více proměnných stupně nejvýše dva. Díky tomu, že polynomy, pomocí kterých $D_{\mathrm{LPCP}}$ rozhoduje o platnosti důkazu, jsou právě tohoto druhu, mohou párování nahradit linear-only encryption v tomto modelu.

## 1.5 Konstrukce 1-dotazového LPCP z k-dotazové LPCP

V kapitole 1.3 o LPCP jsme nahlédli, že LPCP je sice mocný nástroj, ale funguje jen díky dvěma silným předpokladům. Prvním z nich je, že dokazovatel odpovědi počítá jako nějaké lineární zobrazení. Tento předpoklad budeme schopni zaručit díky linear-only šifrování z kapitoly 1.4. Druhým předpokladem je, že dokazovatel používá k tvorbě odpovědí vždy jedno a to samé lineární zobrazení. Tento předpoklad zaručíme tak, že k-dotazové LPCP, upravíme na 1-dotazové LPCP. Dokazovatel nebude moc používat různá lineární zobrazení k odpovědi na různé dotazy, protože prostě nedostane různé dotazy, ale jen jeden dotaz.

K důkazu soundness zkonstruovaného LPCP budeme potřebovat Schwartz–Zippelovo lemma, které uvádíme jako fakt ze základního kurzu algebry bez důkazu.

**Lemma 6.** *(Schwartz–Zippel) Mějme nenulový polynom $f \in \mathbb{F}[x_1,\ldots,x_n]$ stupně $d$ a $M$ konečnou podmnožinu tělesa $\mathbb{F}$. Zvolme $r_1,\ldots,r_n$ nezásvisle z rovnoměrného rozdělení na $M$. Pak* $\Pr[f(r_1,\ldots,r_n) = 0] \leq \frac{d}{|M|}$.

**Věta 7.** *Pokud je $(P_{(k)}, V_{(k)})$ k-dotazový LPCP nad $\mathbb{F}$ pro relaci $\mathcal{R}$ s knowledge error $\varepsilon_{(k)}$, délkou dotazů $n$ a délkou odpovědí 1, pak $(P_{(1)}, V_{(1)})$ získané konstrukcí 5 je 1-dotazový LPCP nad $\mathbb{F}$ pro $\mathcal{R}$ s knowledge error $\varepsilon_{(1)} = \max\{\varepsilon_{(k)}, \frac{2}{|\mathbb{F}|}\}$ s délkou dotazů $n(k+1)$ a délkou odpovědí $k+1$.*

**Konstrukce 5.** *(1-dotazové LPCP) Buď $(P_{(k)}, V_{(k)})$ k-dotazový LPCP nad $\mathbb{F}$ s délkou dotazů $n$ a délkou odpovědí 1. Pro $(P_{(k)}, V_{(k)})$ zkonstruujeme 1-dotazové LPCP $(P_{(1)}, V_{(1)})$ následovně:*

**Dotazovatel** *$Q_{(1)}$ nejprve spustí $Q_{(k)}$, čímž získá prvních k-dotazů a stavovou informaci $\boldsymbol{u}$. Dále zvolí $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$ nezávisle z rovnoměrného rozdělení na $\mathbb{F}$ a spočte*

$$\boldsymbol{q}_{k+1} = \sum_{i=1}^{k} \alpha_i \boldsymbol{q}_i$$

*(poslední dotaz budeme nazývat test linearity). Nakonec dotazy zřetězí a odešle dotaz*

$$\begin{aligned} \boldsymbol{q} &= (\boldsymbol{q}_1^T | \ldots | \boldsymbol{q}_k^T | \boldsymbol{q}_{k+1}^T)^T = \\ &= (q_{1,1}, \ldots, q_{1,m}, q_{2,1}, \ldots, q_{2,m}, \ldots\ldots, q_{k+1,1}, \ldots, q_{k+1,m})^T. \end{aligned} \tag{1.4}$$

*Jako stavovou informaci $\boldsymbol{u}'$ $Q_{(1)}$ vrátí $\boldsymbol{u}$ zřetězené s $\alpha_1, \ldots, \alpha_k$.*

**Dokazovatel** *$P_{(1)}(x, w)$ nejprve spustí $P_{(k)}(x, w)$, čímž získá vektor $\boldsymbol{\pi} \in \mathbb{F}^{1 \times n}$. Následně vrátí matici*

$$\Pi = \left( \begin{array}{c|c|c|c} \boldsymbol{\pi} & \mathbf{0} & \ldots & \mathbf{0} \\ \hline \mathbf{0} & \boldsymbol{\pi} & \ddots & \vdots \\ \hline \vdots & \ddots & \ddots & \mathbf{0} \\ \hline \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{\pi} \end{array} \right).$$

*Tj. i-tá složka odpovědi $\boldsymbol{a}$ se spočte jako $a_i = \langle \boldsymbol{\pi}, \boldsymbol{q}_i \rangle$.*

**Ověřovatel** *$D_{(1)}(x, \boldsymbol{u})$ pro odpověď $\boldsymbol{a} = (a_1, \ldots, a_{k+1})$ vrátí 1 právě tehdy, když $V_{(k)}(x, \boldsymbol{u}, a_1, \ldots, a_k) = 1$ a zároveň $a_{k+1} = \sum_{i=1}^{k} \alpha_i a_i$. Jinak vrátí 0.*

*Důkaz.* Správnost syntaxe a délka dotazů a odpovědí vyplívají přímo z konstrukce. K důkazu completeness si stačí uvědomit, že

$$a_{k+1} = \langle \boldsymbol{\pi}, \boldsymbol{q_{k+1}} \rangle = \langle \boldsymbol{\pi}, \sum_{i=1}^{k} \alpha_i \boldsymbol{q}_i \rangle = \sum_{i=1}^{k} \alpha_i \langle \boldsymbol{\pi}, \boldsymbol{q}_i \rangle = \sum_{i=1}^{k} \alpha_i a_i.$$

Zbývá dokázat knowledge soundness. Zafixujme trvzení $x$. Dále fixujeme $\Pi^* \in \mathbb{F}^{n(k+1) \times k+1}$ nějakou strategii podvádějícího dokazovatele takovou, že

$$\Pr\left[V_{(1)}(x, \boldsymbol{a}) = 1\right] > \max\{\varepsilon_{(k)}, \frac{2}{|\mathbb{F}|}\}, \tag{1.5}$$

kde $\boldsymbol{a} = \Pi^* \boldsymbol{q}$ je náhodná veličina na pravděpodobnostním prostoru všech možných dotazů $\boldsymbol{q} \leftarrow Q_{(1)}$. S tímto prostorem budeme pracovat i nadále. Strategii můžeme

zafixovat už nyní, protože závisí jen na $x$ a $w$. Dále můžeme $\Pi^*$ rozepsat jako

$$\Pi^* : \boldsymbol{q} \mapsto \left(\begin{array}{c|c|c} \boldsymbol{\pi}_{1,1} & \dots & \boldsymbol{\pi}_{1,k+1} \\ \hline \vdots & \ddots & \vdots \\ \hline \boldsymbol{\pi}_{k+1,1} & \dots & \boldsymbol{\pi}_{k+1,k+1} \end{array}\right) \boldsymbol{q}$$

kde pro $\forall i,j \in [k+1]$ $\boldsymbol{\pi}_{i,j} \in \mathbb{F}^{1\times m}$. i-tá složka odpovědi $\boldsymbol{a}$ lze zapsat jako

$$a_i = \sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle, \tag{1.6}$$

kde $\boldsymbol{q}_i$ dostáváme ze vztahu 1.4.

Dokážeme, že $\Pi^*$ je blokově diagonální matice tvaru

$$\Pi^* : \boldsymbol{q} \mapsto \left(\begin{array}{c|c|c|c} \boldsymbol{\pi}_{k+1,k+1} & \mathbf{0} & \dots & \mathbf{0} \\ \hline \mathbf{0} & \boldsymbol{\pi}_{k+1,k+1} & \ddots & \vdots \\ \hline \vdots & \ddots & \ddots & \mathbf{0} \\ \hline \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{\pi}_{k+1,k+1} \end{array}\right) \boldsymbol{q}.$$

Pro spor ať existuje $i^* \in [k+1]$ t.ž. $a_{i^*} \neq \langle \boldsymbol{\pi_{k+1,k+1}}, \boldsymbol{q}_{i^*} \rangle$. Spočteme pravděpodobnost, že $\boldsymbol{a}$ splňuje lineární test

$$a_{k+1} = \sum_{i=1}^{k} \alpha_i a_i. \tag{1.7}$$

Dosazením 1.6 do 1.7 získáme

$$\sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle = \sum_{i=1}^{k} \alpha_i \left( \sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle \right).$$

Po přehození sum vychází

$$\sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle = \sum_{j=1}^{k+1} \sum_{i=1}^{k} \alpha_i \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle.$$

Dále substituujeme $\boldsymbol{q}_{k+1} = \sum_{l=1}^{k} \alpha_l \boldsymbol{q}_l$, čímž získáme

$$\sum_{j=1}^{k} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle + \langle \boldsymbol{\pi}_{k+1,k+1}, \sum_{l=1}^{k} \alpha_l \boldsymbol{q}_l \rangle = \sum_{j=1}^{k} \sum_{i=1}^{k} \alpha_i \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle + \sum_{i=1}^{k} \alpha_i \langle \boldsymbol{\pi}_{i,k+1}, \sum_{l=1}^{k} \alpha_l \boldsymbol{q}_l \rangle.$$

Použitím linearity a vhodným přeuspořádáním členů této rovnice získáme

$$\begin{aligned} &\sum_{j=1}^{k} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle + \sum_{l=1}^{k} \alpha_l \langle \boldsymbol{\pi}_{k+1,k+1}, \boldsymbol{q}_l \rangle = \\ &= \sum_{j=1}^{k} \sum_{i=1}^{k} \alpha_i \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle + \sum_{i=1}^{k} \sum_{l=1}^{k} \alpha_l \alpha_i \langle \boldsymbol{\pi}_{i,k+1}, \boldsymbol{q}_l \rangle \end{aligned}$$

$$\begin{aligned} 0 = &\sum_{i=1}^{k} \sum_{l=1}^{k} \langle \boldsymbol{\pi}_{i,k+1}, \boldsymbol{q}_l \rangle \alpha_l \alpha_i + \sum_{i=1}^{k} \left( \sum_{j=1}^{k} \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle - \langle \boldsymbol{\pi}_{k+1,k+1}, \boldsymbol{q}_i \rangle \right) \alpha_i \\ &- \sum_{j=1}^{k} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle. \end{aligned} \tag{1.8}$$

