Zero-knowledge protokol v kryptografii

Bc. & Bc. Martin Mikala

Diplomová práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. et Bc. Martin Mikala**
Osobní číslo:	**A20141**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Kybernetická bezpečnost**
Forma studia:	**Kombinovaná**
Téma práce:	**Zero-knowledge protokol v kryptografii**
Téma práce anglicky:	**Zero-Knowledge Protocol in Cryptography**

Zásady pro vypracování

1. Vypracujte literární rešerši na zadané téma.
2. Vysvětlete základní principy zero-knowledge protokolů.
3. Popište možnosti praktické aplikace zero-knowledge protokolů.
4. Demonstrujte praktické použití vybraného interaktivního zero-knowledge protokolu.
5. Proveďte ukázku praktického použití vybraného neinteraktivního zero-knowledge protokolu.
6. Zhodnoťte dosažené výsledky a proveďte závěr.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. FIAT, Amos a Adi SHAMIR. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: *Advances in Cryptology —CRYPTO’ 86* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, s. 186-194. Lecture Notes in Computer Science. ISBN 978-3-540-18047-0. Dostupné z: doi:10.1007/3-540-47721-7_12
2. BRANDT, Jørgen, Ivan DAMGÅRD, Peter LANDROCK a Torben PEDERSEN. Zero-Knowledge Authentication Scheme with Secret Key Exchange. In: *Advances in Cryptology —CRYPTO’ 88* [online]. New York, NY: Springer New York, 1990, 1990-12-1, s. 583-588. Lecture Notes in Computer Science. ISBN 978-0-387-97196-4. Dostupné z: doi:10.1007/0-387-34799-2_43
3. ANGEL, Sebastian a Michael WALFISH. Verifiable auctions for online ad exchanges. In: *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM* [online]. New York, NY, USA: ACM, 2013, 2013-08-27, s. 195-206. ISBN 9781450320566. Dostupné z: doi:10.1145/2486001.2486038
4. PAUWELS, Pieter. *ZkKYC: A solution concept for KYC without knowing your customer,leveraging self-sovereign identity and zero-knowledge proofs* [online]. Dostupné z: https://eprint.iacr.org/2021/907.pdf
5. PREUKSCHAT, Alex a Drummond REED. *Self-Sovereign Identity: Decentralized digital identity and verifiable credentials*. Manning, 2021. ISBN 9781617296598.

Vedoucí diplomové práce: **doc. Ing. Roman Šenkeřík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. prosince 2021**
Termín odevzdání diplomové práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně ..... Martin Mikala, v.r. .....

podpis autora

ABSTRAKT

Hlavním cílem práce je popsat principy fungování a základní vlastnosti zero-knowledge protokolů, vysvětlit druhy těchto protokolů a shrnout možnosti jejich praktické aplikace. Součástí práce je také ukázka možného využití interaktivního zero-knowledge protokolu pro účely autentizace, a také demonstrace možného využití neinteraktivního zero-knowledge protokolu v kontextu elektronických dokladů pro prokázání, že je držitel dokladu starší, než je požadovaná věková hranice, bez vyzrazení data narození.

Klíčová slova: zero-knowledge, interaktivní protokol, neinteraktivní protokol, autentizace, důkaz znalosti, důkaz věku

ABSTRACT

The Main goal of this thesis is to describe principles and basic attributes of zero-knowledge protocols, explain types of these protocols and sum up possible practical applications. Thesis also contains demonstration of possible use of interactive zero-knowledge protocol for authentication purposes and also demonstration of possible use of non-interactive zero-knowledge protocol in the context of electronic documents for proving, that document holder is older than required without revealing his date of birth.

Keywords: zero-knowledge, interactive protocol, non-interactive protocol, authentication, proof of knowledge, proof of age

OBSAH

ÚVOD

V 80. letech 20. století začali kryptografové hledat odpověď na otázku: Je možné prokázat platnost nějakého teorému bez jeho vyzrazení? Odpověd na tuto otázku byla nalezena v roce 1985, kdy byl poprvé definován takzvaný Zero-knowledge protokol, případně Zero-knowledge důkaz *(Zero-knowledge proof)*. Tyto důkazy se nejprve zabývaly zejména složitými matematickými problémy, jako jsou důkaz existence hamiltonovské cesty v grafu či důkaz splnitelnosti booleovské formule.

Zero-knowledge protokoly se obecně zabývají prokázáním platnosti teorému, což je spíše teoretický problém, v praxi je často důležitější dokázat fakt, že jedna strana zná nějaké tajemství bez jeho vyzrazení. K tomu vznikly takzvané důkazy znalosti *(Proof of Knowledge)*, kombinace těchto dvou protokolů pak dala za vznik Zero-knowledge důkazu znalosti *(Zero-knowledge Proof of Knowledge)*.

I když výzkum těchto protokolů pokračoval i nadále, byla tato problematika dlouho na okraji zájmu, neboť nebylo známo mnoho praktických aplikací. Zero-knowledge bylo považováno za, sice zajímavý a důležitý, ale spíše teoretický obor výzkumu.

Renezance tématiky Zero-knowledge nastala po roce 2010 s rozvojem kryptoměn, jako jsou Zerocash či Monero. Protokoly se dostaly do středu zájmu a do jejich výzkumu bylo investováno čím dál více prostředků. Mimo kryptoměny se také začalo zkoumat možné využití Zero-knowledge například v systémech elektronických voleb či pro online autentizaci.

V neposlední řadě se termín Zero-knowledge začal skloňovat v souvislosti s elektronickými doklady, ať už jde o systém Self-sovereign Identity, který je založený na redukci předávaných dat na nutné minimum, a na co největší kontrole uživatele nad svými daty, či o právě vznikající koncept Evropské digitální idenity, která umožní všem občanům Evropské unie jednoduše elektronicky sdílet své osobní doklady, a to vnitrostátně i v rámci celé evropské unie.

V dalších letech bude výzkum Zero-knowledge nepochybně pokračovat a budou objeveny lepší protokoly a nové způsoby jejich využití.

Cílem této práce je vysvětlit základní principy Zero-knowledge protokolů a poskytnout shrnutí jejich postupného vývoje a možnosti praktické aplikace. Součástí je také demonstrace praktického použití vybraného interaktivního a neinteraktivního Zero-knowledge protokolu. Práce je rozdělena do 4 částí.

První část, Předmluva, obsahuje zjednodušený úvod do problematiky Zero-knowledge a používaných pojmů. Cílem je poskytnout základní vhled do problematiky. K tomu slouží také intuitivní příklady využití Zero-knowledge protokolů, které se v tomto kontextu uvádějí. K demonstraci toho, že Zero-knowledge protokoly nemusí být nutně

postaveny na složitých matematických problémech, obsahuje tato část také popis Zero-knowledge důkazu vyřešení Sudoku za použití klasického balíčku karet.

Poté následuje Teoretická část, která obsahuje literární řešerži a formální představení interaktivních i neinteraktivních Zero-knowledge protokolů, jejich vlastností, a také souvisejících kryptografických nástrojů a pojmů. Pozornost je věnována mimo jiné i Turingovým strojům a Teorii výpočetní složitosti, neboť práce na téma Zero-knowledge se těmito oblastmi často zabývají. Závěr kapitoly je věnován možnostem praktické aplikace Zero-knowledge protokolů.

Praktická část práce je rozdělena na dvě kapitoly. Část 3 se věnuje demonstraci praktického využití interaktivního Zero-knowledge protokolu v kontextu prokázání identity, resp. online autentizace, které je odvozeno od Fiat-Shamirova identifikačního schématu a postaveno na problematice diskrétního logaritmu. Kromě vysvětlení fungování základní verze protokolu je zde také ukázán možný způsob zapojení ekvivalentu kryptografického pepře a také druhého autentizačního faktoru. Součástí je také vzorová implementace protokolu v jazyce Python.

Poslední část se věnuje ukázce vybraného neinteraktivního zero-knowledge protokolu, konkrétně demonstruje použití neinteraktivního Zero-knowledge protokolu v kontextu elektronických dokladůa to formou ukázky možnosti prokázání dosaženého věku bez prozrazení data narození za pomocí hash chainů. Tento a podobné problémy budou řešeny právě v kontextu vznikající Evropské Digitální identity. Součástí práce je také vzorová implementace v jazyce Python.

I. PŘEDMLUVA

1 Slovo úvodem

Zero-knowledge protokoly jsou poměrně teoretické téma, které se kromě kryptografie dotýká také výpočetní komplexity či teorie informace, a kterému bylo věnováno mnoho vědeckých prací. Z důvodu získání základního vhledu do této problematiky obsahuje tato kapitola neformální shrnutí a zjednodušenou definici základních pojmů, a také intuitivní příklady, které přináší vhled do fungovaní zero-knowledge protokolů a které se v tomto kontextu uvádějí.

Z důvodu lepší čitelnosti jsou originální, anglické verze citovaných pasáží, stejně jako originální anglické názvy zmiňovaných vědeckých prací uvedeny v poznámce pod čarou, v textu jsou uvedeny pouze přeložené verze.

Vzhledem k tomu, že vědecké práce k tomuto tématu vznikají zpravidla v angličtině, bylo nutno v rámci této diplomové práce tyto termíny přeložit do češtiny. Aby bylo možno rozlišit obecně využívané termíny, je u prvního použití daného termínu vždy uvedena i jeho anglická verze, a zároveň jsou tyto termíny v rámci textu vždy psány s velkým počátečním písmenem. Přehled anglických termínů a jejich použitých českých překladů je uveden na konci této práce.

1.1 Důkaz versus důkaz

V rámci práce je často využíván termín důkaz, resp. Důkaz, ať už ve vlastním textu či v citacích. Tyto dva výrazy mají každý jiný význam.

Termín „Důkaz“ *(Proof)* znamená, že jde o Statisticky Solidní *(Statistical Soundness)* protokol jakožto silnější verze Argumentu (který je pouze Výpočetně Solidní *(Computational Soundess)*.

Termín „důkaz“ je pak používán jako synonymum pro slovo protokol, důkazem tedy může být jak Důkaz, tak Argument. Druhým možným způsobem využití termínu „důkaz“ je užití v jeho běžném významu, tedy data, která něco dokazují.

2 Neformální shrnutí

Protokolem je myšlen definovaný postup, podle kterého probíhá elektronická komunikace - neboli, v jakém pořadí si posílají strany komunikace zprávy a co je obsahem jednotlivých zpráv.

Cílem zero-knowledge[1)] protokolů je umožnit jedné straně, zpravidla označované

[1)] V praxi se v angličtině používá jak forma „zero knowledge“, tak forma „zero-knowledge“. Do češtiny se někdy tento termín překládá jako „nulová znalost“, což je, dle mého názoru, sice formálně správný překlad, ale není stejně elegantní a vypovídající, jako jeho anglická verze. Proto je v rámci této práce brán zero-knowledge jako „terminus technicus“ a je používán v nepřeložené verzi jako zero-knowledge (nebo zero knowledge v citacích, kde je tak použito)

termínem Dokazovatel *(Prover)*, dokázat druhé straně, Ověřovateli *(Verifier)*, platnost nějakého tvrzení bez toho, aby mu o tomto tvrzení cokoli prozradil. V praxi je často potřeba dokázat nikoli platnost nějakého tvrzení, ale fakt, že Dokazovatel zná nějaké tajemství. Toho je možné dosáhnout pomocí Zero-knowledge Důkazu znalosti *(Zero-knowledge Proof of knowledge)*.

Tyto protokoly mohou nabývat formy Důkazu *(Proof)* nebo Argumentu *(Argument)*. Důkazy jsou teoreticky bezpečnější, v praxi je i bezpečnost Argumentů dostatečná, a proto se používají častěji, neboť je jejich implementace zpravidla jednodušší.

Obecně od těchto protokolů vyžadujeme několik vlastností. K tomu, aby protokol fungoval je nutné, aby Čestný *(Honest)* Dokazovatel, neboli ten, který nepodvádí, dokázal Ověřovatele o své pravdě přesvědčit. Této vlastnosti se říká Úplnost *(Completeness)*.

Dále je nutné, aby Dokazovatel nemohl podvádět, tedy přesvědčit Ověřovatele že je pravdou něco, co ve skutečnosti pravda není. Toto je označováno jako Solidnost *(Soudness)*.

V ideálním případě by měl čestný Dokazovatel přesvědčit Ověřovatele vždy, a podvádějící Dokazovatel by neměl být schopen přesvědčit Ověřovatele nikdy, ale některé protokoly připouští i určitou pravděpodobnost chyby (teoreticky až 1/3)[2)]. Proto může být potřeba, aby protokol proběhl vícekrát, a tím byla chyba snížena na přípustnou hodnotu - pokud dokáže Dokazovatel podvádět v polovině případů, pak je pravděpodobnost, že se mu podaří podvádět dvakrát za sebou, $1/2 \times 1/2 = 1/4$, pravděpodobnost, že se mu podaří podvádět desetkrát za sebou, je $1/1024 = 0.1\%$, atd.

V rámci těchto protokolů je také nutné, aby se Ověřovatel nedozvěděl nic, co se dozvědět nemá. Tato vlastnost je pojmenována Zero-knowledge. Někdy není nutné, aby se Ověřovatel nedozvěděl vůbec nic, ale může se dozvědět například část tajemství (nesmí se jej ale dozvědět celé).

V neposlední řadě je potřeba, aby byl protokol Efektivní *(Efectivity)*, tedy aby nevyžadoval složité výpočty, které by na běžných počítačích trvaly dlouhou dobu.

Často se v těchto protokolech používají Závazky *(Commitments)*. Závazek si můžeme představit jako uzamykatelnou skříňku, která je uzamčena dvěma zámky - od jednoho zámku má klíč Zavazující se strana, klíč ke druhému zámku má Přesvědčovaný. Skříňka má úzkou šterbinu, kterou je do ní možno vložit papír, ale ten již poté není možno vytáhnout bez odemčení skříňky. Zavazující se strana napíše na papír tajemství - Závazek, a papír hodí do skříňky. V ten okamžik již nedokáže Závazek nijak změnit, neboť se k němu Zavázal *(Binding)*. Přesvědčovaný zase nedokáže toto tajemství zjistit, neboť nedokáže skříňku otevřít sám bez pomoci Zavazující se strany. Této vlastnosti se říká

[2)] Čestný Dokazovatel musí přesvědčit Ověřovatele nejméně ve 2/3 případů, Podvádějící *(Cheating)* Dokazovatel musí být odhalen nejméně ve 2/3 případů

Skrytí*(Hiding)*. Ve správné chvíli je poté možno provést Zrušení Závazků *(Decommitment)*, skříňku otevřít a Přesvědčovaný tak může zjistit, k čemu se Zavazující se strana Zavázala.

U některých vlastností (například Solidnosti, Zero-knowledge, Skrytí či Zavázání) se v rámci zkoumání konkrétních protokolů řeší, jaké úrovně bezpečnosti dosahují z ohledem na to, jak silný protivník se je snaží prolomit. Jako příklad si vezměme Solidnost, tedy schopnost Dokazovatele podvádět. Pokud má Dokazovatel k dispozici „reálné množství" výpočetní síly a přesto nedokáže podvádět, mluvíme o Výpočetní *(Computational)* Solidnosti. Pokud má Dokazovatel k dispozici libovolné množství výpočetní síly a přesto nedokáže podvádět, pak jde o Statistickou *(Statistical)* Solidnost. Nejvyšším stupněm je pak Perfektní Solidnost, u které musí být dokázáno, že neexistuje možnost podvádět.

Obecně můžeme protokoly dělit na dvě skupiny. Protokoly z první skupiny vyžadují, aby se do průběhu výpočtu zapojil i Ověřovatel, zpravidla tím, že vygeneruje a pošle Dokazovateli nějakou náhodnou hodnotu, kterou tento musí ve svém výpočtu použít. Kvůli této probíhající interakci se se tyto protokoly nazývají Interaktivní protokoly *(Interactive protocols)*.

Druhou skupinou jsou protokoly, které tuto interakci nevyžadují - Dokazovatel provede všechny výpočty sám a poté pošle výsledky Ověřovateli v jedné zprávě, aby si je mohl ověřit. Těmto protokolům se říká Neinteraktivní protokoly *(Non-interactive protocols)*. V praxi jsou užitečnější, ale mají své nevýhody, například často vyžadují počáteční nastavení nějakou Důvěryhodnou třetí stranou *(Trusted party)*.

Existují metody, kterými je možno převést Interaktivní protokoly na Neinteraktivní.

Zero-knowledge protokoly mají své využití například v Blockchainech (hlavně v kryptoměnách), ale také například v systémech online voleb či u elektronických dokladů. V neposlední řadě je možno tyto protokoly využít při autentizaci.

3 Intuitivní příklady

Tato kapitola obsahuje intuitivní příklady, které mají za cíl přinést základní vhled do fungování Zero-knowledge protokolů. Kromě běžně uváděných příkladů Ali Babovi tajemné jeskyně a Dvou míčků a barvoslepého kamaráda je také uveden třetí praktický příklad - Zero-knowledge důkaz Sudoku řešený bez použití počítače za pomoci klasického balíčku karet.

3.1 Dva míčky a barvoslepý kamarád

Představme si, že má Daniel (Dokazovatel) dva míčky, jeden červený a druhý zelený, ale jinak identické. Danielův barvoslepý kamarád, Ondřej (Ověřovatel), mu ale nevěří,

že jsou míčky různých barev, protože jemu připadají stejné. Daniel chce Ondřeje přesvědčit, a proto zvolí následující postup:

1. Ondřej si vezme oba míčky, každý do jedné ruky, a ukáže Danielovi, který míček drží ve které ruce
2. Ondřej schová ruce s míčky za zády tak, aby je Daniel neviděl, a náhodně se rozhodne, jestli míčky prohodí, nebo je nechá tak, jak jsou
3. Ondřej ukáže Danielovi ruce s míčky
4. Daniel řekne Ondřejovi, jestli míčky podle něj prohodil, nebo jestli je nechal tak, jak byly

Pokud Daniel nelže, dokáže vždy odpovědět správně. Pokud lže, tak si musí tipnout jednu ze dvou možností, což se mu povede v průměru v polovině případů.

Z pohledu Ondřeje, pokud Daniel určil správně, tak mu může věřit na 50%. To není mnoho, proto může Ondřej vyžadovat, aby Daniel experiment zopakoval vícekrát (například desetkrát). Pravděpodobnost, že Daniel určí správně výsledek všech deseti pokusů, je 100%, pokud nelže, ale pouze 2^{-10}%, tedy asi 0.1%, což už Ondřeje může přesvědčit.

Vzhledem k tomu, že Daniel vždy Ondřejovi řekne pouze to, zda míčky prohodil či neprohodil (což je informace, kterou Ondřej zná), ale nikdy mu neřekne, který míček je zelený a který červený, nedozví se Ondřej nic než to, že Daniel mluví pravdu, jde tedy o Zero-knowledge Důkaz.

Autorem tohoto intuitivního příkladu je pravděpodobně Kostas Kryptos [1].

3.2 Ali Babova jeskyně

Intuitivní příklad fungování Zero-knowledge protokolu přinesl také Jean-Jacques Quisquater v práci Jak vysvětlit Zero-knowledge protokoly svým dětem[1)][2], kde vypráví příběh nazvaný Tajemná Ali Babova jeskyně.

Příběh začíná popisem starého muže jmeném Ali Baba, který pronásleduje zloděje do tajemné jeskyně. Hned u vstupu narazil na křižovatku typu T. Ali Baba neviděl, kterou cestou lupič utekl, proto se vydal tou levou. Po chvíli se ukázalo, že je cesta nejen prázdná, ale i slepá - zloděj uprchl. Druhé den se situace opakovala, Ali Baba si ale na křižovatce zvolil pravou cestu. Ta byla také prázdná a slepá - zloděj opět uprchl. Další a další dny se situace opakovala, zloděj vždy Ali Babovi utekl.

[1)] How to Explain Zero-Knowledge Protocols to Your Children

Po deseti dnech si Ali Baba řekl, že není možné, aby měl takovou smůlu, protože šance, že by si každý den zvolil špatnou cestu, je jedna k bilionu, a rozhodl se zvolit jiný přístup. Schoval se v jeskyni a čekal na lupiče. Po několika hodinách do chodby skutečně vběhl zloděj. Zastavil se u jedné zdi, řekl tajné heslo, zeď se odsunula a spojila obě chodby. Ali Baba se dozvěděl tajemství jeskyně.

Tajné slovo se v rodině Ali Baby přenášelo z generace na generaci, až se jej dozvěděl jeho vzdálený potomek, Mick Ali[2)]. Ten se rozhodl ukázat světu existenci tajemné jeskyně a také to, že zná tajné heslo, nechtěl ale toto heslo prozradit. Proto si pozval reportéra s kamerou. Vešel do jeskyně do jedné z chodeb, ale tak, aby reportér neviděl, do které. Reportér se poté postavil na křižovatku, náhodně si vybral levou nebo pravou chodbu a zavolal, ze které chodby má Mick vyjít. Experiment mnohokrát opakovali.

Protože Mick znal tajné heslo, dokázal vždy vyjít z chodby, ze které reportér chtěl. Pokud by heslo neznal, dokázal by vyjít ze správné chodby jen v polovině případů a reportéra by nepřesvědčil.

Reporér se nedozvěděl o tajném heslu nic, kromě toho, že jej Mick zná, jde tedy o Zero-knowledge protokol.

Příběh dále pokračuje:

Žárlivý reportér z konkurenční televize chtěl natočit stejnou reportáž, Mick s ním ale nechtěl spolupracovat. Proto si našel Mickova dvojníka, se kterým se domluvil na tom, že reportáž natočí. Dvojník ale neznal tajné heslo, proto přišli konkurent a dvojník s následujícím řešením - nasimulují si náhodný výběr chodby, ze které má dvojník vyjít, dopředu. Dvojník si zapomatoval pořadí chodeb, ve kterých se má schovat, a konkurent s ním poté natočil falešnou reportáž.

Obě reportáže, opravdová i simulovaná, byly odvysílány na konkurenčních stanicích ve stejný čas, což vedlo k soudnímu sporu. Obě strany předložily své důkazy a natočené materiály, ale žádný soudce, expert ani porotce nedokázal rozpoznat opravdovou reportáž od té simulované.

„Mick Ali dosáhl svého opravdového cíle. Chtěl, ve skutečnosti dokázat, že je možné přesvědčit bez vyzrazení, a také bez odkrytí jeho tajemství.“[3)][2]

3.3 Sudoku a balíček karet

Aby předvedli, že k využití Zero-knowledge protokolu není nutné spoléhat na složité matematické výpočty, vydali autoři Ronen Gradwohl, Moni Naor, Benny Pinkas a Guy N. Rothblum práci Kryptografické a fyzické zero-knowledge systémy důkazů pro řešení

[2)] narážka na jednoho z autorů původní práce o Zero-knowledge protokolech [3] jménem Silvio Micali

[3)] Mick Ali had achieved his real objective. He wanted, in fact, to show that it is possible to convince without revealing, and so without uveiling his secret.

Sudoku[4)][4]. V této práci ukazejí, kromě klasického protokolu za použití počítače, také možné vytvoření Zero-knowledge důkazu využitím zapečetěných obálek, nůžek a papíru či balíčku karet. Právě poslední z jmenovaných možností popsal jeden z autorů, Benny Pinkas ve své přednášce Zero knowledge pro Sudoku[5)][4] na Zimní škole kryptografie v roce 2019.

Sudoku je číselný hlavolam původem z Japonska. Na plochu o velikosti 9×9 čtverců je nutné doplnit číslice od 1 do 9 tak, aby v každém z 9 řádků, 9 sloupců a 9 podčtverců byla každá číslice zastoupená právě jednou. Na počátku jsou některé čtverce předvyplněny (od počtu předvyplněných čtverců se odvíjí náročnost hlavolamu) a cílem je doplnit odstatní, prázdné čtverce.

Daniel (Dokazovatel) chce přesvědčit Ondřeje (Ověřovatele), že umí vyřešit konkrétní Sudoku, nechce mu ale prozradit žádnou informaci navíc (například hodnotu nějakého čtverce). K dispozici má pouze klasický balíček karet (obsahující 108 karet). Z balíčku vybere 81 karet tak, že každá hodnota od 1 do 9 bude zastoupena devětkrát. Tyto vybrané karty ukáže Ondřejovi, aby se mohl ujistit, že je vybraný balíček v pořádku, a že je s jeho pomocí Daniel schopen úspěšně demonstrovat řešení Sudoku (pokud jej skutečně zná).

Daniel poté vezme vybrané karty a rozmístí je do čtverce 9×9 karet tak, že umístění karet bude odpovídat řešení Sudoku (v podstatě postaví vyřešené Sudoku na stole). Karty, které odpovídají zadání hlavolamu, položí číslem nahoru (aby se Ondřej mohl přesvědčit, že se řešení týká toho konkrétního Sudoku), zbytek karet položí číslem dolů.

Ondřej si poté může zvolit libovolný řádek, sloupec nebo podčtverec (má tedy 27 možností). Daniel vezme vybraných 9 karet, zamíchá je a předá Ondřejovi. Ondřej poté může ověřit, že v balíčku jsou skutečně karty všech hodnot od 1 do 9.

Čestný Daniel (který se nepokouší podvádět) dokáže karty poskládat správně vždy, a přesvědčit tak Václava, proto je tento protokol Úplný.

Vzhledem k tomu, že Ondřej dostane do ruky pouze zamíchané karty, může sice jednoduše ověřit, že balíček obsahuje všechny hodnoty, ale nedokáže žádným způsobem zjistit jejich pozici před zamícháním. Proto jde o (Perfektní) Zero-knowledge protokol.

Daniel ovšem může také lhát a pouze předstírat, že Sudoku umí vyřešit. V takovém případě buď zamíchaný balíček karet neobsahuje všechny hodnoty (a Onděj získá jistotu, že Daniel lže), nebo měl Daniel štestí a zrovna v té vybrané možnosti chyba nebyla. Vzhledem k tomu, že Ondřej zkontroloval jen jednu možnost z 27, mohl Daniel teoreticky podvádět s pravděpodobností $26/27 = 96.3\%$.

To je sice hodně, ale i zde existují možnosti, jak tuto pravděpodobnost snížit na

[4)] Cryptographic and Physical Zero-Knowledge Proof Systems for Solutions of Sudoku Puzzles
[5)] Zero Knowledge for Sudoku

akceptovatelnou úroveň. Daniel může například vrátit karty zpět na svá místa a Ondřej si může vybrat znovu řádek, sloupec nebo podčtverec (klidně ten stejný) a postup opakovat, dokud nebude přesvědčený.

Zde je nutno podotknout, že [5] obsahuje i popis lepších možností, jak snížit možnost Danielova podvádění, výše je popsána pouze základní verze protokolu.

II. TEORETICKÁ ČÁST

4 Úvod do související teorie

K formální definici Zero-knowledge protokolů je potřeba nejprve vysvětlit pojem formální jazyk a obecný princip fungování Turingova stroje.

4.1 Jazyk

V teoretické informatice je možno definovat abecedu jako neprázdnou konečnou množinu znaků. Například množina obsahující prvky 0 a 1 může být označena jako abeceda $\Sigma : \{0, 1\}$.

Z této abecedy je možno sestavit konečné množství řetězců konečné délky, které obsahují pouze znaky abecedy. Množina řetězců (označovaných také jako slov) nad abecedou Σ obsahuje prvky $\Sigma^\star : \{0, 1, 00, 01, 10, 11, 000, \dots\}$. Termínem jazyk L je pak možno označit podmnožinu $\Sigma^\star$, která obsahuje pouze slova splňující určitou podmínku či soustavu podmínek.

Například je možno definovat jazyk L tak, že obsahuje všechna slova, které začínají znakem 0, končí znakem 0 a mezi nimi obsahují libovolné množství znaků 1 (můžeme zapsat také jako $01^\star0$. Jazyk pak obsahuje prvky: $L : \{00, 010, 0110, 01110, \dots\}$.

O prvku $x = 0110$ pak můžeme říci, že patří do jazyka L, neboli $x \in L$, o prvku $y = 1001$ naopak, že do jazyka nepatří - $y \notin L$.

Jako další příklad můžeme definovat jazyk L_1, který je postavený nad abecedou $\Sigma_1 : \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Můžeme pak říci, že $\Sigma_1^\star : \{0^\star || \mathbb{N}^0\}$, tedy že množina všech slov nad abecedou Σ_1 obsahuje nulu a dále všechna přirozená čísla s libovolným množstvím nul na začátku. Jazyk L_1 pak může být definován tak, že obsahuje pouze prvočísla, tedy $L_1 : \{2, 3, 5, 7, \dots\}$.

Potom platí, že $x = 13 \in L_1$ a $y = 12 \notin L_1$.

4.2 Turingův stroj

Roku 1936 popsal známý britský matematik a kryptoanalytik Alan Mathison Turing ve svém díle O vypočítatelných číslech s aplikací na rozhodovací problém[1)][6] koncept „vypočítatelných“čísel, tedy reálných čísel, které je možno kompletně spočítat v konečném čase pomocí algoritmu. K tomuto účelu definuje teoretický model takzvaného Turingova stroje: „Můžeme srovnat proces výpočtu reálného čísla ke stroji, který je schopen být v konečném množství stavů $q_1, q_2, \dots, q_n$; toto nazveme m-konfiguracemi[2)]. Stroj má k dispozici pásku (analogie k papíru), která jím prochází a která je rozdělená na sekce (nazývané čtverce), kde každý může nést symbol. V každém okamžiku exis-

[1)] On Computable Numbers, with an Application to the Entscheidungsproblem

[2)] v současnosti se místo termínu m-konfigurace používá spíše termín (vnitřní) stav

tuje ve stroji pouze jeden čtverec, řekněme r-tý, který nese symbol $\mathfrak{G}(r)$. Tento čtverec můžeme nazvat skenovaný čtverec.“[6][3)] Součástí stroje je „skenovací hlava“, která se nachází nad právě skenovaným čtvercem, a provádí čtení i zápis. Na počátku výpočtu je skenovací hlava na čtverci nejvíce vlevo, páska je jednostranně nekonečná.

Stroj má binární výstup, při ukončení výpočtu tedy vrací hodnotu ANO nebo NE. Jedním z typů problémů, které Turingův stroj může řešit, je, zda nějaká vstupní hodnota x patří či nepatří do jazyka L.

Chování stroje je deterministické; další akce závisí pouze na vnitřním stavu a skenovaném symbolu. Každá akce sestává z několika kroků:

1. je načten symbol ze skenovaného čtverce
2. na základě načteného symbolu a současného stavu:
   (a) do skenovaného čtverce je zapsána nová hodnota (případně stará hodnota, pokud nemá dojít k její změně)
   (b) je změněn vnitřní stav (ma-li být změněn)
   (c) skenovací hlava je posunuta o jeden čtverec doleva nebo o jeden čtverec doprava
3. pokud výpočet:
   (a) neskončil - stroj pokračuje znovu od akce 1
   (b) skončil - stroj vrátí výsledek výpočtu

Výpočet prováděný Turingovým strojem může dopadnout 3 různými způsoby:

- výpočet skončí a stroj vrátí hodnotu ANO (ACCEPT), což znamená, že $x \in L$
- výpočet skončí a stroj vrátí hodnotu NE (REJECT), tedy že $x \notin L$
- výpočet zůstane v nekonečné smyčce a nikdy neskončí

Tato klasická verze stroje bývá také nazývána Deterministrickým Turingovým strojem. V průběhu času bylo navrhnuto mnoho různých verzí Turingova stroje - stroj, který umožňuje při akci nepohnout hlavou, stroj, který používá více pásek či stroj,

[3)] We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \ldots, q_n$; which will be called "m-configurations". The machine is supplied with a "tape"(the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r-th, bearing the symbol $\mathfrak{G}(r)$ which is "in the machine". We may call this square the "scanned square".

který používá oboustranně nekonečnou pásku. Důležitou verzí je takzvaný Nedeterministický Turingův stroj. V kontextu Zero-knowledge protokolů se dále využívají například Pravděpodobnostní či Interaktivní Turingův stroj.

4.2.1 Nedeterministický Turingův stroj

Tato verze stroje pracuje jako klasický Turingův stroj s jedním podstatným rozdílem - zatímco klasická verze stroje vykoná v závislosti na vnitřním stavu a naskenovaném symbolu vždy právě jednu akci, Nedeterminstický Turingův stroj může vykonat paralelně více akcí, které se navzájem neovlivňují. Tato vlastnost může být jinak vyjádřena tak, že v každé akci se stroj „rozdělí“ na potřebné množství vzájemně nezávislých větví (pro každou možnou akci jeden). Každá větev je opět reprezentována Nedeterministickým Turingovým strojem a každý tento stroj prozkoumá právě jednu možnost. Alternativně je možno si běh Nedeterministického stroje představit tak, že při každém rozhodnutí si stroj „magicky“ zvolí vždy tu volbu, která jej povede ke správnému výsledku (existuje-li).

Výpočet prováděný Nedeterministickým strojem může také dopadnou třemi různými způsoby:

- pokud je při výpočtu nalezena alespoň jedna větev, která vrací hodnotu ANO, výpočet skončí a stroj vrátí hodnotu ANO (ACCEPT), tedy že $x \in L$
- pokud všechny větve výpočtu skončí, a všechny větve vrátí hodnotu NE, výpočet skončí a stroj vrátí hodnotu NE (REJECT), tedy že $x \notin L$
- pokud není nalezena větev, která vrací hodnotu ANO, a zároveň nejméně jedna větev zůstane v nekonečné smyčce, výpočet zůstane v nekonečné smyčce a nikdy neskončí

4.2.2 Pravděpodobnostní Turingův stroj

Zatímco klasický i Nedeterministický Turingův stroj pracují determinsticky v tom smyslu, že výsledek výpočtu je vždy stejný, Pravděpodobnostní Turingův stroj v sobě obsahuje prvek náhody, výsledek výpočtu je tedy stochastický. Stejně jako Nedeterminstický Turingův stroj může v závislosti na vnitřním stavu a skenovaném symbolu existovat více možných proveditelných akcí, na rozdíl od Nedeterminstické verze, která zkoumala vždy všechny možnosti, si ale Pravděpodobnostní Turingův stroj zvolí náhodně v závislosti na definovaných pravděpodobnostech k provedení pouze jednu akci.

4.2.3 Interaktivní Turingův stroj

Autoři [3] pro definici Interaktivních systémů důkazu využívají speciální verze Turingova stroje, Interaktivní Turingův stroj. „Interaktivní Turingův stroj (ITM) je Turingův stroj vybavený jednou vstupní páskou pouze pro čtení, pracovní páskou, náhodnou páskou, jednou komunikační páskou pouze pro čtení a jednou komunikační páskou pouze pro zápis.“[4)][3]

Interaktivní protokol pak využívá páru těchto Interaktivních strojů, které sdílí komunikační pásky takovým způsobem, že první stroj může pouze číst z první komunikační pásku a pouze zapisovat na druhou komunikační pásku, zatímco druhý stroj může pouze zapisovat na první komunikační pásku a pouze číst z druhé komunikační pásky.