Tento výraz může chápat také jako polynom z $\mathbb{F}[\alpha_1,\ldots,\alpha_k]$. Všimněme si si, že se jedná o nenulový polynom, jelikož monom $\alpha_{i^*}$ má díky předpokladu $a_{i^*} \neq \langle \boldsymbol{\pi}_{\boldsymbol{k+1,k+1}}, \boldsymbol{q}_{i^*} \rangle$ nenulový koeficient ($a_{i^*} = \sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{\boldsymbol{i^*,j}}, \boldsymbol{q}_j \rangle$). Dále víme, že stupeň tohoto polynomu je nejvýše 2 a také, že $\alpha_1,\ldots,\alpha_k$ byly voleny náhodně. Ze Schwartz-Zippelova lemmatu tedy vyplývá, že pro náhodná $\alpha_1,\ldots,\alpha_k$ rovnost 1.8 nastane s pravděpodobností menší, než $\frac{2}{|\mathbb{F}|}$. Jenže

$$\begin{aligned} &\max\{\varepsilon_{(k)}, \frac{2}{|\mathbb{F}|}\} < \Pr\left[V_{(1)}(x,\boldsymbol{a}) = 1\right] \leq \\ &\leq \Pr\left[V_{(k)}(x,\boldsymbol{a}) = 1 \quad \wedge \quad \boldsymbol{a} \text{ splňuje lineární test}\right] \leq \\ &\leq \Pr\left[\boldsymbol{a} \text{ splňuje lineární test}\right] < \frac{2}{|\mathbb{F}|}, \end{aligned} \tag{1.9}$$

což je spor.

Tím dostáváme $\boldsymbol{\pi} := \boldsymbol{\pi}_{k+1,k+1}$ takové, že $\Pr\left[V^{\pi}_{(k)}(x) = 1\right] > \varepsilon_{(k)}$. Proto můžeme použít extraktora $E^{\pi}_{(k)}(x)$ k získání svědka $w$.

□

Ukážeme, že konstrukce 5 zachovává perfect zero-knowledge.

**Pozorování 8.** *Pokud je $(P_{(k)},V_{(k)})$ HVPZK-LPCP se simulation error $\varepsilon_s$, pak $(P_{(1)},V_{(1)})$ získané konstrukcí 5 je také HVPZK-LPCP se simulation error $\varepsilon_s$.*

*Důkaz.* Mějme simulátor $S$ dosvědčující HVZK $(P_{(k)},V_{(k)})$. Zkonsruujeme simulátor $S'$ pro $(P_{(1)},V_{(1)})$. Pro dané $x$ a stavovou informaci $\boldsymbol{u'}$, simulátor $S'$ nejprve použije $S(x,\boldsymbol{u})$ čímž získá s pravděpodobností $\varepsilon_s$ symbol selhální $\perp$ nebo odpovědi $a_1,\ldots,a_k$. Pokud $S(x,\boldsymbol{u})$ vrátil $\perp$ $S'$ rovněž vrátí $\perp$, čímž simulace končí. Jinak dále spočte $a_{k+1} := \sum_{i=1}^{k} \alpha_i a_i$. To může, protože $\alpha_1,\ldots,\alpha_k$ jsou součástí $\boldsymbol{u'}$. Nakonec $S'$ vrátí $(a_1,\ldots,a_k,a_{k+1})^T$.

Vzhledem k tomu, že $\alpha_1,\ldots,\alpha_k$ byly zvoleny nezávisle z rovnoměrného rozdělení na $\mathbb{F}$, tak i $a_{k+1}$ je z rovnoměrného rozdělení na $\mathbb{F}$. Proto má přidaná hodnota $a_{k+1}$ v simulovaném a nesimulovaném přepisu protokolu stejnou distribuci.

□

## 1.6 Konstrukce zk-SNARK

V předchozích kapitolách jsme popsali konstrukci 5-ti dotazového LPCP, ukázali jsme, že k němu lze zkonstruovat 1-dotazovou variantu, a upravili jsme Paillierův kryptosystém tak, aby mělo linear-only vlastnost. Díky tomu nyní už máme dost nástrojů na to, abychom zaručili předpoklady modelu LPCP. Spojíme 1-dotazové LPCP s linear-only encryption, čímž získáme výsledný zk-SNARK.

**Značení:** Pro vektor plaintextů $\boldsymbol{q} = (q_1,\ldots,q_m)$ označíme $\widehat{\boldsymbol{q}} := (\widehat{q_1},\ldots,\widehat{q_m})$.

**Lemma 9.** *(o rozlišení vektorů) Mějme linear-only encryption schéma $\mathcal{E}$ = (`Gen`, `Enc`, `Dec`, `Add`, `ImVer`) s množinou plaintextů $\mathcal{P}$ a polynom $m(x) \in \mathbb{Z}[x]$.*

*Pak pro každého útočníka platí*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{r}(\mathtt{sk},\mathtt{pk}) \leftarrow \mathtt{Gen}(1^\lambda)\\ (\boldsymbol{q}^0,\boldsymbol{q}^1) \leftarrow A(\mathtt{pk})\\ b \leftarrow \{0,1\}\\ b' \leftarrow A(\mathtt{pk},\widehat{\boldsymbol{q}}^b)\end{array}\right] < \frac{1}{2} + \mathrm{negl}(\lambda),$$

*kde* $\boldsymbol{q}^0,\boldsymbol{q}^1 \in \mathcal{P}^{m(\lambda)}$.

*Důkaz.* Idea důkazu je následující: Z protivníka $A$, který umí rozeznávat zašifrované vektory, potřebujeme zkonstruovat protivníka $A'$, který bude schopen rozeznávat dva ciphertexty. Nejprve ukážeme, že z $\boldsymbol{q}^0$ a $\boldsymbol{q}^1$ lze vždy zkonstruovat dva vektory, které se liší jen v jedné složce a zároveň jejich ciphertexty je stále možné rozeznat pomocí $A$. Nebudeme ale obecně vědět, v které složce se vektory liší. Proto tuto pozici budeme hádat a ukážeme, že i přesto je tato metoda dostatečně spolehlivá k účelu vyvrácení sémantické bezpečnosti $\mathcal{E}$.

Mějme polynom $p(x) \in \mathbb{Z}[x]$ a útočníka $A$, pro kterého platí

$$\Pr\left[b' = b \;\middle|\; \begin{array}{r}(\mathtt{sk},\mathtt{pk}) \leftarrow \mathtt{Gen}(1^\lambda)\\ (\boldsymbol{q}^0,\boldsymbol{q}^1) \leftarrow A(\mathtt{pk})\\ b \leftarrow \{0,1\}\\ b' \leftarrow A(\mathtt{pk},\widehat{\boldsymbol{q}}^b)\end{array}\right] > \frac{1}{2} + \frac{1}{p(\lambda)}.$$

Položíme

$$\begin{aligned}\mathrm{Succ}(\boldsymbol{p}) &:= \Pr\left[A(\mathtt{pk},\widehat{\boldsymbol{p}}) = 1 \;\middle|\; (\boldsymbol{q}^0,\boldsymbol{q}^1) \leftarrow A(\mathtt{pk})\right],\\ Q &:= (\boldsymbol{q}^0,\boldsymbol{q}^1),\\ \boldsymbol{p}_Q^0 &:= \boldsymbol{q}^0,\\ \forall i \in [m-1] \;:\; \boldsymbol{p}_Q^i &:= (q_1^0,\ldots,q_{i-1}^0,q_i^1,q_{i+1}^1,\ldots,q_m^1)^\top \text{a}\\ \boldsymbol{p}_Q^m &:= \boldsymbol{q}^1.\end{aligned}$$

Nyní dokážeme, že pro každé $\lambda \in \mathbb{N}$ a každý pár klíčů $(\mathtt{sk},\mathtt{pk}) \in \mathtt{Gen}(1^\lambda)$

$$\exists i \in \{0,\ldots,m\} : \mathrm{Succ}(\boldsymbol{p}_Q^i) - \mathrm{Succ}(\boldsymbol{p}_Q^{i-1}) > \frac{1}{mp(\lambda)}.$$

Pro spor předpokládejme, že existuje $\lambda \in \mathbb{N}$ a pár klíču $(\mathtt{sk},\mathtt{pk}) \in \mathtt{Gen}(1^\lambda)$, pro které

$$\forall i \in \{0,\ldots,m\} : \mathrm{Succ}(\boldsymbol{p}_Q^i) - \mathrm{Succ}(\boldsymbol{p}_Q^{i-1}) \leq \frac{1}{m(\lambda)p(\lambda)}.$$

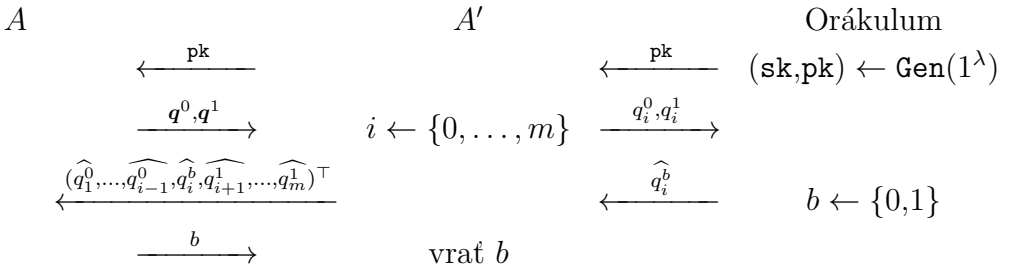
Pak ale

$$\begin{aligned}\frac{1}{p(\lambda)} <& \mathrm{Succ}(\boldsymbol{q^1}) - \mathrm{Succ}(\boldsymbol{q}^0) =\\ =& \mathrm{Succ}(\boldsymbol{p}_Q^{m(\lambda)}) - \mathrm{Succ}(\boldsymbol{p}_Q^0) =\\ =& \sum_{i=1}^{m(\lambda)} (\mathrm{Succ}(\boldsymbol{p}_Q^i) - \mathrm{Succ}(\boldsymbol{p}_Q^{i-1})) \leq\\ \leq& \sum_{i=1}^{m(\lambda)} \frac{1}{m(\lambda)p(\lambda)} = \frac{1}{p(\lambda)}\end{aligned}$$

První nerovnost protivníkovy úspěšnosti. Pro toto $i$,$\lambda$, $\mathtt{pk}$ a $\mathtt{sk}$ bychom mohli zkonstruovat útočníka $A'$ s úspěšností $\frac{1}{m(\lambda)p(\lambda)}$ takto:

$$A'(\mathtt{pk}) := A(q_i^0, q_i^1)$$
$$A'(\mathtt{pk},\widehat{q_i^b}) := A(\mathtt{pk},(\widehat{q_1^0},\ldots,\widehat{q_{i-1}^0},\widehat{q_i^b},\widehat{q_{i+1}^1},\ldots,\widehat{q_m^1})^T).$$

My ale víme jenom, že $i$ existuje, nikoliv jeho hodnotu. Proto potřebujeme sestrojit $A'$ nezávisle ma $i$. Cesta, jak se této nezávislosti docílit, je, že necháme $A'$ volit index $i$ náhodně. Celá redukce bezpečnostních experimentů vypadá takto:




Vzhledem k tomu, že $A'$ uhádne správný index s pravděpodobností $1/m(\lambda)$, lze jeho úspěšnost zezdola ostře omezit funkcí $\frac{1}{(m(\lambda))^2 p(\lambda)}$. To je nezanedbatelná úspěšnost. Proto $A'$ vyvrací sémantickou bezpečnost $\mathcal{E}$.

□

**Konstrukce 6.** *Mějme*

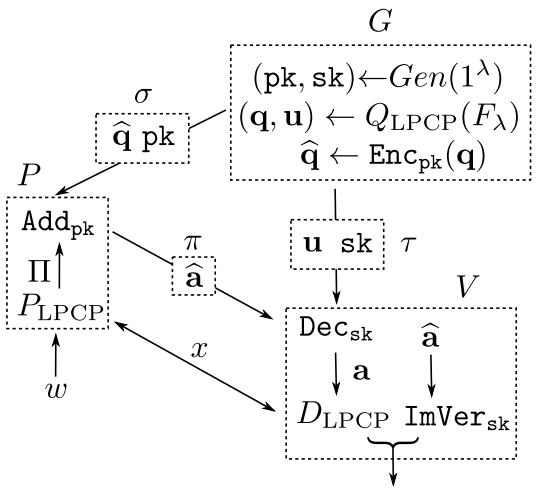
- *posloupnost konečných těles* $(\mathbb{F}_\lambda)_{\lambda\in\mathbb{N}}$ *takovou, že* $|F_\lambda| \in [2^\lambda, 2^{\lambda+1}]$,
- *aritmetický obvod* $C(x,w)$
- *1-dotazové LPCP* $(P_{LPCP},V_{LPCP})$ *pro relaci* $\mathcal{R}_C$ *nad tělesem* $F_\lambda$ *s délkou dotazů* $n$ *a délkou odpovědí* $m$,
- *semanticky bezpečné linear-only encryption* $\mathcal{E} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Add}, \mathtt{ImVer})$ *s linear only vlastností.*

*Zkounstuujeme trojici algoritmů* $(G,P,V)$ *následovně:*

**Generátor** $G(1^\lambda)$ *nejříve spustí* $Q_{LPCP}(\mathbb{F}_\lambda)$, *čímž získá dotaz* $\boldsymbol{q} \in \mathbb{F}_\lambda^n$ *a tajný stav* $\boldsymbol{u} \in \mathbb{F}_\lambda^{n'}$. *Dále vygeneruje pár klíčů* $(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{Gen}(1^\lambda)$ *a položí* $\sigma := (\mathbb{F}_\lambda,\mathtt{pk},\widehat{\boldsymbol{q}})$ *a* $\tau := (\mathbb{F}_\lambda,\mathtt{sk},\boldsymbol{u})$. *Výstup* $G$ *je dvojice* $(\sigma,\tau)$.

**Dokazovatel** $P(\sigma,x,w)$ *pro dané* $(x,w) \in \mathcal{R}_C$ *spustí* $P_{LPCP}(\mathbb{F}_\lambda,x,w)$, *čímž získá matici* $\Pi \in \mathbb{F}_\lambda^{m\times n}$. *Dále pomocí funkce* $\mathtt{Add}$ *homomorně spočte* $\widehat{\boldsymbol{a}} = \Pi\widehat{\boldsymbol{q}}$ *a vrátí* $\pi := \widehat{\boldsymbol{a}}$.

**Ověřovatel** $V(\tau,x,\pi)$ *nejprve* $\forall i \in [m]$ *ověří, že* $\mathtt{ImVer}_{\mathtt{sk}}(\widehat{\boldsymbol{a}}_i) = 1$ *a pokud ano, vrátí stejnou hodnotu, jako* $D_{LPCP}(\mathbb{F}_\lambda,x,\boldsymbol{u},\boldsymbol{a})$




Obrázek 1.6: konstrukce zk-SNARKu

**Věta 10.** *Ať* $(P_{LPCP},V_{LPCP})$ *je 1-dotazové LPCP s knowledge error* $\varepsilon(\lambda)$. $(G,P,V)$ *získáné z* $(P_{LPCP},V_{LPCP})$ *konstrukcí 6 je designated-verifier SNARK s adaptive knowledge error* $\varepsilon(\lambda) + \mathrm{negl}(\lambda)$.

*Důkaz.* Completeness takto zkonstruovaného SNARGu přímo vyplývá z completeness LPCP a korektnosti $\mathcal{E}$.

Nyní nastíníme efektivitu jednotlivých algoritmů. Celková diskuze výpočetních složitostí zkonstruovaných algoritmů je nad rámec této práce. Diskuzi výpočetní složitosti s použitím několika optimalizací jako FFT lze nalézt v [3].

Složitost $G$ a $P$ roste lineárně s velikostí $C$. Jejich složitost je $poly(\lambda + |C|)$.

Složitost $P$ závisí pouze na $|x|$ a $\lambda$. Nicméně, pokud chceme, aby knowledge error $\varepsilon(\lambda) = \frac{|C|}{|F_\lambda|}$ byla vždy menší než nějaká konstanta $c$, musíme s rostoucím obvodem zvyšovat i $\lambda$.

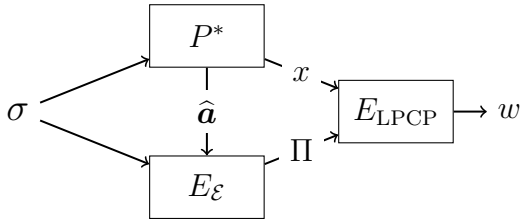
$$\frac{|C|}{|F_\lambda|} < c \quad \rightarrow \quad |C| < c|\mathbb{F}_\lambda| < c2^{\lambda+1} \quad \rightarrow \quad log|C| - 1 < \lambda,$$

proto $\lambda$ musí růst lineárně s $\log|C|$. Čili složitost $P$ je $poly(\lambda + |x| + \log|C|)$.

Velikost výsledného argumentu je 6 prvků tělesa $\mathbb{F}_\lambda$. Tedy $|\pi| = O(\lambda + \log|C|)$.

Dále chceme ověřit, že takto zkonstruovaný SNARG má adpative knowledge soundness. Zkonstruujeme extraktora $E$ takto: Mějme $x$ dané. Pro daného protivníka $P^*$, který pro dokazovací klíč $\sigma = (\mathbb{F}_\lambda, \mathtt{pk}, \hat{\boldsymbol{q}})$ a dodatečný vstup $z \in \{0,1\}^{poly(\lambda)}$ vytvořil argument $\pi = \hat{\boldsymbol{a}}$

1. E použije $E_{\mathcal{E}}^{P^*}(\mathtt{pk}, \hat{\boldsymbol{q}}, \hat{\boldsymbol{a}}, x, z)$ k získání matice $\Pi$ (kde $x$ je součástí dodatečného vstupu),
2. E použije $E_{\mathrm{LPCP}}^{\Pi}(x))$ k získání svědka $w$ t.ž. $(x,w) \in \mathcal{R}$.



Ukážeme, že je tako konstrukce korektní. Mohlo se totiž stát, že jsme nesplnili předpoklady pro použití $E_{\text{LPCP}}$ v druhém kroku. Tedy potřebujeme ukázat, že

$$\Pr\left[D_{\text{LPCP}}(x,\boldsymbol{u},\boldsymbol{a}) = 1 \;\middle|\; \begin{array}{r}(\sigma,\tau) \leftarrow G(1^\lambda)\\ \widehat{\boldsymbol{a}} \leftarrow P^*(\sigma,x)\\ \Pi \leftarrow E_{\mathcal{E}}^{P^*}(\mathtt{pk},\widehat{\boldsymbol{q}},\widehat{\boldsymbol{a}})\end{array}\right] > \varepsilon_{\text{LPCP}}, \tag{1.10}$$

kde $\boldsymbol{u}$, které $V_{\text{LPCP}}$ přijímá, je součástí $\tau$ a $\widehat{\boldsymbol{q}}$, které příjíma $E_{\mathcal{E}}^{P^*}$, je součástí $\sigma$.

Zaprvé víme, že se zanedbatelným počtem výjimek, díky linear-only vlastnosti $\mathcal{E}$, extraktor $E_{\mathcal{E}}$ vrátí matici $\Pi$ takovou, že $\Pi\boldsymbol{q} = \boldsymbol{a}$.

Dále potřebujeme ukázat, že takto vyextrahované zobrazení nejen že přesvědčí ověřovatele $D_{\text{LPCP}}$ pro toto jedno $\boldsymbol{q}$, ale pro všechna $\boldsymbol{q'} \leftarrow Q_{\text{LPCP}}(1^\lambda)$ až na zanedbatelný počet výjimek. Kdyby tomu tak nebylo, uměli bychom pomocí $E_{\mathcal{E}}$ rozlišit ciphertexty od nějakých $\boldsymbol{q}_1,\boldsymbol{q}_2 \in \mathbb{F}^n$. Podle lemmatu 9 by tedy šlo $E_{\mathcal{E}}$ použít k prolomení sémantické bezpečnosti $\mathcal{E}$.