4.2.4 Příklad fungování Turingových strojů

Jako intuitivní příklad rozdílu mezi Deterministickým a Nedeterministický Turingovým strojem můžeme uvést test dělitelnosti / prvočíselnosti[5)].

Klasický Turingův stroj může řešit například problém, zda je vstupní hodnota x dělitelná určitým číslem dle konfigurace stroje. Může tedy existovat Turingův stroj, který řeší dělitelnost číslem 7 (jazyk L_1 tedy obsahuje všechna přirozená čísla dělitelná 7), a pro vstupní hodnotu $x = 35$ vrátí ANO, tedy že $x \in L_1$, pro $y = 37$ vrátí NE, tedy že $y \notin L_1$.

Nedeterministický stroj může řešit, zda je vstupní hodnota x složeným číslem (jazyk L_2 tedy obsahuje všechna přirozená čísla, která nejsou prvočísla). Pak se při výpočtu pro vstupní hodnotu $x = 17$ stroj „rozdělí“ na několik strojů, kde každý ověří dělitelnost jedním číslem z množiny $\{2, 3, \ldots, x-1\}$). Pokud alespoň jeden stroj vrátí ANO (tedy že x je dělitelné daným dělitelem), pak výpočet skončí a stroj odpoví, že jde o složené číslo, neboli že $x \in L_2$. Naopak pokud všechny výpočty skončí odpovědí NE, stroj ověřil, že vstupní hodnota není složené číslo, tedy $x \notin L_2$.

5 Teorie výpočetní složitosti

„Teorie výpočetní složitosti je podobor teoretické informatiky, jehož hlavním cílem je klasifikovat a srovnat praktickou obtížnost řešení problémů týkajících se konečných kombinatorických objektů - například máme-li dvě přirozená čísla n a m, jsou to rela-

[4)] An interactive Turing machine (ITM) is a Turing machine equipped with a read-only input tape, a work tape, a random tape, one read-only communication tape, and one write-only communication tape. The random tape contains an infinite sequence of random bits, and can be scanned only from left to right.

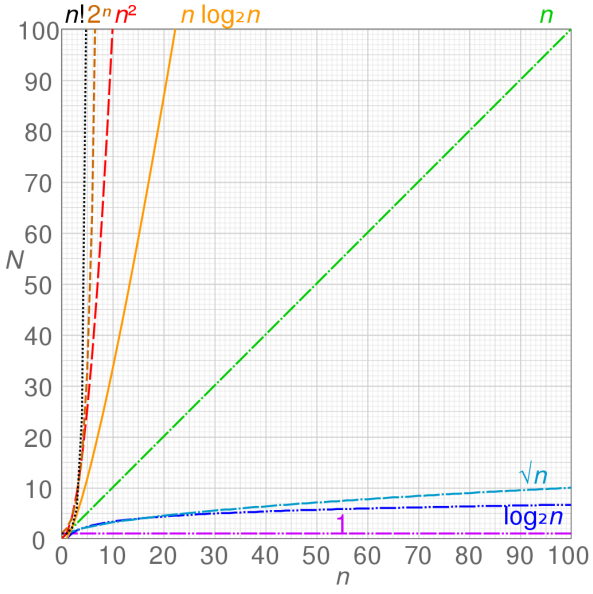
[5)] Při demonstraci testu prvočíselnosti je použit z důvodu jednoduchosti naivní algoritmus testování prvočíselnosti, který ověřuje, zda je testované číslo x dělitelné libovolným číslem z intervalu $\{2, 3, ..., x-1\}$

tivní prvočísla? Máme-li rovnici Φ, má tato rovnice řešení? Pokud bychom hráli šachy na šachovnici $n \times n$ polí, existuje z dané počáteční pozice výherní strategie pro bílého? Z pohledu klasické teorie vyčíslitelnosti jsou tyto problémy stejně složité v tom smyslu, že jsou efektivně rozhodnutelné. Nicméně z pohledu praktické složitosti se zdají razatně rozdílné.“[1)][7]

Hlavním cílem tohoto podoboru je pro daný algoritmus zjistit závislost délky výpočtu (jako referenční model je zpravidla použit klasický Turingův stroj) na délce vstupu (vyjádřené v bitech, značeno n). Tato funkce složitosti je značena řeckým písmenem O. Sledován je hlavně typ funkce (x^n, n^x, $nlog(n)$, ...), konkrétní hodnoty jsou méně důležité. Zpravidla je uvážován pouze nejrychleji rostoucí člen (např. u $O = n^2 + 3n$ se zpravidla člen $3n$ zanedbá a uvede se pouze složitost algoritmu $O = n^2$.

Na základě těchto funkcí je možno rozdělit algoritmy do takzvaných tříd složitosti. Obrázek 5.1 obsahuje srovnání průběhů pro tento účel běžně využívaných funkcí.

Obr. 5.1 Průběh běžně využívaných funkcí v teorii výpočetní složitosti [8]



Nutno podotknout, že, ačkoli jsou obecně problémy z vyšší třídy složitosti výpočetně náročnější, nemusí to být nutně pravda pro v praxi využívané hodnoty n. Například může existovat algoritmus z exponenciální třídy se složitostí $O = 1.0001^n$, který bude pro prakticky užitečná n rychlejší, než polynomický algoritmus se složitostí $O = n^{30}$.

[1)] Computational complexity theory is a subfield of theoretical computer science one of whose primary goals is to classify and compare the practical difficulty of solving problems about finite combinatorial objects – e.g. given two natural numbers n and m, are they relatively prime? Given a propositional formula Φ, does it have a satisfying assignment? If we were to play chess on a board of size $n \times n$, does white have a winning strategy from a given initial position? These problems are equally difficult from the standpoint of classical computability theory in the sense that they are all effectively decidable. Yet they still appear to differ significantly in practical difficulty.

Pro účely Zero-knowledge protokolů je nutno zmínit zejména 3 třídy složitosti označené P, NP a BPP.

5.1 Třída složitosti P

Třída P je jednou ze základních tříd složitosti. „Obsahuje rozhodovací problémy, které mohou být vyřešeny Deterministickým Turingovým strojem za použití polynomického množství výpočetního času, neboli polynomického času.“[2)][9]

Tato třída je důležitá, neboť dle takzvané Cobham-Edmondsovy teze (pojmenované po autorech Alanu Cobhamovi a Jacku Edmonsovi) je možné v praktickém nasazení řešit pouze ty výpočetní problémy, které je možno spočítat v polynomickém čase.

Polynomickým časem je myšleno, že délku běhu algoritmu v závislosti na délce vstupu v bitech je možno vyjádřit rovnicí $O = \sum_{i=1}^{c} a_i n^{j_i}$ kde a, j jsou reálná čísla.

Do této třídy složitosti patří například v současnosti nejlepší algoritmus pro test prvočíselnosti (AKS test prvočíselnosti), který představili v roce 2002 autoři Manindra Agrawal, Neeraj Kayal a Nitin Saxena v práci Prvočísla jsou v P[3)][37].

Mimo to patří do třídy P také klasické matematické operace (násobení či dělení).

5.2 Třída složitosti NP

Třída složitosti NP obsahuje problémy, které mohou být vyřešeny Nedeterministickým Turingovým strojem v polynomickém čase. Ze způsobu fungování Nedeterministického Turingova stroje vyplývá, že pokud je možné problém v NP vyřešit, je možné tento výsledek pomocí klasického Turingova stroje v polynomickém čase ověřit.

Problémy v této třídě, jako například výpočet diskrétního logaritmu či faktorizace poloprvočísel, jsou často velmi důležité v oboru kryptografie.

5.2.1 NP-úplný

Třídy složitosti obsahují podmnožinu problémů, které jsou pro danou třídu „úplné“ *(complete)*. „Neformálně řečeno je problém X redukovatelný na jiný problém Y právě tehdy, když by metoda pro řešení Y zároveň nesla metodu pro řešení X. Redukovatelnost X na Y může být tedy brána jako ukázka toho, že vyřešení Y je nejméně tak složité, jako vyřešení X. Problém X je označený jako úplný pro třídu složitosti C právě tehdy, když X je členem C a všechny problémy v C jsou redukovatelné na X. Úplnost X pro C tedy může být chápána jako demonstrování, že X je zástupce nejtěžších

[2)] It contains all decision problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time, or polynomial time.

[3)] PRIMES is in P

	$P(ANO)$	$P(NE)$
$x \in L$	$P \geq 2/3$	$P < 1/3$
$x \notin L$	$P < 1/3$	$P \geq 2/3$

Tab. 5.1 Pravděpodobnosti odpovědí pravděpodobnostního Turingova stroje

problémů v C.“[4)][7]

V kontextu Zero-knowledge protokolů je důležitá zejména třída složitosti NP-úplný. Z faktu, že je možné NP-úplné problémy efektivně převádět na jiné NP-úplné vyplývá, že existuje-li (či naopak neexistuje-li) určitý protokol pro některý z NP-úplných problémů, existuje (či naopak neexistuje) pro všechny NP-úplné problémy.

5.3 Třída složitosti BPP

Další důležitou třídou složitosti z pohledu Zero-knowledge protokolů je BPP, neboli pravděpodobnostní polynomický čas s omezenou chybou *(bounded-error probabilistic polynomial)*. Tyto problémy mohou být vyřešeny pravděpodobnostním Turingovým strojem v polynomickém čase s chybou menší, než 1/3. To znamená, že v případě, že $x \in L$, pravděpodobnost, že stroj odpoví ANO je $\geq 2/3$ a v případě, že $x \notin L$, stroj odpoví NE s pravděpodobností $\geq 2/3$.

Pravděpodobnosti odpovědi pravděpodobnostního Turingova stroje v závislosti na tom, zda $x \in L$, přehledně shrnuje tabulka 5.1.

5.4 Vztahy mezi třídami

Tyto a další zde nezmíněné třídy složitosti jsou vzájemně provázané a existují mezi nimi vztahy, kdy je jedna třída podmnožinou druhé třídy a podobně.

5.4.1 P a BPP

O vztahu mezi P a BPP se ví to, že $P \subseteq BPP$, neboli že P je podmnožina BPP. Nicméně jednou z nevyřešených otázek je, zda platí, že $P = BPP$.

5.4.2 P a NP

Pro vztah mezi P a NP platí, že $P \subseteq NP$. I zde, není jisté, zda platí $P = NP$.

Takzvaný P versus NP problém je jeden ze sedmi Problémů tisíciletí - nevyřešených zásadních matematických otázek, za jejichž vyřešení byla v roce 2000 vyhlášena

[4)] Informally speaking, a problem X is said to be reducible to another problem Y just in case a method for solving Y would also yield a method for solving X. The reducibility of X to Y may thus be understood as showing that solving Y is at least as difficult as solving X. A problem X is said to be complete for a complexity class C just in case X is a member of C and all problems in C are reducible to X. The completeness of X for C may thus be understood as demonstrating that X is representative of the most difficult problems in C.

odměna milion dolarů (zatím byl vyřešen z těchto sedmi problémů jeden). Platnost $P = NP$ by měla v oboru kryptografie dalekosáhlé důsledky, neboť by to implikovalo existenci jednoduchého řešení pro všechny „složité“ matematické problémy, na kterých jsou postaveny současné asymetrické algoritmy.

5.4.3 NP a BPP

O vztahu mezi třídami NP a BPP se mnoho neví. Je možné, že je jedna třída podmnožinou druhé, nebo že mají pouze společný průnik (součástí tohoto průniku je třída P). Ví se nicméně, že $BPP \neq NP$.

6 Terminologie

Tato část obsahuje definice termínů využívaných v Zero-knowledge protokolech.

6.1 Veřejná a Soukromá mince

V Zero-knowledge protokolech se často využívají zdroje náhodných čísel. Tento zdroj bývá označovaný termínem „mince“ *(coin)*, neboť jde o náhodný řetězec bitů, který je možno si představit jako hody mincí.

V závislosti na tom, zda jsou či nejsou tyto řetězce tajné (tj. zda se je druhá strana může dozvědět) je možno dělit protokoly na protokoly se Soukromou mincí *(Private coin)*, kdy je tento náhodný řetězec udržován v tajnosti, a protokoly s Veřejnou mincí *(Public coin)*, kdy se tento náhodný řetězec, nebo jeho část, druhá strana i v případě dodržení protokolu ve správný moment dozví.

6.2 Nerozlišitelnost náhodných proměnných

V [3] jsou definovány úrovně Nerozlišitelnosti *(Indistinguishability)* náhodných proměnných. Z těchto definic se v Zero-knowledge protokolech vychází.

Nerozlišitelnost demonstrují na příkladu dvou náhodných rozdělení U a V. Z jednoho náhodně zvoleného rozdělení je náhodně vybráno x hodnot, které jsou předány „soudci“. Úrovně nerozlišitelnosti jsou pak stanoveny na základě toho, zda dokáze soudce rozlišit, zda vzorek pochází z rozdělení U nebo V.

6.2.1 Perfektní Nerozlišitelnost

Pokud nedokáže soudce rozlišit původ vzorku ani v případě, že má k dispozici libovolně velký vzorek a libovolně dlouhý výpočetní čas, jsou si oba vzorky rovné, respektive jsou Perfektně Nerozlišitelné *(Perfect Indistinguishability)*.

6.2.2 Statistická Nerozlišitelnost

Pokud má soudce k dispozici libovolně dlouhý výpočetní čas, ale počet vzorků je omezen polynomicky, a nedokáže rozlišit původ vzorku, pak jsou vzorky Statisticky Nerozlišitelné *(Statistical Indistinguishability)*.

6.2.3 Výpočetní Nerozlišitelnost

Nedokáže-li soudce rozlišit původ vzorku za použití polynomického počtu vzorků a polynomického času, jsou vzorky Výpočetně Nerozlišitelné *(Computational Indistinguishability)*.

6.2.4 Vzájemný vztah

Z výše zmíněných definic vyplývá, že jsou-li vzorky Perfektně Nerozlišitelné, jsou i Statisticky Nerozlišitelné, a jsou-li Statisticky Nerozlišitelné, jsou i Výpočetně Nerozlišitelné.

6.3 Náhodné orákulum

Model s Náhodným orákulem *(random oracle)* byl poprvé publikován v roce 1993 Mihirem Bellarem a Phillipem Rogawayem v práci Náhodná orákula jsou praktická: paradigma pro tvorbu efektivních protokolů[1)][11]. Autoři považují Náhodná orákula za: „most mezi kryptografickou teorií a praxí“.[2)][11], a tvrdí, že je vhodné při návrhu protokolu nejprve pracovat v módu s Náhodným orákulem, a až poté toto Náhodné orákulum nahradit vhodně zvolenou funkcí.

Náhodné orákulum je teoretický stroj, ke kterému mají přístup všechny strany komunikace. Tento stroj funguje jako černá skřínka *(black-box)*, která pro každý unikátní vstup vrací náhodně zvolený výstup z množiny možných výstupů (pro stejný vstup vrací vždy stejný výstup). Stroj má k dispozici nekonečnou paměť, do které si ukládá jím spočítané kombinace vstup → výstup.

Jakmile Náhodné orákulum obdrží vstup, nejprve v paměti ověří, zda již tento vstup v minulosti obdrželo:

- pokud ano - vrátí jako výstup stejnou hodnotu, jako vrátilo minule
- pokud ne - vygeneruje nový, opravdu náhodný *(true random)* výstup, uloží si do paměti tuto novou kombinaci vstup → výstup a vrátí výstup

[1)] Random Oracles are Practical: Paradigm for designing Efficient Protocols
[2)] bridge between cryptographic theory and cryptographic practice

Náhodné orákulum si můžeme představit jako idealizovanou kryptograficky bezpečnou hashovací funkci.

Je nutné zmínit, že v roce 1998 Ran Canneti, Oded Goldreich a Shai Halevi v díle Nový pohled na metodologii náhodného orákula[3)] demonstrují, že: „existují podpisová a šifrovací schémata, která jsou bezpečná v modelu Náhodného orákula, ale pro které každý pokus o implementaci Náhodného orákula vede k schématům, která nejsou bezpečné.“[4)][12]

7 Kryptografický závazek

Kryptografický Závazek *(Commitment)* je způsob, kterým se může Zavazující se strana *(Committer)* v určitém bodě komunikace zavázat k nějaké hodnotě x tím způsobem, že předá pouze tento vypočítaný závazek $c = C(x)$. V pozdějším fázi komunikace pak může provést Zrušení Závazku *(Decommitment)*, například tím, že předá Přesvědčovanému původní hodnotu x.

Je například možné se zavázat k hodnotě x tím, že je spočítán hash $h = H(x)$ a jako Závazek je poslána pouze hodnota h. Při Zrušení Závazku pak může být předána Přesvědčovanému hodnota x.

Cílem Závazků v kryptografii je omezit Zavazující se straně při komunikaci možnost podvádění v případě, že je nutné, aby protokol probíhal a hodnoty byly generovány striktně v daném pořadí.

Kryptografické Závazky mají dvě hlavní vlastnosti - Zavázání *(Binding)* a Skrytí *(Hiding)*.

7.1 Zavázání

Aby dávalo využití Závazku smysl, nesmí mít Zavazující se strana možnost změnit hodnotu, ke které se zavázala, a úspěšně tak předstírat, že se zavázala k jiné hodnotě, než ke které se opravdu zavázala.

Zaváže-li se k hodnotě x vytvořením závazku $c = C(x)$, nesmí existovat hodnota y taková, že $C(x) = C(y)$. Při splnění této podmínky mluvíme o Perfektně Zavazujícím Závazku *(Perfectly Binding Commitment)*.

V praxi často není potřeba, aby byl závazek Perfektně Zavazující. V případě, že sice technický existují hodnoty y takové, že $C(x) = C(y)$, ale ani za použití libovolného množství výpočetního času není Zavazující se strana schopna tuto hodnoty najít, mluvíme o Statisticky Zavazujícím Závazku *(Statistically Binding Commitment)*.

[3)] The Random Oracle Methodology, Revisited

[4)] There exist signature and encryption schemes that are secure in the Random Oracle Model, but for which any implementation of the random oracle results in insecure schemes.

Pokud existuje y takové, že $C(x) = C(y)$, ale Zavazující se strana jej nedokáže najít v polynomicky omezeném čase (v závislosti na délce Závazku v bitech), pak jde o Výpočetně Zavazující Závazek *(Computationally Binding Commitment)*.

7.2 Skrytí

Druhou důležitou vlastností Závazků je Skrytí *(Hiding)*, neboli nemožnost Přesvědčovaného zjistit před Zrušením Závazku hodnotu, ke které se Zavazující se strana Zavázala.

Pokud je pravděpodobnost, že se i za použití libovolného množství výpočetního času podaří Přesvědčovanému získat $x = C^{-1}(c)$ nulová, mluvíme o Perfektně Skrývajícím Závazku *(Perfectly Hiding Commitment)*.

Je-li tato pravděpodobnost nenulová, ale zanedbatelná, pak jde o Statisticky Skrývající Závazek *(Statistically Hiding Commitment)*.

Pokud není možné získat $x = C^{-1}(c)$ za použití polynomického množství času (v závislosti na délce Závazku v bitech) s větší, než zanedbatelnou pravděpodobností, pak jde o Výpočetně Skrývající Závazek *(Computationally Hiding Commitment)*.

7.3 Vztah zavázání a skrytí

Vzhledem k tomu, že tyto dvě vlastnosti jsou protichůdné, Závazek, který je Perfektně (Statisticky) Zavazující je zpravidla Výpočetně Skrývající, a naopak Perfektně (Statisticky) Skrývající Závazek je zpravidla Výpočetně Zavazující.

Jak shrnují Ran Canetti a Marc Fischlin v práci Univerzálně poskládatelné závazky[1)]: „V jednoduchém modelu (tj. bez přístupu k nějaké ideální funkcionalitě) nemůže existovat univerzálně poskládatelný protokol Závazku, který nezahrnuje využití třetí strany v komunikaci, a zároveň umožňuje úspěšné dokončení, když jsou odesílatel i příjemce čestní.“[2)][13]

7.3.1 Demonstrace vztahu

Jedním z principů, pomocí kterého je možno Závazek vytvořit, je za pomoci klasické symetrické blokové šifry označené A. Obecně má tato šifra vstupy - otevřený text O, klíč K a šifrovaný text C. Zavazující se strana si zvolí náhodné O a K a spočítá $C = A_K(O)$. Nyní má dvě možnosti.

První možností je poslat jako Závazek hodnoty O, C a při Zrušení Závazku poté poslat hodnotu K. Spoléhající strana pak může klasicky spočítat $O = A^{-1}(C)$ a tím

[1)] Universally Composable Commitments

[2)] In the plain model (i.e., without access to some ideal functionality) there cannot exist universally composable commitment protocols that do not involve third parties in the interaction and allow for successful completion when both the sender and the receiver are honest.

ověřit, že hodnota Závazku skutečně odpovídá.

V tomto případě jde o Perfektně Zavazující Závazek, neboť existuje právě jedna hodnota K, pomocí které je možno získat C z O.

Na druhou stranu je ale Závazek pouze Výpočetně Skrývající, protože by Spoléhající strana, která by měla k dispozici libovolné množství výpočetního času, by postupným zkoušením všech možných K dokázala Závazek jednoznačně zjistit.

Druhou možností je poslat jako Závazek pouze hodnotu O a při Zrušení Závazku pak hodnoty C, K. Spoléhající se strana může pak opět klasicky ověřit hodnotu Závazku.

Obecně můžeme říci, že pro každé K existuje O takové, že platí $C = A_K(O)$. Pokud bude mít Zavazující se strana k dispozici libovolné množství výpočetního času, dokáže najít K' takové, že bude platit $C = A_{K'}(O')$. Proto je závazek pouze Výpočetně Zavazující. Ze stejného důvodu je ale Perfektně Skrývající.

7.4 Částečné Zrušení Závazků

U některých závazků je možno provést pouze Částečné Zrušení Závazku *(Partial Decommitment)*. Například je-li Závazek dlouhý 10 bitů, tak, zatímco při úplném Zrušení Závazku odhalí Zavazující se strana Přesvědčovanému hodnotu všech deseti bitů, u Částečného Zrušení Závazku může odhalit pouze jejich podmnožinu (v závislosti na použitém protokolu), například bity číslo 1, 3 a 7.

Závazky umožňující Částečné Zrušení je možno postavit například pomocí hashování vektorů nebo za použití Merklova stromu.

7.5 Pedersenův Závazek

V roce 1992 popsal Torben Pryds Pedersen ve své práci Neinteraktivní a z pohledu teorie informace bezpečně ověřitelné sdílení tajemství[3)][14] schéma Závazku, který je znám pod názvem Pedersenův Závazek.

Jde o Perfektně Skrývající Výpočetně Zavazující Závazek, který je postaven na problému diskrétního logaritmu. Této problematice se blíže věnuje kapitola 14.1.

7.5.1 Princip

Před přípravou Závazku je nutné určit dva generátory g, h: „Nechť jsou g a h prvky z G_q takové, že nikdo nezná $\log_g h$. Tyto prvky mohou být zvoleny Důvěryhodnou třetí stranou při inicializaci systému, nebo (některým) účastníkem komunikace pomocí protokolu házení mincí.“[4)][14]

[3)] Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing

[4)] Let g and h be elements of G, such that nobody knows $\log_g h$. These elements can either be chosen by a trusted center, when the system is initialized, or by (some of) the participants using a

Zavazující se strana se poté může zavázat k hodnotě $s \in \mathbb{Z}_q$ tak, že vygeneruje náhodné $t \in \mathbb{Z}_q$ a spočítá: $c = g^s h^t \pmod q$.

Při Zrušení Závázku pak dojde k předání hodnot s, t.

7.5.2 Vlastnosti

Bezpečnost Závazku stojí na faktu, že hodnoty g, h jsou zvoleny čestně (ať už jedním z účastníků či Důvěryhodnou třetí stranou), a že Zavazující se strana nezná hodnotu $\log_g h$.

Pedersenův Závazek je Perfektně Skrývající, neboť pro každý pár s, t existuje s', t' takové, že $g^s h^t \equiv g^{s'} h^{t'} \pmod q$. Proto se libovolně silný Přesvědčovaný nedokáže bez Zrušení Závazku dozvědět jeho hodnotu.

Závazek je pouze Výpočetně Zavazující, neboť zná-li Zavazující se strana $\log_g h$ (případně dokáže-li jej spočítat), pak může libovolně změnit hodnotu, ke které se zavázala.

Zavazující se strana, která má k dispozici pouze polynomický čas, logaritmus spočítat nedokáže (proto je Závazek Výpočetně Zavazující), ale Zavazujcící se strana neomezená polynomickým časem toto dokáže (proto nejde o Statisticky Zavazující Závazek).

8 Vlastnosti Zero-knowledge protokolů

Obecně musí Zero-knowledge protokoly splňovat několik základních podmínek. Tato základní sada podmínek je známá pod termíny Úplnost *(Completeness)*, Solidnost *(Soundness)*, Efektivita *(Efficiency)* a Zero-knowledge *(Zero-knowledge)*.

Tyto podmínky je možno formálně definovat například takto: „NP systém důkazu členství v L je algoritmus V, kde platí pro $\forall x$:“ [1)][15][2)]

8.1 Úplnost

Formálně je Úplnost definována takto: „Pokud platí: $x \in L$, pak $\exists \pi$, $V(x, \pi) = \text{ANO}$.“ Jinak řečeno, pokud je tvrzení x platné, pak existuje takový důkaz π, který přesvědčí Ověřovatele o platnosti tvrzení.

Protokol je Úplný právě tehdy, když je Čestný Dokazovatel schopný (s pravděpodobností $P \geq 2/3$) přesvědčit Čestného Ověřovatele o platnosti tvrzení. Jde o základní

coin-flipping protocol.

[1)] NP proof system for membership in L is algorithm V such that $\forall x$:

- Completeness: If $x \in L$, then $\exists \pi$, $V(x, \pi) = \text{ACCEPT}$
- Soundess: If $x \notin L$, then $\forall \pi$, $V(x, \pi) = \text{REJECT}$
- Efficiency: $V(x, \pi)$ halts after at most $poly(|x|)$ steps

[2)] Jednotlivé body citace jsou umístěny v relevantních podkapitolách

vlastnost, z níž se odvíjí, zda má daný protokol vůbec smysl.

8.2 Solidnost

Následuje definice Solidnosti: „Pokud platí: $x \notin L$, pak $\forall\pi$, $V(x,\pi) = \text{NE}$.“ Jinak řečeno, pokud je tvrzení x neplatné, pak pro všechny možné důkazy π Ověřovatel důkaz odmítne.

Solidnost protokolu se váže k možnostem Dokazovatele podvádět. Protokol je Solidní, pokud dokáže Nečestný Dokazovatel přesvědčit Ověřovatele o platnosti tvrzení, které platné není, pouze s malou pravděpodobností ($P < 1/3$).

Podobně jako u nerozlišitelnosti (viz kapitola 6.2) jsou i zde definovány úrovně Solidnosti. Pokud v rámci daného protokolu nedokáže výpočetně vázaný Dokazovatel podvádět, pak jde o protokol s Výpočetní Solidností. Pokud nedokáže Dokazovatel podvádět, ani když má k dispozici libovolné množství výpočetní síly, pak jde o protokol se Statistickou Solidností. Pokud je dokázáno, že pro daný protokol neexistuje pro Dokazovatele možnost podvádět, pak jde o Perfektně Solidní protokol.

8.2.1 Kompozice protokolů

Pravděpodobnost úspěšného podvádění Dokazovatele je možno snížit vícenásobným spuštěním protokolu. Je-li pravděpodobnost, že se Dokazovateli podaří přesvědčit Ověřovatele o nepravdě při jednom běhu protokolu $1/n$, pak při spuštění protokolu x krát může být pravděpodobnost snížena na $1/n^x$.

Obecně je možno toto vícenásobné spuštění protokolu vyřešit třemi různými způsoby.

První možností je tyto protokoly zřetězit sériově, tedy začít další kolo vždy po skončení předchozího kola. Nevýhodou tohoto systému je, že značně narůstá počet komunikačních kol.

Druhou možností je paralelní zpracování, kdy jsou protokoly vyhodnocovány zároveň.

Poslední možností je současné zpracování, což je speciální případ paralelního zpracování, ve kterém je nutno, aby všechny paralelní větve probíhaly zároveň, tj. všichni Ověřovatelé pošlou první zprávu, a až poté jim Dokazovatel všem zároveň odpoví.

Vhodnou kompozici protokolu je nutno řešit v závislosti přímo na konkrétním protokolu, neboť některé z těchto kompozic mohou být pro daný protokol nevhodné či přímo nebezpečné.

8.3 Efektivita

Pro efektivitu platí: „$V(x, \pi)$ se zastaví nejvíce po $poly(|x|)$ krocích.“ Jinak řečeno, Ověřovatel je schopný ověřit důkaz při vykonání počtu kroků, který je polynomický ve vztahu k bitové délce x.

Aby byl protokol v praxi použitelný, je nutné, aby Dokazovatel i Ověřovatel byli schopni v praxi provést výpočty, které dle protokolu provést mají.

8.4 Zero-knowledge

Důležitým přínosem [3] je definice termínu Zero-knowledge: „Tvrdíme, že Interaktivní systém důkazu pro L je Zero-knowledge, pokud pro každé $x \in L$ neprozradí Dokazovatel Ověřovateli v podstatně nic jiného, než že $x \in L$; a to i v případě, že se Ověřovatel nedrží systému důkazu, ale místo toho se pokusí (v polynomickém čase) podvést Dokazovatele a získat nějakou informaci.“[3)][3]

Na příkladu důkazu kvadratického zbytku *(quadratic residue)* (tedy zda je možno dané y zapsat jako $y \equiv w^2 \pmod{x}$ pro nějaké w), kterým se v této práci zabývají, uvádějí, že při Zero-knowledge by se ani Podvádějící Ověřovatel nejen, že neměl dozvědět w či prvočíselný rozklad x, ale žádnou informaci, která by mu umožnila spočítat cokoli lépe, než před touto interakcí. Je tedy možno říci, že při Zero-knowledge by se měl být Ověřovatel schopen dozvědět pouze to, co by si mohl spočítat sám i bez interakce s Dokazovatelem.

Obecně tedy Zero-knowledge vyjadřuje schopnost Ověřovatele podvádět, tedy pokusit se získat informaci, kterou získat nemá. I zde jsou, podobně jako u nerozlišitelnosti (viz kapitola 6.2), definovány úrovně Zero-knowledge v závislosti na dostupném výpočetním času Ověřovatele a jeho schopnosti podvádět. Pokud má Ověřovatel k dispozici polynomické množství času (v závislosti na délce důkazu v bitech), a přesto není schopen podvádět, pak jde o Výpočetní Zero-knowledge protokol. Pokud nedokáže Ověřovatel podvádět ani za použití libovolného množství výpočetního času, pak jde o Statistickou Zero-knowledge. Pokud je dokázáno, že pro Ověřovatele neexistuje možnost podvádět, pak jde o Perfektní Zero-knowledge protokol.

V praxi není vždy potřeba, aby použitý protokol neprozradil vůbec žádnou informaci navíc, stačí například, když neprozradí jen její klíčovou část. Při tomto uvolnění Zero-knowledge definice je možno získat například níže zmíněné verze protokolu.

[3)] We say an interactive proof system for L is zero-knowledge if for each $x \in L$, the prover tells the verifier essentially nothing, other than that $x \in L$; this should be the case even if the verifier chooses not to follow the proof system but instead tries (in polynomial time) to trick the prover into revealing something.

8.4.1 Zero-knowledge s Čestným Ověřovatelem

Pokud se jedná o Zero-knowledge protokol pouze tehdy, pokud se Ověřovatel drží přesně postupu protokolu, pak jde o Zero-knowledge s Čestným Ověřovatelem *(Honest Verifier Zero-knowledge)*. Typickým zástupcem je například Schnorrův protokol (viz kapitola 10.2).

8.4.2 Svědek

Svědek *(Witness)* je hodnota w, která je v takovém vztahu k x, že je pomocí ní možno prokázat, že $x \in L$.

V roce 1992 definovali Uriel Feige a Adi Shamir v práci Protokoly s nerozlišitelností svědků a skrytím svědků[4)][16] termíny Nerozlišitelnost Svědků *(Witness Indistinguishability)* a Skrytí Svědků (Witness Hiding).

„Neformálně mají tyto protokoly Nerozlišitelnost svědků pokud V nemůže rozlišit mezi dvěma běhy protokolu které se liší pouze v specifickém svědkovi, kterého P použil. Například nechť je z hamiltonský graf s několik hamiltonskými cykly. Důkaz hamiltonismu je WI, pokud je protokol z pohledu V stejný bez ohledu na to, který cyklus w z x P zná a použil.“[5)][16]

Skytí Svědků pak definují jako: „Protokol (P, V) je WH, pokud účast na protokolu nepomůže V spočítat žádné nové svědky, které na začátku protokolu neznal.“[6)][16]

9 Interaktivní Zero-knowledge protokoly

U interaktivních protokolů je k vytvořneí důkazu potřeba vstup Ověřovatele, zpravidla ve formě náhodného řetězce - Výzvy.

9.1 Protokol Arthur-Merlin

Za předchůdce Interaktivních systémů důkazu je možno považovat takzvaný Arthur-Merlin systém důkazu, který byl zveřejněn v roce 1985. Autorem práce Výměna teorie grup za náhodnost[1)][17] je maďarský profesor matematiky László Babai.

V protokolu definuje dvě strany, krále Artuše a kouzelníka Merlina: „Král Artuš zná nadpřirozené intelektuální schopnosti Merlina, ale nevěří mu. Jak může Merlin

[4)] Witness Indistinguishable and Witness Hiding Protocols

[5)] Informally, these protocols are witness indistinguishable if V cannot distinguish between two executions of the protocol which differ in the specific witness P is using. For example, let z be a Hamiltonian graph with several Hamiltonian cycles. A proof of Hamiltonicity is WI if V's view is the same no matter which cycle w in x party P knows and uses.

[6)] Protocol (P, V) is WH if participating in the protocol does not help V to compute any new witnesses to the input which he did not know at the beginning of the protocol.