Tím dostáváme platnost rovnice 1.10 a proto můžeme použít $E_{\text{LPCP}}$ extraktora na $\Pi$ k získání hledaného svědka $w$.

□

**Pozorování 11.** *Pro $|\mathbb{F}_\lambda| \in [2^\lambda,2^{\lambda+1}]$ je funkce $\frac{1}{|\mathbb{F}_\lambda|}$ zanedbatelná v $\lambda$. Proto je simulation error $\varepsilon_s = \frac{m}{|\mathbb{F}_\lambda|}$ u LPCP z konstrukce 2 rovněž zanedbatelná funkce v $\lambda$.*

**Pozorování 12.** *Ať $(P_{LPCP},V_{LPCP})$ je HVPZK-LPCP s simulation error $\varepsilon_s(\lambda) = \text{negl}(\lambda)$. Pak $(G,P,V)$ získáné z $(P_{LPCP},V_{LPCP})$ konstrukcí 6 je zero-knowledge SNARK.*

*Důkaz.* Mějme simulátor $S$ dosvědčujícího zero-knowledge LPCP. Setrojíme simulátor $S'$ pro SNARK takto:

1. $S'$ dostane vstup $1^\lambda$.
2. $S'$ vygeneruje parametry $(\sigma,\tau)$ stejně, jako $G$ z konstrukce 6 a uschová si nezašifrovaný stavový vektor $\mathtt{trap} = \boldsymbol{u}$.
3. $S'$ dostane $x$ a spustí $S(x,u)$ čímž dostane, až na zanedbatelný počet výjimek, vektor $(a_1,\ldots,a_6)$. Jinak $S'$ vrátí $\perp$.
4. $S'$ zašifruje $(a_1,\ldots,a_6)$ a vrátí $\pi = (\widehat{a}_1,\ldots,\widehat{a}_6)$.

Vzhledem k tomu, že jsou původní distribuce přepisu protokolu a simulace identické, jsou i tytéž zašifrované distribuce identické. Proto je $(G,P,V)$ zero-knowledge SNARK.

□

# 2. Aplikace zk-SNARK

*'Just the place for a Snark! I have said it twice:*
*That alone should encourage the crew.*
*Just the place for a Snark! I have said it thrice:*
*What I tell you three times is true.'*

— Lewis Carol, *The Hunting of the Snark*

V této kapitole se nejprve budeme zabývat problémem anonymity Bitcoinu a poté popíšeme, jak tento problém řeší Zcash pomocí zk-SNARK. Uvidíme přitom, že i když se zk-SNARK jeví jako mocný nástroj, jejich slabinou skutečnost, že k jejich implementaci potřebujeme, aby nějaká důvěryhodná autorita vygenerovala veřejné parametry $(\sigma,\tau)$.

V druhé části této kapitoly vybudujeme model mezilidské důvěry, který nazýváme Síť důvěry. Pro tento model pak popíšeme způsob, jak se uživatelé Sítě důvěry pomocí zk-SNARK mohou autentizovat a přitom zůstat anonymní v rámci nějakého okruhu ostatních uživatelů.

## 2.1 Krátce o kryptoměnách

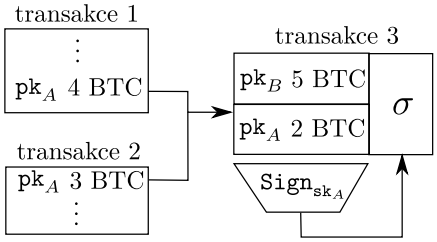
V roce 2009 pod pseudonymem Shatoshi Nakamoto byl zveřejněn článek [4] vysvětlující, jak pomocí několika kryptografických primitiv vytvořit digitální měnu, která ke svému fungování nevyžaduje přítomnost centrální autority. Zároveň s tímto článkem vznikla první taková měna se jménem Bitcoin.

Dvě součásti bitcoinové sítě tvoří uživatelé, kteří tvoří transakce, a těřaři, kteří zajišťují, že je těžké upravovat záznam všech již stvrzených transakcí. Těžaři spojí příchozí transakce do většího bloku a tento blok potvrdí vyřešením výpočetně náročné operace [1]. Za nalezení nového bloku je těžař odměněn nově vzniklými bitcoiny. Každý nový blok navazuje na předchozí. Čím delší řetězec bloků na blok navazuje, tím méně je pravděpodobné, že tento blok někdy někdo dokáže přepsat. Výsledný řetězec všech bloků se nazývá blockchain. Bezpečnost bitcoinu, tedy neměnost blockchainu, je podmíněná předpokladem, že nikdo v síti nevlastní více než polovinu výpočetního výkonu celé sítě.

V současnosti existuje mnoho dalších návrhů, jak tvořit decentalizovanou databázi s neměnnou historií. Rozdíly mezi nimi ale nejsou pro tuto práci důležité. Vystačíme si s představou, že blockchain je způsob, jak i bez centrální autority vytvořit záznam o transakcích v síti, jenž je těžké zpětně měnit.

Důležitý pro nás bude především princip, jak na sebe transakce navazují. Každá transakce se skládá ze vstupů a výstupů, přičemž každý vstup odpovídá výstupu některé již předchozí transakce a každý výstup je dvojice veřejného klíče příjemce a zaslané částky. Aby odesílatel mohl použít nějaký výstup potvrzené transakce jako vstup nové transakce, musí ji autorizovat podpisem korespondujícím s veřejným klíčem u tohoto výstupu, jako na obrázku 2.1.

[1] Toto schéma se nazývá Proof of Work. Toto je jedna z možných implementací: Ať $H$ je kryptograficky bezpečná hashovací funkce, $b$ je hash všech transkací v novém bloku, $p$ je hash posledního potvrzeného bloku a $d$ je parametr obtížnosti. Pak potvzení nového bloku znamená najít $n$ takové, že $H(p||b||n) < d$.




Obrázek 2.1: Bicoinová transakce

## 2.2 Anonymita Bitcoinu

Bitcoin je občas chybně označován jako anonymní měna. Ve skutečnosti je celá historie transakcí bitcoinové sítě veřejná i spolu s částkami, odesílateli a příjemci. Příjemce a odesílatele identifikujeme pomocí veřejných klíčů (adres). Jakmile známe něčí adresu (např. nám někdo pošle bitcoiny), můžeme sledovat jeho příchozí a odchozí transakce. Bitcoin tedy není anonymní, ale pseudonymní, kde uživatelův pseudonym je právě jeho adresa.

Kvůli snahám o regulaci kryptoměn je v ČR od 1.1.2017 dle zákonu 368/2016 Sb., který upravuje zákon 253/2008 Sb. *o opatřeních proti legalizaci výnosů z trestné činnosti a financování terorismu*, nutné při nákupu kryptoměn v automatech nad 1000 euro poskytnout svoji identitu. Při nákupu ve směnárnách je to nutné vždy. To znamená, že v případě potřeby má policie možnost dohledat identitu původního vlastníka bitcoinů použitých ke konání trestné činnosti.

Uživatelé Bitcoinu můžou částečné anonymity docílit pomocí míchání svých mincí. Mějme skupinu lidí, kteří chtějí změnit své adresy tak, aby po záměně bylo těžké spojit jejich původní adresu s jejich novou adresou. Zvolí si nějakou důvěryhodnou třetí stranu a té všichni pošlou své bitcoiny a své nové adresy. Tato třetí strana následné pošle uživatelům bitcoiny zpět na jejich nové adresy. Přitom je důležité, aby každý uživatel zaslal stejné množství bitcoinů. Jinak bychom si mohli spojit jeho adresy dle částky, co mu na novou adresu přišla. Samozřejmě existují i chytřejší verze mixérů [14] s kryptografickými zárukami anonymity a toho, že bitcoiny zůstanou v rukou jejich původního vlastníka.

Sledovatelnost toku bitcoinů dosvědčuje článek [15], jehož autoři si stáhli celou databázi Bitcoinu a následně zmapovali všechny transakce s velkými objemi bitcoinů.

## 2.3 Zerocash

V této kapitole rozlišujeme Zerocash (návrh protokolu) a Zcash (implementace protokolu Zerocash). Protokol Zerocash [1] je navržen jako kompatibilní rozšíření Bitcoinu s cílem poskytnout anonymní transakce. Zerocash používá čtyři typy transakcí. Prvním typem jsou věřejné transakce strukturou identické bicoinovým transakcím. Dalším typem jsou tzv. skryté [2] transakce. To je blockchainová transakce, u které nevidíme vstupy, výstupy ani utracené částky. Zato je k ní přiložený zk-SNARK argument o tom, že je tato transakce korektní tj. splňuje následující:

[2] v originále *shielded transaction*, doslovně *krytá transakce*, v češtině nabývá jiného významu

**Návaznost:** Vstupy transakce odpovídají výstupům nějakých předchozích transakcí.

**Autentizace:** Použité vstupy náleží tvůrci transakce.

**Čerstvost:** Vstupy transakce ještě nebyly utraceny.

**Rovnováha:** Součet částek na vstupu je roven součtu částek na výstupu.

Nakonec Zerocash poskytuje dva typy poloskrytých transakcí, které převádí veřejně viditelné mince na skryté a obráceně.

Nyní popíšeme, jak Zerocash funguje. Vyhneme se přitom technickým podrobnostem, které čtenář nalezne v [1]. Budeme potřebovat následující kryptografická primitiva:

- $H : \{0,1\}^* \to \{0,1\}^\ell$ kolizivzdornou hashovací funkci a
- $(G,P,V)$ zk-SNARK.

Pro jednoduchost budeme v tomto textu uvažovat blockchain, ve kterém jsou pouze skryté transakce. Konstrukce zbylých tří typů transakcí je spíše technická záležitost. Slovem transakce od nyní označujeme skrytou transakci. Dále pro jednoduchost popíšeme jen transakci, která má právě dva vstupy a dva výstupy. Struktura transakce s větším či menším počtem vstupů a výstupů je analogická.

**Definice 16.** *Transakce* `tx` *je* $(\pi, s_1^{\text{old}}, s_2^{\text{old}}, c_1^{\text{new}}, c_2^{\text{new}})$ *kde* $s_1^{\text{old}}, s_2^{\text{old}}, c_1^{\text{new}}, c_2^{\text{new}} \in \{0,1\}^\ell$ *a* $\pi$ *má typ shodný s výstupem zk-SNARK dokazatele* $P$*.*

Transakce se tedy skládá jen ze čtveřice hashů a důkazu nějakého tvrzení. Pojďme se nejdříve z vrchu podívat, jak Zerocash zaručuje *návaznost*, *autentizaci*, *čerstvost* a *rovnováhu* každé transakce.

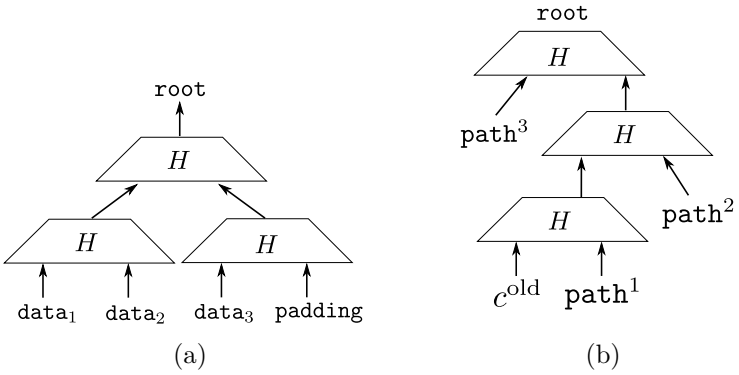
**Návaznost:** Do blockchainu ukládáme pouze hashe transakcí. Hash transakce budeme nazývat $c$ jako commitment. S každým novým blokem spočteme hašovací strom ze všech dosud potvrzených transakcí a uložíme si jeho kořen. Když chce uživatel použít výstup transakce s hashem $c^{\text{old}}$ jako vstup nějaké nové transakce, stačí mu prokázat, že existuje cesta od kořene hashovacího stromu k listu $c^{\text{old}}$. Díky zero knowledge už ale nemusí odhalit jednotlivé úseky této cesty a tím pádem ani to, o jakou transakci se jedná.

**Autentizace:** Stačí použít nějaké podpisové schéma. Zero knowledge našeho SNARKu nabízí toto schéma: Uživatel si zvolí soukromý klíč `sk` z rovnoměrného rozložení na $\{0,1\}^\ell$. Veřejný klíč k `sk` je `pk` $:= H(sk)$. Uživatel se autentizuje prokázaním znalosti vzoru k `pk` při funkci $H$. Díky zero knowledge přitom neodhalí `sk` a vzhledem k tomu, že `pk` bude zakrytý pod hashem $c$, nemusí odhalit ani `pk`.

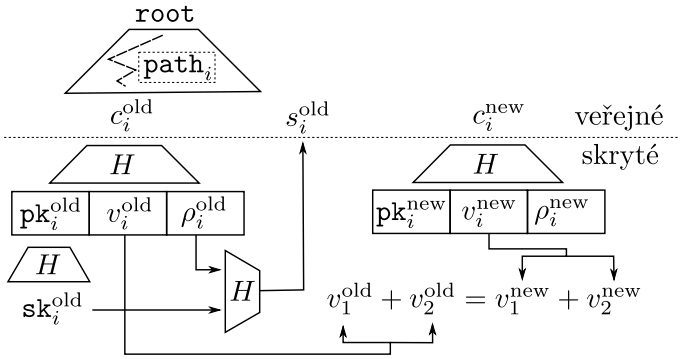
**Čerstvost:** Každý výstup má navíc sériové číslo $s$. Pro utracení vstupu musí být toto číslo odhaleno. Pokud spolu s každým vstupem transakce nebylo odhaleno nové sériové číslo, transakce je zamítnuta.

**Rovnováha:** Částky reprezentujeme 32-bitovými integery. To, že součet vstupů je roven součtu výstupů, opět může bez vyzrazení příslušných částek dokázat pomocí zk-SNARKu. Musíme přitom kontrolovat přetečení.

Nyní zadefinujeme tvrzení, jehož zk-SNARK $\pi$ je přiložen k transakci.




Obrázek 2.2: (a) hashovací strom (b) svědek `path`




Obrázek 2.3: schéma zerocash `POUR`

**Definice 17.** *Tvrzení* `POUR`$(x,w)$ *je definováno následovně:*

**Instance:** $x = (r,s_1^{\text{old}},s_2^{\text{old}},c_1^{\text{new}},c_2^{\text{new}})$ *kde* $r \in \{0,1\}^\ell$ *je kořen hashovacího stromu transakcí tj. společný veřejný parametr*

**Svědek:** $w = (\mathtt{sk}_1,\mathtt{sk}_2,\mathtt{path}_1,\mathtt{path}_2,c_1^{\text{old}},c_2^{\text{old}},v_1^{\text{old}},v_2^{\text{old}},v_1^{\text{new}},v_2^{\text{new}},\rho_1^{\text{old}},\rho_2^{\text{old}},\rho_1^{\text{new}},\rho_2^{\text{new}})$

**Tvrzení:** $\forall i \in \{1,2\}$

1. $\mathtt{path}_i$ *je validní cesta od kořenene* $r$ *k listu* $c_i^{\text{old}}$
2. $c_i^{\text{old}} = H(H(\mathtt{sk}_i)||v_i^{\text{old}}||\rho_i^{\text{old}})$
3. $c_i^{\text{new}} = H(\mathtt{pk}_i^{\text{new}}||v_i^{\text{new}}||\rho_i^{\text{new}})$
4. $s_i^{\text{old}} = H(\mathtt{sk}_i||\rho_i^{\text{old}})$
5. $v_i^{\text{new}} \leq v_{\max}$
6. $v_1^{\text{old}} + v_2^{\text{old}} = v_1^{\text{new}} + v_2^{\text{new}}$

Když omezíme počet vstupů transakce na $2^6$, vhodná hodnota pro $v_{\max}$ je $2^{32-6} = 2^{26}$. Popsali jsme konstrukci anonymních transakcí v Zerocash. Na obrázku 2.3 schéma tvrzení `POUR`. Nyní se zaměříme na parametry konkrétní implementace Zerocash s názvem Zcash (popsaná v [1]).

Jako první nás bude zajímat počet součinových hradel v obvodu pro tvrzení `POUR`. Zcash používá hashovací funkci SHA256, kterou je možné spočíst po bitech

| # | část obvodu pro POUR | počet hradel |
|---|---|---|
| 1 | $c_1^{\text{old}}$,$c_2^{\text{old}}$ jsou v hašovacím stromu s kořenem $r$ | $2 \times 64 \times 28161$ |
| 2 | výpočet $c_1^{\text{old}}$,$c_2^{\text{old}}$,$c_1^{\text{new}}$,$c_2^{\text{new}}$ | $2 \times 111316$ |
| 3 | výpočet $c_1^{\text{new}}$,$c_2^{\text{new}}$ | $2 \times 83712$ |
| 4 | výpočet $s_1^{\text{old}}$,$s_2^{\text{old}}$ | $2 \times 27904$ |
| 5 | kontrola $v_1^{\text{old}} < v_{max}$ a $v_2^{\text{old}} < v_{max}$ | $2 \times 65$ |
| 6 | kontrola $v_1^{\text{old}} + v_2^{\text{old}} = v_1^{\text{new}} + v_2^{\text{new}}$ | 1 |
| | pomocná hradla | 2384 |
| | celkem | 4052987 |

Tabulka 2.1

| Algoritmus | velikost výstupu | Intel Core i7-2620M 2.70GHz 12GB RAM 1 vlákno | Intel Core i7-4770M 3.40GHz 16GB RAM 1 vlákno | Intel Core i7-4770M 3.40GHz 16GB RAM 4 vlákna |
|---|---|---|---|---|
| Generování společných parametrů | dokazovací klíč 896MB, ověřovací klíč 749 B | 7 min 48 s | 5 min 11 s | 1 min 47 s |
| Generování důkazu | 288 B | 2 min 55s | 1 min 59 s | 46 s |
| Ověření důkazu | | 8.5 ms | 5.4 ms | 5.4 ms |

Tabulka 2.2: Časová náročnost Zcash

obvodem s 27 904 násobicími hradly. Tabulka 2.1 uvádí počty hradel v jednotlivých částech obvodu pro POUR, přičemž tyto hodnoty přejímáme hodnoty z [1].

Dále přikládáme tabulku časové složitoti algoritmů a velikostí klíču a důkazu. Hodnoty jsou opět převzaté z [1]. Vygenerovat společné paremetry trvá dlouho, ale vzhledem k tomu, že je stačí vygenerovat jen jednou, toto není kritické. Zarážející je velikost dokazovacího klíče, která se blíží 1 GB. Čas potřebný k vygenerování důkazu korektnosti jedné transakce je kritický a pohybuje se v jednotkách minut. I paměťová složitost tohoto procesu je vysoká. Zcash dělá měnu nepoužitelnou pro každodenní transakce. Naopak velmi příznivá je časová složitost ověření korektnosti transakce, která se pohybuje v jednotkách milisekund.

Nakonec je třeba zmínit, že bezpečnost Zcash závisí na způsobu, kterým byly vygenerovány veřejné parametry, tedy dokazovací a ověřovací klíč, dále jen CRS (Common Reference String). Přímo z definice zero-knowledge SNARKu totiž vyplývá, že ten, kdo vygeneroval CRS, může vytvořit platný důkaz pro libovolnou instanci $x$ bez znalosti svědka $w$. V případě Zcash by si tedy tato osoba mohla nagenerovat libovolné množství mincí.

Proto autoři Zcash nenechali výpočet CRS na jedné osobě, ale uspořádali tzv. Tau ceremonii, během které 6 kryptografů z různých míst na světě spočetlo CRS na počítačích odpojených od všech sítí pomocí Multi-party computation[3] [16].

[3] Mějme $n$ stran a funkci $F : \mathbb{F}^n \to \mathbb{F}$. Každá strana $i$ má své $x_i \in \mathbb{F}$. Multi-party compu-

Po zveřejnění výsledných CRS a transkriptu veškeré komunikace mezi účastníky výpočtu, byly tyto počítače zničeny spolu s daty, která by mohla být použita k falzifikaci důkazů v Zcash síti.