[1)] Trading Group Theory for Randomness

přesvědčit inteligentního, ale netrpělivého krále, že řetězec x patří do daného jazyka L?“[2)][17]

Práce popisuje více verzí protokolu v závislosti na komunikačním schématu (kolik se posílá zpráv a kdo posílá první zprávu) ve všech je strana „Artuš“ reprezentována klasickým počítačem s generátorem náhodných čísel, zatímco strana „Merlin“ je zastoupena Nedeterministickým Turingovým strojem (který zná náhodná čísla vygenerované Artušem).

9.2 Interaktivní systémy důkazu

„Interaktivní systém důkazu *(Interactive proof system)* je metoda, pomocí které jedna strana s neomezenými zdroji, nazývaná Dokazovatel *(Prover)* může přesvědčit druhou stranu s omezenými zdroji, nazývanou Ověřovatel *(Verifier)*, o pravdivosti tvrzení. Ověřovatel může házet mincí, ptát se Dokazovatele opakovaně na otázky a spouštět efektivní testy nad odpovědmi Dokazovatele před tím, než se rozhodne, jestli je přesvědčen.“[3)][18]

Tento termín byl poprvé popsán v [3]: „Stále povolujeme Ověřovateli pouze polynomický čas a Dokazovateli libovolnou výpočetní sílu, nyní ale povolujeme oběma stranám házet nezaujatou mincí. Výsledkem je pravděpodobnostní verze NP, která umožňuje s malou pravděpodobností chybu. Nicméně k získání toho, co by mohlo být úplné zobecnění nápadu, je nutné povolit Dokazovateli a Ověřovateli interakci (například posílání zpráv tam a zpět) a možnost udržet své hody v tajnosti.“[4)][3]

Zatímco protokol popsaný v [3] je protokol se Soukromou mincí, v případě protokolu Arthur-Merlin jde o protokol s Veřejnou mincí. Goldwasser a Sipser ve své práci Soukromé versus Veřejné mince v Interaktivních systémech důkazu[5)][18] dokázali, že Interaktivní systém důkazu pro libovolný jazyk existuje právě tehdy, když pro tento jazyk existuje Arthur-Merlin protokol.

Na toto tvrzení dále navázali ve své práci z roku 1991 s názvem Důkazy, které nepředávají nic než jejich platnost nebo všechny jazyky v NP mají zero-knowledge

[2)] King Arthur recognizes the supernatural intellectual abilities of Merlin but doesn't trust him. How should Merlin convince the intelligent but impatient King that a string x belongs to a given language L?

[3)] An interactive proof system is a method by which one party of unlimited resources, called the prover, can convince a party of limited resources, call the verifier, of the truth of a proposition. The verifier may toss coins, ask repeated questions of the prover, and run efficient tests upon the prover's responses before deciding whether to be convinced.

[4)] We will still allow the verifier only polynomial time and the prover arbitrary computing power, but will now allow both parties to flip unbiased coins. The result is a probabilistic version of NP, where a small probability of error is allowed. However, to obtain what appears to be the full generality of this idea, we must also allow the prover and verifier to interact (i.e., to talk back and forth) and to keep secret their coin tosses.

[5)] Private coins versus public coins in interactive proof systems

systémy důkazu.[6)][19] Goldreich, Micali a Wigderson, kde dokázali, že pro všechny jazyky v NP existuje Zero-knowledge systém důkazu.

9.3 Zero-knowledge Důkaz

Termín Zero-knowledge Důkaz *(Zero-knowledge proof)* definovali roku[7)] 1985 v práci s názvem Komplexita znalosti Interaktivního systémů důkazu[8)][3] kryptografové Shafi Goldwasser, Silvio Micali a Charles Rackoff.

V abstraktu práce tvrdí: „Důkaz teorému zpravidla obsahuje více informací než pouhý fakt, že je teorém pravdivý. Například k důkazu toho, že je graf hamiltonovský stačí předložit hamiltonovskou cestu; nicméně toto obsahuje více informací, než pouze jediný bit znamenající je hamiltonovský / není hamiltonovský. V této práci je definována teorie výpočetní složitost informace obsažené v důkazu. Zero-knowledge důkaz je definován jako důkaz, který neobsahuje žádnou další informaci kromě informace o platnosti diskutovaného tvrzení.“[9)][3] Práce se zabývá termínem Zero-knowledge v kontextu interaktivních protokolů.

9.4 Zero-knowledge Argument

V závislosti na Solidnosti protokolu je možno rozlišovat „silnější“ Důkazy *(Proofs)* a „slabší“ Argumenty *(Arguments)*. „Je užitečné rozlišovat mezi Zero-knowledge Důkazy se Statistickou Solidností a Zero-knowledge Argumenty s Výpočetní Solidností. Obecně Důkazy mohou mít pouze Výpočetní Zero-knowledge, zatímco Argumenty mohou mít Perfektní Zero-knowledge.“[10)][20]

V roce 1988 v práci Důkazy Znalosti s minimálním odhalením[11)][21] autoři Gilles Brassard, David Chaum a Clade Crépeau dokázali, že pro všechny jazyky v NP existuje Zero-knowledge Argument s Perfektní Zero-knowledge.

[6)] Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems

[7)] revidovaná verze byla vydána v roce 1989

[8)] The Knowledge Complexity of Interactive Proof Systems

[9)] Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian. In this paper a computational complexity theory of the "knowledge"contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question.

[10)] It is useful to distinguish between zero-knowledge proofs, with statistical soundness, and zero-knowledge arguments with computational soundness. In general proofs can only have computational zero-knowledge, while arguments may have perfect zero-knowledge.

[11)] Minimum Disclosure Proofs of Knowledge

10 Důkaz Znalosti

Ačkoli [3] mluví o konceptu Důkazu Znalosti *(Proof of Knowledge)*, formálně tento termín nedefinuje. V práci Zero-knowledge důkazy identity[1)][22] na toto reagují Uriel Feige, Amos Fiat a Ali Shamir takto: „Termín 'zero knowledge důkazy' je mírně zavádějící, neboť dokazovatel A odhaluje jeden bit znalosti ověřovateli B (konkrétně že I patří do L). Naším prvním cílem v této práci je ukázat, že je možné rozšířit tuto myšlenku na 'opravdové zero knowledge důkazy', které neodhalují ani tento jeden bit. Základní myšlenkou je nahradit 'znalost' 'znalostí o znalosti': cílem A není dokázat, že I náleží do L, ale dokázat, že zná vztah I a L. Z pohledu B se nedozvěděl žádnou informaci o 'reálném světě' (I, L a jejich vztah) - pouze o stavu znalosti A o reálném světě.“[2)][22]

Tuto myšlenku rozšiřují Mihir Bellare a Oded Goldreich v roce 1993 v práci O definici Důkazů Znalosti[3)][23], kde formálně definují termín Zero-knowledge Důkaz Znalosti *(zero-knowledge proof of knowledge)*.

10.1 Sigma protokol

Sigma protokoly jsou obecnou kategorií Interaktivních Zero-knowledge protokolů s Čestným Ověřovatelem, které jsou postaveny na třech komunikačních kolech (tj. v rámci protokolu jsou poslány 3 zprávy).

Sigma protokoly vychází ze schématu popsaného Amosem Fiatem a Adi Shamirem v [26].

Tyto protokoly byly pojmenovány Sigma protokoly, neboť schéma komunikace připomíná řecké písmeno Sigma - Σ.

Sigma protokoly používají komunikační schéma, které se skládá ze tří zpráv. Jsou to Závazek→Výzva→Odpověď *(Commitment→Challenge→Response)*. Dokazovatel v první zprávě učiní Závazek k nějaké hodnotě, Ověřovatel mu poté pošle náhodou Výzvu. V poslední zprávě pošle Dokazovatel Odpověď, kterou spočítá pomocí Závazku a Výzvy. Ověřovatel poté může ověřit vztah Závazku a Odpovědi a tím zjistit, že je poslaná hodnota platná.

Typickým zástupcem Sigma protokolů je Schnorrův protokol.

[1)] Zero Knowledge Proofs of Indentity

[2)] The name "zero knowledge proofs"is slightly misleading, since the prover A reveals one bit of knowledge to the verifier B (namely that I belongs to L). Our first objective in this paper is to show that it is possible to extend this notion to "truly zero knowledge proofs"which do not even reveal this single bit. The basic idea is to replace "knowledge"by "knowledge about knowledge": A's goal is not to prove that I belongs to L, but to prove that he knows the status of I with respect to L. From B's point of view, he didn't get any information whatsoever about "the real world"(I,L and their relationships) - only about A's state of knowledge concerning the real world.

[3)] On Defining Proofs of Knowledge

10.2 Schnorrův protokol

Jedním z prvních protokolů pro Interaktivní Důkaz Znalosti je takzvaný Schnorrův protokol. Jeho autorem je německý matematik a kryptograf Claus-Peter Schnorr. Ten v roce 1990 vydal práci Efektivní identifikace a podpisy pro čipové karty[4)][24], kde popisuje využití Důkazu Znalosti diskrétního logaritmu v kontextu elektronické identifikace a elektronických podpisů optimalizovaných pro použití na čipových kartách. Bezpečnost algoritmů je postavena na problému diskrétního logaritmu.

Tento protokol byl v roce 1989 patentován pod číslem US4995082A - Metoda identifikace uživatelů a generování a ověřování elektronických podpisů v systému výměny dat.[5)]. Patent vypršel v roce 2008.

Ačkoli je Schnorrův protokol často uváděný jako příklad Zero-knowledge protokolu, technicky nejde o Zero-knowledge protokol, neboť podvádějící Ověřovatel je schopen manipulací dat získat informaci navíc. Na toto upozorňuje samotný Schnorr: „Schéma není Zero-knowledge protože trojice (x, y, e) může být konkrétní řešení rovnice $x = \alpha^y v^e \pmod p$ díky faktu, že výběr e může být závislý na x.“footnoteThe scheme is not zero-knowledge because the tripel (x, y, e) may be a particular solution of the equation $x = \alpha^y v^e \pmod p$ due to the fact that the choice of e may depend of x.[24]

Jedná se tedy o příklad Zero-knowledge s Čestným Ověřovatelem.

10.2.1 Schéma

Obrázek 10.1 shrnuje průběh Schnorrova protokolu pro důkaz znalosti diskrétního logaritmu x z rovnice $h = g^x$. Veškeré výpočty probíhají modulo q. Dokazovatel je označen písmenem P, Ověřovatel písmenem V.

- P si nejprve zvolí náhodný Závazek r, spočítá $u = g^r$ a pošle jej V
- V zvolí náhodnou Výzvu c a pošle ji zpět P
- P spočítá Odpověď $z = r + cx$ a pošle ji V
- V ověří, zda platí rovnice $g^z = uh^c$. Pokud ano, je Závazek platný

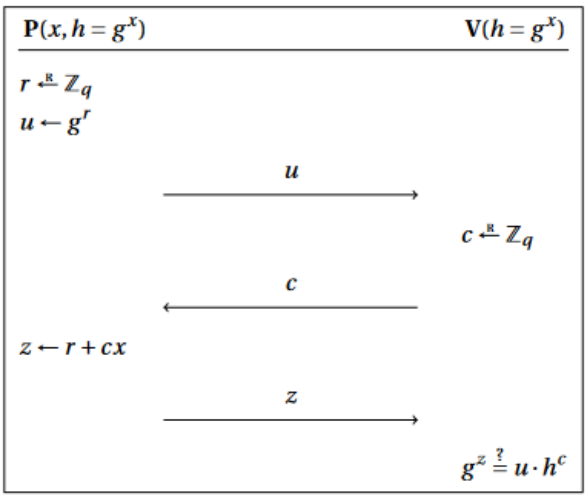
11 Fiat-Shamirova transformace

Amos Fiat a Adi Shamir v [26] také definují obecný způsob, kterým je možno změnit Interaktivní Důkaz Znalosti s Veřejnou mincí na Neinteraktivní Důkaz Znalosti. Tento

[4)] Efficient identification and signatures for smart cards

[5)] Method for identifying subscribers and for generating and verifying electronic signatures in a data exchange system

Obr. 10.1 Schéma Schnorrova protokolu [25]



protokol je známý pod názvem Fiat-Shamirova heuristika nebo také Fiat-Shamirova transformace. V [26] nebyl uveden důkaz bezpečnosti protokolu, ten doplnili v roce 1996 David Pointcheval a Jacques Stern v práci Důkazy bezpečnosti pro podpisová schémata[1)][27], kde prokázali bezpečnost Fiat-Shamirovy transformace v modelu s Náhodným orákulem.

„Myšlenkou za Fiat-Shamirovou transformací je, že namísto toho, aby Ověřovatel poslat náhodnou výzvu Dokazovateli, Dokazovatel může spočítat tuto hodnotu sám za použití náhodné funkce, například kryptografické hashovací funkce.“[2)][28]

„Tato transformace se může zdát jasná, ale bohužel se ukazuje, že její praktická aplikace je velmi ošidná. Konkrétně - Dokazovatel vygeneruje náhodnou hodnotu Výzvy za použití kryptografické hashovací funkce - ale jaké jsou vstupy této funkce? Ukazuje se, že pokud vyberete špatné vstupy, většinou to znamená, že je systém rozbitý.“[3)][28] V závislosti na vstupech této náhodné funkce můžeme rozlišit slabou Fiat-Shamirovu transformaci , kde je jako vstup použita pouze hodnota Výzvy, a silnou Fiat-Shamirovu transformaci, kde je to hodnota Výzvy a tvrzení, jehož platnost chceme dokázat.

Bezpečností Fiat-Shamirovy transformace se věnují mimo jiné například David Bernhard, Olivier Pereira a Bogdan Warinschi v práci Jak se neprokázat - nástrahy Fiat-Shamirovy heuristiky s aplikací na Helios[4)][29], kde tvrdí: „V situacích, kde si může

[1)] Security Proofs for Signature Schemes

[2)] The idea behind the Fiat-Shamir transformation is that instead of having the verifier send a random challenge value to the prover, the prover can compute this value themselves by using a random function, such as a cryptographic hash function.

[3)] This transformation may seem straightforward, but unfortunately, it tends to be very tricky in practice. In particular, the prover generates the random challenge value using a cryptographic hash function- but what are the inputs? It turns out that if you choose the wrong inputs, it usually means your proof system is broken.

[4)] How not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios

podvádějící Dokazovaelt volit tvrzení adaptivně, nese Fiat-Shamirova transformace důkazy, které nejsou Solidní či Extrahovatelné. A při tom tyto situace přirozeně nastávají v systémech, kde jsou použity Zero-knowledge důkazy k vynucení čestného chování.“[5)][29]

Práce dále prokazuje bezpečnost silné Fiat-Shamirovy transformace v modelu s Náhodným orákulem.

12 Neinteraktivní Zero-knowledge protokoly

Druhou velkou skupinou Zero-knowledge protokolů jsou Neinteraktivní protokoly. Na rozdíl od Interaktivních protokolů nevyžadují obousměrnou komunikaci mezi Dokazovatelem a Ověřovatelem, komunikace probíhá striktně ze strany Dokazovatele Ověřovateli. Všechny nutné informace jsou předány zpravidla zároveň v jedné zprávě.

Neinteraktivní protokoly představily v roce 1988 Manuel Blum, Paul Feldman a Silvio Micali v práci Neinteraktivní Zero-knowledge a jeho aplikace[1)][30]. K tomuto využívají takzvaný CRS, neboli „Společný referenční řetězec“ *(common refrence string)* což je náhodný řetězec známý oběma stranám před započetím komunikace (například vygenerovaný Důvěryhodnou třetí stranou). „Sdílí-li Ověřovatel a Dokazovatel společný náhodný řetězec, může Dokazovatel Neinteraktivně, a přesto v Zero-knowledge, přesvědčit Ověřovatele o platnosti libovolného teorému, který objeví.“[2)][30]

V práci využívají koncept náhodného Společného referenčního řetězce k definici prvního systému asymetrické kryptografie bezpečného proti CCA útoku, tj. se zvoleným zašifrovaným textem *(Chosen-ciphertext attack)*, čímž vyřešili v té době otevřenou otázku, zda takový kryptosystém může existovat.

Výše zmíněná práce předpokládá použití Společného referenčního řetězce pro Důkaz jednoho teorému, neboli na jedno použití. Použití jednoho Společného referenčního řetězce pro více důkazů se věnují Uriel Feige, Dror Lapidot a Adi Shamir v práci Mnohonásobné Neinteraktivní Zero-knowledge Důkazy za obecných předpokladů.[3)][31]. „V této práci ukazujeme, jak postavit Neinteraktivní Zero-knowledge Důkazy pro jakýkoli NP problém za použití obecných předpokladů (namísto předpokladů teorie čísel), a jak dovolit polynomicky mnoha Dokazovatelům dát polynomické množství takových Důkazů založených na jednom řetězci. Naše konsturkce může být použita v kryptografických aplikacích, ve kterých je Dokazovatel omezen polynomickým časem.“[4)][31]

[5)] in situations where malicious provers can select their statements adaptively, the weak Fiat-Shamir transformation yields unsound/unextractable proofs. Yet such settings naturally occur in systems when zero-knowledge proofs are used to enforce honest behavior.

[1)] Non-Interactive Zero-Knowledge and Its Applications

[2)] If the prover and the verifier share a common random string, the prover can non-interactively and yet in zero-knowledge convince the verifier of the validity of any theorem he may discover.

[3)] Multiple Noninteractive Zero Knowledge Proofs Under General Assumptions

[4)] In this paper we show how to construct noninteractive zero knowledge proofs for any NP statement

12.1 zk-SNARK

V roce 2011 představili Nir Bitansky, Ran Canetti, Alessandro Chiesa a Eran Tromer svou práci Z extrahovatelné odolnosti proti kolizím k Stručnému Neinteraktivnímu Argumentu Znalosti a zase zpět[5)][32]. V této práci definují termín SNARK, neboli Stručný Neinteraktivní Argument Znalosti *Succint Non-Interactive Argument of Knowledge*.

Termín Stručný autoři vysvětlují takto: „Naše konstrukce zajišťuje, že komunikační komplexita a časová komplexita Ověřovatele jsou svázány polynomicky s bezpečnostním parametrem, velikostí instance a logaritmem času, který je potřeba k ověření platného Svědka pro danou instanci. Tento polynom je nezávislý na použitém NP jazyce, je tedy 'univerzální'.“[6)][32]

Autoři dokazují existenci SNARKu za předpokladu existence extrahovatelné hashovací funkce odolné proti kolizím[7)], což je autory definované zesílení předpokladu pro klasickou hashovací funkci odolnou proti kolizím.

Ze SNARKu je možno kombinací s Neinteraktivním Zero-knowledge protokolem postaveným na modelu se Společným referenčním řetězcem získat zk-SNARK, neboli Zero-knowledge Stručný Neinteraktivní Argument Znalosti.

12.2 Bulletproofs

V roce 2017 byla vydána práce Bulletproofs: Krátké důkazy pro důvěrné transakce a další[8)][33], ve které její autoři představují nový typ Neinteraktivního Zero-knowledge Argumentu pojmenovaný Bulletproof. Jde o Zero-knowledge Argument Znalosti postavený na předpokladu náročnosti výpočtu diskrétního logaritmu. Interaktivita protokolu je odstraněna aplikací Fiat-Shamirovi transformace.

Výhodou tohoto protokolu jsou krátké důkazy (s logaritmickou velikostí s ohledem na délku Svědka) a také fakt, že nevyžadují přípravu Důvěryhodnou třetí stranou.

Tento protokol se hodí zejména k důkazu, že se určitá číselná hodnota nachází v definovaném rozsahu. „Navíc, Bulletproofs podporují agregaci intervalových důkazů, tedy že Dokazovatel může dokázat, že m závazků leží v daném intervalu přidáním k

under general (rather than number theoretic) assumptions, and how to enable polynomially many provers to give polynomially many such proofs based on a single random string. Our constructions can be used in cryptographic applications in which the prover is restricted to polynomial time.

[5)] From Extractable Collision Resistance to Succinct Non-Interactive Argument of Knowledge, and Back Again

[6)] Our construction ensures that the communication complexity and the verifier's time complexity are bounded by a polynomial in the security parameter, the size of the instance, and the logarithm of the time it takes to verify a valid witness for the instance; this polynomial is independent of the specific NP language at hand, i.e., is "universal"

[7)] Extractable collision-resistant hash function

[8)] Bulletproofs: Short Proofs for Confidental Transactions and More

délce důkazu pouze $O(\log(m))$ elementů grupy.“[9)][33]

12.3 zk-STARK

V roce 2018 byl představen další protokol zk-STARK, neboli Zero-knowledge Škálovatelný Transparentní Argument Znalosti *(Zero-knowledge Scalable Transparent Argument of Knowledge)*. Autory práce Škálovatelná, transparentní a postkvantově bezpečná výpočetní integrita[10)][34] jsou Eli Ben-Sasson, Iddo Bentov, Yinon Horesh a Michael Riabzev. „Zde demonstrujeme první realizaci transparentního ZK systému (ZK-STARK), ve kterém je verifikace roste exponenciálně rychleji než velikost databáze, a navíc toto exponenciální zrychlení ve verifikaci je možno sledovat konkrétně pro smysluplné a sekvenční výpočty, jak je popsáno dále.“[11)][34]

Hlavními výhodami tohoto Neinteraktivní Argumentu Znalosti je, že je Transparentní, neboli že nevyžaduje počáteční nastavení Důvěryhodnou třetí stranou, a také že je Škálovatelný, neboť čas potřebný pro tvorbu důkazu roste kvazi-lineárně a čas potřebný na ověření roste polylogaritmicky.

Kvazi-lineárním časem je myšlena výpočetní náročnost ($O(n\log(n))$, tedy horší, než lineární, ale stále lepší, než kvadratický. Polylogaritmický čas pak znamená ($O(\log(n)^k)$, tedy horší, než logaritmický, ale lepší, než lineární výpočetní čas.

13 Praktické použití zero-knowledge protokolů

Ačkoli z teoretického hlediska byly Zero-knowledge protokoly zkoumány již od 80. let 20. století, praktických aplikací těchto protokolů příliš nebylo.

Toto se změnilo zejména po roce 2010 s příchodem praktických neinteraktivních Zero-knowledge protokolů využitých spolu s Blockchainy zejména v oblasti kryptoměn. Kromě toho mají tyto protokoly také možné využití pro autentizaci, v elektronických volbách či v systémech elektronických dokladů.

13.1 V rámci jiných protokolů

Zero-knowledge protokoly mají své využití například i v rámci jiných protokolů. Jako příklad můžeme uvést MPC, neboli výpočet mnoha stran *(Multi party computing)*,

[9)] Moreover, Bulletproofs supports aggregation of range proofs, so that a party can prove that m commitments lie in a given range by providing only an additive $O(\log(m))$ group elements over the length of a single proof.

[10)] Scalable, transparent and post-quantum secure computational integrity

[11)] Here we report the first realization of a transparent ZK system (ZK-STARK) in which verification scales exponentially faster than database size, and moreover, this exponential speedup in verification is observed concretely for meaningful and sequential computations, described next.

které tyto protokoly využívají k důkazu, že se strany komunikace drží protokolu a nepokoušejí se podvádět.

13.2 Identifikace a autentizace

Problematice využití Zero-knowledge protokolů pro identifikaci či autentizaci se věnovali již v 80. a 90. letech například Fiat a Shamir v [26], Feige, Fiat a Shamir v [22] či Schnorr v [24].

Později byl definován termín Zero-knowledge Důkaz hesla *(Zero-knowledge password proof)* jako: „Interaktivní zero knowledge důkaz znalosti dat derivovaných z hesla sdílených mezi Dokazovatelem a odpovídajícím Ověřovatelem.“[1)][35] Tyto důkazy se používají například v rámci autentizačního protokolu EKE, neboli Šifrovaná výměna klíčů *(Encrypted Key Exchange)*, který byl demonstrován v práci Stevena M. Bellovina a Michaela Merritta s názvem Šifrovaná výměna klíčů: Protokoly založené na heslu odolné proti slovníkovým útokům[2)][36].

Část III této práce obsahuje praktickou ukázku možného využití Interaktivního Zero-Knowledge Argumentu Znalosti hesla pomocí protokolu odvozeného od Fiat-Shamirova identifikačního schématu.

13.3 Kryptoměny

Vlastnosti Zero-knowledge protokolů jsou často využívány v kombinaci s distribuovanou decentralizovanou databází - Blockchainem. Často jsou tyto protokoly využívány v souvislosti s kryptoměnami.

13.3.1 Zcash

Protokol Zerocoin vznikl v roce 2013, jeho autoři jsou Matthew D. Green, Ian Miers a Chrstina Garman.

Zerocoin původně vznikl jako rozšíření pro kryptoměnu Bitcoin, které si kladlo za cíl umožnit v rámci Bitcoinu anonymní transakce. „Tento projekt začal jako návrh rozšíření, nazvané Zerocoin, Bitcoinu, které umožňuje uživatelům míchat své vlastní mince. Spolupráce mezi původními členy project Zerocoin, kryptografy MIT, Technionu a Univerzity Tel Aviv přinesla mnohem efektivnější protokol, který umožňuje přímé soukromé platby jiným uživatelům bez vyzrazení částky.“[3)][37]

[1)] An interactive zero knowledge proof of knowledge of password-derived data shared between a prover and the corresponding verifier.

[2)] Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks

[3)] This project began with a proposed extension, called “Zerocoin”, to the Bitcoin protocol that allowed users to mix their own coin. A collaboration between the the original Zerocoin project members and cryptographers at MIT, The Technion, and Tel Aviv University, has produced a far more efficient protocol that allows for direct private payments to otherusers of hidden value.

Ve stejném roce Matthew D. Green zveřejnil vylepšený protokol Zerocash.

Zerocash využívá Zero-knowledge protokol zk-SNARK (viz kapitola 12.1).

„Zerocash rozšiřuje protokol Bitcoinu přidáním nových druhů transakcí, které přidávají oddělenou měnu zachovávající soukromý, v níž transakce neodhalují zdroj platby, její cíl ani částku. Zerocash vytváří vlastní anonymní měnu, které existuje vedle (neanonymní) základní měny, kterou označujeme Basecoin. Každý uživatel může převést (neanonymní) basecoiny na (anonymní) mince Zerocash protokolu, které nazýváme Zerocoiny.“[4][38]

Na tomto protokolu vznikla v roce 2016 nová kryptoměna, Zcash.

13.3.2 Monero

Druhým příkladem kryptoměny, která využívá Zero-knowledge protokoly, je kryptoměna Monero. „Monero je decentralizovaná kryptoměna. Využívá veřejnou distribuovanou knihu záznamů s technologiemi zvyšujícími soukromí, které skrývají transakce, aby dosáhli anonymity a zastupitelnosti. Pozorovatelé nedokáží dešifrovat adresy, které obchodují s monerem, částky transakcí, zůstatky na adresách ani historii transakcí. Protokol je open source a založený na technologii CryptNote.“[5][39]

Monero využívá Zero-knowledge protokol Bulletproofs (viz kapitola 12.2).

13.4 Elektronické volby

Klasický systém voleb je z uživatelského hlediska nepraktický, neboť vyžaduje fyzickou přítomnost na konkrétním místě v definovaném časovém intervalu, což je v první řadě nepraktické a dále může nést další problémy zejména pro osoby z omezenou pohyblivostí či dlouhou pracovní dobou. Proto se i zde objevuje tlak na přechod do online světa, umožnění voleb z pohodlí domova a zavedení systému elektronických voleb, což již bylo v některých státech (například Austrálie, Estonsko, Švýcarsko, Rusko či Spojené státy) alespoň částečně umožněno (nutno podotknout, že ve velkém množství těchto systému byly objeveny vážné bezpečnostní nedostatky).

Obecně jsou od volebního systému očekávány tyto základní vlastnosti [40]:

- anonymita - nesmí být možné zjistit, kdo jak volil

[4] Zerocash extends Bitcoin's protocol by adding new types of transactions that provide a separate privacy-preserving currency, in which transactions reveal neither the payment's origin, destination, or amount. Zerocash creates a separate anonymous currency, existing alongside a (non-anonymous) base currency, which we refer to as Basecoin. Each user can convert (non-anonymous) basecoins into (anonymous) Zerocash coins, which we call zerocoins.

[5] Monero is a decentralized cryptocurrency. It uses a public distributed ledger with privacy-enhancing technologies that obfuscate transactions to achieve anonymity and fungibility. Observers cannot decipher addresses trading monero, transaction amounts, address balances, or transaction histories. The protocol is open source and based on CryptoNote.

- integrita - hlas musí být započítán přesně tak, jak byl zadán
- důvěrnost - nejsou prozrazena žádná osobní data o voličích
- ověřitelnost hlasu - musí být ověřitelné, že byl hlas skutečně zaznamenán

Právě tyto vlastnosti může do volebního systému přinést použití zero-knowledge protokolu.

Zero-knowledge volební systémy zpravidla využívají blockchainů a neinteraktivních protokolů. Na to téma vznikla například práce Plně anonymní e-volební protokol užívající obecný zk-SNARK a Chytré kontrakty[6)][41] či práce Anonymní volební systém založený na blockchainu užívající zkSNARK[7)][42].

Proces voleb popsaný v [42] je zjednoduše vysvětlen v [40] takto[8)]:

1. volební komise přidělí každému voliči ID, PIN a veřejný klíč (ten je vygenerován například na základě údajů na dokladu totožnosti). Tyto tokeny jsou podepsány volební komisí
2. všechny tokeny jsou zapsány do veřejného dostupného blockchainu a ten je zapečetěn
3. při samotné volbě:
   (a) volič předloží svůj doklad, zadá PIN a provede volbu
   (b) volební stroj na základě veřejného klíče a PINu vygeneruje náhodné řetězce S, R. Tyto mohou být použity k identifikaci voličova ID (a tím ověření, že již toto ID nemůže znovu hlasovat).
   (c) na základě S, R je vygenerován Zero-knowledge důkaz znalosti R. Všechny tyto transakce jsou vloženy do blockchainu, aby mohly být veřejně ověřitelné
   (d) hlas podepsaný voličovým soukromým klíče je za použití ZKP nahrán do blockchainu
4. volič může díky znalosti S a R ve veřejném Blockchainu kdykoli ověřit svůj hlas

Tento protokol splňuje výše zmíněné požadované vlastosti, následujícím způsobem:

- anonymita - protokol je anonymní, pouze pokud je ID generováno důvěryhodnou stranou a důvěryhodným způsobem

[6)] A Fully Anonymous e-Voting Protocol Employing Universal zk-SNARKs and Smart Contracts
[7)] Blockchain Based Anonymous Voting System Using zkSNARKs
[8)] nutno podotknout, že je v článku popisován elektronický volební systém, který vyžaduje fyzickou přítomnost na místě konání voleb)

- integrita - integrita hlasu je zajištěna elektronickým podpisem voliče
- důvernost - volič volí na základě svého náhodně vygenerováného ID, ze kterého není možné získat další osobní údaje
- ověřitelnost hlasu - volič může díky znalosti S a R kdykoli ověřit svůj hlas

Aritra Banerjee v [41] využívá navrhuje jiný přístup: „Využijeme koncept zk-SNARKu v Zcash, abychom provedli volební fázi voleb a zkHawk protokol pro chytré kontrakty k sečtení výsledků voleb.“[9)][41]

13.5 Elektronické doklady

Rozvoj elektronizace Státní správy přináší mimo jiné tlak na zavedení elektronických verzí osobních dokladů, jako jsou například občanský průkaz, řidičský průkaz či cestovní pas. Díky sílícímu tlaku na ochranu osobních údajů (například nařízení GDPR - Obecné nařízení o ochraně osobních údajů *(General Data Protection Regulation)*) jsou hledány způsoby, jak u těchto elektronických dokladů minimalizovat předávané informace. Jedním z těchto způsobů je selektivní zveřejňování *(selective disclosure)*, které umožňuje s dokladu předat jen několik vybraných atributů (například předání pouze jména a příjmení, ale nikoli adresy bydliště, i když doklad obsahuje všechny tyto informace).

Druhým způsobem, jak minimalizovat přenášená data, jsou právě Zero-knowledge protokoly. V tomto kontextu je často citován příklad prokázání dosaženého věku (například při nákupu alkoholických nápojů) bez vyzrazení data narození, nicméně není to jediné možné využití těchto protkolů v kontextu elektronických dokladů.

Část IV této práce obsahuje praktickou ukázku možného využití neinteraktivního Zero-Knowledge Argumentu k prokázání dosaženého věku pomocí hash chainů.

13.5.1 Evropská Digitální identita

Dne 3.6.2021 byl zveřejněn návrh novely nařízení Evropského parlamentu a rady č. 910/2014 (eIDAS) [43]. Tato novela si klade za cíl ustanovit rámec pro Evropskou Digitální identitu - celoevropský rámec pro sdílení elektronických dokladů. Technické a provozní specifikace zatím nejsou známy (měly by být známy nejdříve na podzim 2022), nicméně existuje předpoklad, že bude tento systém využívat mimo jiné i právě Zero-knowledge protokoly.

Jedním z možných rámců, které mohou být k tomuto účelu využity, je koncept Self-Sovereign Identity.

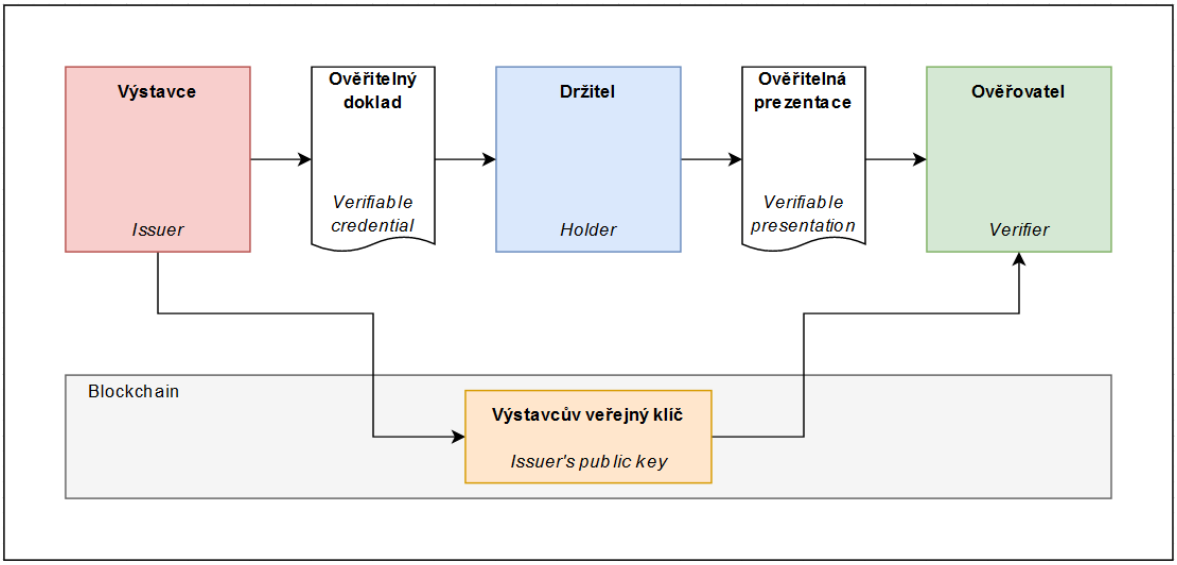
[9)] We will leverage the concept of zk-SNARKs in Zcash to carry out the voting phase of the election and the zkHawk smart contract protocol to tally the results of the election.