Přes veškeré snahy autorů Zcash vytvořit kvalitní CRS je důvěryhodnost výsledku této ceremonie stále zpochybňována. Do budoucna očekáváme v oblasti anonymních kryptoměn nahrazení zk-SNARK pomocí zk-STARKs [17], jejichž kvalita CRS je veřejně ověřitelná pomocí kryptografie založené na kódech. Navíc jsou zk-STARKs odolné proti kvantovým algoritmům.

Alternativním návrhem kryptoměny s anonymními transakcemi je CryptoNote [5]. Výhodou Cryptonote je, že se obejde bez inicializační fáze generování CRS. V rámci každého bloku využívá kombinaci ring signatures [6], one-time public keys [5] a tvz. confidental transactions [18] k utajení návaznosti transakcí a utracených částek. Díky ring signatures může odesílatel ke vstupům transakce přidat 20 dalších nesouvisejících vstupů a následně transakci podepsat tak, aby nebylo jasné, jaké vstupy byly použity. Příkladem kryptoměny založené na CryptoNote protokolu je Monero. ZCash jednoznačně dominuje nad Monerem mírou mixování tansakcí. Počet transakcí, se kterými se míchá nová Monero transakce, je totiž kvůli výpočetní složitosti ring signatures nízký. Zatímco v Moneru se transakce míchá s dalšími 20 transakcemi, v Zerocash je každá skrytá transakce smíchána s každou následující skrytou transakcí.

## 2.4 Síť důvěry

V této kapitole provedeme diskuzi nad tím, jak modelovat důvěru, jaké možnosti by nám přineslo zdigitalizování důvěry, a nakonec, jak lze pomocí zk-SNARK v našem modelu sítě důvěry zaručit částečnou anonymitu.

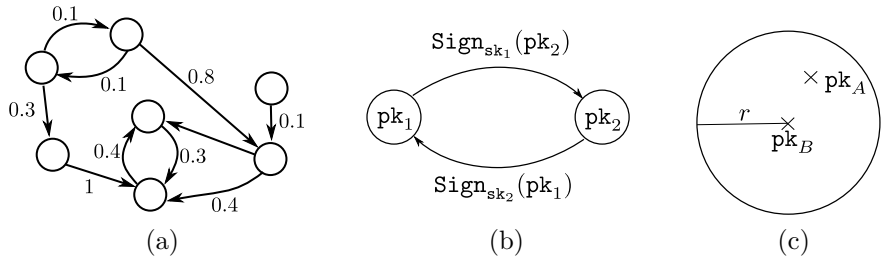
Pokusme se vytvořit nějaký model mezilidské důvěry. Jako vhodná struktura se nabízí orientovaný graf s váženými hranami, kde vrcholy symbolizují jednotlivce a váha hrany z A do B znamená, jak moc A důvěřuje B. Tento graf budeme nazývat síť důvěry.

**Definice 18.** ***Síť důvěry*** $\mathcal{N}$ *je trojice* $(G,w,\texttt{data})$*, kde* $G = (V,E \subset V^2)$ *je orientovaný graf na nějaké množině* $V$*,* $w : E \to [0,\infty)$ *je váhová funkce a* $\texttt{data} : E \cup V \to \{0,1\}^*$ *jsou jakákoliv další data k vrcholům a hranám* $G$*.*

**Značení.** *Pro síť důvěry* $\mathcal{N} = (G,w,\texttt{data})$ $v \in \mathcal{N}$ *znamená* $v \in V(G)$*.*

Zajímavá vlastnost důvěry je určitá tranzitivita: když A důvěřuje B a B důvěřuje C, tak i A má sklon s určitou mírou důvěřovat C. Důvěru A k C nazveme **nepřímou**. Vede-li od A k B hrana s velkou vahou, nazveme důvěru, kterou symbolizuje, **přímou** důvěrou. V současné době velkou roli v budování důvěry hraje stát. My věříme státu, stát provádí kontroly a tuto důvěru přenáší na další instituce a osoby. Diskuze toho, jaké výhody a nevýhody má síť důvěry se silnou centrální autoritou, je nad rámec této práce. My se budeme zabývat tím, jak bychom mohli pomocí kryptografických primitiv a blockchainu takovou síť vybudovat i bez centrální autority.

tation je kryptografický protokol, který umožňuje spočíst $F(x_1,\ldots,x_n)$ tak, aby se žádná strana nedozvěděla o zbývajích vstupech víc, než lze zjistit ze znalosti funkce $F$, výsledné hodnoty a lokálního $x_i$.




Obrázek 2.4: (a) síť důvěry (b) podepisování hran (c) důvěryhodné okolí

V první řadě můžeme použít asymetrickou kryptografii k vybudování decentralizované autentizace v této síti. Mějme nějaké podpisové schéma (`Gen`,`Sign`,`Ver`). `Gen` je generátor klíčů, `Sign` je podepisovací algoritmus a pomocí `Ver` lze ověřovat podpisy. Vrcholy $V$ budou veřejné klíče. Každý, kdo se chce zapojit do síťe důvěry, jednoduše zveřejní svůj veřejný klíč. Aby hrana z Alice k Bobovi byla platná, musí ji Alice podepsat svým soukromým klíčem, čili $\texttt{data}((A,B)) = \texttt{Sign}(\texttt{sk}_A,\texttt{pk}_B)$. Tato topologie z veřejných klíčů a podpisů klíčů bude veřejná informace.

Na stejném principu je založena PGP Web of Trust[19]. Sémantickýcm rozdílem je, že v PGP Web of Trust uživatel podpisem potvrzuje důvěryhodnost certifikátu druhého uživatele, nikoliv samotnou důvěryhodnost majitele certifikátu. Také výpočet důvěryhodnosti certifikátu neodpovídá modelu důvěry, protože nezavisí na ověřovateli. Model předpokládá, že graf sítě je silně spojitý a tedy se lze dostat z každého vrcholu do každého. Pro každého uživatele najdeme nejkratší cestu k cílovému uživateli. Důvěryhodnost certifikátu cílového uživatele je pak průměr délek těchto cest.

**Značení.** *Pro každý index $i$* $\texttt{pk}_i$ *je veřejný klíč odvozený od soukromého klíče* $\texttt{sk}_i$ *a naopak* $\texttt{sk}_i$ *je soukromý klíč, ke kterému náleží veřejný klíč* $\texttt{pk}_i$.

Nepřímou důvěru pak reprezentujeme pomocí jakékoliv funkce $d$, která pro síť důvěry a dva její vrcholy vrátí nějaké nezáporné reálně číslo. Pro přirozený jazyk by bylo příjemnější, kdyby míra nepřímé důvěry byla přímoúměrná funkci $d$. Z matematického hlediska je mnohem přirozenější představovat si ji jako metriku, čili s rostoucí hodnotou míra nepřímé důvěry klesá. Budeme se držet matematické intuice. $d$ tedy vyjadřuje nedůvěru mezi dvěma uživateli sítě. Navíc $d$ budeme nesprávně nazývat *metrika*, i když nepožadujeme, aby splňovala vlastnosti metriky už jenom proto, že náš graf důvěry je orientovaný.

Přirozenou *metrikou* $d$ je nejkratší vzdálenost mezi dvěma vrcholy ve váženém orientovaném grafu. Označíme ji $d_g$.

Dále pomocí *metriky* $d$ můžeme v síti důvěry $\mathcal{N}$ nadefinovat důvěryhodné okolí s poloměrem $r \in \mathbb{R}$ jako

$$B_d^{\mathcal{N}}(u,r) := \{v \in V(G) \mid d(\mathcal{N},u,v) \leq r\}.$$

Vytvořili jsme pseudonymní prostor. To znamená, že pokud Alici někdo dokáže spojit s jejím veřejným klíčem, může se i podívat, komu důvěřuje. To je citlivá informace. Předpokládejme, že Alice zná veřejný klíč Boba. Dokud si Alice potřebuje jen ověřit, že může důvěřovat Bobovi, stačí jí prohlédnout si topologii a spočíst hodnotu nepřímé důvěry. Nemusí přitom nikomu vyzrazovat svůj veřejný

klíč. Co když ale zároveň potřebuje Bobovi dokázat, že Bob může důvěřovat jí? Může to udělat bez vyzrazení svého veřejného klíče? Formálněji: může Alice Bobovi dokázat, že $\mathtt{pk}_A \in B_{d_g}^{\mathcal{N}}(\mathtt{pk}_B,3)$ bez toho, aby se Bob dozvěděl o jejím klíči cokoliv navíc?

S pomocí zk-SNARK toho lze docílit. Pro jednoduchost nyní uvažujeme případ, ve kterém mají všehny hrany váhu 1. Nejprve zkonstuujeme obvod $C(x,w)$ pro tvrzení:

$$\text{znám } \mathtt{sk}_A \text{ takové, že } d_g(\mathcal{N},\mathtt{pk}_B,\mathtt{pk}_A) \leq 3. \tag{2.1}$$

Alice prokáže, že zná $\mathtt{sk}_A$ tím, že podepíše nějakou Bobovu výzvu $r$. Po krátkém rozboru dostáváme kryptografičtější zápis tvrzení 2.1, které už lze přeložit do aritmetického obvodu:

$$\begin{aligned} & \mathtt{Ver}(\mathtt{pk}_0,\sigma_1,\mathtt{pk}_1) \\ \wedge\ & \mathtt{Ver}(\mathtt{pk}_1,\sigma_2,\mathtt{pk}_2) \\ \wedge\ & \mathtt{Ver}(\mathtt{pk}_2,\sigma_3,\mathtt{pk}_3) \\ \wedge\ & \mathtt{Ver}(\mathtt{pk}_3,\sigma_4,r), \end{aligned} \tag{2.2}$$

kde instance $x = (\mathtt{pk}_0,r)$, svědek $w = (\mathtt{pk}_1,\mathtt{pk}_2,\mathtt{pk}_3,\sigma_1,\sigma_2,\sigma_3,\sigma_4,\mathtt{sk})$ a $\mathtt{Ver}(\mathtt{pk},\sigma,m)$ je algoritmus ověřující, že $\sigma$ je validní podpis zprávy $m$ uživalem s veřejným klíčem $\mathtt{pk}$. Pro tento obvod pak vygenerujeme zk-SNARK parametry. Alice nyní už jen spočte důkaz tvrzení 2.2 a pošle ho Bobovi.