13.5.2 Self-sovereign Identity

Self-sovereign identity[10] je koncept postavený na decentralizaci (díky využití Blockchainu) a na maximalní kontrole uživatele nad svými údaji. Ačkoli se o tomto konceptu zpravidla mluví v kontextu elektronických dokladů, je praxi využíván i například pro sledování zásob v průběhu celého dodavatelského cyklu.

SSI se skládá ze 4 hlavních částí. Hlavní stranou komunikace je Držitel *(Holder)*. Ten ve své digitální peněžence (aplikace na mobilním telefonu či v počítači) ukládá elektronické doklady, které se řídí standardem W3C [44] a jsou nazvány Ověřitelné doklady *(Verifiable Credentials)*. Tyto doklady vystavuje Výstavce *(Issuer)*, který uloží informace nutné k ověření pravosti dokladů do distribuované databáze - Blockchainu. Držitel se poté může prokázat svými doklady Ověřovateli *(Verifier)* posláním takzvané Ověřitelné prezentace *(Verifiable presentation)*, což je nadstavbová vrstva nad Ověřitelnými doklady, která umožňuje kromě jiného využít právě Zero-knowledge protokoly k předání dat. Zjednodušené schéma fungování SSI je možno vidět na obrázku níže.

Obr. 13.1 Zjednodušené schéma fungování SSI



13.5.3 Srování terminologie

Ačkoli jsou si rámce Self-sovereign identity a Evropské Digitální identity velmi podobné, každý využívá svou vlastní terminologii. Tabulka 13.1 obsahuje srovnání základní terminologie těchto dvou rámců, a také terminologie využité v praktické implementaci v části 25, které z těchto rámců myšlenkově vychází.

[10] Self-sovereign je možné zhruba přeložit jako „autonomní“ či „samořídící“

Self-sovereign identita	Evropská digitální identita	Zero-knowledge Argument věku
Výstavce *Issuer*	Poskytovatel služeb vytvářejících důvěru *Trust service provider*	Důvěryhodná třetí strana *Trusted party*
Držitel *Holder*	Uživatel *User*	Dokazovatel *Prover*
Ověřovatel *Verifier*	Spoléhající se strana *Relying party*	Ověřovatel *Verifier*
Ověřitelný doklad *Verifiable credential*	Elektronické potvrzování atributů *Electronic attestation of attributes*	Toolbox *Toolbox*
Ověřitelná prezentace *Verifiable presentation*	Elektronické potvrzování atributů *Electronic attestation of attributes*	Důkaz *Proof*

Tab. 13.1 Srovnání terminologie SSI, EDI a ZKP

III. ZERO-KNOWLEDGE AUTENTIZACE

14 Modulární aritmetika

„Modulární aritmetika je aritmetikou na množině celých čísel $\mathbb{Z}$, v níž se čísla opakují po dosažení určité hodnoty n, již nazýváme modul[1)]. Na rozdíl od běžných celočíselných operací se zde po každé operaci provede ještě celočíselné dělení modulem n a výsledkem operace je zbytek po tomto dělení.“[45]

Množina, se kterou modulární aritmetika pracuje, je označována jako $\mathbb{Z}_n$ a obsahuje celá čísla v intervalu{0,1,...,n-1}.

Modulární aritmetika je také známa pod pojmem „hodinová aritmetika“, neboť čísla můžeme uspořád do kruhu, podobně jako na ciferníku hodin, kde například vteřiny nabývají hodnot v množině $\mathbb{Z}_{60}$ (modulo 60), a můžeme tak říci, že $59+1 \equiv 0 \pmod{60}$ - po 59. vteřině následuje na hodinách opět 0. vteřina, nikoli 60.

Modulární aritmetika má svá specifika, která ji odlišují od klasické aritmetiky (pracující v oboru reálných čísel $\mathbb{R}$). Jedním z nich je modulární umocňování, tedy výpočet $y \equiv g^x \pmod{p}$. To je možno poměrně jednoduše provádět i pro velmi velké hodnoty g, x, p (v kryptografii se běžně využívají čísla, která mají 2048 bitů, tj. cca 667 číslic v běžném zápise).

14.1 Diskrétní logaritmus

Druhým specifikem modulární aritmetiky využívaným v kryptografii je praktická nemožnost výpočtu takzvaného diskrétního logaritmu.

V $\mathbb{R}$ je jednoduché získat odpověď na rovnici $20 = 7^x$. Rovnici můžeme zlogaritmovat na $x = log_7(20) \doteq 1.54$. Obrázek 14.1 ukazuje graf této exponeciální funkce.

V $\mathbb{Z}_n$ je naopak získání výsledku rovnice $20 \equiv 7^x \pmod{31}$ velmi složité a pro velká čísla prakticky nemožné. Na této vlastnosti jsou postaveny některé asymetrické kryptografické systémy, jako například DSA, elGamal či klasická Diffie-Hellmanova výměna klíčů. Obrázek 14.2 ukazuje graf této modulární exponenciální funkce.

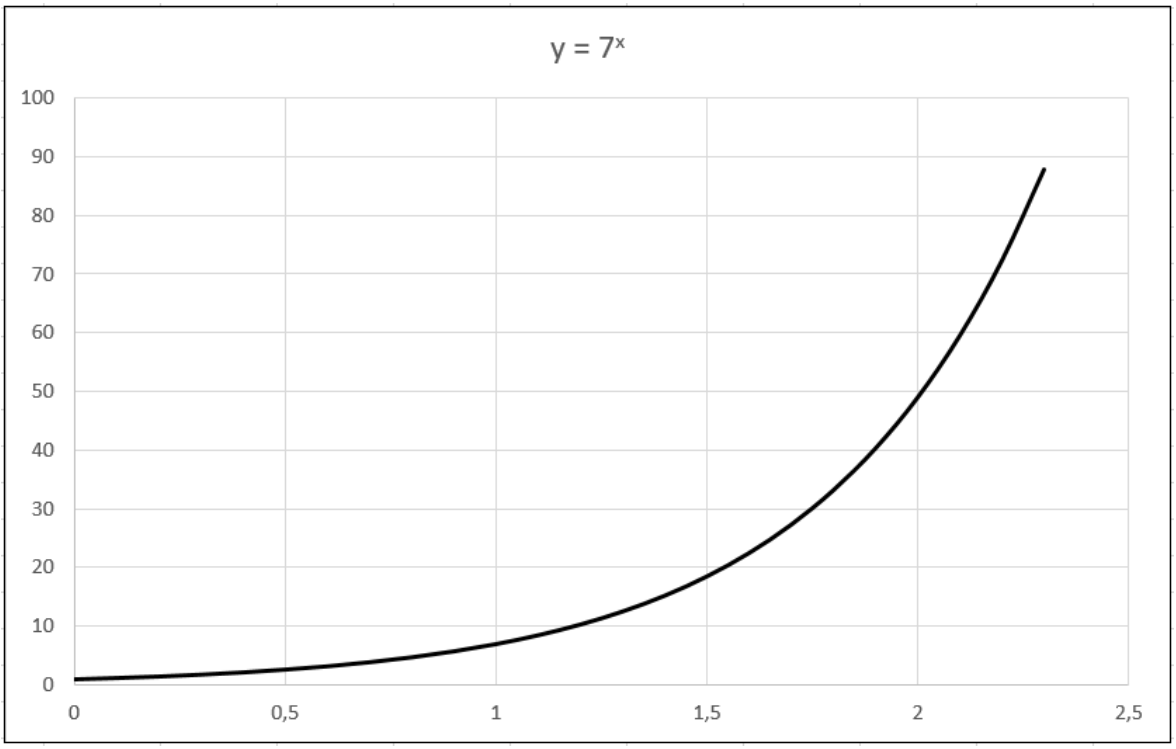
Rozdíl mezi klasickým a modulárním umocňování je možno vidět na grafech průběhu výše zmíněných funkcí - zatímco exponenciální funkce v oboru reálných čísel (obr. 14.1) je monotónně rostoucí, a proto je možno odhadnout, zda při růstu x bude y růst či klesat, v $\mathbb{Z}_n$ (obr. 14.2) toto není obecně možné.

14.2 Lambda funkce

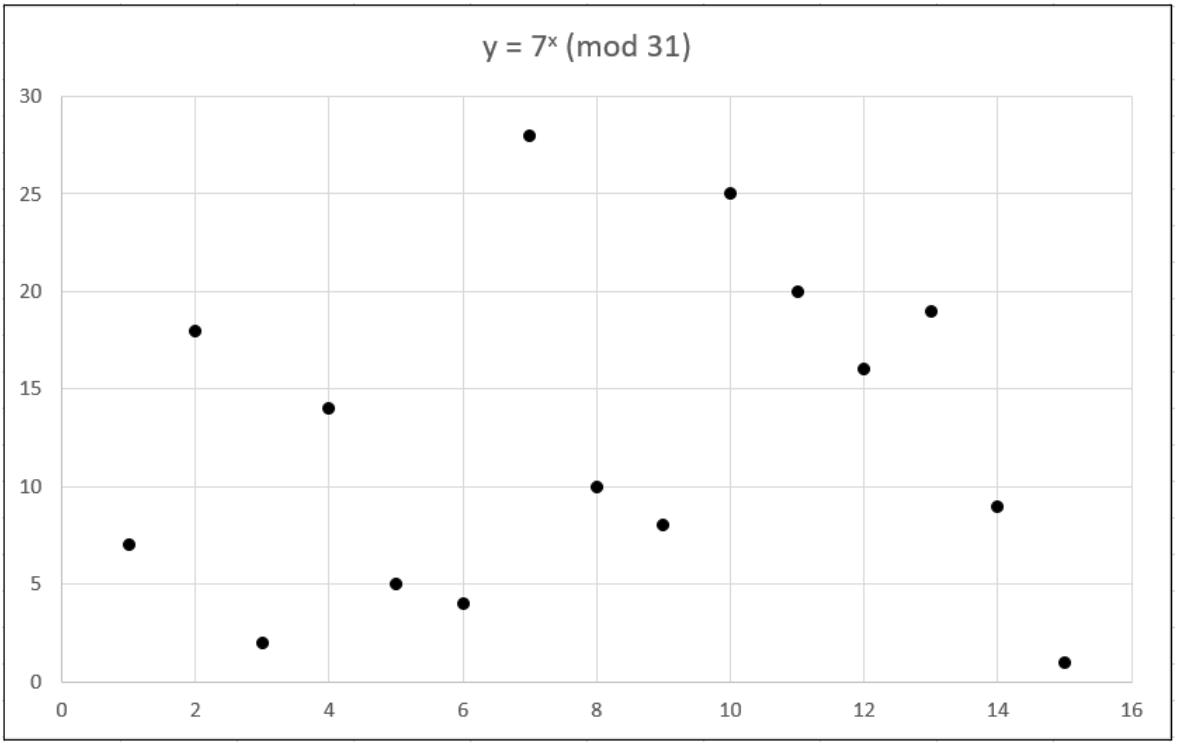
„Carmichaelova funkce, pojmenovaná po Robertu Danielovi Carmichaelovi, je funkce z oboru teorie čísel značená $\lambda(n)$, která pro přirozené číslo n vrátí nejmenší m takové,

[1)] [sic], v praxi se spíše, podobně jako v angličtině, používá tvar „modulo“

Obr. 14.1 Graf exponenciální funkce (klasický)



Obr. 14.2 Graf exponenciální funkce (modulární aritmetika)



že $a^m \equiv 1 \pmod{n}$.“[46]

Tato funkce může být mimo jiné použita k redukci exponentu pro modulo n, neboť platí:

$$\begin{aligned} x &\equiv m \pmod{\lambda(n)} \\ a^x \pmod{n} &\equiv a^m \pmod{n} \end{aligned} \tag{14.1}$$

15 Autentizace

Zatímco termín „identifikace“ znamená získání identity jiného subjektu, tedy zjištění, kdo daný subjekt je, „autentizace“ je proces ověření proklamované identity, jinými slovy ověření, že daný subjekt je skutečně ten, kdo tvrdí, že je.

15.1 Autentizační faktory

Autentizaci je možno provést mnoha různými způsoby, obecně můžeme tyto způsoby (označované zpravidla jako „faktory“) rozdělit do několika skupin:

- Faktor znalosti (Něco, co znám) - například heslo či PIN, obecně nějaká hodnota, kterou je nutno si pamatovat
- Faktor vlastnictví (Něco, co mám) - například elektronický čip, bankovní karta, telefonní číslo (SMS) či emailová adresa, obecně něco, k čemu má přístup pouze jedna osoba
- Faktor inheritance (Něco, co jsem) - biometrické údaje, jako například otisk prstu či sken obličeje
- Faktor lokality (Někde, kde jsem) - konkrétní místo, ať už fyzické či virtuální (IP adresa), ze kterého probíhá autentizace
- Faktor schopnosti (Něco, co umím) - například určitá dovednost, kterou je autentizovaný schopen demonstrovat

Standardně se uvádí jako autentizační faktury uvádí pouze první tři skupiny (znalost, vlastnictví a inheritance), zbylé dva (lokalita a schopnost) buď nejsou uvažovány, nebo jsou považovány spíše jako pomocné faktory sloužící spíše k odhalení podezřelých pokusů o autentizaci. Internetové standardy [47] i legislativa uvažují až na výjimky při řešení problematiky autentizace vždy pouze první tři zmíněné faktory.

15.1.1 Vícefaktorová autentizace

Autentizace může probíhat buď pomocí jednoho faktoru (například přihlášení heslem či odemknutí obrazovky mobilního telefonu pomocí otisku prstu), nebo za použití více faktorů zároveň (vložení bankovní karty do bankomatu a zadání PINu či zadání hesla do internetového bankovnictví a poté přepsání kódu z SMS zprávy). Toto je označováno jako dvoufaktorová (2FA), případně vícefaktorová (MFA) autentizace.

Aby byla autentizace považována za vícefaktorovou, je nutné, aby byly použity různé faktory (například znalost + vlastnictví nebo znalost + inheritance), v případě použití více metod jednoho faktoru (například PIN a heslo) se nejedná o vícefaktorovou autentizaci.

15.2 Autentizace heslem

Jednou z klasických metod přihlášení je využití hesla. Kromě samotné síly hesla je u této metody z hlediska bezpečnosti kladen důraz zejména na riziko odposlechnutí hesla při komunikaci[1)] a na riziko úniku hesel uložených v databázi na straně služby, která poskytuje autentizaci.

V průběhu let bylo vyvinuto mnoho způsobů, jak bezpečně pracovat s přihlašovacími údaji při jejich přenosu i uložení, které nahrazují absolutně nevhodný způsob posílání hesel i jejich ukládání v běžně čitelné podobě (otevřeném textu), kdy se útočník, pokud se mu podaří odposlechnout komunikaci či získat přístup do databáze, okamžitě dozví všechna hesla.

15.2.1 Hashovaní hesel

Standardním způsobem zabezpeční hesel uložených v databázi je využití klasických kryptografických hashovacích funkcí (jako například SHA256) či v lepším případě využití hashovacích funkcí vytvořených speciálně pro tento účel (například Argon2id či bCrypt). Důvodem jejich využití je zvýšení bezpečnosti, neboť i v případě úspěšného útoku se útočníkovi podaří získat pouze hash hesla. Získání hesla z jeho hashe se provádí útokem hrubou silou, případně za použití databází předvypočítaných hodnot (takzvaných rainbow tables), a za normálních okolností je možné pouze u slabých hesel.

Výstup hashe je deterministický, mají-li různí uživatelé stejné heslo, bude toto heslo v databázi zahashováno stejně.

[1)] Skupina takzvaných man-in-the-middle (muž uprostřed) útoků

15.2.2 Kryptografická sůl

Kryptografická sůl je náhodný řetězec unikátní pro každého uživatele. Tato sůl s při hashování přidává k heslu, čímž je zajištěna unikátnost i v případě, že více klientů použije stejné heslo. Hodnota soli je uložena v databázi spolu s heslem (respektive s jeho „osoleným hashem“), a poskytuje prvek ochrany při uložení v databázi, případně také při komunikaci (dle konkrétního způsobu implementace).

Přidání soli také podstatně zkomplikuje snahu získat z hashované hodnoty zpět původní heslo, neboť není možno využít předem vypočítané databáze hashů, a je nutné útočit na každé heslo zvlášť.

15.2.3 Kryptografický pepř

Spolu se solí je možno použít i tzv. kryptografický pepř[2)]. Jde opět o náhodný řetězec, v tomto případě je ale pro všechny uživatele stejný. Dalším rozdílem je, že tento pepř není uložený v databázi, ale v bezpečném prostředí (například HSM[3)]), a chrání pouze záznamy v databázi, kdy i v případě úniku uložených přihlašovacích údajů velmi znesnadňuje (či znemožňuje) snahy o odhalení hesel v čitelném tvaru.

Pepř je aplikován na „osolený hash“ hesla pomocí hashovací funkce, případně je možno jej použít jako klíč ke klasické symetrické šifře (v závislosti na tom, zda je či není nutné získat z „opepřeného hashe“ zpět hodnotu „osoleného hashe“).

15.2.4 Nonce

Použitím nonce[4)] při autentizaci heslem je možno vnést prvek náhody do autentizace, tedy že při jednotlivých autentizačních pokusech jednoho klienta jsou i bez změny hesla posílány pokaždé jiná data.

Tato vlastnost slouží jako ochrana proti „replay“ útokům, kdy se útočník pokouší serveru předložit data získaná v minulosti (při některém z minulých platných autentizačních pokusů).

Na bezpečnost uložení hesel v databázi nemá použití nonce vliv.

15.3 HMAC

HMAC, neboli Autentizační kód zprávy založený na hashi[5)] je mechanismus, který umožňuje zaručit a ověřit autenticitu zprávy pomocí libovolné kryptografické hashovací

[2)] Někdy (např. [47]) bývá označeno termínem tajná sůl („secret salt“).

[3)] Hardware Security Module - hardwarový bezpečnostní modul je specializované zařízení, jehož cílem je provádět kryptografické operace (například šifrování či dešifrování) a sloužit jako zabezpečené úložiště kryptografického materiálu (klíčů) k tomu použitých

[4)] number only used once - číslo použito pouze jednou

[5)] Hash-Based Message Authentication Code

funkce v kombinaci se symetrickým klíčem. Autory tohoho schématu jsou Mihir Bellare, Ran Canetti a Hugo Krawczyk, kteří jej představili v roce 1996 na kryptografické konferenci CRYPTO '96 ve své práci Použití klíče u hashovacích funkcí pro zaručení autenticity zprávy[6)][48].

Standard pro tento protokol byl vydán organizací IETF[7)] pod číslem RFC2104 [49]. Ačkoli se standard věnuje specificky hashovacím funkcím MD5 a SHA1, popisuje obecnou konstrukci HMAC, kterou je možno využít i s novějšími hashovacími algoritmy.

Tato obecná konsturkce vypadá následovně:

$$HMAC = H(K \oplus opad||H(K \oplus ipad||text)) \tag{15.1}$$

Kde:

- H - použitá hashovací funkce
- || - symbol pro spojení textových řetězců ($te||xt = text$)
- $\oplus$ - XOR, neboli exkluzivní disjunkce
- K - symetrický klíč, který je použit k výpočtu HMAC a také k jeho ověření
- text - vstup, pro který se HMAC počítá
- ipad - vnitřní padding (byte $0x36$ opakovaný v délce bloku dané hashovací funkce)
- opad - vnější padding (byte $0x5C$ opakovaný v délce bloku dané hashovací funkce)

15.4 OTP

Zkratka OTP pochází z anglického One-Time Password (případně One-Time PIN), což v překladu znamená jednorázové heslo (jednorázový PIN). Jde o zpravidla 6-8 místný numerický kód, který se používá při vícefaktorové autentizaci. Toto OTP je klientovi předáno jiným komunikačním kanálem (nejčastěji pomocí SMS či emailu, může ale jít také například o softwarovou aplikaci či přímo pro toto určené jednoúčelové hardwarové zařízení), než který je použit k autentizaci, a slouží k ověření faktoru vlastnictví.

15.4.1 HOTP

V roce 2005 byl vydán ve spolupráci s organizací OATH standard IETF RFC4226 s názvem HOTP: Algoritmus pro jednorázové heslo založený na HMAC[8)][50]. Standard

[6)] Keying Hash Functions for Message Authentication
[7)] Internet Engineering Task Force
[8)] HOTP: An HMAC-Based One-Time Password Algorithm

popisuje použití HOTP za použití hashovacího algoritmu SHA1, v praxi je nicméně možno využít libovolný kryptografický hashovací algoritmus.

Obecná konstrukce HOTP je následující:

$$HOTP = D_x(HMAC(K, C)) \tag{15.2}$$

Kde:

- K - symetrický klíč, který je použit ve výpočtu HMAC a také k jeho ověření
- C - hodnota čítače, která postupně roste, a musí být synchronizována mezi klientem a serverem
- D - funkce, která převádí spočítaný HMAC na číslo v desítkové sestavě s požadovaným počtem číslic (popsáno v kapitole 5.3 standardu [50])
- x - požadovaný počet číslic HOTP

15.4.2 TOTP

Roku 2011 byl vydán standard IETF RFC6238 s názvem TOTP: Algoritmus pro jednorázové heslo založený na čase[9)][51]. Jde o modifikovaný HOTP, kde je místo čítače využit datum a čas přihlášení, primárně z důvodu jednodušší případné resynchronizace mezi klientem a serverem.

Algoritmus zpravidla vychází z unixového času[10)] a je u něj možno nastavit dobu platnosti TOTP kódu (doporučenou hodnotou je 30 sekund).

$$TOTP = D_x(HMAC(K, T)) \tag{15.3}$$

Kde t je časová hodnota (počet celých uplynulých časových období doby platnosti TOTP kódu od daného počátečního okamžiku).

Například 1.1.2022 o půlnoci nabýval unixový čas hodnoty 1640991600. V takovém případě by pro dobu platnosti kódu 30 sekund byla spočítána hodnota t takto:

$$\begin{aligned} t &= 1640991600/30 \\ t &= 54699720 \end{aligned} \tag{15.4}$$

[9)] TOTP: Time-Based One-Time Password Algorithm

[10)] počet sekund, které uběhly od 1.ledna1970

16 Fiat-Shamirova identifikace

V roce 1986 byla na konferenci CRYPTO 1986 zveřejněna dvěma izraelskými kryptografy, Amosem Fiatem a Adim Shamirem, práce s názvem Jak se prokázat: Praktická řešení identifikace a problémů s podpisem[1)][26].

Autoři v abstraktu tvrdí: „V této práci popisujeme jednoduchá identifikační a podpisová schémata, která umožní jakémukoli uživateli prokázat svou identitu a autenticitu svých zpráv komukoli jinému bez použití sdílených či veřejných klíčů.“[2)][26]

Práce popisuje 3 schémata - autentizační, identifikační a podpisové. Tato schémata jsou „založena na náročnosti spočítání odmocniny v modulární aritmetice při neznalosti faktorizace n“[3)][26]. V důsledku jsou tedy schémata založena na problematice faktorizace (tj. na praktické nemožnosti faktorizovat, neboli rozložit na prvočíselné dělitele, velké poloprvočíslo), podobně jako například algoritmus RSA (kterého je Adi Shamir jedním ze spoluautorů).

Nejen na tuto práci navazují v roce 1997 kryptografové Jan Camenisch a Markus Stadler ve své práci Systémy důkazu pro obecná tvrzení o diskrétních logaritmech [4)][52], kteří podobný identifikační protokol staví na matematickém principu složitosti výpočtu diskrétního logaritmu (viz kapitola 14.1).

16.1 Popis protokolu

Cílem tohoto Interaktivního Zero-knowledge Důkazu Znalosti hesla je umožnit Dokazovateli prokázat Ověřovateli svou identitu tím, že prokáže znalost předem domluveného tajemství, jinak řečeno, autentizovat se za použití hesla.

Bezpečnost protokolu je postaven na praktické nemožnosti spočítat diskrétní logaritmus. V rámci této kapitoly jsou termíny Dokazovatel a Ověřovatel nahrazeny výrazy Klient a Server. V komunikaci tedy Klient prokazuje Serveru znalost hesla.

Protokol se skládá ze dvou částí, první část se zabývá registrací. Klient zde předává Serveru své heslo, jehož znalostí mu bude v další fázi prokazovat svou identitu.

V druhé, autentizační části, pak Klient za pomoci Sigma protokolu prokazuje Serveru, že stále zná své heslo.

[1)] How To Prove Yourself: Practical Solutions to Identification and Signature Problems

[2)] In this paper we describe simple identification and signature schemes which enable any user to prove his identity and the authenticity of his messages to any other user without shared or public keys.

[3)] It is based on the difficulty of extracting modular square roots when the factorization of n is unknown

[4)] Proof Systems for General Statements about Discrete Logarithms

16.1.1 Generování p, g

V rámci protokolu jsou vygenerovány a použity dvě velká prvočísla p, g tak, že $p > g$. Z pohledu protokolu nezáleží na tom, která strana tyto hodnoty vygeneruje (zda obě vygeneruje Klient, obě Server, nebo jednu Klient a jednu Server).

Z pohledu filozofie Zero-knowledge dává smysl, aby obě hodnoty generoval Klient. Klient má své tajné heslo, které si chce chránit, a chce mít jistotu, že se jej Server nepokusí úmyslně chybným vygenerování těchto hodnot získat. Registrační fáze je v tomto případě také kratší o jednu zprávu. Z hlediska bezpečnosti protokolu jde v tomto případě o Výpočetní Zero-knowledge protokol (viz kapitola 18.3).

Z praktického hlediska dává smysl, aby tyto hodnoty generoval Server, neboť má zpravidla přístup k výkonějšímu a lepšímu generátoru velkých prvočísel, což je poměrně výpočetně náročná operace, se kterou mohou mít některá zařízení problémy. V takovém případě jde nicméně pouze o Zero-knowledge s Čestným Ověřovatelem (viz kapitola 18.3).

Kompromisní řešení, tedy že Server vygeneruje větší z prvočísel p a Klient poté vygeneruje menší prvočíslo g je z implementace nejsložitější, jde, podobně jako v prvním případě, o Výpočetní Zero-knowledge protokol, nicméně je o jednu zprávu delší, stejně jako v druhém případě.

V rámci základní verze protokolu demonstrované dále jsou obě hodnoty generovány Klientem, v rozšířené verzi pak obě generuje Server.

17 Základní verze protokolu

17.1 Registrační fáze

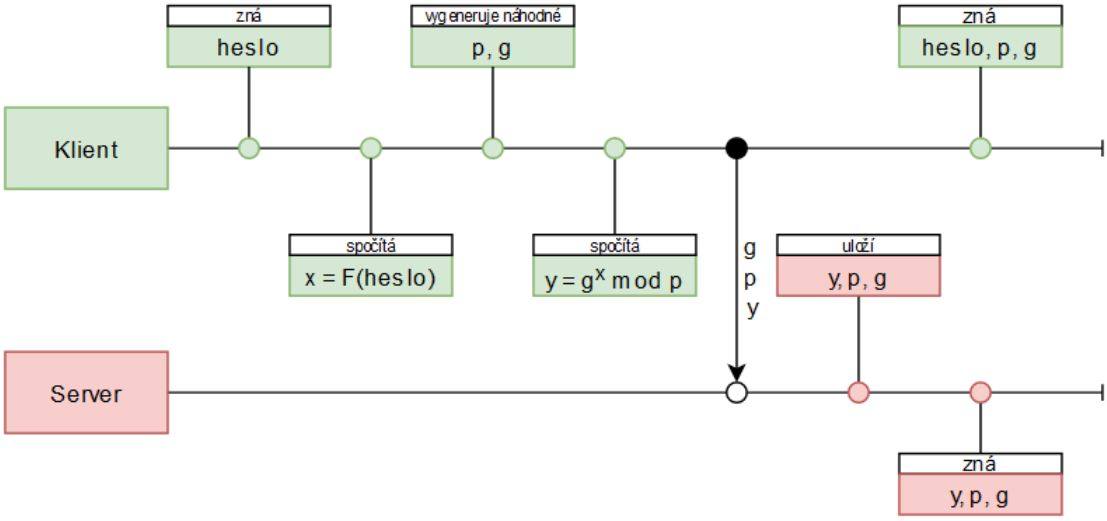
Cílem registrační fáze pro Klienta je zvolit si své heslo, z něj spočítat hodnotu y, kterou poté předá Serveru, a kterou bude moci v budoucnu použít k sestavení důkazu znalosti tohoto hesla.

Klient si nejprve vygeneruje dostatečně dlouhé prvočíslo p a vhodný (prvočíselný) generátor g. Poté použije své heslo jako vstup funkce F, čímž získá hodnotu x. Cílem funkce F je převést heslo, což je řetězec znaků, který může obsahovat velká a malá písmena, číslice i speciální znaky, na přirozené číslo. Tato funkce není ve Fiat-Shamirově identifikačním schématu specifikována. Pro jednoduchost nyní předpokládejme, že heslo je rovnou ve tvaru přirozeného čísla, funkce F tedy bude pracovat jako $x = F(x)$.

$$\begin{aligned} x &\equiv F(heslo) \pmod{\lambda(p)} \\ y &\equiv g^x \pmod{p} \end{aligned} \tag{17.1}$$

Hodnoty y, p, g jsou poté předány Serveru. Schéma registrace shrnuje obrázek 17.1.

Obr. 17.1 Schéma registrace (základní verze)



17.2 Autentizační fáze

V této fázi se Klient autentizuje Serveru, tedy prokazuje svou identitu tím, že demonstruje znalost svého hesla.

Autentizační fáze probíhá následovně:

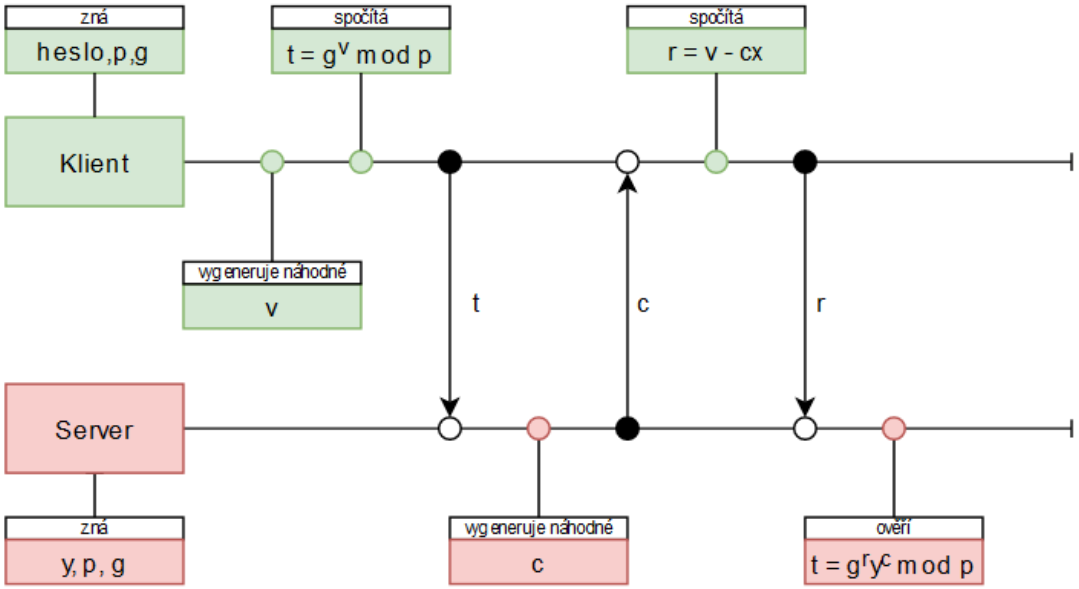
1. Klient náhodně zvolí vhodně velké přirozené číslo v, spočítá Závazek $t \equiv g^v \pmod{p}$ a tuto hodnotu předá Serveru
2. Server si nezávisle zvolí svou Výzvu c, tedy vhodně velké náhodné přirozené číslo, a předá ji klientovi
3. Klient spočítá Odpověď $r \equiv (v - cx) \pmod{\lambda(p)}$ a tuto hodnotu předá Serveru
4. Server ověří, že platí $t = g^r y^c \pmod{p}$. Pokud ano, prokázal Klient znalost hesla a tím svou identitu

Autentizační schéma shrnuje obrázek 17.2.

17.3 Důkaz platnosti protokolu

Následuje důkaz platnosti protokolu:

Obr. 17.2 Schéma autentizace (základní verze)



$$
\begin{aligned}
t &\equiv g^r y^c \pmod{p} \\
g^v \pmod{p} &\equiv g^r y^c \pmod{p} \\
g^v \pmod{p} &\equiv g^{v-cx} g^{xc} \pmod{p} \\
g^v \pmod{p} &\equiv g^{v-cx+xc} \pmod{p} \\
g^v \pmod{p} &\equiv g^v \pmod{p}
\end{aligned}
\tag{17.2}
$$

Obrázek 17.3 ukazuje vzájemný vztah použitých proměnných s ohledem na strany komunikace.

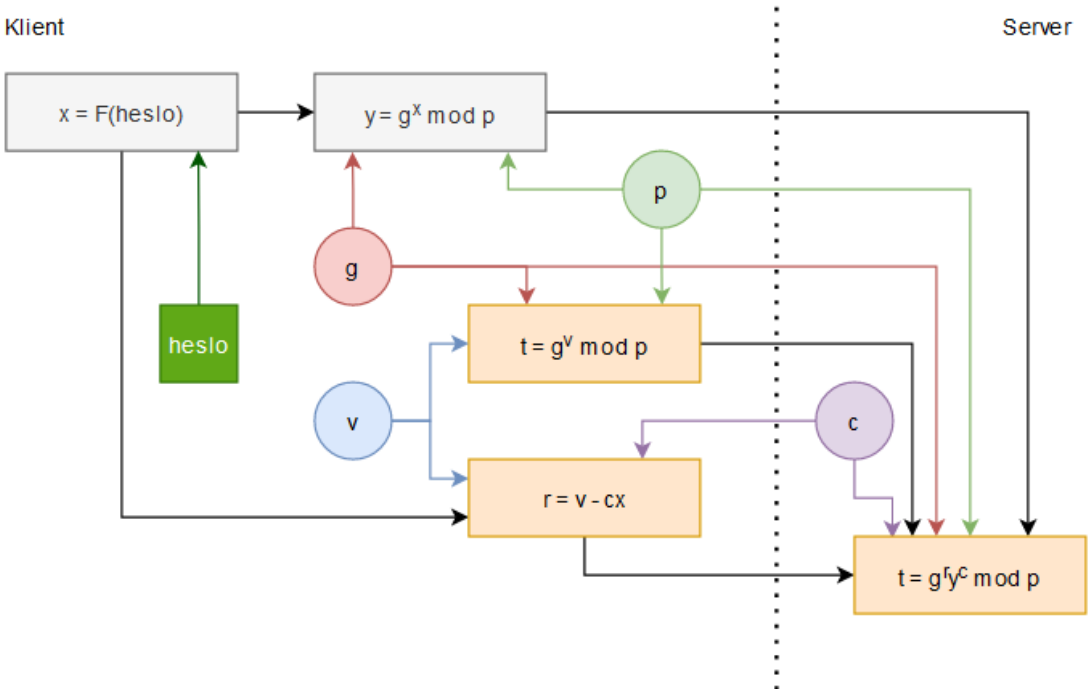
17.4 Příklad

Následuje vzorový příklad využití protokolu k důkazu identity. Klient si zvolil heslo 12345.

17.4.1 Registrační fáze

Klient si náhodně zvolí prvočíslo $p = 1009$ a generátor $g = 3$. Z toho hodnoty p vyplývá, že $\lambda(p) = 1008$. V registrační fázi tedy Klient spočítá:

Obr. 17.3 Vztah použitých proměnných (základní)



$$
\begin{aligned}
x &\equiv F(heslo) \pmod{\lambda(p)} \\
x &\equiv 12345 \pmod{1008} \\
x &\equiv 249 \pmod{1008}
\end{aligned}
\tag{17.3}
$$

$$
\begin{aligned}
y &\equiv g^x \pmod{p} \\
y &\equiv 3^{249} \pmod{1009} \\
y &\equiv 710 \pmod{1009}
\end{aligned}
$$

Klient poté předá serveru hodnoty $p = 1009$, $g = 3$, $y = 710$ a server si všechny tři hodnoty uloží.

17.4.2 Autentizační fáze

Klient si náhodně zvolí hodnotu $v = 613$, a spočítá Závazek t, který předá Serveru.

$$
\begin{aligned}
t &\equiv g^v \pmod{p} \\
t &\equiv 3^{613} \pmod{1009} \\
t &\equiv 836 \pmod{1009}
\end{aligned}
\tag{17.4}
$$

Server si zvolí svou náhodnou Výzvu $c = 443$ a předá ji Klientovi.

Klient spočítá Odpověď r a pošle tuto hodnotu zpět Serveru:

$$\begin{aligned} r &\equiv (v - cx) \pmod{\lambda(p)} \\ r &\equiv (613 - 443 \times 249) \pmod{1008} \\ r &\equiv -109694 \pmod{1008} \\ r &\equiv 178 \pmod{1008} \end{aligned} \tag{17.5}$$

Server ověří:

$$\begin{aligned} t \pmod{1009} &\equiv g^r y^c \pmod{p} \\ 836 \pmod{1009} &\equiv 3^{178} \times 710^{443} \pmod{1009} \\ 836 \pmod{1009} &\equiv 527 \times 82 \pmod{1009} \\ 836 \pmod{1009} &\equiv 836 \pmod{1009} \end{aligned} \tag{17.6}$$

Rovnost platí, Klient tedy prokázal svou identitu dokázáním znalosti hesla (bez jeho prozrazení).

18 Bezpečnost protokolu

Cílem této kapitoly je zhodnotit protokol z hlediska požadavků na Zero-knowledge protokoly - jednotlivé podkapitoly se věnují Úplnosti, Solidnosti a Zero-knowledge.

18.1 Úplnost

Protokol je Úplný, neboť čestný Klient dokáže vždy na spočítat Odpověď na Výzvu. Zná totiž jak heslo x, tak korektně zvolený Závazek t.

18.2 Solidnost

Protokol je Perfektně Solidní, neboť nečestný Klient nedokáže bez znalosti x spočítat správnou Odpověď r tak, aby odpovídala Závazku t, a aby jej Server mohl ověřit.

18.3 Zero-knowledge

Server zná pouze proměnné y, g, p z rovnice $y \equiv g^x \pmod{p}$, a k získání x by musel spočítat diskrétní logarimus, což Server dokáže pouze za předpokladu, že má k dispozici libovolně dlouhý výpočetní čas. Proto jde o Výpočetní Zero-knowledge protokol.

Pokud by hodnoty p, q generoval Server, mohl by si je teoreticky úmyslně zvolit chybně s cílem získat hodnotu x - použité heslo (které nemusí být nutně chráněno jednosměrnou funkcí, a proto může být Server schopen z hodnoty x získat heslo v otevřené podobě). Vzhledem k tomuto faktu je v takovém případě pro bezpečnost protokolu nutné, aby byl Server důvěryhodný - jde tedy o protokol Zero-knowledge s Čestným Ověřovatelem.

19 Rozšířená verze protokolu

V praxi je nutno vyřešit způsob transformace hesla na celočíselnou hodnotu x a také komunikaci rozšířit o další "pomocné"hodnoty, jako je například uživatelské jméno. Do protokolu je také z důvodu zvýšené ochrany uložených přístupových údajů zavedena obdoba kryptografického pepře. Posledním rozšířením protokolu, které bude předvedeno, je ukázka možného využití TOTP v rámci dvoufaktorové autentizace.

19.1 Transformace hesla

Ačkoli by pro tuto transformaci bylo nejvhodnějším řešením využití hashovacího algoritmu, bude zvolen jiný způsob, protože jedním z vedlejších cílů této kapitoly je demonstrovat možnost praktického využití metody přihlášení pomocí hesla bez využití hashovacích algoritmů[1)].

Jednou z možností je řetězec interpretovat jako číslo v 256kové soustavě, kde hodnoty jednotlivých znaků jsou vyjádřeny jejich ASCII hodnotou. Hodnotu je pak možno vyjádřit následujícím vzorcem:

$$x = \sum_{i=0}^{len(h)-1} 256^{len(h)-i-1} \times ord(h[i]) \tag{19.1}$$

Kde:

- h - heslo
- h[i] - i-tý znak hesla
- len(h) - počet znaků hesla
- ord() - funkce, která vrátí ASCII hodnotu daného znaku

Hodnotu řetězce Heslo je tedy možno vyjádřit takto:

$$\begin{aligned} x &= 256^4 \times ord('H') + 256^3 \times ord('e') + 256^2 \times ord('s') + 256^1 \times ord('l') + 256^0 \times ord('o') \\ x &= 256^4 \times 72 + 256^3 \times 101 + 256^2 \times 115 + 256^1 \times 108 + 256^0 \times 111 \\ x &= 310939708527 \end{aligned} \tag{19.2}$$

19.2 Použití kryptografického pepře

Hodnota kryptografického pepře s je náhodná, stejná pro všechny uživatele a tajná, proto musí být uložena mimo databázi s přihlašovacími údaji (například v bezpečném

[1)] Jedinou výjimkou je způsob generování OTP, který je založen na výše popsaných běžně využívaných standardech a hashovací funkci.

HSM[2)] modulu, který může veškeré výpočty s touto hodnotou provádět a vracet pouze jejich výsledky).

Při registrační fázi Server za normálních okolností ukládá pro Klienta hodnoty y, p, q. Při použití kryptografického pepře místo y uloží $z = y^s \pmod p$.

Při autentizační fázi poté Server ověřuje upravenou rovnici:

$$t^s \equiv g^{r^s} z^c \pmod p \tag{19.3}$$

Následuje důkaz platnosti protokolu při použití kryptografického pepře:

$$\begin{aligned} t^s &\equiv g^{r^s} z^c \pmod p \\ g^{v^s} \pmod p &\equiv g^{rs} y^{s^c} \pmod p \\ g^{vs} \pmod p &\equiv g^{s(v-cx)} g^{x^{sc}} \pmod p \\ g^{vs} \pmod p &\equiv g^{sv-scx} g^{xsc} \pmod p \\ g^{vs} \pmod p &\equiv g^{sv-scx+xsc} \pmod p \\ g^{vs} \pmod p &\equiv g^{sv} \pmod p \end{aligned} \tag{19.4}$$

V případě úniku dat se pak musí útočník pokoušet hrubou silou spočítat x z rovnice $z = g^{xs} \pmod p$ kde nezná x ani s, což je prakticky nemožné. Více se této problematice věnuje kapitola 21.1.

19.3 Použití TOTP jako druhého faktoru

Nahrazením náhodné serverem generované hodnoty Výzvy c kódem OTP je možno zkrátit autentizaci o dvě zprávy. V tomto případě Klient při autentizaci vygeneruje stejným způsobem náhodné v a spočítá z něj Závazek t. Poté, namísto odesílání Závazku serveru, sám získá z OTP kódu Výzvu c, spočítá Odpověď r a pošle hodnoty r, t dohromady. Server postupuje stejně, jako při klasickém způsobu, pouze namísto generování náhodné Výzvy c spočítá také stejný OTP kód.

19.3.1 Inicializace OTP

Nutným krokem k použití OTP je, bez ohledu na způsob, kterým bude OTP generováno, jeho inicializace ze strany Serveru. K tomu je nutné, aby Server vygeneroval sdílené tajemství o, k jehož získání je nutné znát Serverový klíč m_s, který je, stejně jako kryptografický pepř, uložené na bezpečném místě (např. v HSM modulu), a dále náhodný vygenerovaný řetězec m_c, který je unikátní pro každého Klienta, a který je spolu s přihlašovacími údaji uložen (ve formě otevřeného textu) v databázi. Sdílené tajemství je pak spočítáno dle vzorce:

$$o = HMAC(m_s, m_k) \tag{19.5}$$

[2)] Hardware Security Module - kryptografické zařízení, které slouží k bezpečnému uložení hesel a provádění kryptografických operací

Toto sdílené tajemství je poté nahráno do (hardwarového či softwarového) TOTP generátoru, který při použití vrátí OTP kód spočítaný pomocí vzorce $OTP = TOTP(o, T)$, kde T je čas vydání OTP kódu.

TOTP generátor je poté nutné vhodným komunikačním kanálem (například klasickou poštou, případně fyzicky) předat klientovi.

Server musí při ověření získat stejný OTP kód. Toho dosáhne pomocí vzorce:

$$OTP = TOTP(HMAC(m_s, m_k), T) \tag{19.6}$$

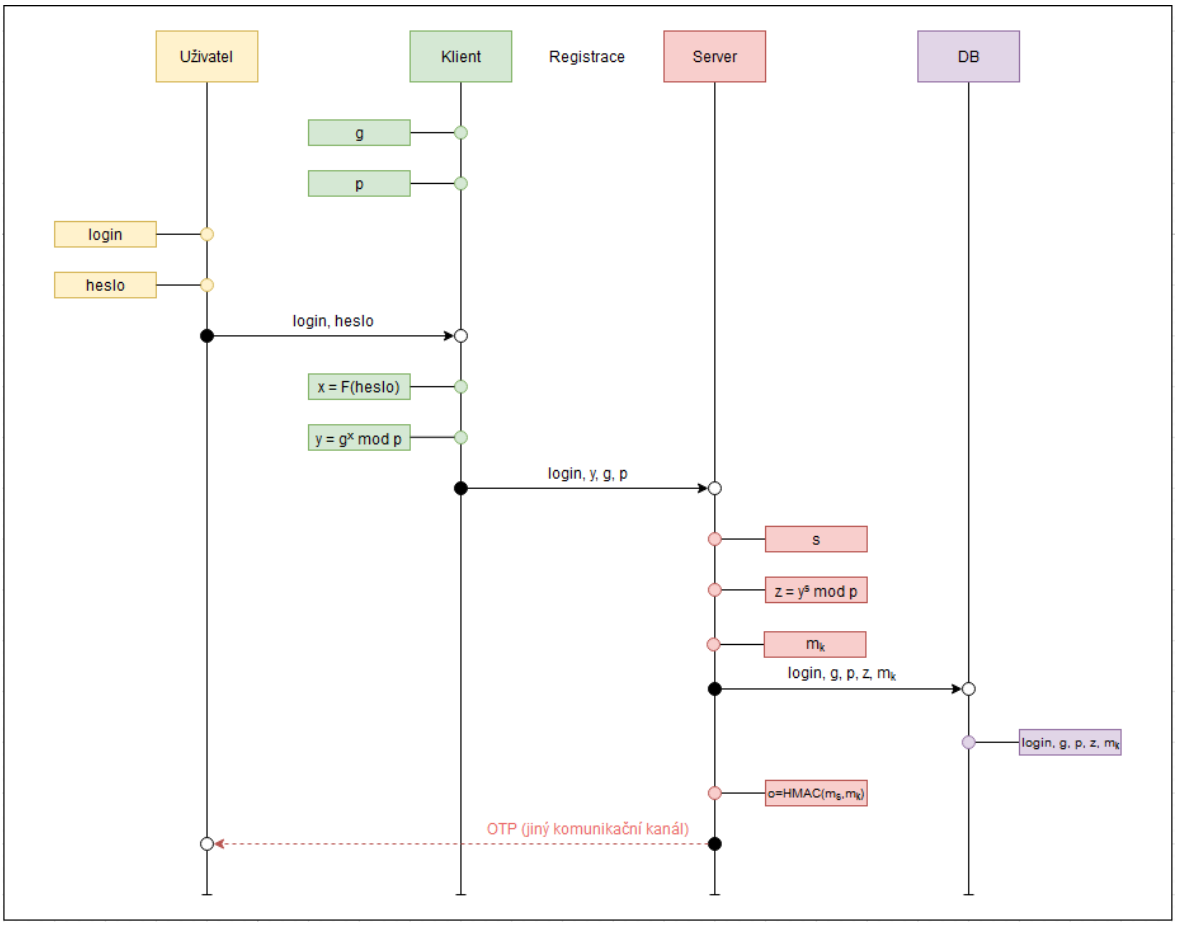
19.4 Registrační fáze

Před začátkem registrační fáze je předpokladem, že Server zná svůj tajný kryptografický pepř s a také klíč pro výpočet sdíleného tajemství pro OTP označený m_s. Registrační fáze probíhá takto:

- Klient vygeneruje prvočísla p, g
- Klient si zvolí své uživatelské jméno (dále login)
- Klient si zvolí své heslo
- Klient spočítá $x = F(heslo)$
- Klient spočítá $y \equiv g^x \pmod{p}$
- Klient odešle login, y Serveru
- Server spočítá $z \equiv y^s \pmod{p}$
- Při použití jednoho faktoru:
  - Server si uloží login, z, p, g
- Při použití obou faktorů:
  - Server vygeneruje náhodná data pro výpočet OTP m_k
  - Server si uloží login, z, p, g, m_k
  - Server spočítá sdílené tajemství pro OTP $o = HMAC(m_s, m_k)$
  - Server vytvoří OTP generátor se sdíleným tajemstvím o a předá jej vhodným komunikačním kanálem Klientovi

Schéma registrační fáze shrnuje obrázek 19.1.

Obr. 19.1 Schéma registrace (rozšířené)



19.5 Autentizační fáze

Postup při autentizaci se liší v závislosti na tom, zda je použit pouze 1 faktor (heslo) nebo 2 faktory (heslo a OTP).

19.5.1 Použití 1 faktoru

Rozšířená autentizační fáze při využití pouze faktoru znalosti probíhá ve třech komunikačních kolech (5 zpráv).

1.kolo:

- Uživatel zadá login a pošle jej Serveru
- Server na základě loginu získá z databáze hodnoty z, p, g
- Server odešle hodnoty p, g Klientovi

2.kolo:

- Klient vygeneruje náhodné v
- Klient spočítá Závazek $t \equiv g^v \pmod{p}$
- Klient odešle Závazek Serveru
- Server vygeneruje náhodnou Výzvu c a pošle ji Klientovi

3.kolo:

- Klient zadá heslo
- Klient spočítá $x = F(heslo)$
- Klient spočítá Odpověď $r = v - cx$ a pošle tuto hodnotu Serveru
- Server ověří platnost rovnice $t^s \equiv g^{rs} z^c \pmod{p}$

Pokud daná rovnice platí, přesvědčil Klient Server o znalosti hesla.

Schéma autentizační fáze shrnuje obrázek 19.2.

19.5.2 Použití 2 faktorů

Při využití OTP namísto náhodné Výzvy c je možno zkrátit komunikaci na 2 kola (3 zprávy).

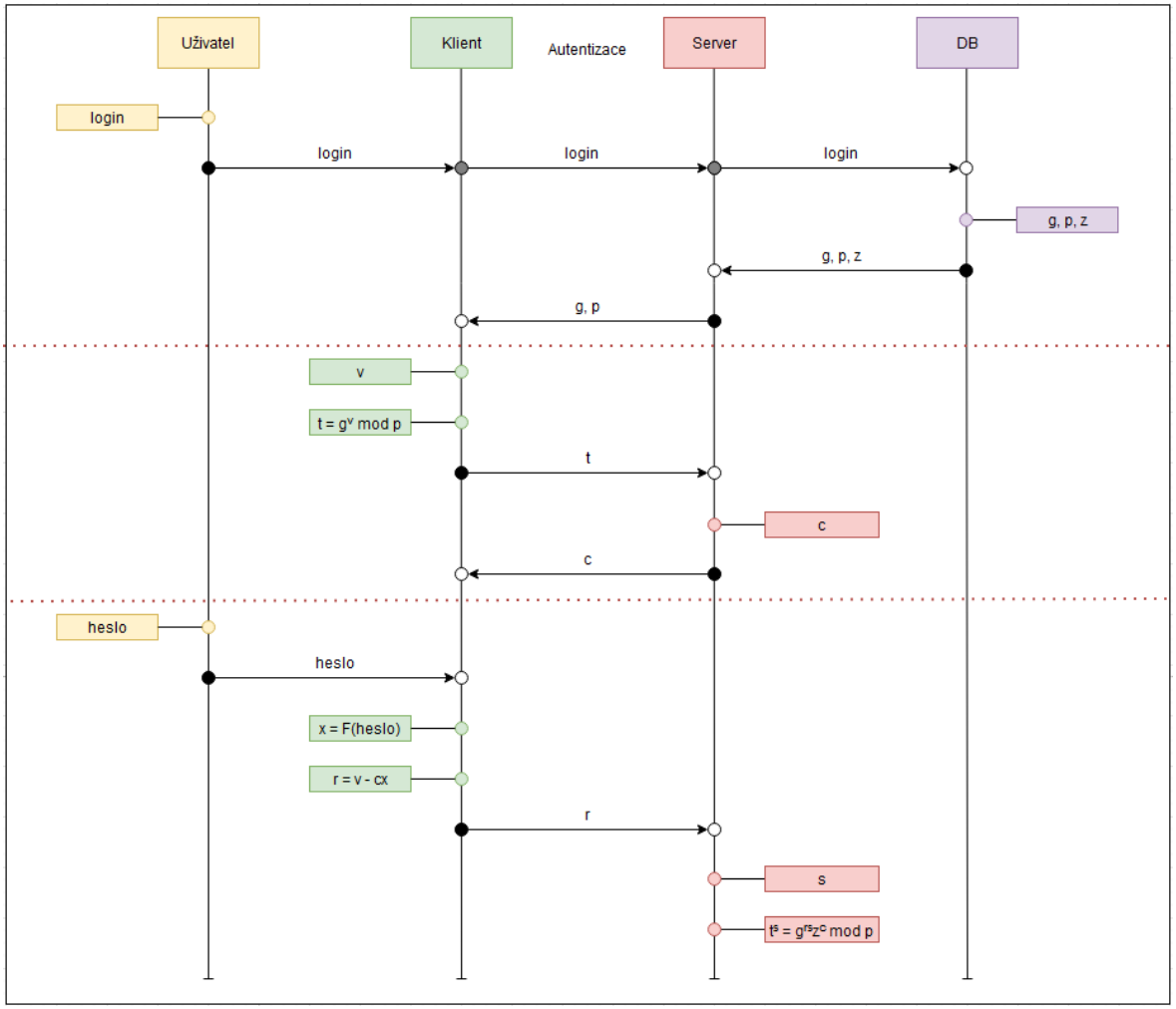
1.kolo:

- Uživatel zadá login a pošle jej Serveru
- Server na základě loginu získá z databáze hodnoty z, p, g, m_k
- Server odešle hodnoty p, g Klientovi

2.kolo:

- Klient vygeneruje náhodné v
- Klient spočítá Závazek $t \equiv g^v \pmod{p}$
- Klient zadá heslo
- Klient spočítá $x = F(heslo)$
- Klient zadá OTP z OTP generátoru jako Výzvu c
- Klient spočítá Odpověď $r = v - c \times x$

Obr. 19.2 Schéma autentizace (1 faktor, rozšířené)



- Klient pošle Serveru hodnoty t, r
- Server spočítá sdílené tajemství pro OTP $o = HMAC(m_s, m_k)$
- Server spočítá $c = TOTP(o, cas)$
- Server ověří platnost rovnice $t^s \equiv g^{rs} z^c \pmod{p}$

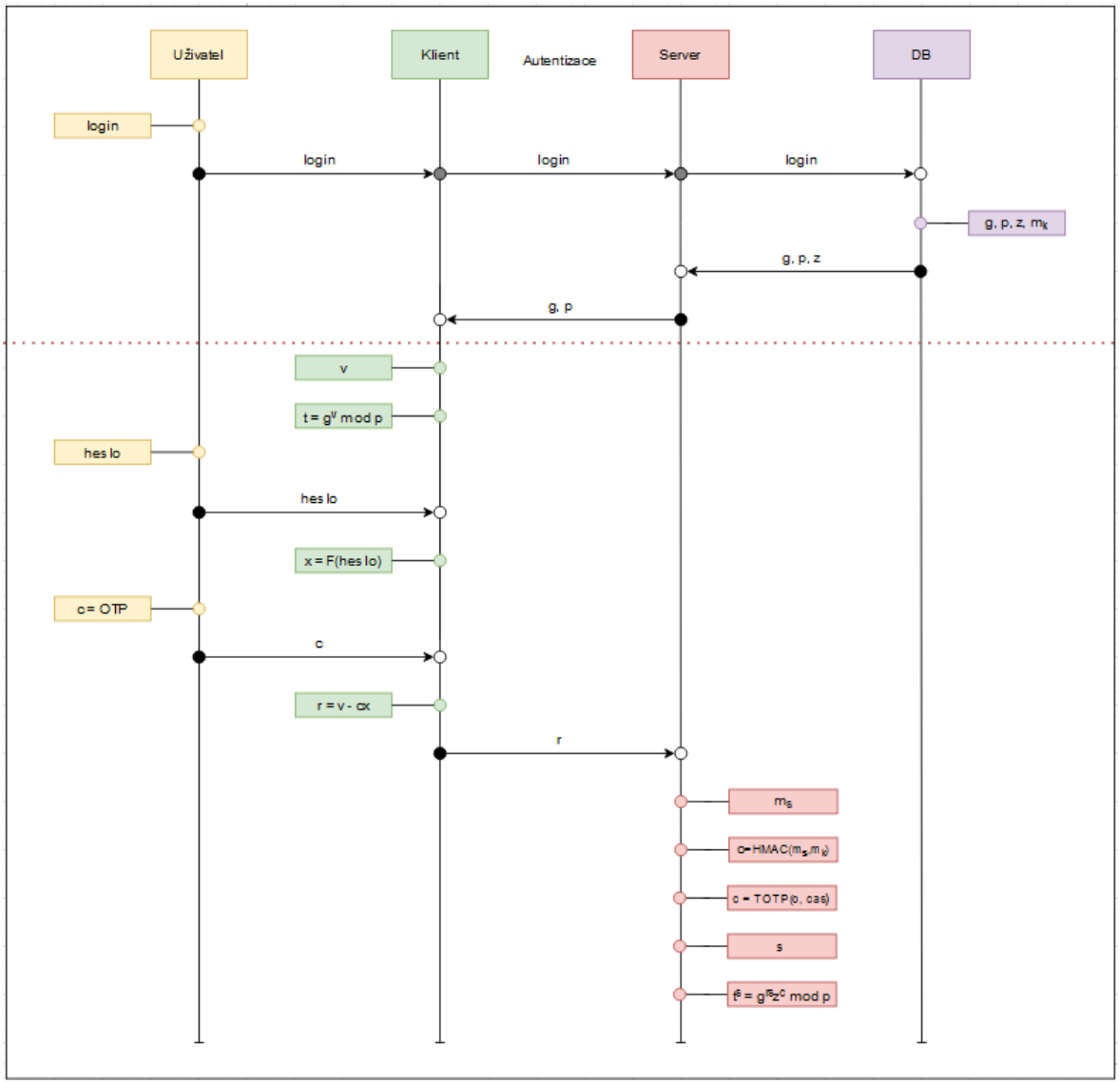
Pokud daná rovnice platí, přesvědčil Klient Server o znalosti hesla.

Schéma rozšířené autentizační fáze shrnuje obrázek 19.2.

20 Vzorová implementace

Cílem vzorové implementace demonstrovat možné bezpečné autentizační schéma, které nevyužívá hashovacích funkcí, ale je postaveno na problematice diskrétního logaritmu. Toto schéma bylo z důvodu rozšíření bezpečnosti rozšířeno o využití kryptografického

Obr. 19.3 Schéma autentizace (2 faktory, rozšířené)



pepře, pomocí kterého jsou chráněna hesla v databázi. Součástí je také ukázka možného zakomponování faktoru vlastnictví ve formě (simulovaného) hardwarového OTP generátoru, čímž vznikne dvoufaktorová autentizace.

20.1 Rozsah práce

Implementace je vytvořena pouze v nezbytně nutném rozsahu, neboť jejím cílem je pouze demonstrovat využití Interaktivního Zero-knowledge protokolu. Proto nejsou implementovány v praxi nezbytné součásti reálné autentizační služby, jako je například databáze, šifrování komunikace či přihlašovací obrazovka. Snahou také bylo minimalizovat množství externích knihoven, které program využívá. Výhodou tohoto přístupu je fakt, že ke spuštění a otestování této implementace není nutné instalovat jiné služby

(kromě Pythonu), ani mnoho externích balíčků.

Ze stejného důvodu je využití hardwarového OTP generátoru také pouze simulované.

20.2 Datový model

Všechny zprávy jsou posílány ve formátu JSON. Jednotlivé zprávy obsahují následující atributy:

- messageid - identifikátor zprávy ve formátu x_y_z, kde x značí typ komunikace ("reg"- registrace, "auth"- klasická autentizace, "auth2fa"- dvoufaktorová autentizace), y značí číslo zprávy v rámci komunikace a z stranu, která zprávu posílá ("c"- klient, "s"- server)
- username - login Klienta
- encryptedpassword - hodnota y
- generator - hodnota g
- prime - hodnota p
- authtype - druh autentizace, možné hodnoty jsou "classic"pro jednofaktorovou (výchozí volba) a "2FA"pro dvoufaktorovou autentizaci
- success - zpráva, zda fáze proběhla úspěšně
- commitment - hodnota t
- challenge - hodnota c
- response - hodnota r

Některé zprávy obsahují strukturu "demonstrationsecrets". Tato struktura obsahuje tajná data Klienta, která se v praxi Server nesmí nikdy dozvědět! Atributy v této struktuře slouží čistě k demonstračním účelům a k ověření správnosti výpočtů. V této struktuře se mohou nacházet následující atributy:

- plaintextpassword - Klientovo heslo v čitelném tvaru
- preparedpassword - hodnota x
- clientrandom - hodnota v
- otp - hodnota otp, resp. hodnota c u "2FA"

Obsah jednotlivých zpráv je možno vidět na příkladu komunikace v kapitole 20.3.

20.3 Příklad komunikace

Tato kapitola obsahuje v následujících dvou podkapitolách příklady komunikace pro využití jednofaktorové a dvoufaktorové autentizace.

20.3.1 Jednofaktorová autentizace

Klient si chce vytvořit nový uživatelský účet s uživatelským jménem zkp a heslem ZKPdemo. Chce využívat pouze tento faktor. Komunikace mezi Klientem a Serverem při registrační fázi bude probíhat následovně:

Klient zasílá Serveru registrační data:

```
{
    "messageid": "reg_1_c",
    "username": "zkp",
    "encryptedpassword": 2365939199,
    "generator": 39551,
    "prime": 3466111739,
    "authtype": "classic",
    "demonstrationsecrets": {
        "plaintextpassword": "ZKPdemo",
        "preparedpassword": 25415556557794671
    }
}
```

Server posílá Klientovi odpověď:

```
{
    "messageid": "reg_2_s",
    "username": "zkp",
    "success": true
}
```

Atribut "success"obsahuje hodnotu True, což znamená, že byla registrace úspěšně dokončena, Klient je zaregistrován.

V autentizační fázi poté proběhne následující komunikace:

Klient Zasílá Serveru svůj login, aby získal hodnoty p, g:

```
{
    "messageid": "auth_1_c",
    "username": "zkp"
}
```

Server si najde Klienta na základě loginu ve své databázi a pošle mu následující odpověď:

```
{
    "messageid": "auth_2_s",
    "username": "zkp",
    "generator": 39551,
    "prime": 3466111739,
    "authtype": "classic"
}
```

Klient zasílá Serveru Závazek:

```
{
    "messageid": "auth_3_c",
    "username": "zkp",
    "commitment": 2214237281,
    "demonstrationsecrets": {
        "clientrandom": 14800
    }
}
```

Server zasílá Klientovi Výzvu:

```
{
    "messageid": "auth_4_s",
    "username": "zkp",
    "challenge": 53552
}
```

Klient zasílá Serveru Odpověď:

```
{
    "messageid": "auth_5_c",
    "username": "zkp",
    "response": 1662026728
}
```

Server ověří hodnotu Odpovědi a odešle zprávu s výsledkem autentizace Klientovi:

```
{
    "messageid": "auth_6_s",
    "username": "zkp",
    "success": true
}
```

Atribut "success"obsahuje hodnotu True, což znamená, že byl Klient úspěšně autentizován.

20.3.2 Dvoufaktorová autentizace

Klient si chce vytvořit nový uživatelský účet s uživatelským jménem zkp2fa a heslem ZKPdemo. Chce využívat dvoufaktorovou autentizaci. Komunikace mezi Klientem a Serverem při registrační fázi bude probíhat následovně:

Klient zasílá Serveru registrační data:

```
{
    "messageid": "reg_1_c",
    "username": "zkp2fa",
    "encryptedpassword": 2322360183,
    "generator": 63617,
    "prime": 2588849591,
    "authtype": "2FA",
    "demonstrationsecrets": {
        "plaintextpassword": "ZKPdemo",
        "preparedpassword": 25415556557794671
    }
}
```

Server posílá Klientovi odpověď:

```
{
    "messageid": "reg_2_s",
    "username": "zkp2fa",
    "success": true
}
```

Atribut "success"obsahuje hodnotu True, což znamená, že byla registrace úspěšně dokončena, Klient je zaregistrován. V této fázi Klient obdrží svůj OTP generátor.

V autentizační fázi poté proběhne následující komunikace:

Klient Zasílá Serveru svůj login, aby získal hodnoty p, g:

```
{
    "messageid": "auth_1_c",
    "username": "zkp2fa"
}
```

Server si najde Klienta na základě loginu ve své databázi a pošle mu následující odpověď:

```
{
    "messageid": "auth_2_s",
    "username": "zkp2fa",
    "generator": 63617,
    "prime": 2588849591,
    "authtype": "2FA"
}
```

Klient pošle Serveru zprávu, která již obsahuje všechny potřebné data, včetně OTP kódu:

```
{
    "messageid": "auth2fa_3_c",
    "username": "zkp2fa",
    "commitment": 145419271,
    "response": 2312987708,
    "demonstrationsecrets": {
        "otp": "30026274",
        "clientrandom": 21942
    }
}
```

Server ověří hodnotu Odpovědi a odešle zprávu s výsledkem autentizace Klientovi:

```
{
    "messageid": "auth2fa_4_s",
    "username": "zkp2fa",
    "success": true
}
```

Atribut "success"obsahuje hodnotu True, což znamená, že byl Klient úspěšně autentizován.

20.4 Implementace

Součástí této akademické práce je vzorová implementace tohoto protokolu v jazyce Python. Tato implementace obsahuje mimo nástrojů potřebných k provedení výpočtů také soubory demonstration.py, app.py a use_otp.py.

Soubor demonstration.py obsahuje statickou a okomentovanou ukázku praktické demonstrace výpočtu od registrace Klienta až po jeho autentizaci pomocí hesla či kombinace hesla a OTP.

Soubor app.py obsahuje velmi jednoduchou interaktivní verzi demonstrace sloužící pro umožnění lepšího testování a pochopení fungování protokolu.

Soubor use_otp.py slouží jako simulovaný otp generátor. V demonstration.py je (kvůli automatizaci) volána přímo funkce, která vrátí OTP, v app.py je při použití dvoufaktorové autentizace použít OTP kód získaný pomocí tohoto simulovaného OTP generátoru. OTP generátor je volán pomocí s jedním argumentem a tím je uživatelské jméno uživatele, pro kterého je OTP generováno.

Nutno podotknout, že ani jeden z těchto souborů nemají za cíl být plnou funkční implementací dané problematiky, ale pouze umožnit lépe pochopit fungování protokolu a poskytnout demonstraci toho, jakým způsobem je možno knihovnu reálně využít.

Ke spuštění obou souborů je nutno mít nainstalovanou Pythonovskou knihovnu simpy.

Ačkoli je tato akademická práce psána v češtině, program včetně komentářů byl psán v angličtině.

21 Zhodnocení protokolu

Ačkoli je autentizační schéma popsané v této práci známé již z 80. a 90. let 20. století, v praxi se příliš neosvědčilo. Důvodem je, že ačkoli nabízí srovnatelnou úroveň bezpečnosti jako autentizace založená na hashích, má několik praktických nevýhod.

První z nich je přílišná interaktivita protokolu - zatímco pro základní autentizaci je v tomto protokolu nutno poslat mezi Klientem a Serverem 5 zpráv, při klasické autentizaci je poslána pouze 1 zpráva (případně 3 zprávy u složitějšího schématu, které využívá soli, nonce či obou hodnot). Tyto zprávy navíc zbytečně komplikují praktickou implementaci. Existuje sice způsob, jak z Interaktivního protokolu udělat Neinteraktivní, a to Fiat-Shamirova transformace, a jeho výsledkem je skutečně protokol, který využívá pro autentizaci 3 zprávy, i tak je ale komplikovanější, než klasická autentizace, neboť využívá hashovací funkce a k tomu ještě stále modulární umocňování.