Alice Bobovi s veřejným klíčem $\mathtt{pk}_B$ dokazuje, že zná soukromý klíč $\mathtt{sk}_A$ k nějakému veřejnému klíči $\mathtt{pk}_A \in B_{d_g}^{\mathcal{N}}(\mathtt{pk}_B,3)$. Tím Bobovi vyzrazuje podmnožinu veřejných klíčů, ve které je i její veřejný klíč. Na druhou stranu díky zero-knowledge vlastnosti SNARKu se Bob nedozví o Alicině veřejném klíči nic víc, než tento fakt. Právě jsme pomocí zk-SNARK zkonstruovali schéma, které nazveme síťový podpis (network signature). Následující definice je inspirovaná definicí ring signatures z [20].

**Definice 19.** Síťový podpis *je čtveřice PPT* (`CRSGen`, `Gen`, `NetSign`, `NetVer`) *splňující tyto podmínky a syntax:*

**Syntax:** *Mějme síť důvěry $\mathcal{N}$, metriku v této síti $d$, uživatele sítě $\mathtt{pk}_B \in \mathcal{N}$ a $r \in \mathbb{R}$. Parametry síťového podpisu* `prm` *jsou čtveřice $(\mathcal{N},d,\mathtt{pk}_B,r)$ specifikující množinu uživatelů $U_{\mathtt{prm}} := B_d^{\mathcal{N}}(\mathtt{pk}_B,r)$, kteří jsou schopni vytvořit validní síťový podpis s parametry* `prm`*. Algoritmy mají nasledující vstupy a výstupy:*

| *algoritmus* | *vstupy* | *popis vstupu* | *výstupy* | *popis výstupu* |
|---|---|---|---|---|
| `CRSGen` | $1^\lambda$ | *bezp. parametr* | `crs` | *veřejné* |
| | $d$ | *metrika* | | *parametry* |
| `Gen` | $1^\lambda$ | *bezp. parametr* | `pk` | *veřejný klíč* |
| | | | `sk` | *soukormý klíč* |
| `NetSign` | `crs` | *veřejné parametry* | | |
| | `sk` | *soukormý klíč* | | |
| | $m$ | *zpráva* | $\sigma$ | *podpis* |
| | `prm` | *parametry podpisu definující okolí* $U_{\texttt{prm}}$ | | |
| `NetVer` | `crs` | *veřejné parametry* | | |
| | $m$ | *zpráva* | | $b = 1$ |
| | $\sigma$ | *podpis* | $b$ | $\leftrightarrow$ |
| | `prm` | *parametry podpisu definující okolí* $U_{\texttt{prm}}$ | | *$\sigma$ je validní* |

*Vrcholy sítě $\mathcal{N}$ jsou veřejné klíče vygenererované generátorem* `Gen`.

**Completeness:** *Pro každé parametry* $\texttt{prm} = (\mathcal{N},d,\texttt{pk}_B,r)$*, každou zprávu* $m$ *a každý* $sk_A$ *takový, že* $\texttt{pk}_A \in U_{\texttt{prm}}$

$$\Pr\left[\texttt{NetVer}(\texttt{crs},m,\sigma,\texttt{prm}) = 1 \;\middle|\; \begin{array}{r} \texttt{crs} \leftarrow \texttt{CRSGen}(1^\lambda,d) \\ \sigma \leftarrow \texttt{NetSign}(\texttt{crs},\texttt{sk},m,\texttt{prm}) \end{array}\right] = 1.$$

**Unforgeability:** *Pro každé parametry* `prm` *takové, že* $|U_{\texttt{prm}}| \geq 1$*, má každý PPT útočník zanedbatelnou úspěšnost v této hře:*

1. *Orákulum* $\mathcal{O}$ *vygeneruje* `crs` *a pošle je* $A$*.*
2. $A(\texttt{crs},\texttt{prm})$ *klade dotazy následujících dvou typů.*
   - *Na dotaz* $(\texttt{SIGN},m,\texttt{pk}_i)$ *kde* $\texttt{pk}_i \in \mathcal{N}$ *orákulum spočte síťový podpis* $\sigma = \texttt{NetSign}(\texttt{crs},\texttt{sk}_i,m,\texttt{prm})$ *a pošle ho* $A$*. Navíc si uloží* $(m,\sigma)$ *do množiny* $\Sigma$*.*
   - *Na dotaz* $(\texttt{CORRUPT},\texttt{pk}_i)$ *kde* $\texttt{pk}_i \in \mathcal{N} \setminus U_{\texttt{prm}}$ *orákulum pošle* $A$ $\texttt{sk}_i$*.*
3. *Nakonec* $A$ *vrátí trojici* $(m',\sigma')$*.*
4. $A$ *vyhrál, pokud* $\texttt{NetVer}(\texttt{crs},m',\sigma',\texttt{prm}) = 1$ *a* $(m',\sigma') \notin \Sigma$*.*

**Anonymita:** *Pro každého PPT protivníka* $A$ *a každé parametry* `prm` *takové, že* $|U_{\texttt{prm}}| \geq 2$

$$\Pr\left[\begin{array}{c} b = b' \\ \wedge \\ \texttt{pk}_0 \neq \texttt{pk}_1 \\ \wedge \\ \texttt{pk}_0,\texttt{pk}_1 \in U_{\texttt{prm}} \end{array} \;\middle|\; \begin{array}{r} \texttt{crs} \leftarrow \texttt{CRSGen}(1^\lambda) \\ (\texttt{pk}_0,\texttt{pk}_1,m) \leftarrow A(\texttt{crs},\texttt{prm}) \\ b \leftarrow \{0,1\} \\ \sigma \leftarrow \texttt{NetSign}(\texttt{crs},\texttt{sk}_b,m,\texttt{prm}) \\ b' \leftarrow A(\texttt{crs},\texttt{prm},\sigma) \end{array}\right] < \frac{1}{2} + \text{negl}(\lambda)$$

Dále se podíváme, jak lze funkci nepřímé důvěry dále zpřesňovat. Rozdílné metriky včetně diskuze jejich vlastností lze nalézt například v [21] a [22]. Popíšeme, jak lze síťové podpisy konstruovat i pro složitější *metriky*.

**Utajení délky cesty:** Je zřejmé, že pokud Alice zná cestu délky právě 3 od Boba k ní, pak umí dokázat tvrzení 2.2. Když by mu chtěla tvrzení 2.2 dokázat pomocí kratší cesty, stačí jí, když vytvoří hranu sama k sobě a tou nahradí chybějící díly cesty.

**Více cest:** *Metrika* může také popisovat, že existuje více vrcholově disjunktních cest určité délky. Pak obvod $C$ zkonstuujeme spojením několika obvodů pro 2.2 a dodatečnou podmínkou, že vstupní klíče (vrcholy) různých obvodů jsou různé (samozřejmě s výjimkou počátečního a koncového vrcholu cesty).

**Váha cest:** Pokusíme se zachytit skutečnost, že důvěra není binární a že její míra se vzdáleností v síti důvěry klesá. V přesnějším modelu sítě důvěry má každá hrana navíc váhu v intervalu [0,1]. Váha cesty může být například

$$w(v_1,\ldots,v_n) = 1 - \prod_{i=1}^{n}(1 - w((v_{i-1},v_i))).$$

Svědek pro skutečnost, že Alice důvěřuje Cecílii s vahou $w(\mathtt{pk}_A,\mathtt{pk}_C)$ (nová hrana), je nyní $\mathtt{sign}(\mathtt{sk}_A, \mathtt{pk}_C||w)$. Novou součástí svědka $\boldsymbol{w}$ musí být váhy jednotlivých cest a novou součástí obvodu $C$ je mechanismus pro finální výpočet váhy cest/y.

**Utajení počtu cest:** Uvažme situaci, kdy Bob bude důvěřovat Alici, jen pokud mu prokáže, že (od něho k ní) existuje buď dostatek slabších cest nebo několik silných cest. Obvod pro takové tvrzení bude vždycky vyžadovat existenci mnoha cest. Má-li tedy Alice k dispozici jen několik silných cest, musí někde vzít zbylé cesty. Tento problém může vyřešit pomocích uměle vzniklých hran s nulovou váhou. Ideálně mohou v síti existovat autority, jimž všichni důvěřují nulovou váhou a jež všem důvěřují.

**Oboustranné utajení:** Ve všech předchozích bodech musel Bob Alici prozradit svůj veřejný klíč. Samozřejmě bychom chtěli, aby tomu tak nebylo, ale nepodařilo se nám vymyslet řešení, které by to nevyžadovalo. Je tu ale pár možností, jak toho docílit částečně.

První možností je, že si Alice a Bob zvolí někoho, komu oba dva důvěřují, Davida, a oba dva Davidovi prozradí svůj veřejný klíč. David si pak prohlédne topologii sítě důvěry, spočte důvěru Boba v Alici a pošle mu toto číslo s příslušným zk-SNARK. Alice ani Bob si sice nemohou být jistí, zda David zachová jejich pseudonymitu, ani jestli k výpočtu nepřímé důvěry použil svědka s nejvyšší hodnotou nepřímé důvěry. Zato má Bob kryptografickou záruku, že číslo, které od Davida dostal, je dolním odhadem jeho nepřímé důvěry k Alici.

Druhá možnost je, že Bob je součástí nějaké komunity, jejíž členové si navzájem důvěřují a chce, aby mu Alice dokázala, že je také součástí této komunity.

**Anonymita v komunitě:** Bob po Alici chce, aby mu dokázala, že je součástí nějaké komunity. Vybere $n$ náhodných členů komunity a chce, aby mu Alice ukázala, že $\mathtt{pk}_A \in \bigcup_{i=1}^{n} B_d^{\mathcal{N}}(\mathtt{pk}_i,r_i)$.

Dále bychom chtěli, aby v síti důvěry bylo možné hrany nejen přidávat, ale i odebírat. Nabízíme dva způsoby, jak to umožnit.

**Expirace hrany:** Součástí nové hrany je doba její platnosti.

**Hrany v blockchainu:** Všechny hrany sítě důvěry jsou uloženy v blockchainu. Tento blockchain má dvě transakce: `Pridej_hranu` a `Odeber_hranu` . Pokud chce dokazovatel nějakou hranu použít jako součást svědka pro nějaké tvrzení, prokáže její existenci pomocí existence cesty v hashovacím stromu všech platných hran v blockchainu (od kořene k hraně) stejně jako v Zerocash protokolu.

Ukázali jsme několik způsobů, jak počítat nepřímou důvěru a jak prokázat svoji náležitost k nějaké podmnožině veřejných klíču pomocí zk-SNARK. Nyní uvedeme několik praktických aplikací sítě důvěry. I když jsou důvěra a anonymita často protichůdné principy, podařilo se nám najít i takové aplikace, kde chceme mít zaručeno obojí.

**Síť důvěry:** Sama o sobě možnost zjistit, že někomu mohu důvěrovat, je zajímavá.

**Ověřování informací:** Důvěryhodnost informace může být strvzena podpisem jednoho či více uživatelů sítě. Každá informace může být vrcholem v naší síti jen s tím rozdílem, že z vrcholu, reprezentující tuto informaci, nevedou žádné další hrany.

**Dynamicky rostoucí skupina s atributem:** Doposud naše síť zachycovala obecnou důvěru. V reálném životě často nastává situace, že někomu sice nevěříme obecně, ale silně mu důvěřujeme v nějaké oblasti. Můžeme tedy předpokládat, že pro každou takovou oblast, kde by to bylo užitečné, by vznikla paralelní síť důvěry, ve které by hrana z A do B znamenala: „A důvěřuje B v oblasti X“ nebo „A tvrdí, že B má atribut Y“.

Tímto způsobem může například budovat decentralizované občanství: A tvrdí, že B je občan zřízení X. Decentralizované potvrzení fyzické existence: A tvrdí, že B je fyzická osoba. Decentralizované řidičské licence: A tvrdí, že B je způsobilý k řízení auta.

**Recenze:** Podobně jako informace může být recenze vrcholem v síti důvěry. Její důvěryhodnost může být doložena přímo propojením s jejím autorem, nebo se autor může anonymizovat v nějaké skupině dle bodu Komunita.

Můžeme tak vytvořit decentralizovaný systém recenzí, do kterého prodejci nemohou vkládat uměle vytvořené recenze.

**Reputace:** Síť důvěry samozřejmě může sloužit k budování reputace. Reputace a anonymita jsou zcela protichůdné principy, proto zde zk-SNARK nenajdou využití.

**Decentralizové obnovení identity:** Pokud uživatel sítě důvěry přijde o svůj tajný klíč, prohlásí svůj stávající veřejný klíč za neplatný a vydá nový. Totéž může udělat útočník. Útočníkovi se ale na rozdíl od oběti nepodaří fyzicky přesvědčit reálné lidi z okruhu přátel oběti, aby vložili důvěru právě v jeho nový veřejný klíč. Naopak oběti stačí, aby její nový veřejný klíč ověřilo jen pár blízkých lidí k tomu, aby v něj opět začal důvěřovat i zbytek lidí díky nepřímé důvěře.

**Porovnání s ring signatures:** Ukázali jsme způsob, jak nadefinovat nějakou skupinu jednotlivců a jak prokázat náležitost k této skupině bez přímého odhalení své identity. Stejnou možnost nabízí takzvané ring signatures [6]. Náš přístup a ring signatures mají rozdílné vlastnosti. Zatímco v ring signatures defininujeme skupinu výčtem jejích členů, v síti důvěry ji definujeme obecněji nějakým pravidlem. Dále, narozdíl od ring signatures, většinu výpočtu delegujeme na dokazovatele. Povšimněme si například, že k ověření náležitosti dokazovatele k pravidlem nadefinované skupině ověřovatel nemusí znát pseudonymy členů této skupiny.

Kombinace těchto dvou rozdílů nám dává jednu významnou výhodu síťových podpisů nad ring signatures. Složitost generování podpisu a složitost jeho následného ověření nezávisí na velikosti skupiny, v rámci které zůstává podepisující anonymní. To je pro aplikaci v síti důvěry naprosto klíčová vlastnost, protože v praxi se zvyšujícím se poloměrem důvěry $r$ roste velikost množiny $B_d^{\mathcal{N}}(\mathtt{pk},r)$

exponenciálně. Tento jev lze sledovat na grafu přátelství na Facebooku.

Uvažujme graf, ve kterém vrcholy jsou uživatelé Facebooku a mezi dvěma uživateli je hrana právě tehdy, když jsou tito uživatelé přátelé. Pak s 95% procentní pravděpodobností bude vzdálenost mezi dvěma náhodně vybranými uživateli nejvýše 5 [23].

Síť důvěry, kterou jsme popsali, fungovala jen za předpokladu, že každý její uživatel zná její topologii. I přesto, že uživatelé mohou pomocí síťových podpisů komunikovat i bez vyzrazení svého pseudonymu v síti, je toto zásadní nedostatek, který může útočníkovi s dalšími znalostmi o uživatelích usnadnit přiřazení pseudonymů k identitám skutečných uživatelů.

V článku [24] je popsán způsob, jak uživatelé nějaké neúplné sítě (topologie peer-to-peer sítě je neúplný graf) mohou spočít jakoukoli funkci bez toho, aby se kdokoliv během výpočtu dozvěděl o topoligii sítě víc, než je mu doposud známo či lze vyčíst z výsledné funkční hodnoty. Je však nutné upozornit, že tento protokol má značnou komunikační složitost a musí se ho účastnit všichni uživatelé sítě narozdíl od síťových podpisů, kde komunikace probíhá jen mezi dokazovatelem a ověřovatelem.

# Závěr

V konstrukční části této práce jsme poskytli robustní definici zk-SNARKs. Dále jsme pomocí několika konstrukcí spojili Paillierův kryptosystém s LPCP, čímž jsme dostali designated-verifier SNARK. Zatímco důkazy týkající se LPCP byly postaveny na Schwartz-Zippelově lemmatu, u linear-only vlatnosti a sémantické bezpečnosti modifikovaného Paillierova kryptosystému jsme museli vycházet z kryptografických předpokladů KEA a NRIA. Díky zaměření na designated-verifier SNARK se nám podařilo lehce zjednodušit některé definice a důkazy. Nakonec jsme rozvedli důkaz knowledge soundness výsledného zk-SNARKu, který byl v článku [2], ze kterého jsme konstrukci přebrali, jen nastíněn.

V aplikační části jsme popsali problém anonymity Bitcoinu a popsali jeho řešení pomocí zk-SNARK implementované v kryptoměně Zcash včetně parametrů této implementace. Důležitým poznatkem z aplikační části je, že potřeba výpočtu veřejných parametrů důvěryhodnou autoritou je hlavní slabinou praktické implementace zk-SNARKs.

V poslední kapitole aplikací jsme postupně budovali decentralizovanou síť důvěry. Pomocí zk-SNARK jsme pro síť důvěry nadefinovali a nastínili konstrukci síťového podpisu, což je kryptografický protokol, který uživatelům z Bobova důvěryhodného okolí umožní se autentizovat bez toho, aby přímo odhalili svojí identitu. Navázali jsme přitom na ideu PGP Web of Trust a ukázali jsme různé způsoby, jak lze pro síťový podpis používat složitější metriky. Nakonec jsme srovnali síťové podpisy s funkcí podobnými ring signatures.

# Seznam použité literatury


[1] Christina Garman Matthew Green Ian Miers Eran Tromer Madars Virza Eli Ben-Sasson, Alessandro Chiesa. Zerocash: Decentralized Anonymous Payments from Bitcoin (extended version). 2014. `http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf`.

[2] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct Non-Interactive Arguments via Linear Interactive Proofs. Cryptology ePrint Archive, Report 2012/718, 2012. `https://eprint.iacr.org/2012/718`.

[3] Alisa Pankova. Succinct Non-Interactive Arguments from Quadratic Arithmetic Programs. 2013. `https://courses.cs.ut.ee/MTAT.07.022/2013_fall/uploads/Main/alisa-report`.

[4] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2009. `https://bitcoin.org/bitcoin.pdf`.

[5] Nicolas van Saberhagen. Cryptonote v 2.0. 2013. `https://cryptonote.org/whitepaper.pdf`.

[6] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret: Theory and applications of ring signatures. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, pages 164–186, 2006.

[7] Georges Gonthier. Formal proof – the four-color theorem. *Notices of the American Mathematical Society*, (55):1382–1393, 2008. `https://arxiv.org/pdf/1708.07442.pdf`.

[8] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[9] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108, 2013.

[10] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[11] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure Two-Party Computation with Low Communication. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 54–74, 2012.

[12] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[13] Alfred Menezes. An introduction to pairing-based cryptography. In *Recent trends in cryptography*, pages 47–65, 2009.

[14] Daniel Genkin, Dimitrios Papadopoulos, and Charalampos Papamanthou. Privacy in Decentralized Cryptocurrencies. *Commun. ACM*, 61(6):78–88, May 2018.

[15] Dorit Ron and Adi Shamir. Quantitative Analysis of the Full Bitcoin Transaction Graph. Cryptology ePrint Archive, Report 2012/584, 2012. `https://eprint.iacr.org/2012/584`.

[16] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model. Cryptology ePrint Archive, Report 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`.

[17] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018:46, 2018.

[18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334, 2018.

[19] William Stallings. PGP web of trust. *BYTE*, 20:161–162, 02 1995.

[20] Michael Backes, Nico Döttling, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Ring signatures: Logarithmic-size, no setup - from standard assumptions. *IACR Cryptology ePrint Archive*, 2019:196, 2019.

[21] Sana HAMDI. *Computational models of trust and reputation in online social networks*. Theses, Université Paris-Saclay ; Université de Tunis. Faculté des sciences de Tunis, January 2016.

[22] Guanfeng Liu, Yan Wang, and Mehmet A. Orgun. Trust Transitivity in Complex Social Networks. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, 2011.

[23] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *CoRR*, abs/1111.4503, 2011.

[24] Tal Moran, Ilan Orlov, and Silas Richelson. Topology-Hiding Computation. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 159–181, 2015.