Druhou z nich je výpočetní náročnost protokolu. Zatímco klasická autentizace využívá hashovacích funkcí (případně jejich v kombinace s náhodnými řetězci), což je

výpočetně jednoduchá operace, autentizace založená na diskrétním logaritmu obsahuje generování velkých prvočísel (což je výpočetně velmi náročná operace) a několik kol modulárního umocňování (což je stále výpočetně náročnější, než využití hashe).

Dalším problémem, který začal být aktuální zejména, v posledních letech je fakt že schémata založená na nemožnosti výpočtu diskrétního logaritmu nejsou postkvantová, tj. bezpečná proti útoku za pomocí kvantového počítače. Kvantový počítač teoreticky dokáže pomocí Shorova algoritmu spočítat diskrétní logaritmus v polynomickém čase, a tím snížit bezpečnost těchto schémat na nulu.

Je nutno podotknout, že zatím stále[1)] neexistuje dostatečně výkonný kvantový počítač, který by mohl být využit pro praktickou kryptoanalýzu.

S hlediska čisté bezpečnosti protokolu je tento protokol v pořádku, dokonce již v základu využívá kryptografickou sůl a nonce. Největším bezpečnostním rizikem protokolu (stejně jako u všech protokolů využívajících klientem vybrané heslo) je využití slabého hesla.

21.1 Srovnání s klasickou autentizací

Bezpečnost protokolu je postavena na nemožnosti spočítat diskrétní logaritmus. Útočník může při odposlechnutí komunikace získat hodnoty p, g, t, c, r. Z rovnice $r = v - cx$ by mohl získat x, pokud by z rovnice $t \equiv g^v \pmod{p}$ získal v. K tomu by ale potřeboval spočítat diskrétní logaritmus, což v praxi nedokáže.

Stejně jako u klasického autentizačního protokolu je i zde slabým místem heslo, které si Klient volí. Možným vektorem útoku (zejména za použití slabého hesla) je tedy získání hesla pomocí útoku hrubou silou či slovníkového útoku. Útočník se může při získání hodnot y, g, p (ať už odposlechnutím komunikace či získáním dat uložených v databázi) pokoušet hrubou silou odhalit x.

Dvojice hodnot g, p v protokolu plní také funkci kryptografické soli. Užívá-li více uživatelů stejné heslo, pak platí $y_1 = y_2 \leftrightarrow p_1 = p_2 \wedge q_1 = q_2$ - bude-li se lišit p nebo q, bude se lišit i y. Útok je tedy nutné provádět na každý uložený přihlašovací údaj zvlášť, což snižuje efektivitu útoku.

Hodnoty v, c v protokolu slouží jako nonce. Uživatel zasílá při každém přihlášení jiné údaje, což velmi snižuje možnost provést takzvaný replay útok, kdy se útočník pokouší autentizovat pomocí dat z komunikace odposlechnuté v minulosti.

Rozšířený příklad ukazuje možnost využití ekvivalentu kryptografického pepře v protokolu, což také velmi snižuje možnost útoku. Útočník se v takovém případě pokouší vyřešit rovnici $z \equiv g^{xs} \pmod{p}$, kde nezná x, ani náhodný řetězec s. Proto, i když zná x, nedokáže odvodit s (musel by spočítat diskrétní logaritmus) a nemůže se pokoušet

[1)] oficiálně

získat x hrubou silou.

IV. ZERO-KNOWLEDGE ARGUMENT DOSAŽENÉHO VĚKU

22 Hashovací funkce

Hashovací[1)] funkce je jednosměrná deterministická funkce, která mapuje vstup libovolné délky na výstup konstantní délky.

Obecně mají hashovací funkce 3 základní vlastnosti:

- Jednosměrnost - je jednoduché spočítat hodnotu hashe pro daný vstup, ale pro daný výstup není možné jednoznačně určit vstup, který vedl k danému výstupu
- Determinizmus - pro každou vstupní hodnotu x je výstup funkce pokaždé stejný $y = H(x)$
- Konstantní délka výstupu - délka výstupu je předem známá a není závislá na délce ani jiných parametrech vstupu

Mezi hashovací funkce patří například také algoritmy pro kontrolu parity či výpočet Cyklického redundantního součtu (CRC). V praxi je nicméně termínem „hashovací funkce“ zpravidla myšlena speciální podkategorie těchto funkcí, kryptografická hashovací funkce.

22.1 Kryptografická hashovací funkce

Aby byla hashovací funkce použitelná v oboru kryptografie, musí kromě výše zmíněných splňovat i další požadavky:

- Dostatečná délka výstupu - délka výstupu (značíme b) hashovací funkce je měřena v bitech. Počet možných výstupů je možno spočítat podle vzorce 2^b. S rostoucí hodnotou b klesá účinnost útoků hrubou silou na hashovací funkci. Za minimální bezpečnou délku hashe je v současnosti považováno 256 bitů[2)]
- Rovnoměrné rozdělení výstupů - pro libovolný vstup musí být rozdíl pravděpodobností všech výstupů zanedbatelný
- Náhodné rozdělení výstupů - nesmí existovat žádný statisticky významný vztah mezi vstupem nebo jeho částí a výstupem nebo jeho částí

[1)] čteno „hešovací“. V některých zdrojích je uváděn počeštěný termín „hašovací“. Výstupem funkce je hash, čteno „heš“.

[2)] Národní úřad pro kybernetickou a informační bezpečnost (NÚKIB) vydává Doporučení v oblasti kryptografických prostředků - Minimální požadavky na kryptografické algoritmy [53]. Ve verzi aktuální ke dni odevzdání této práce (verze z 28.11.2018) je pro hashovací funkce považována délka 224 bitů za „dosluhující“ s doporučením přestat používat hashe s touto délkou do roku 2023.

- Vhodná rychlost výpočtu - v závislosti na očekáváném způsobu použití je nutné, aby byla hashovací funkce dostatečně rychlá (například pro hashování souborů) nebo naopak dostatečně pomalá (pro hashování hesel)

Z definice obecné hashovací funkce vyplývá, že pro všechny hashovací funkce existují takzvané kolize, tedy že několik různých vstupů sdílý stejný hash. Intuitivně, pokud existuje pouze omezené množství možných hashů 2^b, zatímco množina možných vstupů je neomezená, pak zákonitě musí kolize existovat.

Existence kolizí se využívá například v hashovacích stromech či hashovacích tabulkách, což jsou datové struktury umožňující velmi rychlé vyhledávání. Pro kryptografické hashovací funkce jsou kolize naopak nechtěnou vlastností, neboť kryptografické hashovací funkce spoléhají na praktickou nemožnost nalezení kolize, ať už úmyslně či náhodou.

Existují 3 typy kolizí:

- pravá kolize - cílem hledání kolize je nalezení dvou různých vstupů, které mají stejný hash, tedy $\{y_1 \neq y_2 \wedge H(y_1) = H(y_2)\}$
- 1. předobraz - hledáním prvního předobrazu je myšleno hledání takového vstupu, jehož hashem je požadovaný řetězec, tedy hledání y, když známe x, kdy $x = H(y)$
- 2. předobraz - při hledání druhého předobrazu jde o nalezení takového vstupu, který má stejný hash jako jiný známý vstup. Tedy hledání y_2, když známe y_1 v $\{y_1 \neq y_2 \wedge H(y_1) = H(y_2)\}$

Ačkoli to na první pohled vypadá, že je hledání pravé kolize a druhého předobrazu tou samou činností, rozdílem je, že u pravé kolize jsou hledány dva vstupy, které sdílí společný libovolný výstup, zatímco u hledání druhého předobrazu je známý jeden vstup a výstup a pátrá se po druhém vstupu se stejným hashem.

Kryptografické hashovací funkce musí mít odolnost proti všem třem typům kolizí, nesmí být tedy prakticky možno najít kolizi ani předobraz hashe.

22.2 Hash chain

„Hash chain[3)] je sekvence hodnot získaných pomocí postupného aplikování kryptografické hashovací funkce na původní vstup. Díky vlastnostem hashových funkcí je relativně jednoduché spočítat následné hodnoty řetězu, ale, pokud známe určitou hodnotu,

[3)] je možno přeložit jako „řetěz hashů“

je prakticky nemožné získat předchozí hodnotu.“[4)][54]

Pro úplnost je třeba dodat, že americký Národní úřad pro vědu a technologii (NIST) používá jinou definici hash chainu: „Datová struktura, ke které jde pouze přidávat data, kde jsou data zabalena do datových bloků, které v novém bloku obsahují hash předchozího datového bloku. Tato datová struktura poskytuje důkaz nechtěné manipulace, protože jakákoli modifikace bloku dat změní hash uložený v následujícím datovém bloku.“[5)][55] Tato definice odpovídá spíše současnému vnímání termínu Blockchain.

Hash chain dle této druhé definice můžeme považovat za zobecnění hash chainu z první definice, respektive hash chain z první definice je speciálním případem hash chainu z druhé definice, kdy hashovaný datový blok obsahuje pouze hodnotu hashe.

22.2.1 Forma zápisu

V rámci této kapitoly bude použita následující forma zápisu užití hashovacích funkcí:

$$y = H^n(x) \tag{22.1}$$

Kde H značí libovolnou hashovací funkci, n počet použití hashovací funkce, x původní vstup a y finální výstup - hash. Můžeme tedy napsat:

$$\begin{aligned} y = H^1(x) &\iff y = H(x) \\ y = H^2(x) &\iff y = H(H(x)) \\ y = H^3(x) &\iff y = H(H(H(x))) \end{aligned} \tag{22.2}$$

Pro $n \leq 0$ platí:

$$\begin{aligned} y = H^0(x) &\iff y = x \\ y = H^{-1}(x) &\iff x = H^1(y) \\ y = H^{-2}(x) &\iff x = H^2(y) \end{aligned} \tag{22.3}$$

Jak je možno vidět na rovnicích výše, $y = H^{-1}(x)$ znamená, že y je první předobraz hashe x.

Z použité formy zápisu vyplývá následující vlastnost:

[4)] A hash chain is a sequence of values derived via consecutive applications of a cryptographic hash function to an initial input. Due to the properties of the hash function, it is relatively easy to calculate successive values in the chain but given a particular value, it is infeasible to determine the previous value.

[5)] An append-only data structure where data is bundled into data blocks that include a hash of the previous data block's data within the newest data block. This data structure provides evidence of tampering because any modification to a data block will change the hash digest recorded by the following data block.

$$H^n(x) = H^{n-m}(H^m(x)) \tag{22.4}$$

22.3 Ilustrativní hashovací algoritmus

Ačkoli je pro bezpečnost protokolu nezbytně nutné, aby byl použit kryptograficky bezpečný hashovací algoritmus, pro účely demonstrace a snažšího pochopení principu fungování protokolu nejsou klasické hashovací funkce nejvhodnější. Proto je v příkladech dále využit prot tyto účely vytvořený hashovací algoritmus pojmenovaný Ilustrativní hashovací algoritmus.

Je nutno zdůraznit, že tento algoritmus není absolutně použitelný při reálné implementaci, neboť z pohledu bezpečnosti obsahuje kritické chyby, je ale mnohem přehlednější.

22.3.1 Princip fungování

Tento algoritmus vychází z hashovacího algoritmu SHA-256.

Výstup hashovací funkce je možno popsat následující maskou: $\{N\}^n|\{X\}^x$. Jinými slovy n číslic následovaných svisou čárou a poté dalších x znaků. Parametry n i x je možno nastavit dle potřeby.

Hash se skládá ze 3 částí:

- $\{N\}^n$ - počítadlo, které má délku n, a zaznamenává, kolikrát byla použita hashovací funkce k dosažení daného výstupu. V případě potřeby je počítadlo zleva doplněno nulami
- | - znak, který slouží jako oddělovač
- $\{X\}^x$- prvních x znaků hashe získaného aplikací hashovací funkce SHA-256 na požadovaný vstup

Chování hashovací funkce, respektive nastavení hodnoty počítadla N, se liší v závislosti na tom, zda má daný vstup funkce stejnou masku, jako její výstup (tj. $\{N\}^n|\{X\}^x$ s požadovanými parametry n, x):

- Nemá stejnou masku - hodnota počítadla je nastavena jako $N = 1$
- Má stejnou masku - hodnota počítadla je nastavena jako $N = N + 1$

Chování je možno vidět na následujícím příkladu:

$$\begin{aligned} H(seed) &= 001|19b258 \\ H(001|19b258) &= 002|d55102 \\ H(002|d55102) &= 003|aebf56 \end{aligned} \tag{22.5}$$

22.3.2 Srovnání algoritmů

Tato část obsahuje demonstraci rozdílu v čitelnosti a názornosti Ilustrativního hashovacího algoritmu oproti klasickému hashovacímu algoritmu MD5[6]. Jako vstup hashe byl zvolen náhodný řetězec s.

MD5:

$$\begin{aligned} x = H^3(s) &= 1b2ef6acce61b5f6138dbf382c5f379c \\ y = H^5(x) &= 9de986d96fb3e3709aad48bb6e0ef164 \\ z = H^?(y) &= 9f7212afb2f90148ecaa1ea821a8d3e4 \end{aligned} \tag{22.6}$$

Ilustrativní hashovací funkce:

$$\begin{aligned} x = H^3(s) &= 00003|b2d3f34679 \\ y = H^5(x) &= 00008|62ae2d2322 \\ z = H^?(y) &= 00010|d66b042b769 \end{aligned} \tag{22.7}$$

Na příkladu výše je vidět výhodu Ilustrativní hashovací funkce v lepší názornosti vztahů mezi jednotlivými články hash chainu, včetně toho, kolikrát je nutno hashovat hodnotu y, aby bylo dosaženo hodnoty z. Toto je zároveň důvod, proč není možné tento algoritmus použít jinak než pro demonstraci.

U MD5 (resp. u všech kryptografických hashovacích funkcí) není možné z hodnoty výstupu zjistit, kolikátým článkem řetězce je daný hash(což je vlastnost, která je nezbytně nutná pro praktické nasazení).

22.4 Využití hash chainu v Zero-knowledge protokolech

V roce 2013 bya zveřejněna práce Sebastiana Angela a Mechaela Walfishe na téma Ověřitelné aukce pro online aukce reklam[7][56]. V této práci je v části Důvěrné porovnávání přirozených čísel[8] demonstrováno využití hash chainu pro zhodnocení, zda je přirozené číslo jedné strany větší než nebo rovno hraničnímu číslu dodanému druhou stranou bez toho, aby se druhá strana toto číslo dozvěděla. Uvádí také způsob, jak postavit opačný

[6] Ačkoli MD5 již není považován za bezpečný hashovací algoritmus, byl vybrán pro srovnání, neboť má oproti SHA-256 kratší délku výstupu

[7] Verifiable Auctions for Online Ad Exchanges

[8] Private Integer Comparisons

důkaz, tedy že je číslo menší, než hraniční hodnota. „Hlavní myšlenkou protokolu je zakódovat přirozené číslo do délky hash chainu, zveřejnit poslední článek hash chainu jako Závazek a posléze publikovat hodnotu mezičlánku řetězu jako důkaz.“[9)][56]

Na tuto práci navazuje Ankur Shah Delight ze společnosti Stratumn ve článku Zero-knowledge důkaz[10)] věku pomocí hash chainů.[57][11)], kdy aplikuje myšlenku hash chainu na situaci prokázání věku bez vyzrazení data narození. Hlavní změnou je přidání Důvěryhodné třetí strany, která do modelu přináší prvek důvěry v situaci, kdy si Dokazující nemůže svou hodnotu (v tomto případě věk) zvolit libovolně.

Ačkoli jde technicky o Neinteraktivní Zero-knowledge Argument a ne o Důkaz[12)], je pro jednoduchost používán pro popis protokolu termín „důkaz“.

22.4.1 Vlastnosti hash chainu

V základní verzi protokolu vystupují dvě strany. První stranou je Dokazovatel (dále značený písmenem P), který předkládá důkaz druhé straně - Ověřovateli (V).

Vyžaduje-li to situace, je možné do komunikace zapojit Důvěryhodnou třetí stranu (Trusted party, T), kterému obě strany věří, a zároveň zná tajemství P. T v takovém případě provádí základní výpočetní operace a jejich výsledek strvrzuje svým elektronickým podpisem.

Protokol pracuje se dvěma proměnnými - „tajnou“ proměnnou s (*secret* - tajemství), kterou zná pouze P (případně T), a dále proměnnou t (*threshold* - práh). Tuto proměnnou určuje V a předává ji P, aby ten mohl dokázat, že $s \geq t$. Cílem tedy je, aby se V dozvěděl pouze, že $s \geq t$, ale ne hodnotu s.

P se v rámci protokolu vždy dozví t, proto se nedá využít ke srovnání dvou „tajných“ čísel, neboli k porovnání, zda je větší tajné číslo P či tajné číslo V, bez toho, aby se vzájemně tato čísla strany dozvěděly (takzvaný Yaův milionářský problém).

Vzhledem k tomu, že jak s, tak t jsou zakódovány v délce hash chainu, tedy v počtu kroků (hashů), které byly vykonány, musí obě tato čísla patřit do oboru přirozených čísel, nebo musí být do tohoto oboru možné je vhodným způsobem převést. Například [56] využívá tuto metodu ke srovnání příhozů v akci, tedy částek v dolarech a centech. Do vhodného tvaru tyto příhozy převádí vynásobením konstantou 100. Pro srovnání záporných čísel je například možné k oběma číslům přičíst konstantu, která bude větší než převrácená hodnota nejmenšího možného čísla, k jehož srovnání půjde tento důkaz použít, atd.

[9)] The main idea of the protocol is to encode an integer in the length of a hash chain, publish the tail of the chain as the commitment, and later publish an intermediate value in the chain as the proof.

[10)] technicky nejde o Důkaz, ale o Argument, viz níže

[11)] Zero Knowledge Proof of Age Using Hash Chains

[12)] neboť Solidnost protokolu je pouze Výpočetní, viz kapitola 24.2

Výpočetní náročnost protokolu je nepřímo úměrná velikosti srovnávaných čísel, a, ačkoli je hashování z výpočetního hlediska rychlá operace, je možno narazit na limit výkonu zařízení, na kterém je důkaz tvořen či ověřován. V závislosti na implementaci a očekávané velikosti srovnávaných čísel může být nutné v rámci zrychlení výpočtů srovnávaná čísla například vydělit konstantou, čímž dojde k zrychlení protokolu za cenu částečné ztráty přesnosti.

Je také možné postavit „opačný“ důkaz, tedy že $s \leq t$. V takovém případě je do protokolu zavedena limitní proměnná m (maximum), kterou volí P (případně T), a kterou prozradí V jako součást důkazu. Tato proměnná musí patřit do oboru přirozených čísel a musí splňovat podmínku: $\{m > s \wedge m > t\}$. Podmínku je možno vyjádřit také jako $s \leq t < m$.

O tom, který typ důkazu bude tvořen, musí být rozhodnuto již před začátkem výpočtu, proto není možno použít jeden důkaz pro obě srovnání. V takovém případě je nutno připravit zároveň dva důkazy, které jsou pak vyhodnoceny každý zvlášť. Dohromady pak oba důkazy dokazují, že $t_2 > s \geq t_1$.

Bezpečnost protokolu je postavena na bezpečnosti použitého hashovacího algoritmu, konkrétně na jeho odolnosti proti nalezení prvního předobrazu. Protokol není závislý na konkrétním algoritmu a je možno použít libovolný kryptogracký hashovací algoritmus.

Při výpočtu hash chainu si P (případně T) volí náhodný řetězec r. Bezpečnost důkazu závisí na náhodnosti tohoto řetězce, je tedy nutné pro jeho vygenerování využít vhodný kryptograficky bezpečný generátor pseudonáhodných čísel, a tím získat vstup s dostatečnou úrovní entropie. Je vhodné, aby r nepatřil do množiny možných výstupů zvolené hashovací funkce[13)], proto by měl mít jinou délku, než je délka hashe použité hashovací funkce.

V rámci protokolu jsou postupně počítány 4 články hash chainu. Tyto články jsou z důvodu přehlednosti označeny slovně.

První hash nese označení *base*. Jde vždy o první článek hash chainu (H^1), počítá jej P (případně T), používá se k výpočtu důkazu a V se jej nesmí dozvědět.

Další hash je označen *target*. Jde o poslední článek hash chainu, počítá jej P (případně T; v takovém případě jej zajišťuje svým elektronickým podpisem), je poslán V a označuje „cílovou“ hodnotu, ke které se má V při ověřování důkazu dopočítat.

Třetí hash byl pojmenován *proof*. Tento článek počítá P, který jej poté předá V, a slouží jako samotný důkaz dosaženého věku.

[13)] je-li r možným výstupem použité hashovací funkce, jde vlastně o „nultý“ člen hash chainu, tedy $r = H^{-1}(base)$. Znalost r pak teoreticky umožňuje Podvádějícímu P vytvořit důkaz, ve kterém je s o 1 větší, než ve skutečnosti je. V případě použití jiného r je pak výpočet $H^{-1}(base)$ klasickým hledání prvního předobrazu hashe

Poslední hash, *check*, označuje výpočet k ověření důkazu, který provádí V. Pro korektní důkaz platí, že $check = target$.

23 Princip fungování protokolu

Tato kapitola se věnuje popisu principu fungování čtyř verzí protokolu. Protokoly jsou děleny na základní, kde se vyskytují pouze dvě strany, a rozšířené, kde se vyskytují strany 3, a dále na klasické, kde se dokazuje, že $s \geq t$, a opačné, jejíž cílem je demonstrovat, že $s \leq t$.

Následující podkapitoly se věnují těmto verzím protokolu:

- klasická základní verze
- opačná základní verze
- klasická rozšířená verze
- opačná rozšířená verze

23.1 Klasická základní verze

V této verzi vystupují pouze 2 strany - Dokazovatel P a Ověřovatel V. P zná s, hodnotu t znají obě strany. Cílem je dokázat, že $s \geq t$ bez toho, aby se V dozvěděl s.

Průběh protokolu je možno rozdělit na 3 fáze.

23.1.1 Fáze 1

V první, předpřípravné fázi dojde k výpočtu sady pro vytvoření důkazu - takzvaného Proofkitu. Ten bude v další fázi sloužit k vytvoření samotného důkazu. Tento výpočet provádí P.

Nejprve je vygenerován libovolný náhodný řetězec r, který slouží jako základ pro výpočet hash chainu. Poté jsou spočítány články hash chainu označené jako *base* a *target*. Proměnná r je po výpočtu těchto 2 hashů zapomenuta.

$$\begin{aligned} r &= random() \\ base &= H^1(r) \\ target &= H^{s+1}(r) \end{aligned} \tag{23.1}$$

23.1.2 Fáze 2

V druhé fázi dojde k výpočtu samotného důkazu, který je poté předán druhé straně. Tento výpočet také provádí P.

Je spočítán článek hash chainu *proof*, a poté jsou hodnoty *proof* a *target* poslány V.

$$proof = H^{s-t}(base) \tag{23.2}$$

23.1.3 Fáze 3

V poslední fázi dojde k ověření důkazu. Tento výpočet provádí V.

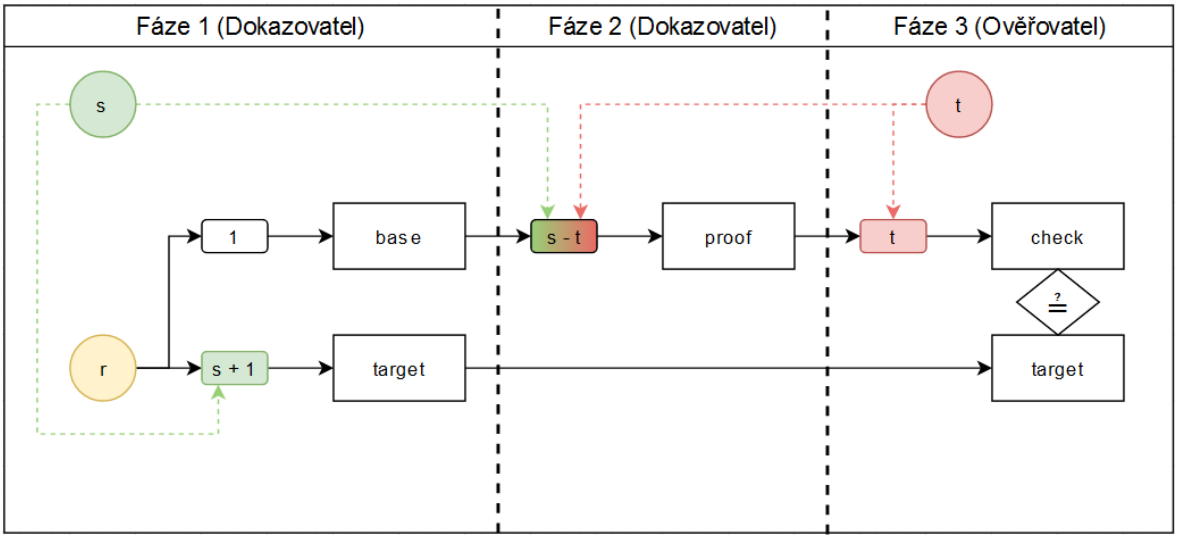
Je spočten hash označený jako *check* a porovnán s hashem *target*. Pokud se tyto hodnoty rovnají, byl protokol úspěšně dokončený a P dokázal přesvědčit V, že $s \geq t$.

$$check = H^{t}(proof) \tag{23.3}$$

23.1.4 Důkaz platnosti protokolu

Obrázek 23.1 ukazuje vztah použitých proměnných a článků hash chainu s označenými stranami komunikace.

Obr. 23.1 Vztah použitých proměnných a článků hash chainu (klasická základní verze)

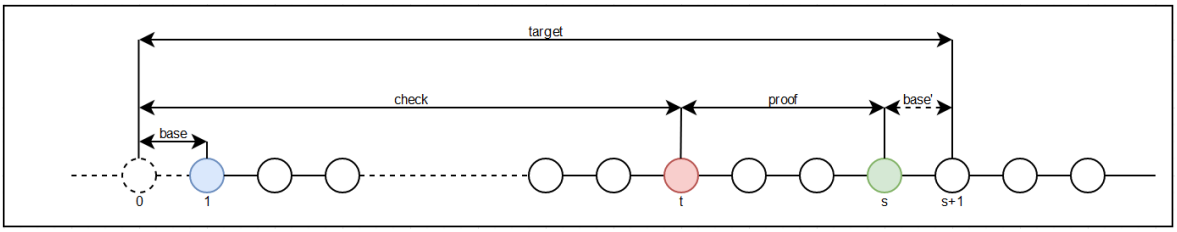


Následuje důkaz platnosti protokolu:

$$\begin{aligned} target &= check \\ H^{s+1}(r) &= H^{t}(proof) \\ H^{s+1}(r) &= H^{t}(H^{s-t}(base)) \\ H^{s+1}(r) &= H^{t}(H^{s-t}(H^{1}(r))) \\ H^{s+1}(r) &= H^{t+s-t+1}(r) \\ H^{s+1}(r) &= H^{s+1}(r) \end{aligned} \tag{23.4}$$

Obrázek 23.2 demonstruje tento důkaz graficky.

Obr. 23.2 Grafické znázornění platnosti klasického důkazu



23.1.5 Příklad

P má své tajné číslo, $s = 10$. V určil hodnotu prahu $t = 8$. P chce prokázat, že je $s \geq t$. Výpočet proběhne následovně:

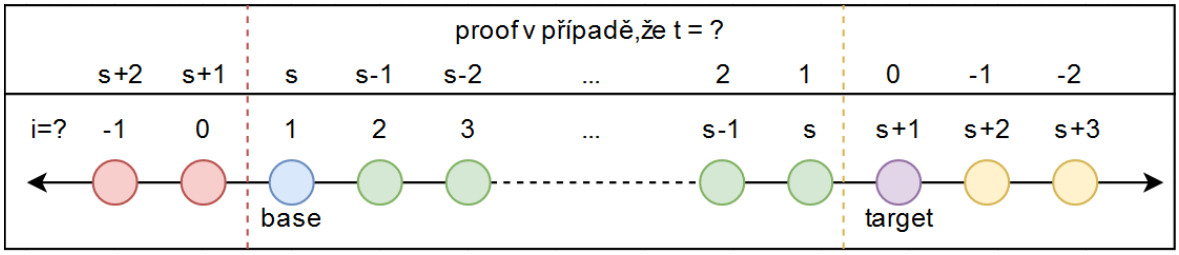
1. P vygeneruje náhodný r
2. P spočítá $base = H^{1}(r) = 001|d1404d$
3. P spočítá $target = H^{10+1}(r) = H^{11}(r) = 011|7eae36$
4. P spočítá $proof = H^{10-8}(base) = H^{2}(001|d1404d) = 003|fdb990$
5. V spočítá $check = H^{8}(proof) = H^{8}(003|fdb990d) = 011|7eae36$
6. V ověří, že $check = target$ $(011|7eae36 = 011|7eae36)$

Tato rovnost platí, P proto přesvědčil V, že $s \geq t$.

23.1.6 Vztah hodnot s, t

články hash chainu, které může P poslat jako $proof$, je možno rozdělit do několika kategorií. Pro klasický protokol tyto kategorie shrnuje obrázek 23.3:

Obr. 23.3 Možné hodnoty *proof* v závislosti na hodnotě t (klasická vetze)



Spodní řada čísel označuje index článku hash chainu(tj. $H^i(r)$), v horní řadě je možno vybrat článek, který může být poslán jako *proof* v závislosti na hodnotě t a jejího vztahu k s. Dále je zde označeno, kde se nachází články *base* a *target*.

Index článku *proof* může v závislosti na vztahu t a s nabývat následujících hodnot:

- $1 \leq t \leq s - 1$ (zelená) - očekávananý rozsah t, pro tyto hodnoty lze běžným způsobem protokol využít. Hodnoty odpovídají podmínce $s > t$. V případě, že se Podvádějící P pokusí poslat namísto očekávané hodnoty některý jiný z těchto článků, pak vlastně posílá důkaz pro jinou hodnotu t (která může být větší i menší než očekávaná)
- $t = s$ (modrá) - specifická (platná) situace, kdy $s = t$, z čehož vyplývá, že $proof = base$. V tomto konkrétním případě nevadí, že se V dozví hodnotu *base*. Odhalením *base* může V sice spočítat tajnou hodnotu s, ale pouze pokud ví, že zná hodnotu *base*
- $t = 0$ (fialová) - protokol není možno použít. V tomto případě se $proof = target$, jeho znalost tedy nedokazuje znalost *base*. Nemá proto smysl tyto hodnoty poslat, neboť nic nedokazují. Nemožnost využití protokolu také vyplývá z faktu, že 0 není přirozené číslo (a protokol slouží pouze ke srovnání přirozených čísel)
- $t < 0$ (žlutá) - protokol není možno použít. Nemá smysl ze stejných důvodů, jako předchozí bod
- $t > s$ (červená) - protokol není možno použít. Odesláním těchto hodnot *proof* by P mohl podvádět (tj. tvrdit, že je $s \geq t$, i když ve skutečnosti není), k jejich získání by ale musel spočítat $H^{-n}(base)$, tedy najít první předobraz použité hashovací funkce, což není v praxi možné

23.2 Opačná základní verze

Hash chain je možno použít také v rámci opačného protokolu, který dokazuje, že $s \leq t$. V základní verzi opět figurují pouze P a V. P si zvolí pomocné číslo m, které prozradí

V. K výpočtům jsou poté použity obrácené proměnné $s' = m - s$ a $t' = m - t$.

Proměnnou m je nutno vybrat z ohledem na to, že musí být větší, než t, jinak nebude možno důkaz vytvořit.

Obrácení proměnných funguje na následujícím principu:

$$\begin{aligned} s &\leq t \\ m - s' &\leq m - t' \\ m + s' &\geq m + t' \\ s' &\geq t' \end{aligned} \tag{23.5}$$

Kromě použití obrácených proměnných funguje protokolo stejným způsobem jako klasická verze.

Výpočet probíhá opět ve 3 fázích.

23.2.1 Fáze 1

Výpočet provádí P, který si v této části zvolí navíc m.

$$\begin{aligned} r &= random() \\ base &= H^1(r) \\ target &= H^{s'+1}(r) \end{aligned} \tag{23.6}$$

23.2.2 Fáze 2

P provede výpočet a pošle V hodnoty $proof$, $target$ a m.

$$proof = H^{s'-t'}(base) \tag{23.7}$$

23.2.3 Fáze 3

Výpočet provádí V. Ten srovná jím vypočítaný článek $check$ s obdrženým článkem $proof$ a pokud se tyto hashe rovnají, dokázal P, že $s \leq t$.

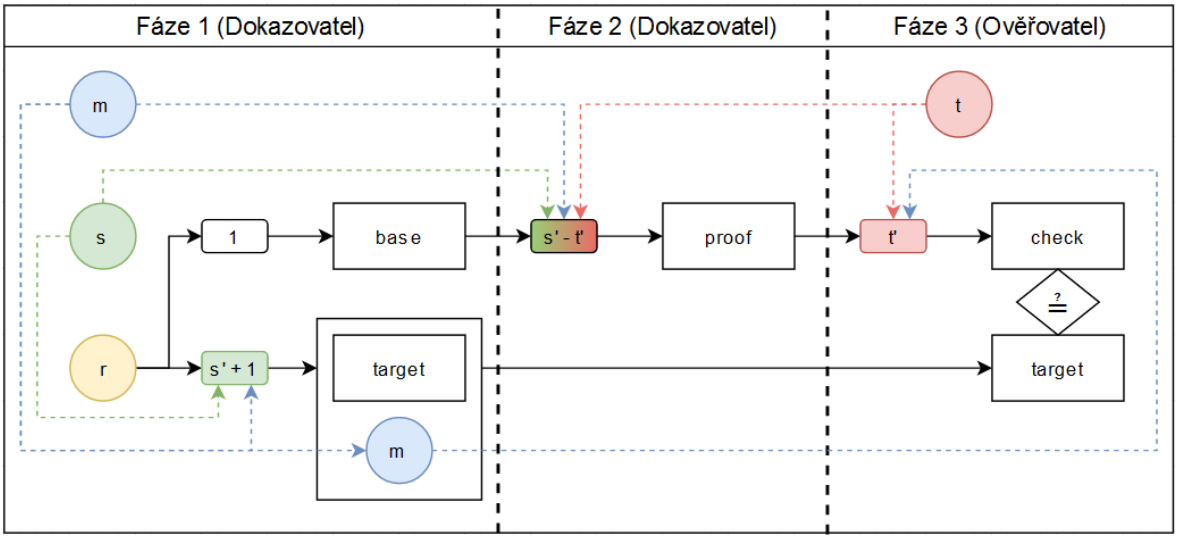
$$check = H^{t'}(proof) \tag{23.8}$$

23.2.4 Důkaz platnosti protokolu

Obrázek 23.4 ukazuje vztah použítých proměnných a článků hash chainu s označenými stranami komunikace.

Následuje důkaz platnosti protokolu:

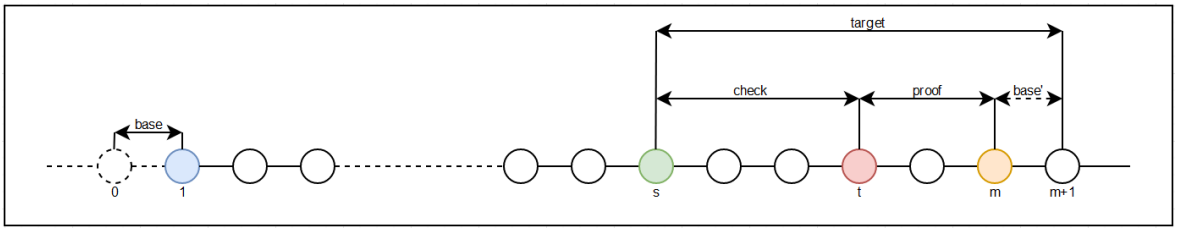
Obr. 23.4 Vztah použitých proměnných a článků hash chainu (opačná základní verze)



$$
\begin{aligned}
target &= check \\
H^{s'+1}(r) &= H^{t'}(proof) \\
H^{s'+1}(r) &= H^{t'}(H^{s'-t'}(base)) \\
H^{s'+1}(r) &= H^{t'}(H^{s'-t'}(H^{1}(r))) \\
H^{s'+1}(r) &= H^{t'+s'-t'+1}(r) \\
H^{m-s+1}(r) &= H^{m-s+1}(r)
\end{aligned}
\tag{23.9}
$$

Obrázek 23.5 demonstruje tento důkaz graficky.

Obr. 23.5 Grafické znázornění platnosti opačného důkazu



23.2.5 Příklad

P má své tajné číslo, $s = 10$. V určil hodnotu prahu $t = 14$. P stanovil hodnotu $m = 20$ a chce prokázat, že $s \leq t$. Výpočet proběhne následovně:

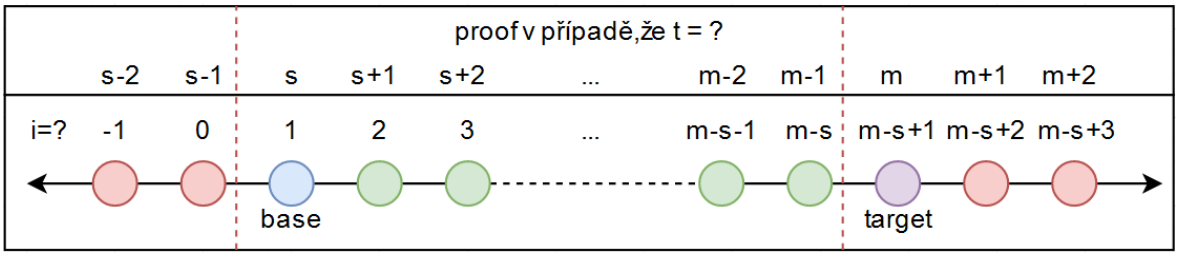
1. P vygeneruje náhodný r

2. P spočítá $base = H^1(r) = 001|d1404d$
3. P spočítá $target = H^{20-10+1}(r) = H^{11}(r) = 011|7eae36$
4. P spočítá $proof = H^{(20-10)-(20-14)}(base) = H^4(001|d1404d) = 005|d24849$
5. V spočítá $check = H^{20-14}(proof) = H^6(005|d24849) = 011|7eae36$
6. V ověří, že $check = target$ $(011|7eae36 = 011|7eae36)$

23.2.6 Vztah hodnot s, t, m

Články hash chainu, které může P poslat jako $proof$, je možno rozdělit do několika kategorií. Pro klasický protokol tyto kategorie shrnuje obrázek 23.6:

Obr. 23.6 Možné hodnoty $proof$ v závislosti na hodnotě t (opačná verze)



Spodní řada čísel označuje index článku hash chainu(tj. $H^i(r)$), v horní řadě je možno vybrat článek, který může být poslán jako $proof$ v závislosti na hodnotě t a jejího vztahu k s. Dále je zde označeno, kde se nachází články $base$ a $target$.

Index článku $proof$ může v závislosti na vztahu t a s nabývat následujících hodnot:

- $s + 1 \leq t \leq m - 1$ (zelená) - očekávananý rozsah t v závislosti na daném m, pro tyto články lze běžným způsobem protokol využít pro dokázání podmínky $s < t$
- $s = t < m$ (modrá) - specifická platná situace, kdy $t = s$. V takovém případě se $proof = base$
- $t = m$ (fialová) - protokol není možno použít. V tomto případě se $proof = target$, jeho znalost tedy nedokazuje znalost $base$
- $t > m$ (červená) - důkaz nelze spočítat, neboť nesplnění nutné podmínky $t < m$ vede ve výpočtu k hashovaní záporným počtem kroků
- $t < s$ (červená) - protokol není možno použít. Odesláním těchto hodnot $proof$ by P mohl podvádět (tj. tvrdit, že je $s \leq t$, i když ve skutečnosti není), k jejich získání by ale musel spočítat $H^{-n}(base)$, tedy najít první předobraz použité hashovací funkce, což není v praxi možné

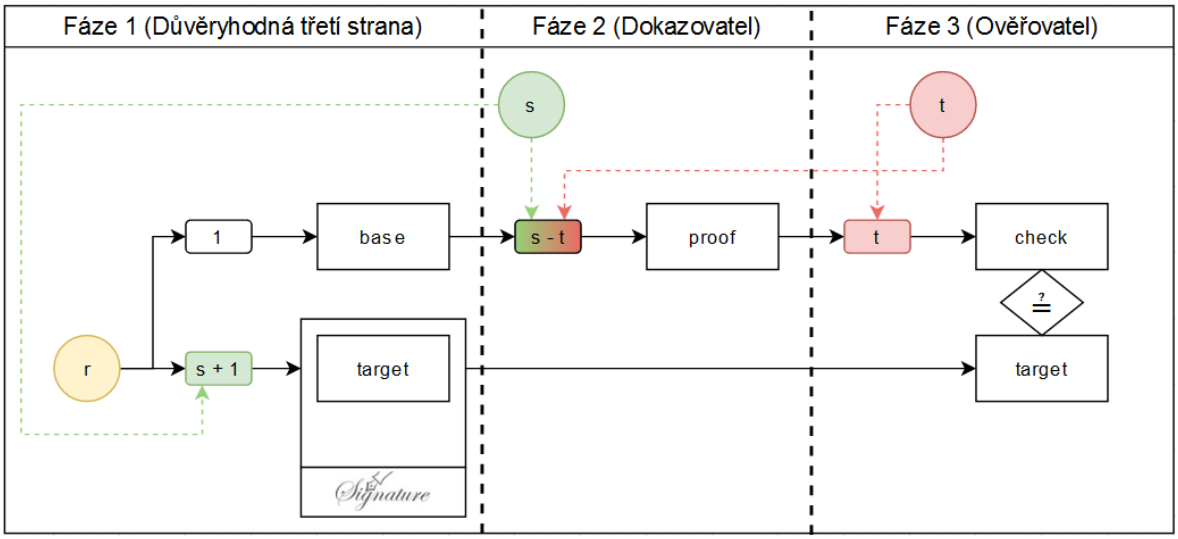
23.3 Rozšířená verze

V rozšířené verzivystupuje kromě P a V také Důvěryhodná třetí strana T. Tento model je vhodný v situaci, kdy P sice prokazuje velikost s, ale nevybírá si jej. T zná s a V mu věří, proto může věřit důkazu předloženému P.

Na rozdíl od jednoduché verze zde první fázi protokolu - výpočet *base* a *target* (a případně generování m) provádí T, nikoli P. T pak hodnoty, které se předávají jako součást důkazu V, opatří svým elektronickým podpisem, čímž umožní V ověřit jejich pravost, a všechny hodnoty předá P ve formě „sady pro tvorbu důkazů“ - takzvaného Proofkitu. Další fáze probíhají beze změny s výjimkou nutnost ověřit pravost elektronického podpisu.

Obrázky 23.7, resp. 23.8 ukazují vztah použitých proměnných a článků hash chainu s označenými stranami komunikace.

Obr. 23.7 Vztah použitých proměnných a článků hash chainu (klasická rozšířená verze)



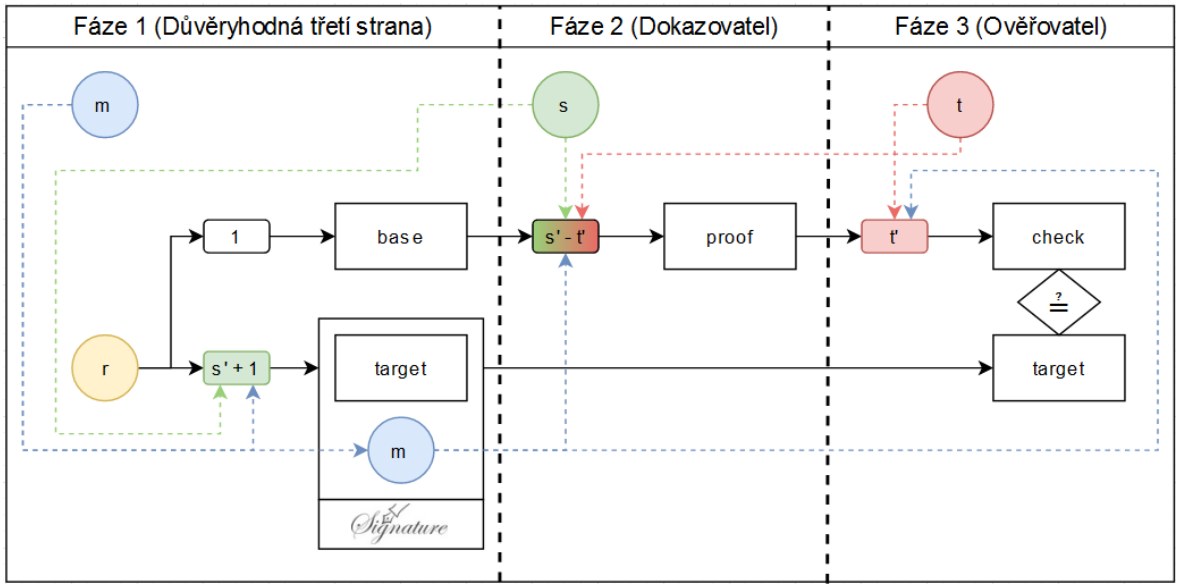
24 Bezpečnost protokolu

Cílem této kapitoly je zhodnotit protokol z hlediska požadavků na Zero-knowledge protokoly - jednotlivé podkapitoly se věnují Úplnosti, Solidnosti a Zero-knowledge.

24.1 Úplnost

Protokol je Úplný, neboť Čestný P dokáže vždy vytvořit takový *proof*, který V přesvědčí, bez ohledu na typ protokolu.

Obr. 23.8 Vztah použitých proměnných a článků hash chainu (opačná rozšířená verze)



24.2 Solidnost

Podvádějící P je takový, který se pokouší prokázat, že jeho tajné číslo splňuje podmínku $s \geq t$, resp. $s \leq t$, i když ji ve skutečnosti nesplňuje.

V případě základního modelu může P poslat libovolnou hodnotu *proof* i *target*, protože generování těchto hodnot má plně pod kontrolou. Z toho vyplývá, že není možné tento model využít pro prokázání pravdivosti informace, která si nemůže P sám zvolit. Tento model je nicméně možné využít například pro vytvořen Závazku a k jeho pozdějšímu Zrušení, případně jako součást jiného protokolu. Tento model je využíván například v [56] v kontextu online aukce.

V případě rozšířeného modelu toto neplatí. P se může pokusit podvádět tím, že pošle jako *proof* jiný článek hash chainu, než by měl[1)]. Z pohledu P dává smysl poslat hodnotu, která by dokazovala, že je s větší (menší u opačného důkazu), než ve skutečnosti je, tedy o „červené“ hodnoty z obrázku 23.3 (resp. z obrázku 23.6). K jejich výpočtu by musel nejen spočítat první předobraz hashe *base* (což samo o sobě je prakticky nemožné), tento předobraz by musel mít stejnou délku, jako výstup dané hashovací funkce (tedy najít jeden konkrétní předobraz), a i při jeho nalezení by mohl předstírat, že je s větší o 1, než ve skutečnosti je. Pro větší rozdíl by musel najít první předobraz několik po sobě.

Z výše uvedeného tedy vyplývá, že P může v praxi podvádět pouze se zanedbatelnou pravděpodobností. Pokud by měl P k dispozici neomezenou výpočetní kapacitu,

[1)] technicky může P poslat i úplně náhodnou hodnotu, ale V v takovém případě není schopen důkaz nijak ověřit, a proto jej vždy odmítne

dokázal by hledat první předobrazy hashů dle potřeby a dokázal by vytvořit důkaz pro libovolnou hodnotu s bez ohledu na to, zda splňuje požadovanou podmínku. Z těchto dvou vlastností vyplývá, že jde o protokol s Výpočetní Solidností.

Výpočetní Solidnost je také důvod, proč jde technicky o Zero-knowledge Argument a nikoli Zero-knowledge Důkaz.

24.3 Zero-knowledge

Podvádějícím V je myšlen ten, který se pokouší zjistit o s více informací, než to, že $s \geq t$, resp. $s \leq t$.

V ze svého pohledu zná tyto dva články hashchainu:

$$\begin{aligned} proof &= H^{x-t}(y) \\ target &= H^{x}(y) \end{aligned} \tag{24.1}$$

Vzhledem k tomu, že nezná x ani y a nemá žádnou možnost, jak kteroukoli z těchto hodnot zjistit či dopočítat, není možné, aby V podváděl, a to ani za předpokladu, že by dokázal spočítat první předobraz použité hashovací funkce.

I kdyby měl V k dispozici neomezenou výpočetní kapacitu a dokázal pro každý hash najít jeho první předobraz, nijak by mu to nepomohlo - ačkoli by při výpočtech předobrazů narazil na opravdovou hodnotu *base*, nijak by nevěděl, že se u této hodnoty má zastavit.

Z tohoto důvodu tento protokol nejen že splňuje Zero-knowledge, jde o Perfektní Zero-knowledge.

Toto platí bez ohledu na to, zda jde o klasický či opačný důkaz, a základní či rozšířený model.

25 Použití hash chainu pro důkaz dosaženého věku

Jedním z možných praktických využití hash chainu je důkaz[1)] dosaženého věku, konkrétně využití rozšířeného modelu s Důvěryhodnou třetí stranou, která dosažený věk P zná a garantuje.

Toto využití má svá specifika, neboť dosažený věk je z principu dynamická, neustále rostoucí, veličina. Model musí umět pracovat s faktem, že P může dnes nesplňovat určitou podmínku, a proto nesmí být schopen vytvořit důkaz, ale zítra už stejnou podmínku splňovat může, a proto musí být schopen důkaz sestavit.

Z toho také vyplývá druhé specifikum - 1. fáze výpočtu, kterou provádí T, zpravidla

[1)] v rámci této kapitoly je slovo „důkaz“ používáno v obecném smyslu, i zde jde o Zero-knowledge Argument a nikoli Důkaz

bude probíhat v jiný den, než 2. a 3. fáze, které provádí P, respektive V. V kombinaci s neustále rostoucím věkem musí být P schopen využít Proofkit k prokázání věku, kterého v okamžiku vystavení Proofkitu ještě dosahovat nemusel.

25.1 Použité proměnné

Tento protokol pracuje s vlastní sadou proměnných, které jsou vyjádřeny ve formě datumu[2)]. Použity jsou následující 4 (5 v případě opačné verze) datumy:

- datum narození (D_n) - den narození P
- datum proofkitu (D_p) - den, ke kterému je tvořen Proofkit
- datum důkazu (D_d) - datum, kdy je věk prokazován
- prokazovaný datum (D_x) - vyjadřuje hraniční den narození nutný ke splnění podmínky
- maximální datum (D_m) - využívá se pouze v opačném modelu a slouží k výpočtu m

25.1.1 Omezení

Výše zmíněné datumy spolu vzájemně souvisí. K tomu, aby bylo možné spočítat důkaz a následně jej ověřit je nutné, aby byly splněny určité podmínky:

1. $D_n < D_p$ - Proofkit musí být vytvořen po datu narození P. Není možné dokázat věk někoho, kdo se ještě nenarodil
2. $D_p \leq D_d$ - důkaz nemůže být vytvořen dříve, než je vytvořen Proofkit. Může být nicméně vytvořen ve stejný den
3. (klasický) $D_n \geq D_x$ - datum narození musí být větší než nebo roven prokazovanému datumu. Toto implikuje splnění podmínky $s \geq t$ a možnost vytvoření důkazu
4. (opačný) $D_n \leq D_x$ - datum narození musí být menší než nebo roven prokazovanému datumu. Toto implikuje splnění podmínky $s \leq t$ a možnost vytvoření důkazu
5. (opačný) $D_m > D_*$ - Aby bylo možno výpočet provést, musí být maximální datum největší ze všech použitých datumů

[2)] Ačkoli by slovo „datum“ mělo být správně skloňováno jako 2.pád „data“, 3.pád „datu“ apod., je v rámci této kapitoly použito alternativní skloňování, které nevypouští slabiku „um“, tedy 2.pád „datumu“, 3.pád „datumu“ atd. Důvodem je rozlišení, zda je mluveno o „datumech“ či o „datech“ (tj. údajích) obecně. Internetová jazyková příručka [58] toto skloňování (právě v tomto případě) umožňuje.

25.1.2 Pomocné proměnné

Ke zjednodušení popisu vzájemného vztahu datumů je možno definovat následující pomocné proměnné:

- $V_x = D_d - D_x$ - věk nutný ke splnění podmínky. Tato hodnota je zpravidla zadána a slouží k výpočtu D_x
- $V_p = D_p - D_n$ - věk P při vystavení Proofkitu
- $V_d = D_d - D_n$ - věk P při tvorbě důkazu
- $O_p = D_d - D_p$ - stáří Proofkitu
- $O_m = D_m - D_p$ - počet dní mezi maximálním datem a datem proofkitu, slouží jako proměnná m

Proměnné s a t je možno vyjádřit následujícími rovnicemi:

$$\begin{aligned} s &= D_p - D_n \\ s &= V_p \\ \\ t &= (D_d - D_x) - (D_d - D_p) \\ t &= D_p - D_x \end{aligned} \tag{25.1}$$

Jako proměnná s slouží věk P v okamžiku vystavení Proofkitu. Proměnnou t je možno chápat jako věk nutný ke splněný podmínky ponížený o stáří Proofkitu. Tento vztah vyjadřuje následující myšlenku: dnešní důkaz, že $s \geq t$, je ekvivaletní s důkazem, že před x časovými jednotkami bylo $s \geq t - x$.

Vzájemné vztahy mezi datumy přehledně zachycuje obrázek 25.1 (klasická verze), resp. 25.2 (opačná verze).

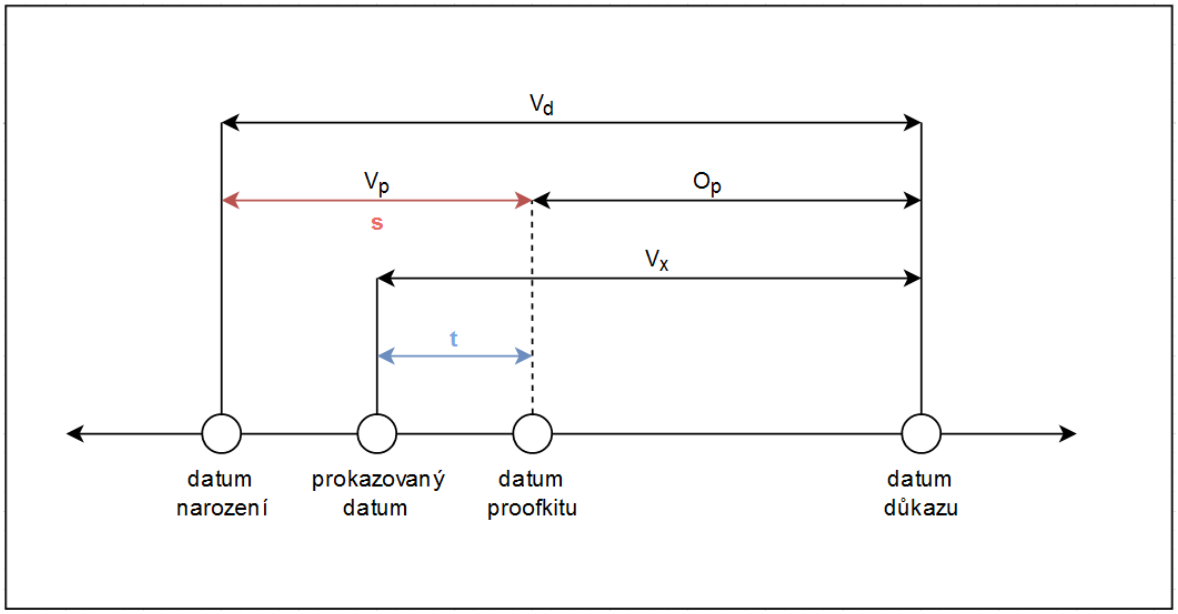
25.2 Klasický důkaz věku

Protokol se, podobně jako předchozí modely, skládá ze tří fází.

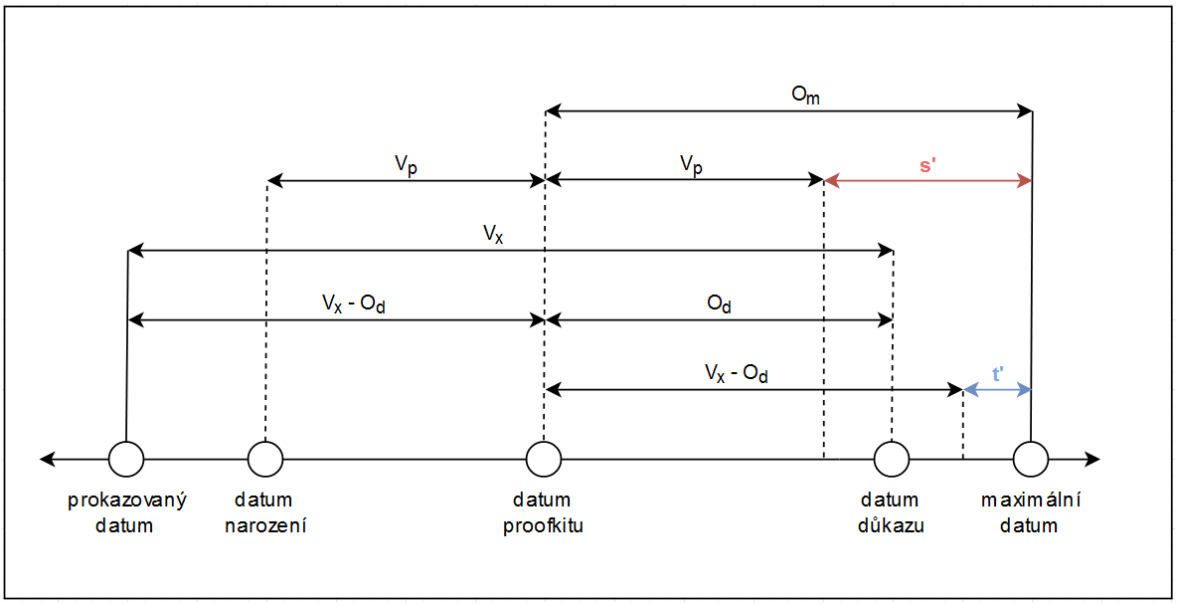
25.2.1 Fáze 1

Výpočet provádí T. Ten zná datum proofkitu D_p a D_n, tedy datum narození P.

Obr. 25.1 Vzájemné vztahy mezi datumy (klasický důkaz věku)



Obr. 25.2 Vzájemné vztahy mezi datumy (opačný důkaz věku)



$$
\begin{aligned}
r &= random() \\
base &= H^{1}(r) \\
\\
target &= H^{D_p - D_n + 1}(r) \\
target &= H^{V_p + 1}(r)
\end{aligned}
\tag{25.2}
$$

T poté elektronicky podepíše hodnoty *target* a D_p, a tyto hodnoty pošle P spolu s článkem hash chainu *base* .

25.2.2 Fáze 2

Výpočet provádí P. Ten zná datum proofkitu D_p, datum narození D_n, datum důkazu D_d i prokazovaný datum D_x.

$$\begin{aligned} proof &= H^{(D_p-D_n)-(D_d-D_x-(D_d-D_p))}(base) \\ proof &= H^{(D_p-D_n)-(D_p-D_x)}(base) \\ proof &= H^{V_p-(D_p-D_x)}(base) \end{aligned} \tag{25.3}$$

P poté pošle V hodnoty *proof* a elektronicky podepsané *target* a D_p, které mu dodal T.

25.2.3 Fáze 3

Výpočet provádí V. Ten zná datum důkazu D_d i prokazovaný datum D_x.

$$\begin{aligned} check &= H^{D_d-D_x-(D_d-D_p)}(proof) \\ check &= H^{D_p-D_x}(proof) \end{aligned} \tag{25.4}$$

V poté srovná *check* a *target* a ověří elektronický podpis T. Pokud je vše v pořádku, je důkaz platný.

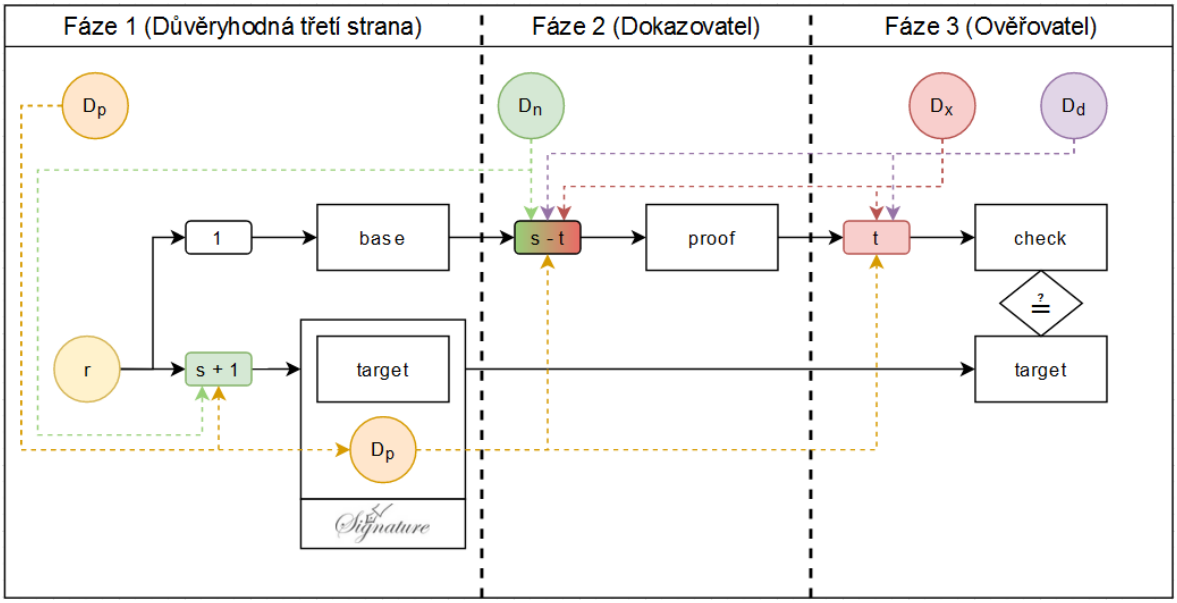
25.2.4 Důkaz platnosti protokolu

Následuje důkaz platnosti protokolu:

$$\begin{aligned} target &= check \\ H^{D_p-D_n+1}(r) &= H^{D_d-D_x-(D_d-D_p)}(proof) \\ H^{D_p-D_n+1}(r) &= H^{D_p-D_x}(proof) \\ H^{D_p-D_n+1}(r) &= H^{D_p-D_x}(H^{(D_p-D_n)-(D_d-D_x-(D_d-D_p))}(base)) \\ H^{D_p-D_n+1}(r) &= H^{D_p-D_x}(H^{D_x-D_n}(base)) \\ H^{D_p-D_n+1}(r) &= H^{D_p-D_x}(H^{D_x-D_n}(H^1(r))) \\ H^{D_p-D_n+1}(r) &= H^{D_p-D_x+D_x-D_n+1}(r))) \\ H^{D_p-D_n+1}(r) &= H^{D_p-D_n+1}(r))) \end{aligned} \tag{25.5}$$

Obrázek 25.3 ukazuje vztah použitých proměnných a článků hash chainu s označenými stranami komunikace.

Obr. 25.3 Vztah použitých proměnných a článků hash chainu (klasický důkaz věku)



25.2.5 Příklad

P se narodil 7.6.2003 (D_n). Důvěryhodná třetí strana mu vystavila Proofkit dne 15.9.2018 (D_p). V té době mu bylo 15 let. P chce tento důkaz použít k tomu, aby dne 27.2.2022 (D_v) prokázal, že mu již bylo 18 let, jinými slovy, že se narodil dříve, než 27.2.2004 (D_x). Výpočet proběhne následovně.

1. T vygeneruje náhodný r
2. P spočítá $base = H^1(r) = 000001|f1893906$
3. P spočítá $target = H^{D_p - D_n + 1}(r) = H^{5580}(r) = 005580|975401c1$
4. P spočítá $proof = H^{(D_p - D_n) - (D_d - D_x - (D_d - D_p))}(base) = H^{265}(000001|f1893906) = 000266|42fce72c$
5. V spočítá $check = H^{D_p - D_x}(proof) = H^{5314}(000266|42fce72c) = 005580|975401c1$
6. V ověří, že $check = target$

Pokud se obě hodnoty rovnají, a elektronický podpis je možno ověřit, pak je důkaz platný a V je přesvědčený,že P je skutečně starší, než 18 let.

25.3 Opačný důkaz věku

I zde je nutno pro výpočet opačného protokolu spočítat proměnné s' a t'. Ty je možno vyjádřit následovně:

$$\begin{aligned} s' &= (D_m - D_p) - (D_p - D_n) \\ s' &= O_m - V_p \\ \\ t' &= O_m - ((D_d - D_x) - (D_d - D_p)) \\ t' &= D_m - D_p + D_x - D_p \end{aligned} \tag{25.6}$$

Výpočet probíhá klasicky ve třech fázích.

25.3.1 Fáze 1

Výpočet provádí T. Ten zná datum proofkitu D_p, datum narození P (D_n) a určí maximální datum D_m .

$$\begin{aligned} r &= random() \\ base &= H^1(r) \\ \\ target &= H^{(D_m - D_p) - (D_p - D_n) + 1}(r) \\ target &= H^{O_m - V_p + 1}(r) \end{aligned} \tag{25.7}$$

T poté elektronicky podepíše hodnoty $target$ a D_p a D_m, a tyto pošle P spolu s článkem hash chainu $base$.

25.3.2 Fáze 2

Výpočet provádí P. Ten zná datum proofkitu D_p, datum narození D_n, datum důkazu D_d, maximální datum D_m i prokazovaný datum D_x.

$$\begin{aligned} proof &= H^{((D_m - D_p) - (D_p - D_n)) - ((D_m - D_p) - (D_p - D_x)))}(base) \\ proof &= H^{(O_m - V_p) - (O_m - (D_p - D_x)))}(base) \end{aligned} \tag{25.8}$$

P poté odešle V hodnoty $proof$ a dalé elektronicky podepsané hodnoty, které obdržel od T ($target$ D_m a D_p).

25.3.3 Fáze 3

Výpočet provádí V. Ten zná datum důkazu D_d i prokazovaný datum D_x.

$$\begin{aligned} check &= H^{(D_m - D_p) - (D_p - D_x)}(proof) \\ check &= H^{O_m - (D_p - D_x)}(proof) \end{aligned} \tag{25.9}$$

V poté srovná hodnoty *check* a *target* a ověří elektronický podpis T. Pokud je vše v pořádku, je důkaz platný.

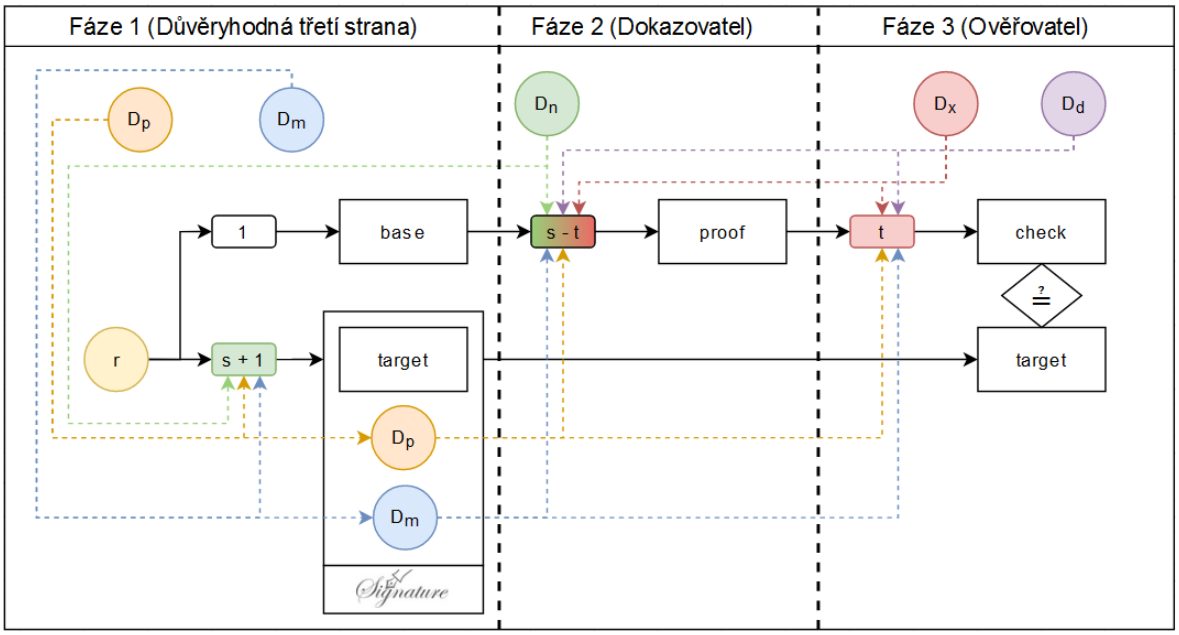
25.3.4 Důkaz platnosti protokolu

Následuje důkaz platnosti protokolu:

$$\begin{aligned} target &= check \\ H^{O_m-V_p+1}(r) &= H^{O_m-((D_d-D_x)-(D_d-D_p))}(proof) \\ H^{O_m-V_p+1}(r) &= H^{D_m-D_p+D_x-D_p}(H^{(O_m-V_p)-(O_m-((D_d-D_x)-(D_d-D_p)))}(base)) \\ H^{O_m-V_p+1}(r) &= H^{D_m-D_p+D_x-D_p}(H^{O_m-V_p+D_p-D_m+D_p-D_x}(H^1(r))) \\ H^{O_m-V_p+1}(r) &= H^{D_m-D_p+D_x-D_p+O_m-V_p+D_p-D_m+D_p-D_x+1}(r))) \\ H^{O_m-V_p+1}(r) &= H^{O_m-V_p+1}(r) \end{aligned} \tag{25.10}$$

Obrázek 25.4 ukazuje vztah použitých proměnných a článků hash chainu s označenými stranami komunikace.

Obr. 25.4 Vztah použitých proměnných a článků hash chainu (opačný důkaz věku)



25.3.5 Příklad

P se narodil 7.6.2003 (D_n). Důvěryhodná třetí strana mu vystavila Proofkit dne 15.9.2018 (D_p). V té době mu bylo 15 let. P chce tento důkaz použít k tomu, aby dne 27.2.2022 (D_v) prokázal, že mu ještě nebylo 35 let, jinými slovy, že se nenarodil dříve, než

27.2.1987 (D_x). Jako D_m bude použito datum 15.9.2118[3)]. Výpočet proběhne následovně:

1. T vygeneruje náhodný r
2. P spočítá $base = H^1(r) = 000001|faf623f6$
3. P spočítá $target = H^{D_p - D_n + 1}(r) = H^{30946}(r) = 030946|2c298bca$
4. P spočítá $proof = H^{(D_p - D_n) - (D_d - D_x - (D_d - D_p))}(base) = H^{5944}(000001|faf623f6) = 005945|d746827a$
5. V spočítá $check = H^{D_p - D_x}(proof) = H^{25002}(005945|d746827a) = 030946|2c298bca$
6. V ověří, že $check = target$

Pokud se obě hodnoty rovnají, a elektronický podpis je možno ověřit, pak je důkaz platný a V je přesvědčený,že P je skutečně mladší, než 35 let.

26 Vzorová implementace

Cílem vzorové implementace je demonstrovat možné použití hash chainů jako důkaz věku v kontextu elektronickém dokladu, a to od vystavení daného dokladu Důvěryhodnou třetí stranou až po ověření důkazu.

Implementace myšlenkově vychází z rámce Self-sovereign identity, který je popsán v kapitole 13.5.2.

26.1 Rozsah práce

Demonstrace je omezena pouze na předání informace o dosaženém věku P. Prokázání totožnosti (aby V věděl, že jde skutečně o doklad P) je mimo kontext této demonstrace, stejně tak je mimo kontext demonstrace předání veřejného klíče T, který může být použit k ověření integrity a autenticity důkazu.

Implementace vychází z předpokladu, že tyto kroky, stejně jako zabezpečení komunikace a další nezbytné záležitosti jsou vyřešeny v rámci „Systému elektronických dokladů“, do kterého je tento protokol zasazen.

26.2 Datum proofkitu

Za normálních okolností je datum vystavení dokladu také datumem proofkitu. Jedinou limitující podmínkou pro datum proofkitu je, že $D_p \leq D_x$ - není možno prokázat, že je

[3)] Vysvětlení výběru datumu viz kapitola 26.3

věk větší, než x let, dokladem, který je $x + n$ let starý. Předložením tohto dokladu se V dozví, že je P starý nejméně $x + n$ let, i když se má dozvědět pouze informaci, že je starší než x let, což porušuje Zero-knowledge vlastnost, nešlo by tedy o Zero-knowledge protokol[1)].

Situaci je možno vyřešit stanovením hodnoty k. Ta značí minimální věk, který má být možno tímto Proofkitem prokázat. Při vystavení dokladu jsou připraveny Proofkity s datumy $D_p + nk, 0 \leq n \leq z$, kde je z určeno s ohledem na požadavky na dobu očekávaného používání (například na platnost dokladu) Proofkitu.

Tato vlastnost se týká pouze klasických důkazů, nikoli opačných.

Má-li být například vystaven doklad s platností na 20 let, tak aby bylo možno kdykoli v době platnosti dokladu prokázat minimální věk 5 let, budou vystaveny celkem 4 Proofkity:

1. Proofkit 1 (D_p^1) - datum vystavení
2. Proofkit 2 (D_p^2) - datum vystavení + 5 let
3. Proofkit 3 (D_p^3) - datum vystavení + 10 let
4. Proofkit 4 (D_p^4) - datum vystavení + 15 let

Při prokázání věku pak použijeme Proofkit s největším datumem, který splňuje podmínku $D_p < D_d$.

26.3 maximální datum

Při určení maximálního datumu, který bude v celém systému použit, musí být přihlédnuto k následujícím podmínkám:

1. $t < m$
2. $s < m$
3. m musí být co nejmenší, neboť výpočetní náročnost je závislá na velikosti m
4. hodnota m nesmí být jakkoli závislá na věku či datumu narození P. Z pohledu V nesmí m prozradit ani napovědět žádnou informaci navíc, a to i přes to, že ji V zná

[1)] V praktickém nasazení v kontextu elektronických dokladů by tento problém byl pravděpodobně ignorován, neboť platnost vystavovaných dokladů (zpravidla 5 či 10 let) bývá většinou mnohem kratší, než věk, který se v praxi dokazuje (zpravidla více než 15 let).

5. hodnota musí být univerzální (tj. použitelná pro všechny kombinace s, t, pro které bude moci být v rámci daného systému teoreticky použita)

Z podmínky 1 vyplývá, že maximální datum musí být větší, než je největší věk, který může být v rámci systému porovnáván. V reálném světě existují situace, kdy musí starší občané prokázat svůj důchodový věk, systém tedy musí počítat s touto situací.

Z podmínky 2 vyplývá, že maximální datum musí být větší, než je nejvyšší možný věk P. Aby systém mohli používat starší spoluobčané, musí být k tomuto přihlédnuto.

Po zvážení výše zmíněných podmínek byl vzorec pro výpočet maximálního datumu určen jako $D_m = D_p + 100$ let[2)].

26.4 Datový model

V rámci praktické ukazky je použit následující datový model pro přenos dat mezi zúčastněnými stranami. Všechny zprávy jsou ve formátu JSON a obsahují nejmenší nutné množství dat k výpočtu důkazu.

Elektronické podpisy v příkladech jsou z důvodu přehlednosti zkráceny.

Obecně vzato struktura Toolbox obsahuje jeden nebo více (zpravidla dva) Proofkitů, každý Proofkit je pak možno využít k sestavení libovolného množství důkazů Proof.

26.4.1 Struktura Public

Část datového modelu označená jako Public obsahuje data, která tvoří T, a která jsou určená pro předání V. Jde o část, která je elektronicky podepisována. Obsahuje následující data:

- prooftype - typ důkazu (možné hodnoty jsou "ge"pro klasický důkaz a "l"pro opačný důkaz)
- maxdate - při klasickém důkazu *null*, při opačném důkazu obsahuje maximální datum D_m ve formátu "den/měsíc/rok"
- proofdate - datum proofkitu D_p ve formátu "den/měsíc/rok"
- target - článek hash chainu *target*

Příklad:

[2)] V praxi by bylo pravděpodobně nutné zvolit číslo ještě větší (za těchto okolností nebudou moci lidé starší, než 100 let, prokázat, že jim ještě nebylo 101 let atd.)

```
            "public": {
                "prooftype": "ge",
                "maxdate": null,
                "proofdate": "15/9/2018",
                "target": "005580|b6607e4b"
            }
```

26.4.2 Struktura Proof

Strukturu Proof posílá P jako podklady pro ověření věku. Jeden Proof obsahuje data potřebná k ověření jednoho důkazu. Proto je struktura Proof obalena strukturou "computedproofs", která obsahuje jednu nebo dvě části "Proof".

Tato část obsahuje data:

- algorithm - použitá hashovací funkce
- issuer - nepovinná identifikace T
- proof - článek hash chainu *proof*
- verificationdate - datum důkazu D_d ve formátu "den/měsíc/rok"
- agetoprove - pole dat, které obsahuje informace o požadovaném věku, k jehož prokázání je možno tento důkaz použít, ve formátu "[rok,měsíc,den]"
- signature - elektronický podpis struktury Public ve formátu base64

Příklad:

```
{
    "computedproofs": [
        {
            "algorithm": "ilustrative_hash",
            "issuer": "",
            "proof": "000266|1a9a3b54",
            "verificationdate": "27/2/2022",
            "agetoprove": [
                18,
                0,
                0
            ],
            "public": { ... },
            "signature": "dwYr59xGI5...sdR0uAJOk="
        },
        { ... }
    ]
}
```

26.4.3 Struktura Proofkit

Proofkit jsou data, která tvoří T a zasílá je P a která slouží k vytvoření důkazu. Struktura je obalena strukturou "Toolbox", která může obsahovat více (zpravidla 2) Proofkitů.

- proofid - identifikátor Proofkitu v rámci Toolboxu
- base - článek hash chainu *base*
- birthdate - datum narození P ve formátu "den/měsíc/rok"
- algorithm - použitá hashovací funkce
- issuer - nepovinná identifikace T
- signature - elektronický podpis struktury Public ve formátu base64

Příklad struktury Proofkit:

```
{
    "toolbox": [
        {
            "proofid": 1,
            "base": "000001|34ae1809",
            "birthdate": "7/6/2003",
            "algorithm": "ilustrative_hash",
            "issuer": "",
            "public": { ... },
            "signature": "dwYr59xGI5...sdR0uAJOk="
        },
        { ... }
    ]
}
```

26.5 Příklad komunikace

P se narodil 15.9.2018. Dne 7.6.2003 mu T vystavuje elektronický doklad, který obsahuje následující Toolbox:

```
{
    "toolbox": [
        {
            "proofid": 1,
            "base": "000001|34ae1809",
            "birthdate": "7/6/2003",
            "algorithm": "ilustrative_hash",
            "issuer": "",
            "public": {
                "prooftype": "ge",
                "maxdate": null,
                "proofdate": "15/9/2018",
                "target": "005580|b6607e4b"
            },
            "signature": "dwYr59xGI5...sdR0uAJOk="
        },
        {
            "proofid": 2,
            "base": "000001|783ba74a",
            "birthdate": "7/6/2003",
            "algorithm": "ilustrative_hash",
            "issuer": "",
```

```
            "public": {
                "prooftype": "l",
                "maxdate": "15/9/2118",
                "proofdate": "15/9/2018",
                "target": "030946|e2f1f68e"
            },
            "signature": "eW6CdGQLT2...olB2XAKg8="
        }
    ]
}
```


Dne 27.2.2002 chce P dokázat, že je mu více, než 18 let, a zároveň méně, než 35 let. Proto vygeneruje následující důkazy, které může odeslat V:


```
{
    "computedproofs": [
        {
            "algorithm": "ilustrative_hash",
            "issuer": "",
            "proof": "000266|1a9a3b54",
            "verificationdate": "27/2/2022",
            "agetoprove": [
                18,
                0,
                0
            ],
            "public": {
                "prooftype": "ge",
                "maxdate": null,
                "proofdate": "15/9/2018",
                "target": "005580|b6607e4b"
            },
            "signature": "dwYr59xGI5...sdR0uAJOk="
        },
        {
            "algorithm": "ilustrative_hash",
            "issuer": "",
            "proof": "005945|a10858a8",
            "verificationdate": "27/2/2022",
            "agetoprove": [
                35,
                0,
                0
            ],
            "public": {
                "prooftype": "l",
                "maxdate": "15/9/2118",
                "proofdate": "15/9/2018",
                "target": "030946|e2f1f68e"
            },
            "signature": "eW6CdGQLT2...olB2XAKg8="
        }
    ]
}
```


V může tyto důkazy ověřit, spolu s elektronickým podpisem T, a tím se přesvědčit, že je věk P skutečně v požadovaném rozsahu bez toho, aby se dozvěděl datum narození P.

26.6 Implementace

Součástí této akademické práce je vzorová implementace tohoto protokolu v jazyce Python. Tato implementace obsahuje mimo nástrojů potřebných k provedení výpočtů také soubory demonstration.py app.py.

Soubor demonstration.py obsahuje statickou a okomentovanou ukázku praktické demonstrace výpočtu od vystavení Toolkitu přes vytvoření důkazu až po jeho ověření.

Soubor app.py obsahuje velmi jednoduchou interaktivní verzi demonstrace sloužící pro umožnění lepšího testování a pochopení fungování protokolu.

Nutno podotknout, že ani jeden z těchto souborů nemají za cíl být plnou funkční implementací dané problematiky, ale pouze umožnit lépe pochopit fungování protokolu a poskytnout demonstraci toho, jakým způsobem je možno knihovnu reálně využít.

Ke spuštění obou souborů je nutno mít nainstalovanou Pythonovskou knihovnu Crypto.

Ačkoli je tato akademická práce psána v češtině, program včetně komentářů byl psán v angličtině.

27 Zhodnocení protokolu

Na prvních pohled není odpověď na otázku jako dokázat věk bez vyzrazení data narození zřejmá, proto je s podivem, že existuje toto jednoduché řešení. Ačkoli existují i jiné Zero-knowledge protokoly, které by mohly být v rámci elektronických dokladů použity, žádný není natolik elegantní jako využití hash chainů.

Hlavním benefitem tohoto řešení je jeho srozumitelnost - k získání náhledu fungování protokolu a pochopení jeho principu není zapotřebí pokročilá znalost kryptografických a matematických nástrojů, ale stačí pouze povědomí o kryptografických hashovacích funkcích a jejich jednosměrnosti, což je jeden ze základních stavebních kamenů moderní kryptografie, znalost existence elektronického podpisu, díky kterému může Ověřovatel důkazu věřit, a matematika na úrovni základní školy (sčítání a odčítání).

I z pohledu praktické implementace jde o jednoduchý protokol, neboť nevyžaduje interaktivní komunikaci (pouze zpracování či vytvoření dat ve formátu JSON) a instalaci základních knihoven, které umožní vytvoření (na straně Důvěryhodné třetí strany) či ověření (na straně Ověřovatele) elektronického podpisu a využití hashovacích funkcí.

Jediný bezpečnostní prvek, na který protokol spoléhá, je jednosměrnost použité hashovací funkce (resp. praktická nemožnost nalezení prvního předobrazu), což je z pohledu kryptografie jedna ze zásadních a podrobně zkoumaných vlastností těchto funkcí, proto existuje u běžně používaných hashovacích funkcí vysoká úroveň důvěry v jejich bezpečnost, z čehož vyplývá i vysoká úroveň bezpečnosti tohoto protokolu. Fakt,

že jde technicky pouze o Neinteraktivní Zero-knowledge Argument, má vliv pouze na bezpečnost v teoretické rovině, nikoli v praktické.

Protokol není postaven na žádné konkrétní hashovací funkci, proto je možné použitou funkci jednoduše změnit za jinou v případě, že je v ní objevena zásadní bezpečnostní zranitelnost.

Současná úroveň výzkumu kvantových algoritmů předpokládá, že v současnosti používané hashovací funkce budou odolné i proti útoku pomocí kvantových počítačů. Protokol by tedy měl být bezpečný i v budoucnu.

Zdánlivou nevýhodou tohoto protokolu je fakt, že vyžaduje k fungování Důvěryhodnou třetí stranu. V kontextu elektronických dokladu se ale tato strana nachází vždy (výstavce dokladů), proto to v tomto případě nehraje roli.

Obecně je tento protokol možno využít k porovnání nějaké číselné hodnoty garantované Důvěryhodnou třetí stranou s hranicí definovanou Ověřovatelem. Kromě důkazu věku to může být v kontextu elektronických dokladů využito například v situaci, kdy zaměstnavatel dokládá zaměstnanci výši mzdy pro banku či finanční společnost. V takovém případě může zaměstnanec pouze prokázat, že jeho mzda splňuje nějakou minimální požadovanou hranici bez toho, aby svou mzdu vyzradil. Případně může například banka vystavit svému klientovi potvrzení o výši zůstatku na účtu k určitému dni, pomocí kterého může tento klient v případě potřeby prokázat, že jeho zůstatek na účtu je vyšší, než požadovaná hranice atd.

ZÁVĚR

Ačkoli výzkum Zero-knowledge protokolů započal již v 80. letech 20. století, šlo spíše o teoretickou oblast a, ačkoli byly navrženy možné způsoby aplikace těchto protokolů, do praxe se moc neprosadily. Toto se změnilo zejména po roce 2010 s rozvojem Blockchainu a na něm založených krypto měn, které ke svému fungování vyžadují vlastnosti, které jim zejména neinteraktivní Zero-knowledge protokoly mohou nabídnout. Nalezení a implemetnace prvních opravdu praktických aplikací odstartovalo novou vlnu zájmu o Zero-knowledge protokoly. Začaly vznikat nové, vylepšené verze protokolů a také byly objeveny nové oblasti možného využití (například elektronické volby či elektronické doklady). V blízké budoucnosti proto můžeme očekávat další rozvoj těchto protokolů a jejich nasazení do širší praxe, ať už například v rámci nově vznikajícího celoevropského systému pro elektronické doklady nazvaného Evropská Digitální identita, kde je zvažována možnost využití těchto protokolů například pro důkaz dosaženého věku bez vyzrazení data narození.

Cílem této práce bylo vysvětlit základní principy Zero-knowledge protokolů, shrnout jejich postupný vývoje a možnosti praktické aplikace. Součástí práce je také demonstrace praktického použití vybraného interaktivního a neinteraktivního Zero-knowledge protokolu.

Práce obsahuje kromě literární rešerže a úvodu do problematiky Zero-knowledge také dvě ukázky praktického užití Zero-knowledge protokolů včetně vzorové implementace v jazyce Python.

První praktická část se věnuje demonstraci možného využití Interaktivního Zero-knowledge Důkazu Znalosti hesla k autentizaci. Představuje protokol, který vychází z Fiat-Shamirova identifikačního schématu, na rozdíl od něj je ale postaven a problematice nemožnosti výpočtu diskrétního logaritmu. V práci je demonstrováno možné rozšíření protokolu o 2 bezpečnostní prvky, prvních z nich je využití kryptografického pepře k zabezpečení přihlašovacích údajů uložených v databázi. Druhým bezpečnostním prvkem je zapojení OTP generátoru jako druhého autentizačního faktoru. Kapitola je uzavřena srovnáním tohoto protokolu s klasickým autentizačním protokolem postaveným na hashovacích funkcích.

Druhá praktická část je věnována výše zmíněnému důkazu dosaženého věku bez vyzrazení data narození v kontextu elektronických dokladů. Využívá k tomu Neinteraktivní Zero-knowledge Argument Znalosti. Protokol je postavený na mnohonásobném využití klasické hashovací funkce, čímž vznikne takzvaný hash chain - "řetěz hashů", a využívá Důvěryhodné třetí strany - výstavce dokladu. V práci je demonstrován postup vystavení dokladu a tvorby a ověření důkazu, které v tomto kontextu zpravidla probíhá jiný den, a je vysvětleno proč je možno dokázat věkovou hranici, kterou dokazující

strana splňuje nyní, ale nesplňovala v okamžiku vystavení dokladu. Součástí kapitoly je také zjednodušená verze protokolu, která může být použita v případě, že je dokazovaná hodnota statická (resp. když je důkaz stavěn k okamžiku vydání dokladu), což může být v praxi použito například pro zaměstnavatelem vystavený důkaz velikosti mzdy, kterým může být například bankovní instituci dokázáno dosažení požadované úrovně mzdy při sjednání úvěru.

SEZNAM POUŽITÉ LITERATURY

[1] KRYPTOS, Kostas. Demonstrate how Zero-Knowledge Proofs work without using math. *LinkedIn* [online]. [cit. 2022-05-02]. Dostupné z: https://www.linkedin.com/pulse/demonstrate-how-zero-knowledge-proofs-work-without-using-chalkias

[2] QUISQUATER, Jean-Jacques a Tom BERSON. *How to Explain Zero-Knowledge Protocols to Your Children* [online]. 1998 [cit. 2022-05-02]. Dostupné z: https://pages.cs.wisc.edu/~mkowalcz/628.pdf

[3] GOLDWASSER, Shafi, Silvio MICALI a Charles RACKOFF. *THE KNOWLEDGE COMPLEXITY OF INTERACTIVE PROOF SYSTEMS* [online]. 1989 [cit. 2022-05-02]. Dostupné z: http://crypto.cs.mcgill.ca/~crepeau/COMP647/2007/TOPIC02/GMR89.pdf

[4] PINKAS, Benny. *Zero Knowledge for Sudoku - Benny Pinkas* [záznam přednášky] [online]. 2019 [cit. 2022-05-02]. Dostupné z: https://www.youtube.com/watch?v=_tGDoys_w5c

[5] GRADWOHL, Ronen, Moni NAOR, Benny PINKAS a Guy N. ROTHBLUM. *Cryptographic and Physical Zero-Knowledge Proof Systems for Solutions of Sudoku Puzzles* [online]. 2009 [cit. 2022-05-02]. Dostupné z: https://www.wisdom.weizmann.ac.il/~naor/PAPERS/sudoku.pdf

[6] TURING, Alan Mathison. *ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM* [online]. 1936 [cit. 2022-05-02]. Dostupné z: https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf

[7] Computational Complexity Theory. *Standford Encyclopedia of Philosophy* [online]. [cit. 2022-05-02]. Dostupné z: https://plato.stanford.edu/entries/computational-complexity/

[8] Time complexity. *Wikipedia: the free encyclopedia* [online]. [cit. 2022-05-02]. Dostupné z: https://en.wikipedia.org/wiki/Time_complexity

[9] P (complexity). *Wikipedia: the free encyclopedia* [online]. [cit. 2022-05-02]. Dostupné z: https://en.wikipedia.org/wiki/P_(complexity)

[10] AGRAWAL, Manindra, Neeraj KAYAL a Nitin SAXENA. *PRIMES is in P* [online]. [cit. 2022-05-02]. Dostupné z: https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf

[11] BELLARE, Mihir a Phillip ROGAWAY. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols* [online]. 1993 [cit. 2022-05-02]. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/168588.168596

[12] CANETTI, Ran, Oded GOLDREICH a Shai HALEVI. *The Random Oracle Methodology, Revisited* [online]. 2002 [cit. 2022-05-02]. Dostupné z: https://eprint.iacr.org/1998/011.pdf

[13] CANETTI, Ran a Marc FISCHLIN. *Universally Composable Commitments* [online]. 2001 [cit. 2022-05-02]. Dostupné z: https://eprint.iacr.org/2001/055.pdf

[14] PEDERSEN, Torben Pryds. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing* [online]. 1992 [cit. 2022-05-02]. Dostupné z: https://link.springer.com/content/pdf/10.1007/3-540-46766-1_9.pdf

[15] ROSEN, Alon. *Introduction to Zero Knowledge - Alon Rosen* [záznam přednášky] [online]. 2019 [cit. 2022-05-02]. Dostupné z: https://www.youtube.com/watch?v=6uGimDYZPMw

[16] FEIGE, Uriel a Adi SHAMIR. *Witness Indistinguishable and Witness Hiding Protocols* [online]. 1990 [cit. 2022-05-02]. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/100216.100272

[17] BABAI, László. *Trading Group Theory for Randomnes* [online]. 1985 [cit. 2022-05-02]. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/22145.22192

[18] GOLDWASSER, Shafi a Michael SIPSER. *Private Coins versus Public Coins in Interactive Proofs Systems* [online]. 1986 [cit. 2022-05-02]. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/12130.12137

[19] GOLDREICH, Oded, Silvio MICALI a Avi WIDGERSON. *Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems* [online]. 1991 [cit. 2022-05-02]. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/116825.116852

[20] BOOTLE, Jonathan, Andrea CERULLI, Pyrros CHAIDOS, Jens GROTH a Christophe PETIT. *Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting* [online]. 2016 [cit. 2022-05-02]. Dostupné z: https://eprint.iacr.org/2016/263.pdf

[21] BRASSARD, Gilles, David CHAUM a Claude CRÉPEAU. *Minimum Disclosure Proofs of Knowledge* [online]. 1988 [cit. 2022-05-02]. Dostupné z: https://www.chaum.com/publications/Minimum_Disclosure_Proofs_of_Knowledge.pdf

[22] FEIGE, Uriel, Amos FIAT a Adi SHAMIR. *Zero Knowledge Proofs of Identity* [online]. 1987 [cit. 2022-05-02]. Dostupné z: https://www.researchgate.net/publication/221591581_Zero_Knowledge_Proofs_of_Identity

[23] BELLARE, Mihir a Oded GOLDREICH. *On Defining Proofs of Knowledge* [online]. 1993 [cit. 2022-05-02]. Dostupné z: https://link.springer.com/content/pdf/10.1007/3-540-48071-4_28.pdf

[24] SCHNORR, Claus-Peter. *EFFICIENT IDENTIFICATION AND SIGNATURES FOR SMART CARDS* [online]. 1990 [cit. 2022-05-02]. Dostupné z: https://link.springer.com/content/pdf/10.1007/0-387-34805-0_22.pdf

[25] KOGAN, Dima. *Lecture 5: Proofs of Knowledge, Schnorr's protocol, NIZK* [online]. [cit. 2022-05-02]. Dostupné z: https://crypto.stanford.edu/cs355/19sp/lec5.pdf

[26] FIAT, Amos a Adi SHAMIR. *How To Prove Yourself: Practical Solutions to Identification and Signature Problems* [online]. 1987 [cit. 2022-05-02]. Dostupné z: https://link.springer.com/content/pdf/10.1007%2F3-540-47721-7_12.pdf

[27] POINTCHEVAL, David a Jacques STERN. *Security Proofs for Signature Schemes* [online].1996 [cit. 2022-05-02]. Dostupné z: https://link.springer.com/content/pdf/10.1007/3-540-68339-9_33.pdf

[28] Fiat-Shamir transformation. *ZKDocs* [online]. [cit. 2022-05-02]. Dostupné z: https://www.zkdocs.com/docs/zkdocs/protocol-primitives/fiat-shamir/

[29] BERNHARD, David, Olivier PEREIRA a Bogdan WARINSCHI. *How not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios* [online]. 2016 [cit. 2022-05-02]. Dostupné z: https://eprint.iacr.org/2016/771.pdf

[30] BLUM, Manuel a Paul FELDMAN. *Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)* [online]. 1988 [cit. 2022-05-02]. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/62212.62222

[31] FEIGE, Uriel, Dror LAPIDOT a Adi SHAMIR. *MULTIPLE NON-INTERACTIVE ZERO KNOWLEDGE PROOFS UNDER GENERAL ASSUMPTIONS* [online]. 1999 [cit. 2022-05-02]. Dostupné z: https://inst.eecs.berkeley.edu/~cs276/fa20/notes/Multiple%20NIZK%20from%20general%20assumptions.pdf

[32] BITANSKY, Nir, Ran CANETTI, Alessandro CHIESA a Eran TROMER. *From Extractable Collision Resistance to Succint Non-Interactive Arguments of Knowledge, and Back Again* [online]. 2011 [cit. 2022-05-02]. Dostupné z: `https://eprint.iacr.org/2011/443.pdf`

[33] BÜNZ, Benedikt, Jonathan BOOTLE, Dan BONEH, Andrew POELSTRA, Pieter WUILLE a Greg MAXWELL. *Bulletproofs: Short Proofs for Confidental Transactions and More* [online]. 2017 [cit. 2022-05-02]. Dostupné z: `https://eprint.iacr.org/2017/1066.pdf`

[34] BEN-SASSON, Eli, Iddo BENTOV, Yinon HORESH a Michael RIABZEV. *Scalable, transparent, and post-quantum secure computational integrity* [online]. 2018 [cit. 2022-05-02]. Dostupné z: `https://eprint.iacr.org/2018/046.pdf`

[35] IEEE Standard Specifications for Password-Based Public-Key Cryptographic Techniques. *IEEE.org* [online]. [cit. 2022-05-02]. `https://webpages.charlotte.edu/yonwang/1363.2-2008.pdf`

[36] BELLOVIN, Steven M. a Michael MERRITT. *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks* [online]. 1992 [cit. 2022-05-02]. Dostupné z: `https://www.cs.columbia.edu/~smb/papers/neke.pdf`

[37] Zerocoin. *Zerocoin Project* [online]. [cit. 2022-05-02]. Dostupné z: `https://zerocoin.org/`

[38] How Zerocash Works. *Zerocash* [online]. [cit. 2022-05-02]. Dostupné z: `http://zerocash-project.org/how_zerocash_works`

[39] Monero. *Wikipedia: the free encyclopedia* [online]. [cit. 2022-05-02]. Dostupné z: `https://en.wikipedia.org/wiki/Monero`

[40] SALAHELDINE, Omar. Zero Knowledge Proof: Basics & Applications. *Medium* [online]. [cit. 2022-05-02]. Dostupné z: `https://medium.com/systems-and-network-security/zero-knowledge-proof-basics-applications-f945a9b30000`

[41] BANERJEE, Aritra. *A Fully Anonymous e-Voting Protocol Employing Universal zk-SNARKs and Smart Contracts* [online]. 2021 [cit. 2022-05-02]. Dostupné z: `https://eprint.iacr.org/2021/877.pdf`

[42] MURTAZA, Malik Hamza, Zahoor Ahmed ALIZAI a Zubair IQBAL. *Blockchain Based Anonymous Voting System Using zkSNARKs* [online]. 2019 [cit. 2022-05-02]. Dostupné z: `https://ieeexplore.ieee.org/abstract/document/8853836`

[43] Návrh NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY, kterým se mění nařízení (EU) č. 910/2014, pokud jde o zřízení rámce pro evropskou digitální identitu. *EUR-Lex: Access to European Union law* [online]. [cit. 2022-05-02]. Dostupné z: `https://eur-lex.europa.eu/legal-content/CS/TXT/HTML/?uri=CELEX:52021PC0281&from=EN`

[44] Verifiable Credentials Data Model. *W3.org* [online]. [cit. 2022-05-02]. Dostupné z: `https://www.w3.org/TR/vc-data-model`

[45] Modulární aritmetika, Malá Fermatova věta. *České vysoké učení technické v Praze* [online]. [cit. 2022-05-02]. Dostupné z: `https://zolotarev.fd.cvut.cz/static/mag/mag-2014-04-handouts.pdf`

[46] Carmichaelova funkce. *Wikipedia: the free encyclopedia* [online]. [cit. 2022-05-02]. Dostupné z: `https://cs.wikipedia.org/wiki/Carmichaelova_funkce`

[47] Digital Identity Guidelines. *National Institute of Standards and Technology* [online]. [cit. 2022-05-02]. Dostupné z: `https://pages.nist.gov/800-63-3/sp800-63b.html`

[48] BELLARE, Mihir, Ran CANNETI a Hugo KRAWCZYK. *Keying Hash Functions for Message Authentication* [online]. 1996 [cit. 2022-05-02]. Dostupné z: `https://link.springer.com/content/pdf/10.1007/3-540-68697-5_1.pdf`

[49] HMAC: Keyed-Hashing for Message Authentication. *RFC Editor* [online]. [cit. 2022-05-02]. Dostupné z: `https://www.rfc-editor.org/rfc/rfc2104`

[50] HOTP: An HMAC-Based One-Time Password Algorithm. *RFC Editor* [online]. [cit. 2022-05-02]. Dostupné z: `https://www.rfc-editor.org/rfc/rfc4226`

[51] TOTP: Time-Based One-Time Password Algorithm. *RFC Editor* [online]. [cit. 2022-05-02]. Dostupné z: `https://www.rfc-editor.org/rfc/rfc6238`

[52] SHAMIR, Adi a Markus STADLER. *Proof Systems for General Statements about Discrete Logarithms* [online]. 1997 [cit. 2022-05-02]. Dostupné z: `https://crypto.ethz.ch/publications/files/CamSta97b.pdf`

[53] Minimální požadavky na kryptografické algoritmy. *Národní úřad pro kybernetickou a informační bezpečnost* [online]. 2018 [cit. 2022-05-02]. Dostupné z: `https://www.govcert.cz/download/doporuceni/Kryptograficke_prostredky_doporuceni_v1.0.pdf`

[54] Hash Chain. *SpringerLink* [online]. [cit. 2022-05-02]. Dostupné z: `https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_780`

[55] Hash chain. *National Institute of Standards and Technology* [online]. [cit. 2022-05-02]. Dostupné z: https://csrc.nist.gov/glossary/term/Hash_chain

[56] ANGEL, Sebastian a Michael WALFISH. *Verifiable Auctions for Online Ad Exchanges* [online]. [cit. 2022-05-02]. Dostupné z: https://cs.nyu.edu/~mwalfish/papers/vex-sigcomm13.pdf

[57] Zero Knowledge Proof of Age Using Hash Chains. *Stratumn: The SaaS solution for your financial processes* [online]. [cit. 2022-05-02]. Dostupné z: https://www.stratumn.com/en/blog/zero-knowledge-proof-of-age-using-hash-chains/

[58] Skloňování jmen středního rodu zakončených na -um, -on, -ma. *Internetová jazyková příručka* [online]. [cit. 2022-05-02]. Dostupné z: https://prirucka.ujc.cas.cz/?id=263

SEZNAM POUŽITÝCH ZKRATEK

SEZNAM POUŽITÝCH TERMÍNŮ A PŘEKLADŮ

Anglický termín	Český překlad
částečné zrušení závazku	partial decommitment
čestný	honest
dokazovatel	prover
důkaz	proof
důkaz znalosti	proof of knowledge
důvěryhodná třetí strana	trusted party
efektivita	efectivity
extrahovatelné	extractable
extraktor znalosti	knowledge extractor
chyba znalosti	knowledge error
interaktivní protokol	interactive protocol
interaktivní systémy důkazu	Interactive proof systems
náhodné orákulum	random oracle
neinteraktivní protokoly	non-interactive protocol
nerozlišitelnost	indistinguishability
odpověd	response
ověřovatel	verifier
podvádějící	cheating
skrytí	hiding
solidnost	soundness
soukromá mince	private coin
společný refrerenční řetězec	common reference string
statistická	statistical
stručný neinteraktivní	succinct non-interactive
škálovatelný transparentn	scalable transparent
úplnost	completeness
veřejná mince	public coin
výpočetní	computational
výzva	challenge
zavázání	binding
závazek	commitment
zavazující se strana	committer
zero-knowledge	zero-knowledge
zero-knowledge důkaz	zero-knowledge proof
zrušení závazku	decommitment

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH