ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kalivoda** Jméno: **Jan** Osobní číslo: **484837**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav:

Studijní program: **Informatika**

Studijní obor: **Bezpečnost a informační technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Bezpečnostní zranitelnosti v Lightning Network

Název bakalářské práce anglicky:

Security Vulnerabilities in the Lightning Network

Pokyny pro vypracování:

1) Popište architekturu Bitcoinu, vysvětlete, v čem spočívá problém jeho škálovatelnosti.
2) Seznamte se s Lightning Network, popište jeho účel, východiska a řešení.
3) Nastudujte známé zranitelnosti v Lightning Network. Popište jejich příčiny, dopady a možná řešení.
4) Zvolte vhodnou zranitelnost a demonstrujte ji v testovacím prostředí.
5) Vyhodnoťte průběh a výsledky svého experimentu, dejte doporučení uživatelům sítě, jak se co nejefektivněji bránit.

Seznam doporučené literatury:

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Josef Kokeš katedra informační bezpečnosti FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **02.02.2022** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: _____

Ing. Josef Kokeš podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	doc. RNDr. Ing. Marcel Jiřina, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Bezpečnostní zranitelnosti v Lightning Network

Jan Kalivoda

Katedra počítačových systémů
Vedoucí práce: Ing. Josef Kokeš

8. května 2022

Poděkování

Děkuji svému vedoucímu práce Ing. Josefu Kokešovi za konzultace k práci a čajovému listí za bdělost při psaní.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2022 .....

Odkaz na tuto práci

Kalivoda, Jan. *Bezpečnostní zranitelnosti v Lightning Network*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Obsahem této práce je analýza protokolu Lightning Network, sloužícího k provádění bitcoinových plateb mimo blockchain (off-chain) rychlým a laciným způsobem. Jedná se o jedno z možných řešení tzv. problému škálovatelnosti Bitcoinu. V případě, že se rozhodneme provádět platby mimo blockchain, ztratíme tím mnoho jeho výhod jako je například solidní řešení dvojitého utracení týž prostředků (double-spending). Bezpečnost protokolu je hlavní náplní této práce, která mimo jiné představuje možné zranitelnosti protokolu a jejich zneužití, které je v nějakých případech i prakticky znázorněno. V neposlední řadě jsou probrány návrhy na opravu zranitelností či alespoň doporučení pro uživatele jak snížit pravděpodobnost jejich zneužití.

Účelem práce je pak rozšířit povědomí o Lightning Network a to především o jeho bezpečnostních aspektech, aby když se uživatelé rozhodnou ho aktivně využívat, tak aby minimalizovali riziko ztrát svých prostředků.

Klíčová slova lightning network, bitcoin, blockchain, kryptoměna, peer-to-peer, platby, problém škálovatelnosti, multisig, smart contract

Abstract

The scope of this work is to analyze the Lightning Network protocol, which is used to make bitcoin payments outside of blockchain (off-chain) in a fast and inexpensive way. It is one of the possible solutions for the so-called scalability problem of Bitcoin. However, this solution, is built on trade-offs, especially in security. In the case of making payments outside blockchain, we are losing many benefits, such as a solid solution for the double-spending problem. Security aspects of this protocol are the main objective of this work. Among other things, the thesis presents possible vulnerabilities, their exploitation, and in some cases also practical demonstration. Finally, remediation steps for mentioned vulnerabilities are discussed, or at least recommendations for users on how to minimize the risk of being hacked.

The purpose of this work is to spread awareness of the Lightning Network, especially its security aspects, to help users who use the network minimize the risk of losing their funds.

Keywords lightning network, bitcoin, blockchain, cryptocurrency, peer-to-peer, payments, scalability problem, multisig, smart contract

Obsah

Úvod **1**
Cíl práce . . . 2

1 Teoretické předpoklady **3**
1.1 Bitcoin . . . 3
1.1.1 Blockchain . . . 3
1.1.2 Transakce . . . 5
1.2 SegWit . . . 9
1.2.1 Transaction Malleability . . . 9
1.2.2 Souvislost s Lightning Network . . . 9
1.3 Problém škálovatelnosti . . . 10
1.3.1 On-chain řešení . . . 10
1.3.2 Off-chain řešení . . . 11
1.4 Smart contract . . . 11

2 Popis Lightning Network **13**
2.1 Historie . . . 13
2.2 Architektura protokolu . . . 13
2.3 Platební kanály . . . 14
2.3.1 Konstrukce kanálu . . . 14
2.3.2 Provoz kanálu . . . 17
2.3.3 Uzavření kanálu . . . 19
2.3.4 Směrování plateb . . . 23
2.3.5 Poplatky síti . . . 27
2.4 Pathfinding . . . 28
2.4.1 Decentralizace sítě . . . 28
2.4.2 Volba cesty . . . 29
2.4.3 Konstrukce grafu . . . 29
2.5 Faktury . . . 31

2.5.1 Formát faktur 31
2.5.2 QR kód formát 32
2.6 Implementace 32
2.7 Provoz uzlu 32

3 Analýza zranitelností **33**
3.1 Podvrhnutí commitment transakce 33
3.1.1 Doporučení 34
3.2 Griefing 34
3.2.1 Doporučení 34
3.3 Flood & Loot 35
3.3.1 Doporučení 37
3.4 Time-dilation Attack 37
3.4.1 Doporučení 38
3.5 Pinning 39
3.5.1 Doporučení 40
3.6 Fee Siphoning Attack 40
3.6.1 Doporučení 41
3.7 Dust limit exploitation 41
3.7.1 Doporučení 42
3.8 Balance probing 42
3.8.1 Doporučení 44

4 Simulace útoků **45**
4.1 Krádež prostředků pomocí podvrhnutí commitment transakce . 45
4.1.1 Nastavení testovacího prostředí 45
4.1.2 Útok 46
4.1.3 Doporučení 55

Závěr **61**

Literatura **63**

A Seznam použitých zkratek **67**

B Obsah přiloženého CD **69**

Seznam obrázků

1.1 Schéma části Bitcoinového blockchainu[1] 4
1.2 Blockchain Trilemma 4

2.1 Sestrojení kanálu [2] 15
2.2 Simulace provozu kanálu 18
2.3 Výměna zpráv při běžném provozu [2] 19
2.4 Asymetrie commitment transakcí [2] 20
2.5 Mutual close [2] 23
2.6 Probíhající HTLC transakce 25
2.7 Zakončení HTLC transakce 26
2.8 Přesná likvidita je neznámá 30

3.1 1. fáze: Rozestavění uzlů při útoku 35
3.2 2. fáze: Zahlcování HTLC transakcemi [3] 36
3.3 3. fáze: Cílový uzel odpovídá 36
3.4 Nekonzistentní HTLC transakce 39
3.5 Nastavení útočníkových uzlů 41
3.6 Odhalení zůstatků v kanálu 43

4.1 Nástroj Polar 46
4.2 Stav kanálu před útokem 52
4.3 Stav uzlů po útoku 55
4.4 Rozestavění uzlů 59

Úvod

Zažíváme enormní nárůst zájmu o kryptoměny a blockchainové technologie. Většina uživatelů za tím pravděpodobně vidí jen peněžitý zisk (investice, těžení, ...), což zdaleka není špatně, ba naopak, síť potřebuje své uživatele a právě Bitcoin a další blockchainové sítě běží na principech odměňování, aby vůbec mohly fungovat. Kdyby populace začala být najednou přehnaně altruistická a člověk tím přestal být ziskuchtivou bytostí, tak je možné, že ten systém selže (pokud by se neadaptoval). Ovšem existuje i skupina lidí[1], kteří kryptoměny nepovažují jen za možný zdroj příjmu, ale věří ve smysl jejich použití jakožto svobodného, na státech nezávislého platidla. Uskutečnitelnost této vize nám ukáže čas, avšak již nyní si můžeme být jisti, že bude nerealizovatelná, pokud daná kryptoměna nesplní určité kvality bezpečnosti, decentralizace a škálovatelnosti (tzv. Blockchain Trilemma). V tuto chvíli se jako nejvhodnější adept nabízí Bitcoin, návrhem jednoduchý, časem prověřený, dostatečně robustní a bezpečný. Nicméně existují oblasti kde je tento platební systém nedostačující, jako jsou například drobné platby či běžné fyzické nakupování. Asi nikdo z nás nechce čekat čtvrt hodiny u pokladny, než mu těžaři po celém světě potvrdí platbu, aby ve výsledku ke dvěma rohlíkům a vlašskému salátu zaplatil dalších 150,- Kč[2] na poplatcích bitcoinové síti. Toto je bohužel nešťastná realita, ale Bitcoin je z podstaty prvotního návrhu neškálovatelný.

Tento problém se snaží řešit Lightning Network (LN). Nákup v obchodě by pak vyžadoval, aby prodejce měl LN terminál, a kupující by mohl rázem provádět okamžité platby s poplatkem v hodnotě haléřů pomocí skenování QR kódu na pokladně a aplikace v jeho telefonu. Avšak není to tak snadné jak se na první pohled zdá. V tomto textu si rozebereme proč.

[1] Její existence budiž potvrzena mojí příslušností k ní.

[2] Pro čtenáře budoucí, tato hodnota aktuálně odpovídá zhruba 1,5 kg vlašského salátu.

Cíl práce

Cílem práce je nejdříve seznámit čtenáře s důležitými pojmy využitými v práci (kapitola 1. Teoretické předpoklady) jako je například základní architektura Bitcoinu a popsání problému škálovatelnosti. V druhé kapitole následuje detailní seznámení s Lightning Network, které je nutné pro pochopení zranitelností a útoků ve třetí kapitole (Analýza zranitelností). Tato kapitola pak obsahuje jednotlivé známé útoky a doporučení pro uživatele jak se proti nim bránit. Poslední, čtvrtá kapitola, je praktická a věnována simulaci útoků v testovacím prostředí.

Kapitola 1

Teoretické předpoklady

Tato kapitola vysvětluje pojmy zmíněné v textu, které vyžadují hlubší pochopení, aby sekce, kde jsou pojmy užity, byly srozumitelně vyloženy.

1.1 Bitcoin

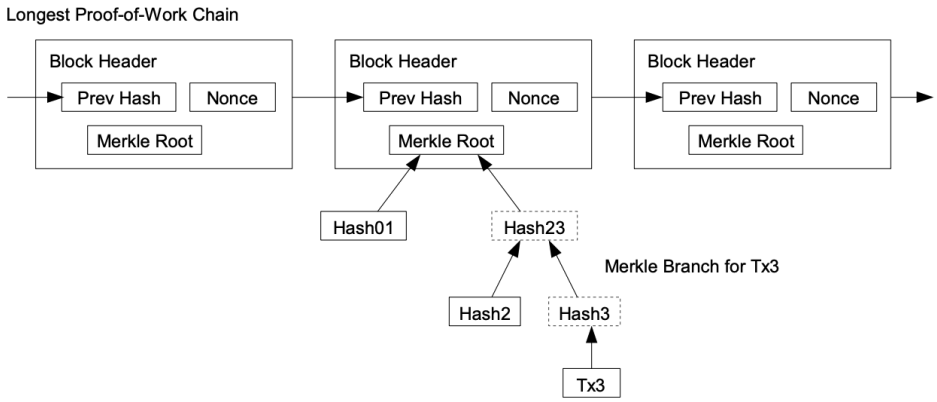
Myšlenkou pro stvoření Bitcoinu[3] bylo vytvořit elektronické peníze, které si budou moci lidé vzájemně posílat bez nutnosti využívání jakýkoliv finančních institucích a třetích stran. Od jiných elektronických řešení se liší v tom, že řeší tzv. double-spending, problém dvojitého utracení týchž prostředků. Toho Satoshi Nakamoto, tvůrce Bitcoinu a jeho oficiálního whitepaperu, docílil pomoci asymetrické kryptografie a technologie blockchainu.[1]

Pro účely této práce v podsekci Blockchain stručně probereme základní princip Bitcoinu a poté se v následující sekci zaměříme především na transakce, jejichž pochopení nám pomůže při práci s Lightning Network.

1.1.1 Blockchain

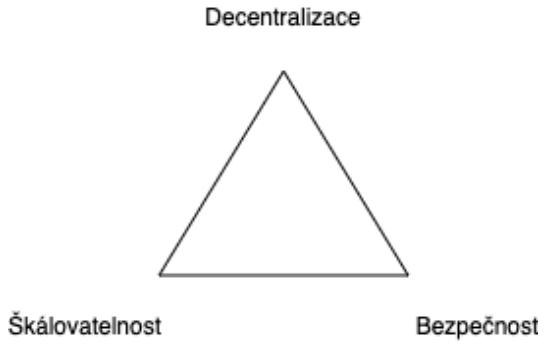
Blockchain v kontextu Bitcoinu je decentralizovaná databáze která obsahuje transakce (záznamy o platbách). Transakce jsou uskladněny v blocích a každý z těchto bloků se odkazuje na ten předchozí a tvoří tak řetězec bloků. Není možné pozměnit jeden z předešlých bloků, aniž by se nemusely přehashovat všechny bloky po něm následující. Klíčovou vlastností daného blockchainu je pak jakým způsobem funguje tvorba nových bloků. Bitcoin využívá konsensus Proof-of-Work, kde platí nejdelší řetězec bloků. Jeho schéma lze spatřit na obrázku 1.1.[1]

[3]Slovem *Bitcoin* s velkým počátečním písmenem označujeme protokol či síť, je-li písmeno malé, označujeme tím minci nebo platidlo, píšeme *bitcoin*.



Obrázek 1.1: Schéma části Bitcoinového blockchainu[1]

Řešení tvorby nových bloků pak představuje Blockchain Trilemma problém, což je problém kde se snažíme vyřešit vyváženost blockchainu mezi bezpečností, decentralizací a škálovatelností (rychlostí transakcí).



Obrázek 1.2: Blockchain Trilemma

Bitcoin svým návrhem preferuje bezpečnost oproti škálovatelnosti a svými vlastnosti z dostupných blockchainů se jeví i jako nejvíce decentralizovaný.[4] Decentralizaci způsobují mimo jiné nízké HW nároky na full-node, popularita a i jeho doba existence. Bezpečnost má na starost vhodně nadimenzovaná tvorba nových bloků společně se zmíněným Proof-of-Work konsensuálním mechanismem.[1]

Proof-of-Work

Proof-of-Work je jedním z klíčových způsobů zabezpečení Bitcoinu. Přináší do protokolu aspekt lidské vynaložené práce a tím chrání jeho hodnotu. Podobně

jako hodnotu zlata chrání obtížnost jeho těžby a omezenost zdrojů na naší planetě. Bitcoin se zde inspiroval a právě tím, že ho je omezené množství (21 milionů bitcoinů) a má obtížnou emisi bloků, je často se zlatem srovnáván[5, 6]. Těžba Bitcoinu závisí na výpočetních operacích a ty jsou syceny elektrickou energií. Výpočetní operací rozumíme hledání neznámého čísla, zvaného nonce (hash preimage), jehož SHA-256 hash vede na předdefinovaný řetězec. Těžař, který toto číslo najde, pak oznámí síti, o jaké číslo se jedná, a celá síť to může jednou hashovací operací zkontrolovat.

Výsledkem je, že máme obtížnou operaci, která je snadno ověřitelná. Otázka vyvstává jak si protokol poradí s tím, když se do sítě připojují těžaři s více výkonnými stroji (vývoj výkonnějších procesorů, Moorův zákon). To by z principu věci měli těžit (nalézat správné hashe) rychleji a tím by urychlili emisi bloků a potenciálně znehodnocovali hodnotu bitcoinu. Tomuto zabraňuje mechanismus, který upravuje náročnost těžby, aby se držela na přibližně 1 bloku za 10 minut. K této úpravě náročnosti dochází přibližně každých 14 dní. Proč jsou tyto hodnoty pouze přibližné je způsobeno tím, že nehovoříme v časových intervalech, ale v počtech bloků, kde úprava náročnosti těžby probíhá po každých 2016 blocích. Podobně, aby se zajistila postupná emise v souladu s Moorovým zákonem, tak se odměna za vytěžený blok každý 210 000 změní na polovinu. To zhruba odpovídá, že se odměna sníží každý 4. rok a k úplnému vytěžení dojde v roce 2140. Tyto mechanismy jsou navrženy, tak aby se zabránilo double-spending problému a tím i 51% útoku.[1]

51% útok

Jedná se o známé úskalí konsensuálních mechanismů blockchainů. V tomto případě útok spočívá v tom, že útočník disponuje více jak polovinou výkonu sítě. Tím, že disponuje takovýmto výkonem je schopen vytvořit delší řetězec bloků než zbytek sítě a tím bude útočníkův blockchain považován za validní, poněvadž platný řetězec bloků je ten nejdelší. To mu umožní provádět double-spending, tedy provést platbu a následnou těžbou vytvořit jiný řetězec s jinou transakční historií. Pravděpodobnost provedení takového útoku je přesně taková, jaká je pravděpodobnost, že útočník bude vlastnit více výpočetního výkonu než zbytek planety participující se na těžbě. Jedním z možných scénářů je, že by se spojily největší těžební pooly, avšak vyvstává otázka jestli ztráta renomé a fungujícího byznysu jim stojí za provedení těchto plateb.

1.1.2 Transakce

Transakce v Bitcoinu představuje přesun peněz (bitcoinu) z jedné adresy na jinou, který je ohlášen do celé sítě a zaznamenám v blocích. Tato transakce může být brána jako nevratná podle toho, kolika bloky byla potvrzena. V případě 0 bloků stále čeká v mempoolu, což je prostor, z kterého si těžaři vybírají transakce ke zpracování, a není tedy ještě zpracována. Pro malé platby

se transakce považuje jako nevratná již při 1 bloku, avšak standardně se za bezpečnou hodnotu uvádí např. 6 bloků, poněvadž se při nynější velikosti sítě již jedná o enormní výpočetní výkon, který by potenciální útočník musel vynaložit (bylo by potřeba najít nonce pro 6 bloků rychleji než zbytek sítě, který by hledal jen jednu). Transakce nejsou šifrovány a kdokoliv si může prohlížet Bitcoin jako veřejnou účetní knihu, kde místo jmen jsou veřejné klíče.[7, 8]

Běžný formát transakce vypadá následovně:

- *version* – má potenciální využití do budoucna, momentálně se rovná 1,
- *flag* – indikuje přítomnost *witness* dat,
- *in-counter* – kladné číslo, počet záznamů v *list of inputs*,
- *list of inputs* – seznam vstupů,
- *out-counter* – kladné číslo, počet záznamů v *list of outputs*,
- *list of outputs* – seznam výstupů,
- *witnesses* – seznam tzv. svědků, jeden záznam pro každý vstup, přeskočeno pokud flag chybí,
- *lock_time* – indikuje dobu uzamčení po vytěžení bloku, typicky je rovno 0.

Vstup je reference na výstup z předchozí transakce. Takže bitcoin jakožto mince je pouze řetězec digitálních podpisů. Vezmeme-li si například první transakci v bloku, odměnu těžaři, tzv. coinbase transakci, tak zde je vstupem 6.25 BTC z žádného výstupu poněvadž je tento bitcoin čerstvě vytěžený. Výstupem této transakce je pak 6.25 BTC (+ poplatky za transakce), které putují těžaři. Chce-li pak těžař někde z těchto peněz zaplatit, tak u této transakce bude vstupem výstup zmiňované coinbase transakce a výstupem osoba které platí. Jakmile ta osoba, které zaplatil, bude chtít tyto prostředky využít, tak vstupem její transakce bude výstup transakce od těžaře a tak dále. Ten výstup který spočívá u té osoby a který předtím spočíval u těžaře, nazýváme UTXO, unspent transaction output. UTXO je v podstatě abstrakcí pro peníze. Bitcoiny, které má uživatel ve své peněžence, jsou pouze UTXO, neutracené výstupy, konec onoho řetězce digitálních podpisů (aktuální vlastník UTXO vždy podpisem stvrdí jeho převod na příjemcův veřejný klíč, adresu).[7, 8]

Nyní si představíme detailněji, jak transakce fungují a jakých jsou různých typů.

Bitcoin skript

Bitcoin skript je skriptovací jazyk inspirovaný jazykem Forth používaný pro operace nad protokolem. Jedná se o Turingovsky nekompletní jazyk (neobsahuje cykly), což je od počátku záměrem, protože pro funkčnost protokolu je tento jazyk dostačující a eliminuje se tím řada rizik spojených s Turingovsky kompletními systémy, poněvadž na těchto systémech je útočník schopen spustit jakýkoliv algoritmus (Church-Turingova teze). Kontrakty na Bitcoinu jsou pak díky této vlastnosti vždy deterministické a nehrozí zde nekonečné smyčky a podobná úskalí.

Tento skript pak poskytuje např. transakcím instrukce, jak a kdo může získat zaslané bitcoiny. Slova jazyka jsou pak nazývány op-kódy a samotný kód pak připomíná jazyky symbolických instrukcí. V této práci se s ním nejednou setkáme.[9]

Pay-to-Public-Key-Hash

Většina transakcí v Bitcoinu je právě tohoto typu (P2PKH). Klasické posílaní prostředků na Bitcoinovou adresu odpovídá následujícímu skriptu. Výstup je uzamčen pomocí P2PKH skriptu a může být odemčen pomocí veřejné adresy a odpovídajícího podpisu. Například Alice chce poslat Bobovi 1 BTC:

```
OP_DUP OP_HASH160 <Hash Bobova veř. klíče> OP_EQUAL OP_CHECKSIG
```

Hash Bobova veřejného klíče odpovídá jeho Bitcoinové adrese bez kódování `Base58Check`. Tímto skriptem se uzavřou prostředky Alice. Bob si je poté může odemknout využitím následujícího skriptu, pokud poskytne svůj klíč s validní podpisem:

```
<Bobův podpis> <Bobův veřejný klíč> OP_DUP OP_HASH160
<Hash Bobova veř. klíče> OP_EQUAL OP_CHECKSIG
```

Instrukce `OP_DUP` zduplikuje první slot na zásobníku a následně dojde k jeho zahashování. Toto se ověří a nakonec se zkontroluje podpis.[7, 8, 10]

Pay-to-Public-Key

Je jednodušší varianta P2PKH a již se tolik nepoužívá poněvadž P2PKH má kratší formát. V tomto případě je veřejný klíč přímo uložen v uzamykacím skriptu.

```
<veřejný klíč Boba> OP_CHECKSIG
```

Odemykací skript poté vypadá následovně:

```
<Bobův podpis> <veřejný klíč Boba> OP_CHECKSIG
```

V podstatě pouze jen zkontroluje, jestli podpis odpovídá veřejnému klíči.[7, 8, 10]

Multi-signature

Multi-signature, neboli více-podpisové skripty se užívají v klasickém M:N schématu, kde N je počet klíčů a M počet klíčů potřebných k validaci. Uzamykací i odemykací (je potřeba mít na zásobníku M validních podpisů) skript pak vypadá následovně:

```
M <veřejný klíč 1> ... <veřejný klíč N> N OP_CHECKMULTISIG
```

Například v LN bychom se mohli setkat s:

```
2  <veřejný klíč Alice> <veřejný klíč Boba> 2 OP_CHECKMULTISIG
```

Pay-to-Script-Hash

Zkráceně P2SH, je efektivnější a používanější varianta Multi-signature skriptu, při více klíčích. Když je součástí multisigu mnoho klíčů, tak to způsobí, že výsledný skript je velmi dlouhý a tím i náročnější na zpracování (multisig transakce o 5 klíčích bude přibližně 5x větší než běžná transakce). Proto přichází na scénu P2SH, který vytvoří tzv. redeem skript z Multi-signature skriptu a nadále se použije pouze hash tohoto redeem skriptu. Tím získáme pouze 20 bytový řetězec, kterým nahradíme jinak zdlouhavé parametry.[7, 8, 10]

Další typy transakcí

Předešlé 4 typy transakcí, které jsme si představili, spadají do skupiny tzv. standardních transakcí. Mezi standardní transakce také patří např. Data output (OP_RETURN), který slouží k ukládání dat přímo na blockchain. Je omezený 40 byte a jedná se od počátku o neutratitelný (a nepřevoditelný) výstup. Příkladem[4] využití by mohlo být vložení zajímavého (avšak velmi krátkého) zenového koánu na blockchain, kde do výstupu dáme 0 BTC, poněvadž tento výstup už nebude možné získat nikdy zpět.

```
OP_RETURN 4A616B20746C65736BE120312072756B613F
```

Zbývající standardní skripty (které jsou součástí od přijmutí SegWitu) jsou Pay to Witness Public Key Hash (P2WPKH) a Pay to Witness Script Hash (P2WSH). V jejich případě se jedná o nejefektivnější skripty, poněvadž se předávají pouze hodnoty ze zásobníku a o ověření se stará witness struktura (viz SegWit), jinak fungují stejně jako P2PKH, resp. P2SH. Dále existuje mnoho další skriptů z tzv. nestandardní třídy, které ale jsou ale již nad rámec této práce.[7, 8, 10]

[4] Otázka je, ale jestli to má nějaký smysl.

1.2 SegWit

Jedná se o jeden z BIP, Bitcoin Improvement Proposal, návrh na vylepšení Bitcoinu, který vždy podstupuje hlasování mezi těžaři, kteří rozhodnou, zdali bude přijat. Tento BIP byl navrhnut v roce 2015 a aktivován byl v roce 2017. [11] Definuje novou strukturu zvanou *witness*, která je v blocích uložena mimo mimo merkle tree (hashový strom pro transakce), čehož se docílí právě díky skriptu OP_RETURN. Tato struktura obsahuje potřebná data k ověření validity transakce a není již potřeba mít tyto data explicitně u každé transakce (stačí mít pouze všechno při sobě v témže bloku). Hlavní motivací pro přijmutí této funkcionality bylo vyřešení Transaction Malleability na úrovni identifikátoru transakce, avšak přineslo to i jiné výhody, jako například zvětšení kapacity bloku zmenšením transakcí.[12, 13]

1.2.1 Transaction Malleability

Tento útok (v našem kontextu) je založen na tom, že kdokoliv může změnit identifikátor transakce, pokud je součástí identifikátoru i podpisový skript (`skriptSig`). V prvotním návrhu byl podpisový skript součástí transakce a tím byl součástí i identifikátoru transakce (otisk transakce). Ve skriptu je pak možné provést změny, které způsobí nový otisk transakce, ale podpis bude stále platný (např. záměna `6` za `06` vede na stejné číslo, ale jako řetězec vede na jiný hash). Toho útočník může docílit jednoduše vybráním transakce z mempoolu a vytvoření nové s větším poplatkem. Útok za využití Transaction Malleability, který se zapsal do dějin, je hack burzy Mt. Gox. Ta používala pro identifikaci transakcí právě pouze transakční identifikátor, takže útočníkovi stačilo si vybrat své bitcoiny z burzy, změnit transakční identifikátor a nahlásit burze, že mu žádné bitcoiny nepřišly. Mt. Gox se podíval na blockchain a vzhledem k tomu, že tu transakci opravdu neviděl, tak útočníkovi zaslal bitcoiny znovu. Tímto způsobem se z burzy vybralo více jak 850 tisíc bitcoinů, což tehdy (rok 2014) odpovídalo zhruba 620 milionům dolarů, dnes (v době psaní tohotu textu) se částka pohybuje v desítkách miliard dolarů.[14]

SegWit přesunul data ze `skriptSig` ven z transakce (v transakci je nyní kvůli soft forku nevyplněná proměnná) do zmiňované struktury *witness* a tím pádem není podpis již součástí otisku transakce.[12]

1.2.2 Souvislost s Lightning Network

Aktivace SegWitu byla klíčová pro Lightning Network, poněvadž LN neustále pracuje s nepotvrzenými transakcemi a byla by tedy velmi zranitelná vůči Transaction Malleability. Zavedení SegWitu akcelerovalo její vývoj.

1.3 Problém škálovatelnosti

Tento termín uvažujeme v kontextu blockchainových sítí. Znamená, že tím, jak blockchain a počet jeho uživatelů roste, tak se zvedají požadavky na počet zpracovávaných transakcí, avšak parametry, které toto číslo ovlivňují, jsou stále stejné, neškálovatelné s nabývajícím provozem.

Pomocí sady nástrojů TradeBlock[5] lze nalézt, že průměrná velikost bitcoinové transakce za poslední 3 roky je cca 455 bytů na jednu transakci. Velikost bitcoinového bloku je 1 MB a složitost těžení bloků je pravidelně algoritmicky upravována[6], aby zabrala přibližně 10 min. Z těchto údajů jsme schopni vypočítat průměrnou hodnotu zpracovaných transakcí za sekundu[7] [15].

$$\begin{aligned} \#\text{ počet transakcí na blok} &= \frac{\text{Velikost bloku [byte]}}{\text{Průměrná velikost transakce [byte]}} \\ &= \frac{1000000}{455} \cong 2197,80 \end{aligned} \tag{1.1}$$

$$\text{TPS} = \frac{\#\text{ počet transakcí na blok}}{\text{Doba těžení jednoho bloku [s]}} = \frac{2197,80}{600} \cong 3,66 \tag{1.2}$$

Z výsledné hodnoty vyplývá, že platební systém, kterému hodnota náleží, není připraven na to, stát se zcela globálním. Pro porovnání, například Visa, denně odbavuje přibližně 150 milionů transakcí, což odpovídá 1736 TPS, a uvádí, že je schopná škálovat do řádů desítek tisíc TPS.[16]

Řešení tohoto problému se obecně dělí do dvou kategorií.

1.3.1 On-chain řešení

Jsou založená na tom, že veškeré transakce musí probíhat na blockchainu a změny se ho tím přímo týkají. Například zvětšování velikosti bloků nebo snižovaní složitosti (zkracování času) vytěžení jednoho bloku.

Kdybychom chtěli v předešlém příkladě dorovnat počet TPS u Visy tím, že zvětšíme velikost bloku, bude to znamenat 452x navýšení, tedy že velikost bloku bude 452 MB. S tím přichází mimo jiné vyšší nároky na velikost blockchainu i propustnost sítě, což může zapříčinit větší centralizaci, protože provozovatelem tzv. full-node už nebude moci být běžný uživatel (nároky na hardware). V případě zkrácení těžby bloku, bychom museli mít z 10 min cca 1,3 s, což je mnohem méně než čas propagace nového bloku do sítě, který byl vyměřen průměrně na 12,6 s. Takové změny mohou způsobovat chyby v síti, nespolehlivost, zahlcenost, případně jiné zranitelnosti.[17]

[5] `https://tradeblock.com/bitcoin/historical/1w-f-tsize_per_avg-01101`
[6] po každém vytěžení 2016 bloků, což je cca 14 dní
[7] TPS – transactions per second

Příkladem akceptovaného on-chain vylepšení je již zmíněný SegWit, který přispěl lehce propustnosti a škálovatelnosti ve smyslu otevření příležitosti pro off-chain řešení.

1.3.2 Off-chain řešení

Spočívají v provádění transakcí mimo blockchain a díky tomu nejsou výrazně omezována jeho neškálovatelností. Je zde ale potřeba vyřešit mnoho jiných problémů. Tato práce se snaží přiblížit ty, které se týkají Lightning Network.

1.4 Smart contract

Smart contract je forma digitální dohody mezi zúčastněnými stranami. Jedná se zpravidla o program, software či protokol. Program zde zastává roli soudce a rozhoduje o chování uživatelů v protokolu, kde je využit. Pro lepší pochopení si uvedeme příklad.

Máme decentralizovanou burzu, která umožňuje půjčky na blockchainu Ethereum, protokol, který implementuje různou řadu smart contractů. Jedním z těchto kontraktů bude program, který se bude starat o to, že když si uživatel chce půjčit 1000 DAI (token na Ethereum blockchainu), tak musí mít dostatečný kolaterál. Kdyby takový kontrakt neexistoval, tak si kdokoliv kdykoliv může půjčit kolik tokenů DAI si zamane. Kontrakt učiní to, že zjistí aktuální cenu tokenu a podle toho bude od uživatelů vyžadovat takové množství kolaterálu. Jakmile uživatel do kontraktu uzamkne 1 ETH (nativní token pro Ethereum), tak je kontrakt napsaný tak, že v tu chvíli uživateli dovolí si půjčit až do 70% jeho hodnoty. To mu při dané ceně ETH bude stačit na půjčku, takže si úspěšně půjčí 1000 DAI. Následně musí kontrakt také obsahovat např. logiku na splacení částky a následné vyjmutí kolaterálu z kontraktu. Soudcem je tedy kód. Poté je už jen na uživatelích, jaké smart contracty shledají spravedlivými a bezpečnými pro jejich finanční operace. Proto jsou (nebo by alespoň měly být) smart contracty zpravidla auditovány několika bezpečnostními společnostmi (či zdatnými jednotlivci) a mít otevřený zdrojový kód.[18]

Kapitola 2

Popis Lightning Network

Z dosavadního textu již zhruba víme, proč Lightning Network existuje a co by nám měl umožnit. Následující odstavce nám však protokol blíže představí, abychom byli schopni posléze lépe porozumět jeho zranitelnostem.

Jedná se o peer-to-peer protokol, navržený jako druhá vrstva nad Bitcoinovým protokolem. Umožňuje provádění plateb pomocí tzv. platebních kanálů, kde v rámci jednoho kanálu je možné provést několik plateb s minimálními poplatky v (téměř) okamžitém čase. Jedná se v podstatě o smart contracty na Bitcoinovém blockchainu. Kanály se otevírají mezi uzly a společně tvoří graf, kde vrcholy grafu jsou uzly a cesty mezi nimi kanály. Tento graf a logiku za ním schovanou můžeme nazvat Lightning Network.[19, 2]

2.1 Historie

Myšlenka platebních kanálů je údajně součástí kódu Bitcoinu již od prvotních návrhů Satoshiho Nakamota (2009)[20], avšak ucelenou podobu Lightning Network získal až ve svém prvním tzv. white paperu v roce 2015.[21] Adopci avšak blokovala zmíněná aktivace SegWitu, takže teprve po roku 2017 začala vznikat první veřejná vydání implementací LN[8]. V této práci nás čeká zhodnocení a popsání stavu sítě po přibližně 4 letech aktivních vývoje.

2.2 Architektura protokolu

LN využívá několik známých protokolů a člení se do pěti základních vrstev.

1. Platební vrstva – Nejvyšší vrstva, která zajišťuje platební rozhraní pro aplikace. Dochází zde například k generování faktur a hledaní cesty v grafu.

[8] Soudě dle historie vydání z GitHubu u implementací jako je LND a c-lightning.

2. Směrovací vrstva – Obsahuje klíčovou logiku pro směrování plateb a např. tvorbu HTLC.

3. Peer-to-peer vrstva – Umožňuje komunikaci mezi uzly. Konečný automat pro kanály, např. logika pro otevírání a zavírání kanálů a Gossip protokol.

4. Vrstva pro zasílání zpráv – V této vrstvě dochází k zakódování zpráv.

5. Síťová vrstva – Implementuje známé síťové protokoly jako například IPv4, IPv6, TOR2 a TOR3.

2.3 Platební kanály

Platební kanál je smart contract, který představuje finanční vztah mezi mezi dvěma zúčastněnými stranami. Realizován je pomocí 2-of-2 multi-signature (více-podpisové) adresy na Bitcoinovém blockchainu, kde každá strana má jeden klíč a pro každou změnu v rámci kanálu je vyžadován souhlas (klíč) obou stran. Této adrese účastníci kontraktu při zakládání poskytnou určitý finanční obnos, s kterým pak může adresa, potažmo kanál hospodařit. Uzavření kanálu znamená zápis na blockchain a vyrovnání obou stran na jejich bitcoinové peněženky. Kanál lze uzavřít společně nebo i jen jednou stranou a legitimitu uzavření (rozdělení finančních prostředků) hlídá samotný smart contract.[19, 2]

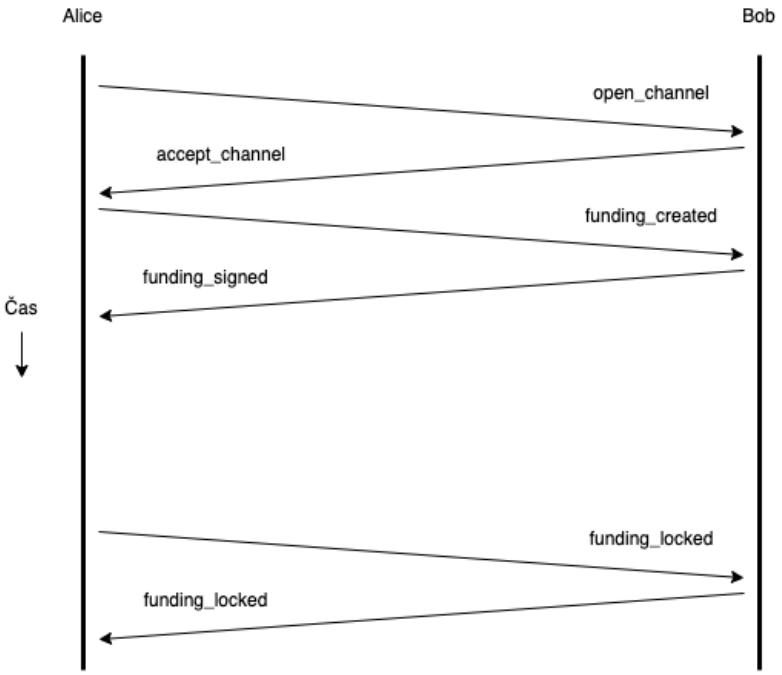
2.3.1 Konstrukce kanálu

Existují dva způsoby otevření kanálu:

- Announced – Po otevření pošle `channel_announcement` zprávu, aby se zapojil do sítě a bylo možné přes něj směrovat

- Unannounced – Kanál se síti nenahlásí, nebude se přes něj směrovat

Otevření kanálu začíná zasláním zprávy `open_channel` a poté následuje výměna dalších pěti zpráv podle obrázku 2.1.[2]



Obrázek 2.1: Sestrojení kanálu [2]

Nejprve Alice sdělí Bobovi, že k němu otevírá kanál (`open_channel`) načež s tím Bob souhlasí a proto Alici odpoví (`accept_channel`). Následuje na straně Alice vytvoření tzv. funding a commitment transakce (`funding_created`) která je zaslána Bobovi. Bob odpoví odesláním potřebných podpisů (`funding_signed`). V tu chvíli má Alice vše potřebné pro založení kanálu a vyšle transakci na blockchain. Multisignature skript vypadá následovně:

```
2 <veřejný klíč Alice> <veřejný klíč Boba> 2 CHECKMULTISIG
```

Tento skript se zakóduje jako Pay-to-Witness-Script-Hash (P2WSH) a to je on-chain adresa kanálu.

Jakmile je on-chain transakce dostatečně zvalidována (počet potřebných bloků lze zadefinovat proměnnou `minimum_depth`) tak Alice tuto skutečnost oznámí pomocí zprávy `funding_locked`. V následujících podsekcích si probereme jednotlivé zprávy dopodrobna. [22, 2]

Zpráva `open_channel`

Tato zpráva indikuje žádost o otevření kanálu a obsahuje následující informace:

- `chain_hash` – Představuje hash genesis bloku daného blockchainu, který slouží k přesné identifikaci, o jaký blockchain se jedná.

- `temporary_channel_id` – Dočasné ID kanálu, dokud není vytvořen (v tu chvíli je nahrazen `channel_id`).
- `funding_satoshis` – Množství satoshi, které se uzavřou v kanálu.
- `push_msat` – Množství satoshi, které odesílatel (uzel otevírající kanál) daruje příjemci.
- `dust_limit_satoshis` – Hranice satoshi, pod kterou nelze vygenerovat commitment nebo HTLC transakci
- `channel_reserve_satoshis` – Množství satoshi, které uzel nemůže poslat a musí si je ponechat na své straně kanálu.
- `htlc_minimum_msat` – Určuje nejmenší možnou hodnotu v milisatoshi, kterou je uzel ochoten směrovat.
- `max_htlc_value_in_flight_msat` – Maximální hranice celkové hodnoty všech HTLC transakcí v milisatoshi, kterou je uzel ochoten směrovat.
- `max_accepted_htlcs` – Maximální počet HTLC transakcí, které je uzel ochoten směrovat.
- `feerate_per_kw` – Určuje počáteční sazbu poplatků (může být časem změnena pomocí `update_fee`).
- `to_self_delay` – Počet bloků, které je třeba počkat před zápisem na blockchain v případě Force close.
- `funding_pubkey` – Veřejný klíč v 2-of-2 multisig bitcoin skriptu.
- Parametry `_basepoint` – 5 parametrů sloužících k odvození unikátních klíčů.
- `first_per_commitment_point` – Hodnota sloužící k odvození první commitment transakce.
- `channel_flags` – Zatím pouze nejméně významný bit této hodnoty je definován a to jako `announce_channel`, který slouží k oznámení, zdali se jedná o veřejný nebo privátní kanál.

Zpráva `accept_channel`

Tato zpráva obsahuje informace o uzlu který souhlasí s vytvořením kanálu. Parametry má stejné jako v `open_channel`. `temporary_channel_id` musí být stejné jako v `open_channel` zprávě, a to i ve všech následujících zprávách po dobu tvorby kanálu.

Zpráva `funding_created`

Tato zpráva obsahuje mimo `temporary_channel_id` také klíčové parametry `funding_txid`, `funding_output_index` a `signature` pro vytvoření commitment transakce (jakmile odesílatel obdrží podpisy příjemce).

Zpráva `funding_signed`

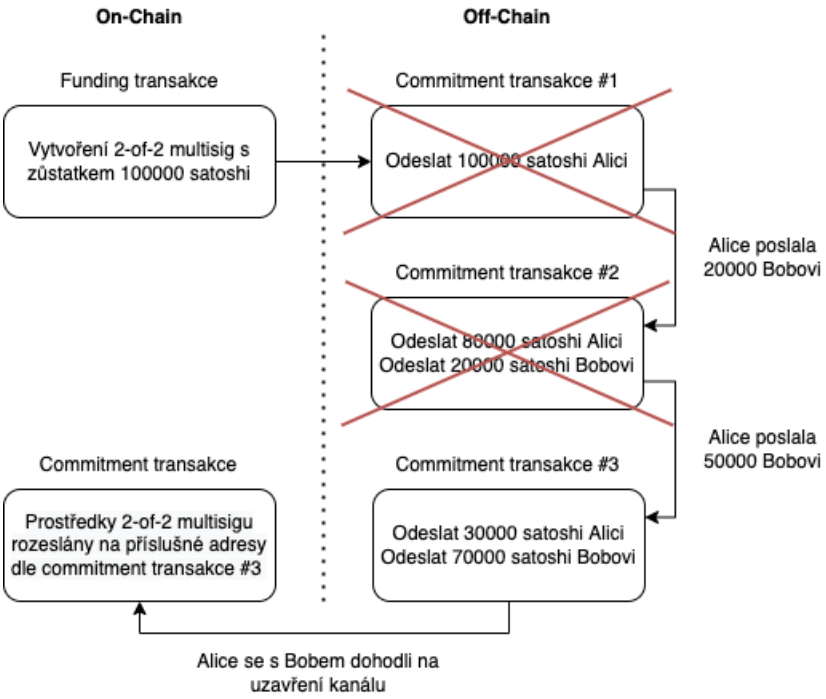
Tato zpráva obsahuje `signature`, podpis příjemce, a vzniká zde `channel_id`, které je odvozené z `funding_txid` a `funding_output_index` z předchozí zprávy (použitím big-endian XOR).

Zpráva `funding_locked`

Tato zpráva indikuje, že funding transakce dosáhla `minimum_depth` a kanál je připraven k provozu. Obsahuje `channel_id` a `next_per_commitment_point`, který bude použit pro příští odvození commitment transakce.

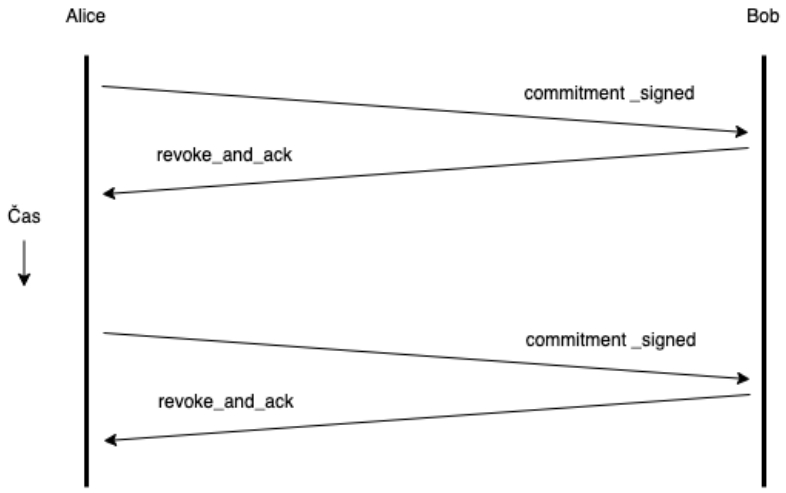
2.3.2 Provoz kanálu

Kanály jsou otevírány jednosměrně, např. Alice má na začátku 100000 satoshi a Bob 0 satoshi. V případě, že Alice pošle Bobovi 40000 satoshi, tak se vytvoří nová commitment transakce, která představuje fakt, že se obě strany shodly na novém zůstatku na odpovídající multisig adrese. Ve chvíli kdy vzniká nová commitment transakce, tak ta předchozí zaniká (zveřejněním `per_commitment_secret`).



Obrázek 2.2: Simulace provozu kanálu

Běžná komunikace obsahuje dvě zprávy. Alice, s požadavkem na změnu zůstatku kanálu posílá zprávu `commitment_signed` na kterou Bob reaguje zprávou `revoke_and_ack`. Zpráva `commitment_signed` obsahuje `channel_id` a podpis, `signature`. Bob může s podpisem Alice vytvořit novou commitment transakci, takže odešle ve zprávě `revoke_and_ack` parametr klíčový pro nalezení `revocationpubkey` pro novou commitment transakci, který nalezneme pod názvem `next_per_commitment_point` a `per_commitment_secret`, kterým se zneplatní původní commitment transakce.



Obrázek 2.3: Výměna zpráv při běžném provozu [2]

Tento způsob komunikace nám zajistí férové jednání obou stran. Příjemce podpisu se nemusí obávat odhalení `per_commitment_secret`, protože s podpisem může zkonstruovat novou commitment transakci, a odesílatel se nemusí obávat, že by neobdržel `per_commitment_secret` aniž by nedošlo k nové commitment transakci.

Pro snadnější sledování již odhalených `per_commitment_secret` protokol zavádí speciální kódování zvané `state hints`, kde vstupem je mimo jiné číslo commitment transakce. Zná-li strana aktuální číslo commitment transakce, je pro ni snadné určit, zdali nějaká transakce je již zneplatněná nebo aktuální a v případě, že je zneplatněná, tak dohledat `per_commitment_secret` v tzv. revocation secret tree. [22, 2]

2.3.3 Uzavření kanálu

Jsou tři způsoby uzavření kanálu:

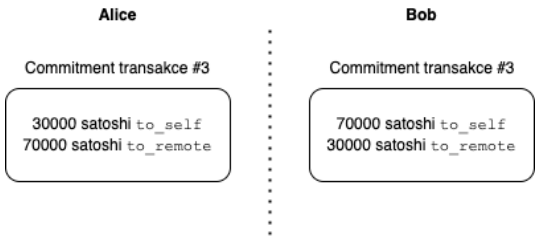
- Mutual close – Obě strany se dohodnou na uzavření pomocí tzv. zavírací transakce. Výhodou je rychlost získání prostředků za cenu jediného poplatku při zápisu na blockchain, který hradí strana, která kanál otevřela.

- Force close – Je možné kanál uzavřít pouze jednou stranou; to se například vyplatí v případě, že druhá strana neodpovídá a není tedy možné kanál uzavřít společně. Nevýhodou je čekání na prostředky po dobu časového zámku a poplatky s tím spojené. Hrozí navýšení nákladů za pokusy spojené se směrováním, protože tyto transakce budou nuceny být vyřešeny on-chain. Další problém souvisí s tím, že není známa výše poplatku v čas uzavření a protokol způsobí, že poplatek může být až 5x vyšší.

- Protocol breach – Je druh Force close případu, kde zavírající strana se snaží druhou podvést. Pokud podvod podváděná strana odhalí, tak je uzavření rychlé a získá zůstatek útočníka, avšak musí zaplatit zápis na blockchain, když druhé straně už nic nezbylo.

Force close & Protocol Breach

K poslednímu případu uzavření existují základní obranné mechanismy, které se snaží narušení protokolu předcházet[9]. Prvním z nich je záměrně omezené rozhraní samotných implementací LN, které neuchovávají staré commitment transakce nebo při pokusu o Protocol breach se (s neupravenou implementací) útočník sám zpenalizuje. Druhou funkcionalitou, na které staví následující dvě, je, že commitment transakce jsou asymetrické. V obrázku 2.4 si ukážeme jak commitment transakce vnímají jednotlivé strany. Využijeme pro to commitment transakci z obrázku 2.2.[22, 2]



Obrázek 2.4: Asymetrie commitment transakcí [2]

Díky čemuž v protokolu jasně rozeznáme dvě strany a můžeme aplikovat penalizační mechanismus. Jeho základním prvkem je parametr `to_self_delay`, který dává oběti více času na odhalení pokusu o krádež prostředků. Avšak ještě více důležité jsou tzv. revocation keys, klíče, pomocí kterých můžeme vytvořit penalizační transakci.

Commitment transakce má následující Bitcoin skript:

```
OP_IF
    # Penalizační transakce
    <revocationpubkey>
OP_ELSE
    <to_self_delay>
    OP_CHECKSEQUENCEVERIFY
    OP_DROP
    <local_delayedpubkey>
OP_ENDIF
OP_CHECKSIG
```

[9] Jak si ale ukážeme v pozdějších kapitolách, tak zjistíme že nedostatečně.

Pokud platí podmínka, tak to umožňuje komukoliv získat prostředky adresy v případě, že jeho podpis je validní oproti `revocationpubkey`. Jinak nastává časový zámek na počet bloků definovaný pomocí `to_self_delay`. Ve chvíli kdy tento čas uplyne, tak prostředky může získat osoba s validním podpisem oproti `local_delayedpubkey`.

Je potřeba zajistit, aby k penalizaci došlo pouze v případě, že jedna ze stran se pokusí o podvod. Toho docílíme pomocí asymetrické kryptografie a eliptických křivek. Jak již víme z předchozí sekce Konstrukce kanálu, na začátku existence kanálů dochází k výměně `_basepoint` parametrů, kde jeden z nich je `revocation_basepoint`. Tento parametr je neměnný po celou dobu životnosti kanálu a unikátní pro každou stranu.

Hodnota `revocationpubkey` je definována následovně:

```
revocationpubkey = revocation_basepoint *
                   SHA256(
                    revocation_basepoint || per_commitment_point
                   )
                   +
                   per_commitment_point *
                   SHA256(
                    per_commitment_point || revocation_basepoint
                   )
```

Kde `per_commitment_point` je odvozený z `per_commitment_secret`, což je jeho privátní klíč. Ten je odhalen pokaždé když se zneplatní původní commitment transakce.

Když se Alice rozhodne, že podvede Boba vysláním zneplatněné commitment transakce do sítě, tak Bob zná odpovídající `per_commitment_secret` (byl v minulosti již odhalen), který využije k penalizaci Alice.

Pro penalizaci potřebujeme vytvořit privátní klíč pro revokaci, kde poskytnutý podpis bude odpovídat výše uvedené formuli definující veřejný klíč `revocationpubkey`.

Tento klíč získáme pomocí následující formule:

```
revocationprivkey = revocation_basepoint_secret *
                    SHA256(
                     revocation_basepoint || per_commitment_point
                    )
                    +
                    per_commitment_secret *
                    SHA256(
                     per_commitment_point || revocation_basepoint
                    )
```

Hodnota `revocation_basepoint_secret` je privátním klíčem náležícím k `revocation_base_point` a má ji každý uzel lokálně při prvním použití `revocation_base_point` (při generování páru).

Jakmile nějaká strana má příslušný `revocation_basepoint_secret` a zná `per_commitment_secret` pro danou commitment transakci, tak je schopna vygenerovat `revocationprivkey`. Tento klíč dané straně poskytne validní digitální podpis a dojde k vykonání první větve podmínky, tedy penalizaci.[23, 22, 2]

Pro úplnost si ukážeme jeden příklad, kde v závorkách je vyznačeno objevení důležitých hodnot:

1. Alice chce otevřít kanál směrem k Bobovi a vložit do něj 100000 satoshi, Bob souhlasí. (`revocation_basepoint_secret`, `revocation_basepoint`, `first_per_commitment_point`).

2. Bob vystaví fakturu na 20000 satoshi a Alice ji obratem zaplatí (předchozí `per_commitment_secret`, `next_per_commitment_point` nový).

3. Bob vystaví fakturu na 50000 satoshi a Alice ji obratem zaplatí (předchozí `per_commitment_secret`, `next_per_commitment_point` nový).

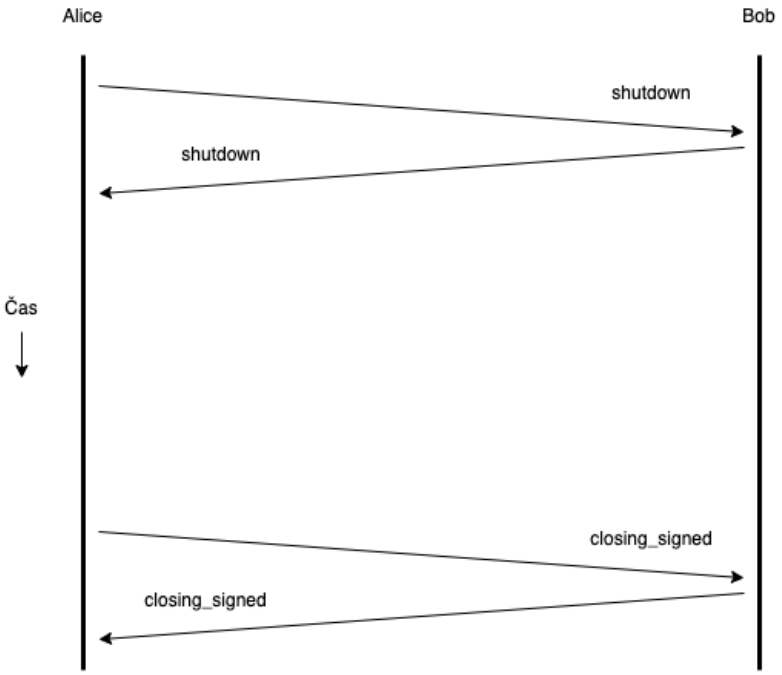
4. Alice se snaží podvést Boba a tvrdí, že má 80000 místo 30000 satoshi. Avšak `per_commitment_secret` tohoto stavu kanálu byl odhalen v 3. kroku.

5. Bob podvod odhalil (`revocationprivkey` u Boba).

6. Bob získal celkem 100000 satoshi, veškerou likviditu kanálu.

Mutual close

Doposud jsme se zabývali tím, jak lze kanál uzavřít jednostranně, avšak doporučený způsob zavření je oboustranný. K oboustrannému uzavření dojde tehdy, když se obě strany shodnou na uzavření kanálu. Strana, která proces uzavření iniciuje, vysílá zprávu `shutdown` a příjemce na ni také reaguje zprávou `shutdown`.[2]



Obrázek 2.5: Mutual close [2]

Zpráva `shutdown` obsahuje `channel_id`, poté `len` a `scriptpubkey`. Parametr `len` je délka adresy dané peněženky do které chce uživatel obdržet svoje prostředky. Samotná adresa je pak v parametru `scriptpubkey` v podobě jedné ze standardních Bitcoinových adres (P2WSH, P2WPKH, etc.).

Ve chvíli, kdy si zúčastněné strany vyměnily svoje `shutdown` zprávy, tak mohou dokončit uzavření kanálu. Zakladatel kanálu pošle druhé straně zprávu `closing_signed`. Tato zpráva obsahuje podpis a návrh na výši poplatku za zpracování transakce on-chain. Příjemce odpoví svým podpisem a výši poplatku zachová nebo se rozhodne změnit. Poté může zakladatel kanálu vyslat transakci na blockchain a po zpracování je kanál uzavřen.[22]

2.3.4 Směrování plateb

LN kanály umožňují směrování/přeposílání plateb. Ve chvíli, kdy existuje více otevřených kanálů a nějaké z nich jsou propojeny, tak v těchto případech hovoříme o síti kanálů, tedy Lightning Network. Tuto síť může uživatel využít k zaplacení faktur jiných uživatelů, přestože s nimi nemá otevřený přímý kanál. Jedná se o hlavní funkci LN. Stačí nám cesta vedoucí k cíli přes jiné uzly a jsme schopni zaplatit bez on-chain platby (Bitcoin transakce, založení kanálu). O to, kudy má daná platba putovat, se stará P2P gossip protokol, který objevuje topologii sítě. Jedná se o hledání cesty v grafu, o jehož topologii můžeme mít (zpravidla vždy máme) neúplnou informaci. Avšak tímto tématem

se budeme zabývat v sekci Pathfinding, nyní nás čeká proces směrování plateb sám o sobě, bez ohledu na jejich cestu.

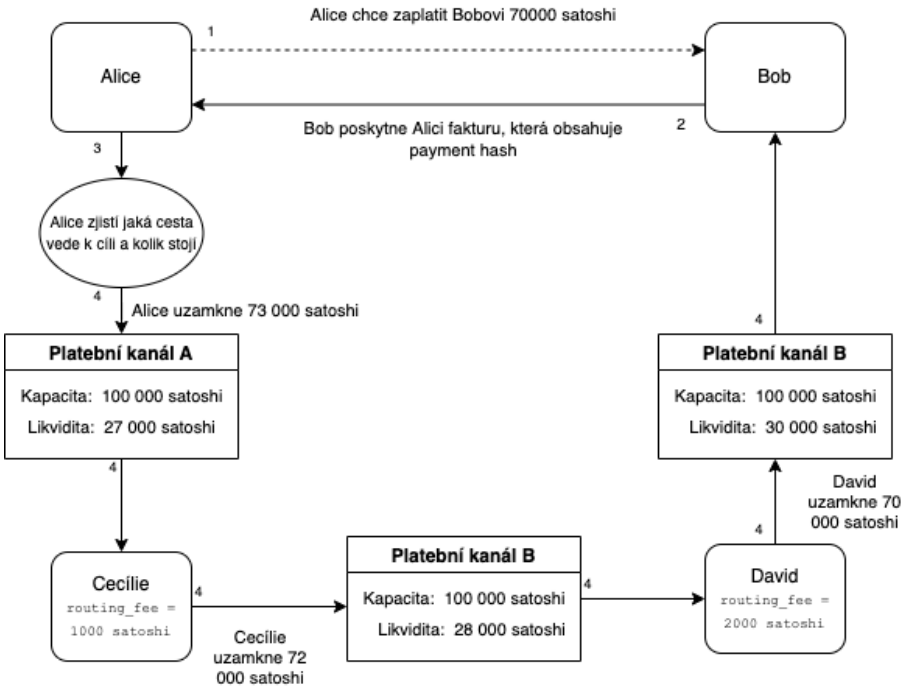
Popíšeme si zde jeden příklad pro lepší uvedení do problematiky.

- Máme 4 uzly: Alice, Bob, Cecílie a David.
- Alice obdržela od Boba fakturu a chce ji Bobovi zaplatit, ale neexistuje mezi nimi platební kanál.
- Místo toho Alice zjistí, že může využít přímé kanály několika uzlů k tomu, aby fakturu Bobovi zaplatila.
- Shodou okolností taková cesta existuje a vede přes Cecílii a Davida.
- Domluví se tedy s Cecílií, že jí pošle peníze, ze kterých si může vzít drobné spropitné, ale musí je poslat Davidovi. Poté se domluví s Davidem, že jakmile mu dorazí peníze od Cecílie, tak si taky může vzít spropitné, ale za každou cenu musí platbu doručit Bobovi.
- Tímto způsobem se zajistí, že každý ze zapojených uzlů bude tímto procesem poznamenán pouze tím způsobem, že obdrží drobné spropitné a jeho vlastní likvidita zůstane nedotčena (pokud přímo nesousedí s odesílatelem nebo příjemcem). Zatímco Alice skutečně zaplatí Bobovi a sníží likviditu svého kanálu směrem k Cecílii a David směrem k Bobovi.

Tento příklad má dvě základní úskalí ve využívání cizích uzlů k doručení platby. První z nich je, jak si můžeme být jistí, že poté, co Cecílie obdrží částku, tak si ji jednoduše neponechá? Výše spropitného (směrovací poplatek) koneckonců nemusí být dostatečnou motivací ke směrování platby.

Aby se tomuto chování předešlo, potřebujeme určitou formu chytré smlouvy mezi zúčastněnými stranami, smart contract. Tento kontrakt bude spočívat v tom, že Alice pošle Cecílii prostředky pouze a jen tehdy, pokud prokáže, že je ona poslala Davidovi a tak dále. Toho docílíme tak, že cílový uzel, v našem příkladě Bob, vytvoří náhodné vygenerované tajné 256-bitové číslo, které zná jen on a stane se předmětem důvěry pro nadcházející transakci. Toto číslo zahashované pomocí SHA-256 se nazývá `payment hash` a předobraz tohoto čísla (Bobovo tajné číslo) se jmenuje `payment secret` nebo `payment preimage`.

Bob nejprve pošle `payment hash` (je součástí faktury) Alici a poté nastane již zmíněný, avšak lehce poupravený řetězec událostí.

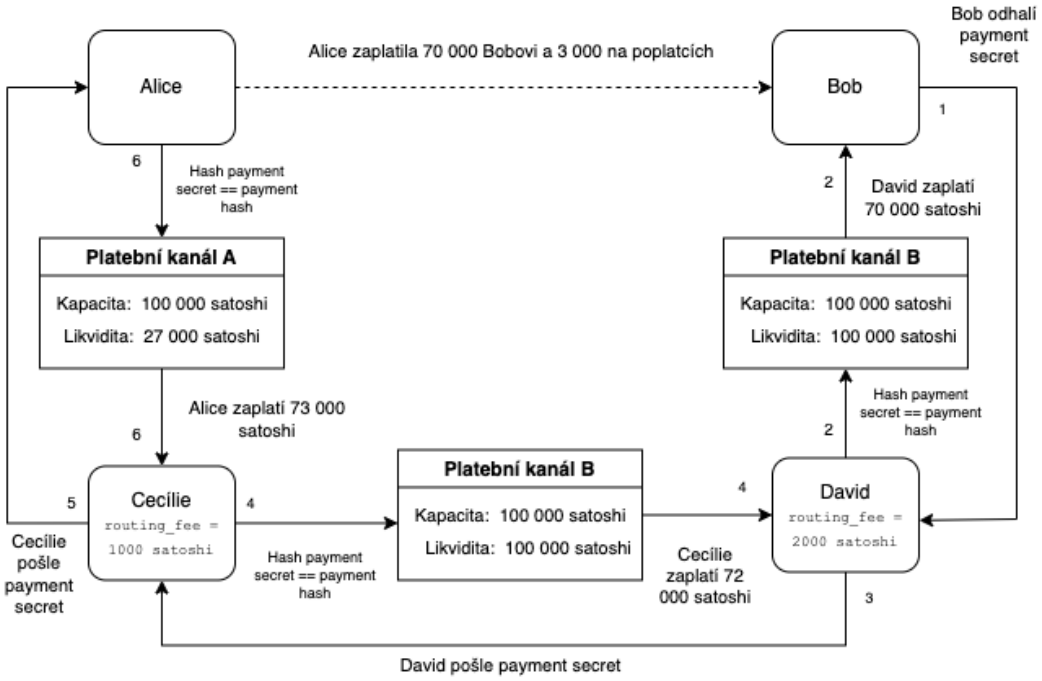


Obrázek 2.6: Probíhající HTLC transakce

Alice pošle svoje prostředky pouze a jen tehdy když jí Cecílie ukáže zprávu (`payment secret`) jejíž SHA-256 hash odpovídá zprávě (`payment hash`), kterou obdržela od Boba. Aby ale toto mohlo nastat, tak se musí prolomit počáteční důvěra. Proto všechny zúčastněné uzly zpočátku pouze uzamykají dané částky do kontraktu a nic neposílají. Alice uzamkne 73000 satoshi do platebního kanálu A a řekne Cecílii, že pokud tuto částku vezme a pošle Davidovi, tak si ještě předtím z toho může odebrat svoje `routing_fee`, 1000 satoshi. Cecílie má před sebou smlouvu, která se neodmítá, poněvadž nemůže nic ztratit[10] (z tohoto důvodu se platby směrují automaticky). Zároveň je incentivizována ziskem poplatku za směrování. Takže nastane stav, že máme v kontraktu uzamčenou určitou likviditu v každém průchozím kanálu až do cílového uzlu (viz obrázek 2.6).

Bob má nyní jistotu, že když odhalí `payment secret` Davidovi, tak že dostane částku o kterou žádal (poněvadž ji David již uzamkl do kontraktu). Podobně David dostane směrovanou částku a poplatek za směrování od Cecílie jakmile jí odhalí pro něj již známý `payment secret` (a tak dále až do výchozího uzlu).

[10]To není zcela pravda, jak si ukážeme v následující kapitole.



Obrázek 2.7: Zakončení HTLC transakce

V tuto chvíli se dostáváme k druhému úskalí směrovaných plateb. Co by se stalo, kdyby se David rozhodl, že nakonec `payment preimage` neodhalí? Bez přidaného mechanismu by to dopadlo tak, že zúčastněné uzly budou mít navždy uzavřené dané prostředky ve smart contractu, ze kterého je nejsou schopny vybrat. Proto se zde zavádí tzv. timelock, časový zámek, a tím získáváme celý název pro tuto třídu kontraktů a tedy HTLC, Hash Time-Locked Contracts. Pokud by nějaká ze stran neukázala té druhé `payment preimage` během předem daného časového intervalu, tak se kontrakt zneplatní a strany budou refundovány z uzamčených prostředků.

Hash Time-Locked Contracts

Již je nám známo, jak funguje směrování plateb v LN, a proto v této podsekci si HTLC rozebereme více podrobněji. Jejich podstatou je tzv. trustless přístup, kde nemusíme žádné zúčastněné straně věřit, protože soudcem je kód. Další důležitou vlastností je atomicita, jak jsme zmiňovali minulý příklad, tak aby se nestalo, že bude např. refundována jen polovina uzlů v kontraktu.

Aby HTLC byly více tolerantní vůči chybám (např. nějaký ze zúčastněných uzlů vypadne), jde platby provádět jak off-chain tak on-chain. Podíváme se nejprve na jeden jejich Bitcoin skript:

```
# To remote node with revocation key
OP_DUP OP_HASH160 <RIPEMD160(SHA256(revocationpubkey))> OP_EQUAL
OP_IF
    OP_CHECKSIG
OP_ELSE
    <remote_htlcpubkey> OP_SWAP OP_SIZE 32 OP_EQUAL
    OP_NOTIF
        # To local node via HTLC-timeout transaction (timelocked).
        OP_DROP 2 OP_SWAP <local_htlcpubkey> 2 OP_CHECKMULTISIG
    OP_ELSE
        # To remote node with preimage.
        OP_HASH160 <RIPEMD160(payment_hash)> OP_EQUALVERIFY
        OP_CHECKSIG
    OP_ENDIF
OP_ENDIF
```

Skript je možné rozdělit do tří základních větví. První je penalizační, podobně jako již známe z přímých kanálů. Pokud k penalizaci nedochází, tak se rozhodujeme mezi vstupem do druhé a třetí větve. Druhá větev představuje HTLC po expiraci a ta třetí odpovídá úspěšné transakci.

Instrukce `OP_HASH160` by mohla být nahrazena instrukci `OP_SHA256` a skript by fungoval stejným způsobem. Důvodem, proč se používá `OP_HASH160`, je úspora místa, poněvadž se jedná o dvojité hashování, nejprve SHA-256 a poté RIPEMD160, které zmenší velikost na 20 bytů.

Nyní si vysvětlíme průchod skriptem, který představuje splácení HTLC vzdáleným uzlem, řádek po řádku v případě validní `payment preimage`. Na vrcholu zásobníku je podpis vzdáleného uzlu a `payment preimage`. Hash `payment preimage` není roven (`OP_EQUAL`) hashi `revocationpubkey` a vstoupíme do větve `ELSE`. Posléze je přidán `<remote_htlcpubkey>` a na zásobníku v tu chvíli máme `<remotehtlcsig>`, `<payment_preimage>`, `<remote_htlcpubkey>`. Poté proběhne `OP_SWAP`, který nám prohodí pořadí položek v zásobníku následujícím způsobem `<remotehtlcsig>`, `<remote_htlcpubkey>` a `<payment_preimage>`. Instrukce `OP_SIZE 32 OP_EQUAL` zkontrolují, že `payment preimage` je velká 32 bytů. Poněvadž v našem případě máme validní `payment preimage`, tak se dostaneme do `OP_IF` větve. Na následujícím řádku se provede kontrola `payment preimage` oproti `payment hash` a skript postoupí dále, poněvadž souhlasí. Na zásobníku máme už jen `<remotehtlcsig>` a `<remote_htlcpubkey>`. Tyto dvě hodnoty předáme instrukci `OP_CHECKSIG`, která zkontroluje jestli podpis souhlasí a výstup HTLC může být úspěšně utracen vzdáleným uzlem.

2.3.5 Poplatky síti

Se směrováním plateb úzce souvisí poplatky síti, ty jsou kalkulovány za průchod uzlem, kde výši poplatku si určí majitel uzlu. V případě nastavení příliš vy-

sokého poplatku stoupne pravděpodobnost, že se jeho uzel bude méně využívat. Další druh poplatků je Bitcoinové síti, ten se platí většinou pouze za otevření a uzavření kanálu.

2.4 Pathfinding

V této sekci se budeme zabývat pathfindingem neboli problémem hledání cesty v grafu, což je klíčová funkce Lightning Network. Pathfinding v LN nemá přesně danou specifikaci formou BOLT standardů jako je zvykem u jiných funkcionalit, tudíž na něj nejsou kladeny nároky jako např. kompatibilita mezi různými implementacemi. Je tedy na tvůrcích implementací, jak jej navrhnou, ale měl by jednoduše plnit svůj účel. Proto také tato sekce nemusí být exaktní pro každou z dostupných implementací. Popíšeme si, jaký problém řešíme a jak je obecně v LN zpracován.

2.4.1 Decentralizace sítě

Pathfinding je především klíčový z hlediska decentralizace. Lightning Network byl stvořen, aby pomohl řešit Blockchain trilemma Bitcoinu. Je možné tvrdit, z dosavadního textu, že LN škálovatelnosti napomáhá a snaží se tím řešit největší problém Bitcoinu. Otázka vyvstává jestli to není za cenu bezpečnosti a decentralizace. Bezpečnost je hlavním aspektem této práce a je zpracována především v následujících kapitolách. Avšak jak si LN stojí na tom s decentralizací? Dle dat z roku 2020[11] bylo v LN 10 % uzlů, které drželi 80 % likvidity celé LN (po dvou letech její existence).[24] Navíc, ve chvíli, kdy by uživatelé začali upřednostňovat mobilní LN peněženky v podobě tenkých klientů, tak lze očekávat, že se toto číslo nebude zmenšovat. Obecně je také pro uživatele snazší otevřít jeden kanál s velkou likviditou k jednomu z největších uzlů, které jsou vyhledatelné například přes 1ml[12], než hledat vhodné sousední peery pro propojení se zbytkem sítě. V tomto centralizovaném scénáři je nanejvýš pravděpodobné, že uzly s nedostatečnou redundancí kanálu budou neoperovatelné v případě pádu nějakého z těchto obrovských uzlů, což může ochromit velkou část sítě. Dalším potenciálním negativním aspektem centralizace je, že cesty k cílovému uzlu mohou být delší, platby se mohou provádět pomaleji a stát více na poplatcích (výši poplatku si stanovuje každý uzel individuálně). Decentralizace LN je silně ovlivněna chováním jednotlivých uživatelů a záleží pouze na nich, jak bude její topologie v budoucnu vypadat. Zajímavou informací je také geografické rozložení LN, kde dle dostupných dat[25] je nejvíce platebních kanálů (pouze snadno dosažitelných, neuvažujeme soukromé a jiné) v USA a to přes 280 tisíc, zatímco na druhém místě je Německo se 3 tisíci kanály.

[11] Další ověřená data, aktuálnější k datu této práce, nebylo možné získat.
[12] https://1ml.com

2.4.2 Volba cesty

Nejvýhodnější cesta k cíli má především následující kritéria:

- cesty s dostatečnou likviditou, aby bylo vůbec možné platbu uskutečnit,
- cesty s nejnižšími poplatky,
- nejrychlejší cesty (nejnižší hodnoty časových zámků).

Pro uspokojení prvního kritéria nám stačí první nalezená cesta k cíli. Pro splnění těch dalších se již jedná o netriviální optimalizační problém, aby při tom zůstala zachována ona proklamovaná rychlost plateb.

Avšak nalezení dostatečné likvidity není tak snadný cíl jak se může zprvu zdát. Likvidita uzlu jedním směrem je zůstatek prostředků po odečtení rezervy kanálu (`channel_reserve`) a aktuálně zpracovávanými HTLC (vzpomeňme si na obrázek 2.6). LN ale pro hledání cesty disponuje pouze počátečním zůstatkem kanálu, který byl ohlášen při tvorbě kanálu. To je způsobeno z hlediska soukromí a škálovatelnosti, poněvadž kdyby každý uzel ohlašoval aktuální zůstatek v daném směru, tak by byla síť podstatně více zahlcená distribucí zpráv. Navíc dostupná likvidita se může měnit každou vteřinu.

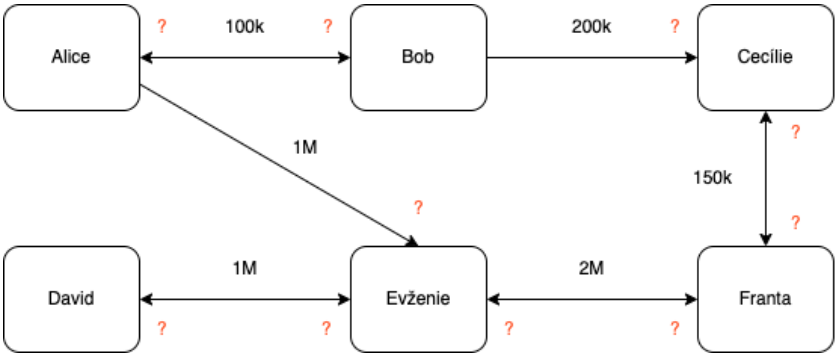
Aktuální implementace se (v ideálním případě) snaží nejprve vytvořit seznam vhodných cest, které posléze seřadí podle výhodnosti a zkouší zdali platba úspěšné dorazí do cíle. Zkoušení těchto cest pak znamená pro uživatele čekání na provedení platby. Nevýhodou oproti on-chain řešení je, že daná dostatečně likvidní cesta nemusí vůbec existovat, zatímco on-chain platba bude provedena vždy a její rychlost záleží na velikosti poplatku a aktuálnímu času do uzavření následujícího bloku (pokud uvažujeme platbu potvrzenou vytěžením jednoho bloku).

2.4.3 Konstrukce grafu

Pro konstrukci grafu je klíčový gossip protokol, který využívá tři důležité zprávy o stavu sítě. Zpráva `node_announcement` obsahuje informaci o uzlu v LN, např. IP nebo Tor adresu, veřejný klíč a další parametry (bod grafu). Druhá zpráva `channel_announcement` nám oznámí vznik kanálu (hrana grafu) a poslední zpráva `channel_update` nám sdělí, pokud došlo ke změně poplatku nebo časovému zámku pro směrování transakcí (váha hrany).

Díky již zmíněným nejistotám spojeným s neznámostí přesné likvidity můžeme tvrdit, že v podstatě pro každý uzel je LN unikátním grafem. [26]

Toto tvrzení si ukážeme na příkladu.



Obrázek 2.8: Přesná likvidita je neznámá

Alice chce poslat 80000 satoshi Frantovi. K Frantovi vedou dvě možné cesty (Alice - Evženie - Franta nebo Alice - Bob - Cecílie - Franta). Předpokládejme, že náš algoritmus upřednostňuje cesty s nejmenšími poplatky a že taková cesta je v našem případě ta kratší. Alice nejdříve vyzkouší poslat peníze přes Boba. K Bobovi platba dorazí, ale k Cecílii již ne. Bob musel v minulosti již Cecílii nějaké satoshi poslat, poněvadž likvidita tohoto kanálu nám nestačila pro přenos 80000 satoshi. V seznamu nám zbývá poslední cesta, takže ji Alice vyzkouší. K Evženii platba dorazí bez problémů a k Frantovi také, takže výsledná cesta byla Alice - Evženie - Franta. Rozhodne-li se nyní Alice, že chce poslat 90000 satoshi Cecílii, tak může využít znalostí z předchozí platby, kterým přiřadí určitou pravděpodobnost pravdivosti, např. podle toho, kolik času od zjištění těchto dat uběhlo. Provádí-li Alice platbu Cecílii bezprostředně potom co ji provedla směrem k Frantovi, tak se vyplatí využít předešlých skutečností. Má-li v plánu poslat Cecílii 90000 a ví že 80000 k ní nedošlo, tak nejkratší cestu ani nezkouší a pošle Cecílii platbu ihned přes Evženii a Frantu.

V dynamickém prostředí když se platba nepodaří (vytížený eshop), tak může být výhodné zkoušet platbu opakovat vícekrát za sebou. Řekněme, že mnoho uzlů chce poslat Davidovi peníze a proto je zde vyčerpaná likvidita jedním směrem. Alice potřebuje Davidovi zaplatit 30000 satoshi, ale opakovaně jí platba selhává. Ve chvíli kdy David zaplatí Evženii 50000 satoshi, tak se zvětší likvidita směrem k Davidovi a v tu chvíli Alici platba projde. Uživatelé obecně a obzvláště pak uzly eshopů a podobných služeb by se měly snažit udržovat své kanály vyvážené.

Z dostupných dat sítě si uzel může být jist jaká je minimální a maximální likvidita kanálů. Poté se pokusy o provedení platby může o síti dozvídat více informací. Může se jevit jako výhodné si tyto nově nabyté informace uchovávat a při příštím hledáním cesty v grafu je uplatnit. Avšak výzvou pro implementaci pathfindingu je právě to, jak s těmito daty efektivně nakládat (rozhodnout, zdali se vyplatí danou informaci uchovávat v paměti).

Pathfinding v LN je postaven na známých algoritmech jako je Dijkstrův nebo A star, avšak již zmíněné úskalí a rostoucí síť mu poskytují dostatek

motivace pro další vývoj a inovace vzhledem k tomu, že se jedná o majoritní část času zpracování transakcí.[2]

2.5 Faktury

V LN uživatelé platí odlišným způsobem, než je zvykem u Bitcoinu. Nelze platit přímým posláním peněz na určitou adresu, ale pouze skrze vystavené faktury. Důvodem, proč LN nepoužívá přímé platby jako Bitcoin, je způsob, jakým probíhají jeho transakce (viz předchozí sekce – příjemce platby musí odhalit `payment hash` všem uzlům vedoucím až k odesílateli).

2.5.1 Formát faktur

Každá faktura obsahuje podpis příjemce, požadovanou částku, volitelný text a payment hash. Zakódovaný řetězec pak může vypadat následovně:

```
lnbc2500u1pvjluezsp5zyg3zyg3zyg3zyg3zyg3zyg3zyg3zyg3zyg3zyg3zyg3
zyg3zygspp5qqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqypqd
pquwpc4curk03c9wlrswe78q4eyqc7d8d0xqzpu9qrsgqhtjpauu9ur7fw2thcl4
y9vfvh4m9wlfyz2gem29g5ghe2aak2pm3ps8fdhtceqsaagty2vph7utlgj48u0g
ed6a337aewvraedendscp573dxr
```

V lidsky čitelné podobě bychom nalezli pouze prefix tohoto řetězce, který značí, o jakou síť se jedná (`ln` + BIP-0173 prefix):

- lnbc – mainnet,
- lntb – testnet,
- lnbcrt – regtest,

a následuje částka transakce:

- m – mili (BTC * 0.001),
- u – micro (BTC * 0.000001),
- n – nano (BTC * 0.000000001),
- p – pico (BTC * 0.000000000001),

v našem příkladu 2500 µBTC což odpovídá 0.0025 BTC nebo 250000 satoshi. Zbytek řetězce je zakódovaná část v bech32 (odděleno znakem `1`), který se používá pro Bitcoin Segregated Witness (SegWit). V této části nalezneme timestamp ve formátu Unix Epoch (`pvjluez`), počet sekund od roku 1970. Dále zde nalezneme zakódovaný `payment_hash` volitelnou zprávu a čas expirace faktury. Nezbytnou součástí je také podpis faktury (512-bit secp256k1),

který slouží k ověření, že byla faktura vystavena cílovým uzlem, a bech32 checksum, kontrolní součet zakódovaného řetězce. Zbývající data jsou key-value páry, které poskytují další důležitá data o transakci a zároveň prostor do budoucnosti, kdyby bylo potřeba nějaká přidat.[27]

2.5.2 QR kód formát

Pro lepší uživatelskou zkušenost se tento řetězec vykresluje zpravidla formou QR kódu. To s sebou může nést bezpečnostní rizika, která již známe z běžného světa internetu, jako je například XSS (cross-site scripting), kde by mohlo dojít ve webové či mobilní aplikaci k podvržení faktury a nepozorný uživatel by tak mohl přijít o své prostředky.

2.6 Implementace

LN je možné dle oficiálního whitepaperu naimplementovat různými způsoby v různých jazycích. Každá z nich má své klady a zápory. Aktuálně mezi nejznámější implementace patří:

- c-lightning (C)
- Eclair (Scala)
- LND (Go)
- Electrum (Python)
- Rust-Lightning (Rust)

2.7 Provoz uzlu

HW nároky jsou doporučovány na 2-4 jádrové CPU, ne méně jak 4GB RAM a 1TB SSD (uvažujeme pro běžné uživatele a ne pro uzly s tisíci kanály). Disk musí pojmout celý blockchain a být dostatečně velký pro další expanzi. V době psaní textu není již instalace LN uzlu záležitostí jen technicky pokročilých uživatelů. Lze využít pomocných installerů a jako počítač Raspberry Pi 4 s nízkou spotřebou elektřiny. Například Umbrel[13] disponuje velmi příjemným uživatelským rozhraním a instalace uzlu rázem zabere jen pár minut. Umbrel využívá implementaci LND, která je obecně nejvíce využívaná (přibližně 87 % uzlů v LN jsou LND).[25]

[13] https://github.com/getumbrel/umbrel

Kapitola 3

Analýza zranitelností

Na základě informací z předchozí kapitoly si představíme bezpečnostní zranitelnosti, které jsou důsledkem návrhu protokolu. Každá následující sekce odpovídající jedné zranitelnosti obsahuje také podsekci, ve které nalezneme doporučení, jak se proti těmto útokům bránit, a pokud to není zcela možné, tak jak alespoň minimalizovat rizika a případně dopady takového útoku.

3.1 Podvrhnutí commitment transakce

Z druhé kapitoly Popis Lightning Network již víme, jak protokol reaguje, když má jeho uživatel úmysly někoho podvést. Co se ale stane, když se Alice pokusí podvést Boba a Bob podvod neodhalí, a co přesně znamená neodhalení podvodu?

Odhalení podvodu závisí pouze na tom zdali je uzel aktivní nebo vypnutý. V případě, že by uzel byl vypnut a útočník by vyslal starou, zneplatněnou commitment transakci[14], tak má Bob přesně specifikovaný čas na to, uzel zapnout a tím podvod odhalit. Tento čas se specifikuje při vytváření kanálu a jeho hodnotou je počet bloků, které je potřeba vytěžit po ohlášení zavření kanálu. Když Bob nestihne uzel zapnout, tak se likvidita kanálu rozdělí mezi účastníky podle commitment transakce, kterou stanovil zavírající kanálu (útočník).

Princip tohoto útoku je jednoduchý, avšak jeho provedení je netriviální. Konkrétně část, kdy při zavření kanálu vyšleme starou commitment transakci. Tuto funkcionalitu nám z dobrého důvodu API známých implementací LN (v běžném sestavení) nenabízí, útočník si ji tedy musí implementovat sám. Navíc jsou implementace nastavené, tak, že pokusí-li se jejich majitel o podvod, tak ho ihned potrestají i bez pomoci druhé strany (původně oběti).

Tento typ útoku je detailněji popsán v sekci 4.1, kde je i názorně předveden[15].

[14] Aby se stále jednalo o útok, tak se předpokládá, že to bude taková transakce, ve které má útočník více prostředků než v čase aktuální commitment transakce.

[15] Útočníkova implementace je ale skryta, aby případný zlomyslný čtenář této práce nemohl útok jednoduše zreprodukovat v reálném světě.

3.1.1 Doporučení

Z hlediska návrhu LN není v tuto chvíli jiné známé řešení, než mít jiný uzel, který hlídá ten druhý, v případě, že by byl vypnut. Tomuto principu se říká Watchtower a je také detailně rozebrán v praktické části této práce.

3.2 Griefing

Podobně jako ve světě běžných sítí, i na LN existují Denial of Service útoky. Základní obranou je, že za využití prostředků, které poskytují jednotlivé uzly, se platí poplatky (směrování plateb). Avšak poplatky jsou dle navržení protokolu vybírány pouze za úspěšné platby, když platba bude neúspěšná tak se poplatky neuplatní. To je pozitivní vlastnost pro uživatele, že nemusí platit za nepovedené platby, ale přináší to další vektor útoku. Útočník může bez nákladů zahlcovat síť neúspěšnými platbami a tím donutit přetížené kanály k jejich uzavření.[28]

K jejich přetížení může dojít následujícím způsobem:

- útočník otevře kanál s částí sítě či jedním uzlem (oběť) který chce přetížit,
- zároveň otevře vlastní další kanál do kterého vede cesta od oběti,
- přes oběť posílá transakce (HTLC) na svůj uzel a na něm neodpovídá s HTLC secret a čeká,
- jakmile útočník přesáhne maximální povolený počet HTLC dotčených kanálů, tak oběť není schopna nadále směrovat platby.

Jedním z útoků, který se řadí také do třídy DoS, je Griefing. Spočívá v tom, že útočník začne zahlcovat oběť (kanály uzlu LN) velkým množství mikro-transakcí. Zpravidla jsou uzly nastaveny, že jsou schopny odbavovat až 483 transakcí najednou (parametr `max_accepted_htlcs`). Útočníkovi stačí přesáhnout tuto hodnotu a udržovat kanály zahlcené. V tomto útoku nedojde ke krádeži prostředků, avšak napadené kanály oběti se stanou neprůchozími a oběť je nucena kanály jednostranně uzavřít (Force close). Ve výsledku je oběť nucena zaplatit vysoké poplatky za jednostranné uzavření kanálů.[29, 28]

3.2.1 Doporučení

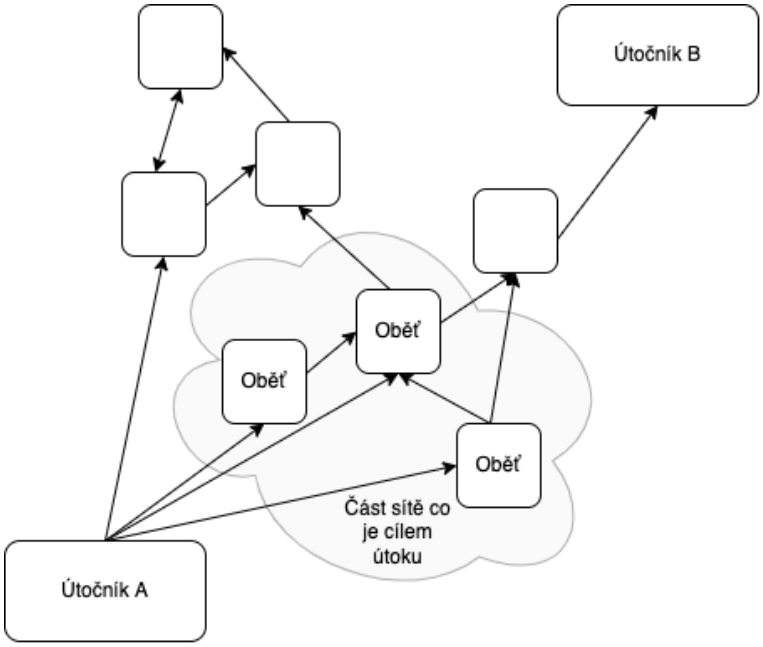
Bohužel implementace LN nejsou v tuto chvíli schopné tento problém řešit globálně, avšak na základě této zranitelnosti vznikl pomocný nástroj, který se snaží tento problém řešit. Jmenuje se Circuit Breaker[16]. Umožňuje jednotlivým uzlům komplexnější správu limitů a funkce jako například rate-limiting,

[16] https://github.com/lightningequipment/circuitbreaker

který je běžně používán pro webové aplikace. Nástroj je ovšem v alpha vydání a neposkytuje žádné záruky.

3.3 Flood & Loot

Pokročilejším útokem ze skupiny DoS je Flood & Loot Attack. Tento útok využívá principů Griefingu a především HTLC timelocků (časových zámků). Příprava na útok opět spočívá v tom, že má útočník dva uzly, mezi kterými si bude posílat transakce, a uzly, přes které transakce prochází, jsou oběťmi útoku. Toto lze požadovat za první fázi útoku.



Obrázek 3.1: 1. fáze: Rozestavění uzlů při útoku

Druhou fází útoku je v optimální čas začít zahlcovat síť uzlů, kterou útočník obklopil. Hodnotu transakce, kterými zahlcuje jednotlivé kanály, je nejvhodnější vybrat tak, aby maximalizoval jeho zisk. Pokud označíme parametr `max_htlc_value_in_flight` (maximální hodnota HTLC) jako H_{max}, `max_accepted_htlcs` (maximální počet HTLC) jako P_{max}, disponibilní zůstatek kanálu jako B a poplatek jako F, dostaneme následující rovnici:

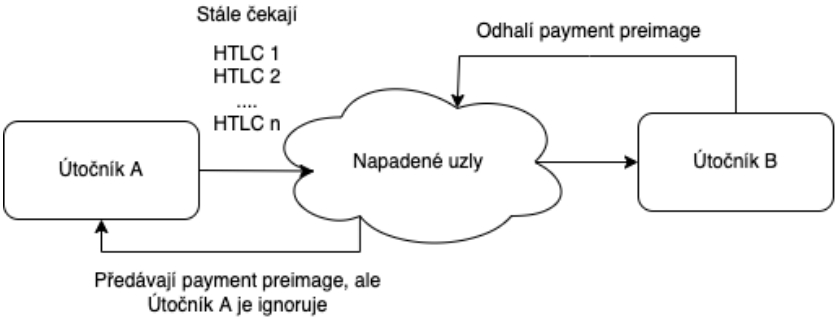
$$\text{ideální hodnota HTLC transakce určené k útoku} = \frac{\min(H_{max},\ B - F)}{P_{max}} \quad (3.1)$$

Dalším důležitým parametrem útoku je, že každá útočníkova HTLC transakce musí mít stejný čas expirace a ne lineárně větší se vzdáleností jako tomu

bývá obvykle. S těmito informace může útočník začít zahlcovat síť, tím že posílá platby na svůj uzel.

Obrázek 3.2: 2. fáze: Zahlcování HTLC transakcemi [3]

Třetí fáze je, že cílový uzel (Útočník B) začne odpovídat. Vyšle zpátky do sítě `payment_preimage`, ale nyní pro změnu přestane odpovídat výchozí uzel (Útočník A).



Obrázek 3.3: 3. fáze: Cílový uzel odpovídá

V tuto chvíli všechny uzly poslaly svoje uzamčené prostředky dál až na ty které mají kanál přímo s Útočníkem A. Z tohoto důvodu jsou postižené uzly jen ty které mají přímý kanál s Útočníkem A. Poněvadž tyto uzly (oběti) nemohou získat svoje prostředky přes LN, tak jsou nuceny jednostranně uzavřít kanál (Force close) a získat svoje prostředky on-chain. Transakce, které vysílají na blockchain, jsou HTLC-success, tedy že HTLC transakce proběhla úspěšně, což je pravda, poněvadž Útočník B již svoje prostředky má a tyto uzly jsou poslední zbývající, které nebyly ještě refundovány. Stále to vypadá, že je všechno v pořádku a LN myslel na všechno a nikdo nebude okraden. To se ve 4. a poslední fázi změní.

Tím, že se útočí na několik uzlů zároveň a Bitcoin neškáluje, tak může nastat situace, že všechny on-chain transakce obětí se nestihnou vytěžit před tím než HTLC vyexpiruje. Jakmile dojde k expiraci HTLC, tak Útočník A může prohlásit všechny čekající HTLC transakce za vyexpirované, HTLC-timeout, a vyšle na blockchain protichůdnou informaci oproti napadeným uzlům. Avšak

útočník má pravdu, že jsou vyexpirované, takže mu pouze stačí využít replace-by-fee mechanismu, přeplatí transakce obětí v mempoolu a tím se jeho transakce vytěží dříve. To je 4. fáze. Při vhodných podmínkách je útočník schopen ukrást veškerou likviditu kanálů.

Důvodem proč je v tomto případě předbíhání v mempoolu tak snadné, je, že oběť vysílá HTLC-success transakci, která potřebuje podpis obou stran, zatímco útočník vysílá HTLC-timeout, které stačí pouze jeden podpis. Kdyby oběť chtěla navýšit poplatek tak potřebuje zároveň svolení útočníka, zatímco útočník si může zvolit poplatek libovolný a navíc přesně ví, jaký poplatek za zpracování transakce platí oběť.[3, 28]

3.3.1 Doporučení

Uživatelé, respektive jejich implementace LN, by se měli snažit, aby neměli příliš mnoho čekajících HTLC transakcí, poněvadž se pak spíše stanou cílem podobného útoku. K tomu jim může mimo jiné pomoci nástroj Circuit Breaker, který jsme již zmiňovali i ve zranitelnosti Griefing. Dalším prvkem, který by mohl pomoci, je rozumné nastavení parametru určujícího expiraci HTLC po vyslání commitment transakce. Čím vyšší hodnota tím menší šance, že se transakce oběti nestihne zpracovat, na druhou stranu v případě čestných uzavření kanálu se bude jednat o nepříjemnost, poněvadž uživatelé budou déle čekat na opětovné získání svých prostředků.

Nejsilnějším způsobem obrany by bylo zamezit výše uvedené nahraditelnosti transakcí, avšak to by vyžadovalo změnu standardů, podle kterých jsou implementace definovány. LN je stále v raném vývoji a tak se této změny (která nevyvolá jiné problémy) třeba v budoucnu dočkáme.

3.4 Time-dilation Attack

Jedná se o třídu útoků na LN, které využívají Eclipse Attack na Bitcoinu pro manipulaci s časem. Nejdříve si popíšeme Eclipse Attack, kde jeho úspěšné provedení nám posléze umožní provést Time-dilation attack na LN.

Eclipse Attack spočívá v tom, že Bitcoinový uzel oběti obklíčíme pomocí uzlů útočníka a tím bude oběť komunikovat (v rámci Bitcoinové sítě) pouze s uzly útočníka. Peer-to-peer sítě nejsou schopné komunikovat se všemi uzly najednou a využívají pouze několika sousedních peerů pro komunikaci se sítí. Ve chvíli, kdy daný uzel (oběť) disponuje ke komunikaci pouze peery, které jsou ovládány například jednou osobou (útočníkem), tak ta daná oběť je schopna oboustranně pracovat pouze s daty které vedou přes útočníka. Útočník pak může poskytovat oběti jiná data o blockchainu než jsou v souladu s aktuálním stavem zbytku sítě, protože je izolována. Dosáhnout této izolace může být sice obtížné, ale dle dostupných studií se nejedná o neproveditelný útok.[30] Zdali se podařilo uzel izolovat, lze zjistit například tak, že se útočník pokusí zpozdit doručení bloku k oběti a bude sledovat, jak na to oběť zareaguje. Pokud bude

propagovat také zpožděný blok, pak lze předpokládat, že byla izolace úspěšná. Je-li pak oběť například obchodník, který akceptuje platby s 0 potvrzeními a má izolovaný uzel, tak útočník si u něj něco nakoupí a zároveň ty samé prostředky utratí někde jinde. Oběť tuto platbu vyšle do sítě, ale útočník ji zdrží. Útočníkova transakce se vytěží ve zbytku sítě a transakce oběti bude zneplatněna. Pokud daný izolovaný obchodník vyžaduje nenulové potvrzení platby, tak musí útočník izolovat i nějaké těžaře, což mu opět umožní provést double-spending a bude se jednat v podstatě o 51 % útok na izolované části sítě.[31]

Tato manipulace s daty je klíčová pro Time-dilation Attack. Útočník si nejdříve podle IP adres nalezne bitcoinové klienty, které mají zároveň LN uzel. Poté využije své schopnosti zpožďovat doručení bloků. Nyní si popíšeme, jak by bylo možné provést známý útok pomocí podvržení commitment transakce ze sekce 3.1, kde nebudeme potřebovat, aby uzel byl vypnut.

Útočník bude doručovat oběti zpožděné bloky, takovým způsobem, aby dosáhl rozdílu, že výška blockchainu pro oběť bude aktuální výška (H) s odečtením počtu bloků potřebných pro odhalení transakce (P).

$$\begin{aligned} \text{výška blockchainu oběti} &= H - P \\ \text{výška blockchainu zbytku sítě} &= H \end{aligned} \tag{3.2}$$

Poté provede s obětí novou platbu (nevratnou, něco od ní za to dostane) a dohodnou se tím na nové commitment transakci. Útočník vyšle do sítě starou zneplatněnou commitment transakci, která bude potvrzena ve výšce H + 1. Útočník pak pokračuje s doručováním bloků oběti a oběť má čas P neplatnou commitment transakci odhalit. Nicméně vzhledem k tomu, že ji odhalí až za H + P + 1, tak již bude pozdě a útočníkovi se podařil double-spending útok. Této schopnosti útočníka lze využít pro celou řadu útoků, jako byl třeba nastíněn v sekci 3.3 Flood & Loot, kde velmi citelně záleží na času (timeout HTLC transakcí).[31]

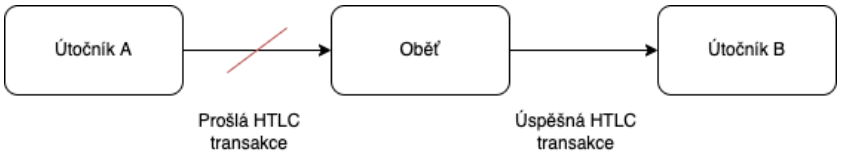
3.4.1 Doporučení

Nejvhodnější obranou se jeví být propojený s dostatkem čestných uzlů. Na druhou stranu ve chvíli, kdy uzly budou komunikovat jen se svými známými uzly, tak to bude brzdit decentralizaci a připojení nových uzlů do čestné sítě. Nadále pro případ útoku pomocí podvržení commitment transakce by nám pomohl Watchtower, který by byl mimo izolovanou síť. Nakonec by mohl pomoci i nějaký heuristický algoritmus, který by se pokoušel identifikovat pokusy o tento útok, avšak je možné, že by znepříjemnil provoz uzlu, poněvadž by generoval mnoho false-positive či false-negative případů (čas těžby bloku je nekonzistentní). Teoreticky je velmi těžké se bránit tomuto útoku v případě, že útočník disponuje velkým množstvím uzlů (IP adres), které mohou potenciálně nahradit důvěryhodné peery. Do Bitcoinu byly ale již naimplementované různé

algoritmy, které se snaží tomuto útoku předcházet, jako je randomizace výběru peerů nebo větší kapacita na uložení adres.[32, 31, 33]

3.5 Pinning

Pinning je poměrně komplexní útok k provedení, avšak stále rizikový. V mnoha aspektech je podobný Flood & Loot útoku, ale na rozdíl od něj nespoléhá na zahlcenost Bitcoinového blockchainu. Útočník potřebuje kanál ze svého uzlu směrem k oběti a také kanál z oběti do svého dalšího uzlu. Útočník A poté pošle HTLC transakci do svého uzlu (Útočník B) skrze oběť. Avšak na svém cílovém uzlu nezareaguje s odhalením payment secret. Za běžných okolností zde mohou nastat přesně dva scénáře ze kterých oběť vždy vyvázne bez újmy. První je, že Útočník B odhalí payment secret, kterou oběť použije, aby získala směrovanou částku. Druhá varianta je, že uplyne daný čas životnosti HTLC a bude si moc nárokovat částku zpět. Nicméně existuje ještě jiná varianta. LN uzly sledují pouze blockchain a nevidí na mempool. Tohoto faktu využije útočník. Vyšle transakci HTLC success (nárokování prostředků z HTLC) se záměrně velmi nízkým poplatkem[17] a zakázaným replace-by-fee mechanismem, kde odhalí payment secret a oběť vyšle HTLC timeout transakci, poněvadž dojde k vypršení HTLC. Těžaři odmítnou těžit transakci s takto nízkým poplatkem a transakci oběti také nevytěží, poněvadž se pokouší utratit stejné UTXO. Oběť nezná payment secret a nemůže protlačit svoji transakci do bloku, kvůli útočníkově HTLC-success transakci v mempoolu. Bitcoin používá dva způsoby navyšování poplatků transakcí. Jedním je již zmíněný replace-by-fee mechanismus, který lze ale zakázat. Druhým způsobem je CPFP (Child Pays For Parent) a ten umožňuje urychlit zpracování jakékoliv transakce, avšak aby byla vykonána druhá transakce (která platí vysoké poplatky pro exekuci té původní) tak musí být vykonána i ta původní. Oběť je tedy tzv. přišpendlena a nemůže nic dělat. Mezitím vyprší HTLC mezi vysílajícím útočníkovým uzlem a obětí, a útočník je refundován vysláním HTLC timeout transakce. Oběť už nemá ani šanci dostat prostředky z vysílajícího uzlu. Jakmile útočník obdržel prostředky z prvního kanálu, tak může přesunout i ty z druhého.



Obrázek 3.4: Nekonzistentní HTLC transakce

[17]Poplatek je záměrně nízký, protože útočník nechce, aby byla transakce vytěžena před tím, než jeho vysílající uzel bude schopen získat prostředky z HTLC mezi vysílajícím uzlem a oběti, protože kdyby se tak stalo, tak bude útok neúspěšný a oběť zůstane neokradena (jak prekérní).

Útok skončí ve stavu, že první část HTLC transakce (od útočníka k oběti) je vyexpirovaná, prostředky se navrátily útočníkovi, zatímco její druhá část (od oběti k útočníkovi) je platná a prostředky se přesunuly od oběti k útočníkovi.[3, 34]

Oběť přišla o částku, kterou v dobré víře směrovala.

3.5.1 Doporučení

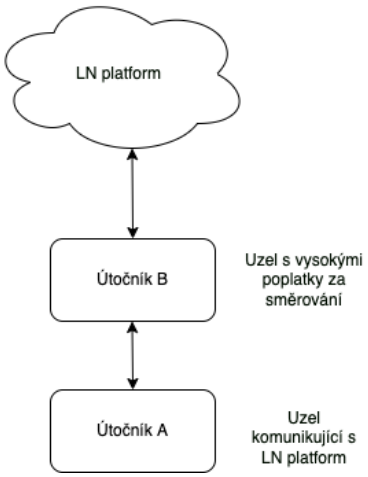
Technika, která může pomoci zabránit tomuto útoku (ale neřeší dokonale celou třídu těchto útoků, kde existuje několik variací jak ho provést) jsou tzv. anchor outputs[18]. Ty jsou již nově součástí implementací (i standardu BOLT 5) a umožňují zvyšovaní poplatků u commitment transakcí, takže oběť by mohla nechat první transakci vytěžit a k útoku by nedošlo. Druhou potenciálně mocnější obranou je (zatím neschválený) návrh pro Bitcoin zvaný Package relay[19]. Tato funkcionalita nabízí uzlům zasílání transakcí v balíčcích, kde je poplatek určován za celý balíček podle toho jaké jsou v něm transakce (podle výše jejich poplatků). Využití této funkcionality by vedlo k tomu, že útočníci nebudou schopni dokončit svůj útok.[3, 34]

3.6 Fee Siphoning Attack

Tento útok je více konkrétní avšak velmi zajímavý tím, jak využívá mechanismy LN a proto si zaslouží zmínku i v této práci. Uživatel Reckless_Satoshi provedl v září 2019 analýzu zranitelnosti následujících platforem vůči popisovanému útoku: Bitfinex, OKex, Muun, WalletOfSatoshi, LNMarkets and Southxchange. Zranitelnost byla oznámena až po opravě na příslušných platformách, takže nadále již není relevantní, ale to neznamená, že by se v budoucnosti nemohla opakovat.

Útok zneužívá fixních poplatků za transakce z burz a podobných služeb. Útočník v podstatě pouze vkládá své prostředky na burzu a opakovaně je vybírá a tím získává více peněz. Háček spočívá v tom, jak. Je třeba si připravit dva uzly s dostatečnou likviditou (podle toho, kolik plánuje získat peněz), kde z uzlu A útočník posílá prostředky přes svůj uzel B do dané platformy.

[18] https://bitcoinops.org/en/topics/anchor-outputs/
[19] https://bitcoinops.org/en/topics/package-relay/



Obrázek 3.5: Nastavení útočníkových uzlů

A pak jen na uzlu B nastaví vysoké poplatky. Takže vložení prostředků na burzu zaplatí vysoké poplatky sám sobě a při výběru peněz z burzy zaplatí burza vysoké poplatky jemu tím, že využívá fixní hodnotu pro stanovení poplatků.

Pro uvedení příkladu zmíníme tehdejší stav Bitfinexu. Bitfinex měl nastavené fixní poplatky na 100 satoshi. Takže stačilo si nastavit na uzlu B poplatky například na 1000 satoshi a tím každý cyklus vložení a výběru peněz uživateli vydělal 900 satoshi.[35]

3.6.1 Doporučení

Platformy využívající LN pro vklady a výběry by měly vždy počítat s tím, že přestože jsou poplatky na LN velmi nízké, tak při nedostatečném ošetření volby poplatků může dojít ke ztrátě jejich prostředků.

3.7 Dust limit exploitation

Kanály disponují parametrem zvaným `dust_limit_satoshis`, který způsobuje, že všechny satoshi pod tímto limitem budou ignorovány v commitment transakci a dostanou je těžaři formou poplatku za transakci. Tohoto parametru bylo v minulosti možné zneužít ke krádeži prostředků, poněvadž byl nedostatečně regulovaný.

Uvedeme si konkrétní scénář útoku. Útočník A má otevřený kanál k Oběti a od ní vede kanál do Útočníka B. Když Útočník A otevírá kanál, tak nastaví `dust_limit_satoshis` na největší možnou hodnotu, kterou ví, že oběť automaticky přijme (`max_dust_limit_satoshis` – záleží na implementaci). V našem případě je maximem 20 % kapacity kanálu. Útočník A nyní může

provést 4 platby přes Oběť do Útočníka B, kterými využije 80 % kapacity kanálu. Útočník B může přijmout platby a Útočník A může vyslat commitment transakci, která neobsahuje žádné HTLC, poněvadž se jedná o částky, které dostanou těžaři. Oběť nemůže nic získat on-chain a už zaplatila Útočníkovi B.

V tomto příkladě útočník nic nevydělal, pouze okradl oběť o částku, kterou nuceně daroval těžařům. Teoreticky by se ale útočník mohl domluvit s těžaři, kteří z těchto operací profitují, a systematicky tímto způsobem okrádat uživatele.[36]

3.7.1 Doporučení

Používat nejnovější stabilní verzi implementací, kde je tato chyba vyřešena. Novější implementace (např. c-lightning $\geq$ v0.10.2) již kontrolují, zdali není `dust_limit_satoshis` příliš velký.

3.8 Balance probing

Jak již víme z předchozí kapitoly tak LN se snaží být více anonymní než Bitcoin. Avšak když se vydáme hlouběji, tak zjistíme, že v mnoha ohledech na to nelze spoléhat, jako jsou například private a public kanály, které se liší jen v tom, že private kanál svoji existenci neoznámí do sítě, ale vlastnosti si vydržuje jinak nadále stejné. V této sekci si předvedeme, že jsme schopni například zjistit aktuální stav prostředků v cizích kanálech, kde tato hodnota je většinou mylně označována za zcela soukromou.[26]

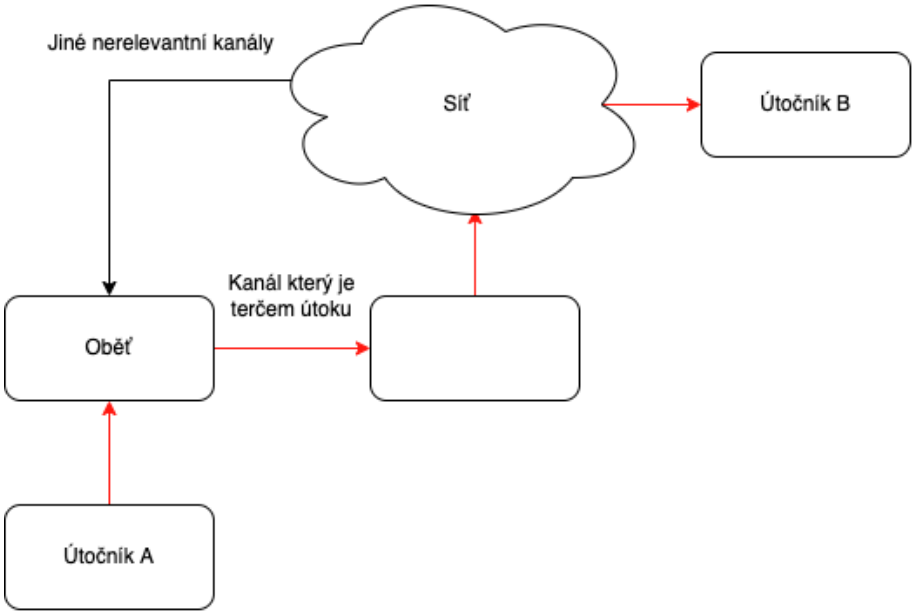
Útočník (v ideálním případě), aby ho útok nestál žádné prostředky (až na poplatky za směrování, případně otevření a kanálů), tak potřebuje dva uzly. Jeden uzel, do kterého vede dostatečně likvidní cesta z sítě, ve které se snaží zjistit, kdo má kolik satoshi ve svém kanálu, a druhý uzel, ze kterého to bude zjišťovat. Jakmile disponuje prvním uzlem, tak dle jeho propojení dostane množinu kanálů, na které může zaútočit. Poté si vybere kanál který ho přesně zajímá, a k němu vytvoří kanál s dostatečnou likviditou. Dostatečná likvidita zde představu hodnotu, do které je útočník schopen odhalit zůstatky kanálu. V případě, že zná kapacitu kanálu, tak ví, že mu stačí vytvořit kanál o takové kapacitě ($K_{Ú}$), kterou disponuje kanál na který útočí (K_O). Zůstatek oběti v daném kanálu můžeme pak označit jako Z_O.

$$Z_O \leq K_O \leq K_{Ú} \tag{3.3}$$

Jediným potenciálně netriviálním krokem je zajistit dostatečnou likviditu k jeho prvnímu kanálu[20], který ale nemusí být přímo spojený s kanálem který

[20]Tento krok, ale lze s upravenou implementací přeskočit, poněvadž můžeme zfalšovat

je terčem útoku. Ve chvíli, kdy má útočník připravené uzly, tak může zahájit útok.



Obrázek 3.6: Odhalení zůstatků v kanálu

Pro útok je potřeba jednoduchý algoritmus naimplementovaný např. v skriptovacím jazyce Bash nebo Python. Cílem algoritmu je obdržení faktur z cílového uzlu, které bude výchozí uzel proplácet. Částka za faktury bude začínat na kapacitě kanálu a poté se bude vhodným způsobem dekrementovat. Pokud první platba proběhne, tak to znamená, že oběť útoku vlastní celkovou kapacitu kanálu, jinak budou platby vraceny kvůli nedostatečné likviditě (nedostatek prostředků oběti pro HTLC). Selhání platby nestojí žádný transakční poplatek a dozvíme se o něm okamžitě poněvadž se jedná o první hop. Faktury lze vystavovat na neurčitou částku což usnadní celý proces, poněvadž skriptu můžeme předat jen seznam několika aktivních faktur. Částka, o kterou budeme iterace dekrementovat, pak odpovídá přesnosti odhadu zůstatku oběti. Pro lepší názornost můžeme nahlédnout do následujícího pseudokódu.

```
n = přesnost odhadu
i = kapacita kanálu

opakuj dokud i >= 0
```

payment hash, což nám zaručí, že platba selže v obou případech, avšak na jiných místech a tyto případy jsou z chybové hlášky rozeznatelné (nedostatek likvidity při prvním hopu, nevalidní payment hash až v cíli).

```
        pokud je na vstupu faktura k proplacení
            proplať fakturu za i
            pokud se podařilo fakturu proplatit
                konec cyklu
            jinak
                i -= n
            konec podmínky
        jinak
            vypiš nedostatek faktur k proplacení
            konec programu
        konec podmínky

    vypiš zůstatek oběti se rovná = i
    konec programu
```

Tento útok může mít třeba jen velmi specifické případy využití, avšak vzhledem k jeho nenáročnosti na vnějších faktorech lze vytvořit komplexnější algoritmus, který potenciálně může sledovat tímto způsobem pohyby v celých sítích a anonymita LN může být rázem pouze jednostranná, poněvadž ti, kteří budou disponovat výpočetním výkonem na exekuci tohoto upraveného algoritmu, mohou systematicky sledovat pohyby uživatelů jako je to možné na Bitcoinu.[37, 26]

3.8.1 Doporučení

Bohužel v tuto chvíli neexistuje nativní ochrana, která by uživatelům poskytla soukromí ohledně jejich zůstatků v kanálech. Avšak řešení, které by mohlo Balance probing ztížit, je například obfuskace směrování, že v našem příkladě by oběť využila jiný likvidnější kanál i za cenu delší cesty k cíli, takže by se ubránila útoku, ale pro uživatele by to znamenalo, že nebude znát cestu své platby, náklady na poplatky se zvýší a síť bude více zatěžována. Jiné, nicméně poměrně komplexní řešení, by byl automatický rebalancing kanálů, tedy vyvažování zůstatků pouze pro účel té dané platby. Uzel by chtěl směrovat platbu útočníka kanálem, na kterou nemá, a tak pro ten moment by si půjčil likviditu z jiného a na konci transakce ji vrátil. Tyto změny by ale byly velmi zásadní pro celý protokol, takže se ukáže časem, jak se komunita LN postaví k anonymitě protokolu. Jinak, bez úpravy protokolu si může potenciálně uživatel naimplementovat nějaký heuristický algoritmus, který bude schopen identifikovat velké množství chybových transakcí na jeho uzlu a blokovat je. Tyto transakce může být poměrně snadné identifikovat např. podle dekrementace částky v našem předchozím příkladě, případně změna částek ve fakturách může následovat vzor Binárního vyhledávání, který bude spíše využit pro systematické skenování sítě.[37]

Kapitola 4

Simulace útoků

Každá následující sekce představuje jeden konkrétní útok na protokol. Nalezneme v nich detailní popis kroků vedoucích k jejich realizaci. Veškeré útoky byly provedeny v testovacím prostředí na lokálním blockchainu a žádný reálný uživatel nepřišel o své prostředky ani nebyl jinak negativně ovlivněn.

4.1 Krádež prostředků pomocí podvrhnutí commitment transakce

Předpokladem tohoto útoku je výpadek připojení k síti na straně oběti. Jeho pravděpodobnost je přesto velmi vysoká (zejména u mobilních peněženek, které zároveň operují jako uzel sítě).

4.1.1 Nastavení testovacího prostředí

Pro simulaci útoku v testovacím prostředí jsem využil open-source software Polar[21]. Tento software využívá k virtualizaci Docker Engine[22] a umožňuje mi využít nejpoužívanější implementace LN, jmenovitě LND, Eclair a c-lightning. Navíc je zde i možnost použít vlastní sestavené implementace v podobě Docker Image, což je klíčová funkcionalita pro provedení tohoto útoku. Pro inicializaci lokálního blockchainu využívá běžného Bitcoin klienta v tzv. regtest módu. Frontend Polaru má zatím omezené funkce, avšak pro účely demonstrace útoku stačí schopnost vytvářet uzly, které jsou napojeny na backend (Bitcoin), a příkazová řádka spuštěná v jednotlivých kontejnerech.

Pro účely tohoto útoku vytvořím dva LN uzly. Jeden náleží útočníkovi a druhý oběti. Mezi útočníkem a obětí je otevřený kanál s likviditou o velikosti 0.01 BTC (1M satoshi). Útočník má na začátku 0.02 BTC a oběť nic. Pro účely tohoto útoku jsem se rozhodl, že útočník použije upravenou implemen-

[21] https://github.com/jamaljsr/polar
[22] https://docs.docker.com/engine/

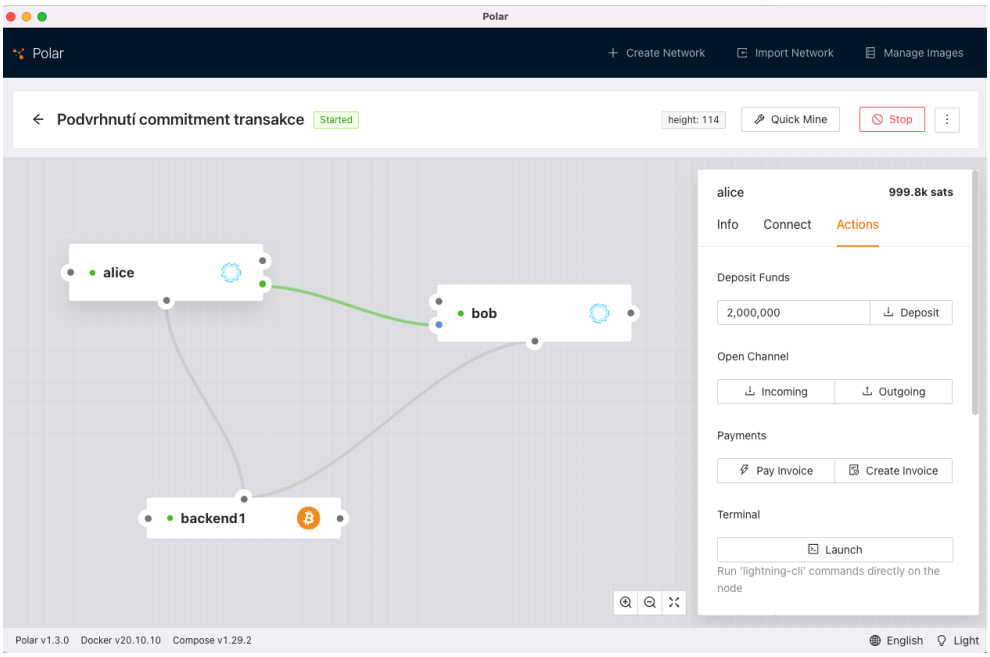
taci c-lightning (verze 0.8.1), poněvadž disponuje několika užitečnými příkazy, které jiné implementace postrádají, a je snadno upravitelná a rozšiřitelná. Oběť může mít jakoukoliv implementaci, ale pro jednoduchost (stejný formát příkazů) použijeme stejnou, jakou má útočník, pouze originální.

4.1.2 Útok

Nejprve si musí útočník (Alice) upravit implementaci svého uzlu, aby byl schopen útok provést, poněvadž penalizační mechanismus funguje oboustranně. Kdyby Alice chtěla zaútočit s originální implementací, tak ve chvíli, kdy by se pokusila podvést druhou stranu, tak sama sebe nahlásí, že podvádí a původní oběť útoku (Bob), jakmile svůj uzel opět připojí do sítě, převezme veškerou likviditu kanálu.

Úprava implementace pro útok je popsána v Příloze C, která není součásti tohoto dokumentu a je zašifrována společně se zdrojovými kódy. Heslo bylo sděleno pouze osobám nutným pro hodnocení této práce.

Jakmile je implementace upravena, lze začít otevírat kanály. V našem případě je veškerá počáteční likvidita směrem od Alice k Bobovi (1M satoshi).



Obrázek 4.1: Nástroj Polar

Alice začne Bobovi proplácet vystavené faktury. Bob vystaví fakturu na 50000 satoshi, kterou mu Alice proplatí. Faktura se pak vystaví následujícím způsobem:

```
clightning@bob:/$ lightning-cli invoice 50000000 "vecere"
↪  "Faktura za veceri"
{
   "payment_hash":
   ↪  "424c77d10c34df567734908577cc5d01b553958aee157a26e6
   ↪  d2d1e4b744efd3",
   "expires_at": 1651230742,
   "bolt11": "lnbcrt500u1p3x9rvkpp5gfx805gvxn04vae5jzzh0nzaqx64
   ↪  89v2ac2h5fhx6tg7fd6yalfsdqugeskkar4wfsjq7npypmx2cm
   ↪  9wf5sxqyjw5qcqp2sp50z44h63danmz3nyflqx7f2uutas30e4
   ↪  rfudkwktj2w3wv04qyttq9qy9qsq7lr0r6aze7xcrfc2lmwk0t
   ↪  gvsehq6juy47s25rfwprw4hp3ur83338r0s5706y6u67kxcl8m
   ↪  j2euhxkdt7ekd02w6dazcla7g3ut g7gq5lxk6c",
   "warning_deadends": "No channel with a peer that is not a
   ↪  dead end"
}
```

U obou stran se můžeme podívat, jak na tom jsou aktuálně se svými prostředky. Výpis vždy obsahuje dva různé typy prostředků. Prvním typem je `outputs`, který obsahuje prostředky na Bitcoinové peněžence propojené s LN uzlem. Druhým typem jsou `channels`, kde jsou znázorněny prostředky, které jsou uzavřené v kanálech. Ty můžou být vyloženě pouze off-chain, ale i on-chain, které jsou v uzamčené v rámci multisig adresy a ještě nebyly delegovány na Bitcoinové adresy uživatelů uzlů. V případě, že jsou on-chain, tak parametr stavu kanálu, `state`, obsahuje řetězec `ON_CHAIN`. Z výstupu příkazu se dozvíme mimo jiné identifikátor kanálu, kapacitu, aktuální zůstatek u dané strany a `peer_id`, které představuje veřejný klíč uzlu s kterým je uzavřen kanál.

```
clightning@bob:/$ lightning-cli listfunds
{
   "outputs": [],
   "channels": [
      {
         "peer_id":
         ↪  "02b3150bbeda79abc37d222cb2e62482c0320c9294db47dc8b
         ↪  6487f30357503960",
         "connected": true,
         "state": "CHANNELD_NORMAL",
         "short_channel_id": "108x1x1",
```

```
            "channel_sat": 0,
            "our_amount_msat": "0msat",
            "channel_total_sat": 1000000,
            "amount_msat": "1000000000msat",
            "funding_txid":
            ↪  "499ff0ee406e3c6ad183d5f87fe244b28a2caa8a574ad980a
            ↪  34f08333125411f",
            "funding_output": 1
        }
    ]
}
```

U Boba vidíme, že aktuálně nedisponuje žádnými bitcoiny (UTXO), ale že má otevřený jeden kanál s kapacitou 1M satoshi a on sám zatím nemá nic (`channel_sat`). Alice měla na začátku 2M satoshi a vytvářela kanál, takže její bilance vypadá následovně:

```
clightning@alice:/$ lightning-cli listfunds
{
   "outputs": [
      {
         "txid":
         ↪  "499ff0ee406e3c6ad183d5f87fe244b28a2caa8a574ad98
         ↪  0a34f08333125411f",
         "output": 0,
         "value": 999846,
         "amount_msat": "999846000msat",
         "address":
         ↪  "bcrt1qpw3vskfa4c54v4wt23x8zj05dg5vdw3d2k9p28",
         "status": "confirmed",
         "blockheight": 108
      }
   ],
   "channels": [
      {
         "peer_id":
         ↪  "034f1e8200a7c82746f7b95ef2e5fcb57e747781971c37cdb
         ↪  e129b389df3db0d3e",
         "connected": true,
         "state": "CHANNELD_NORMAL",
         "short_channel_id": "108x1x1",
         "channel_sat": 1000000,
         "our_amount_msat": "1000000000msat",
```

```
            "channel_total_sat": 1000000,
            "amount_msat": "1000000000msat",
            "funding_txid":
            ↪  "499ff0ee406e3c6ad183d5f87fe244b28a2caa8a574ad980a
            ↪  34f08333125411f",
            "funding_output": 1
        }
    ]
}
```


U záznamu v `outputs` je vidět adresa její Bitcoinové peněženky, zdali zmíněná částka je potvrzena a v jakém bloku je obsažena (`blockheight`).

Alice poté fakturu poctivě zaplatí následujícím příkazem:


```
clightning@alice:/$ lightning-cli pay
lnbcrt500u1p3x9rvkpp5gfx805gvxn04vae5jzzh0nzaqx64
↪  89v2ac2h5fhx6tg7fd6yalfsdqugeskkar4wfsjq7npypmx2cm
↪  9wf5sxqyjw5qcqp2sp50z44h63danmz3nyflqx7f2uutas30e4
↪  rfudkwktj2w3wv04qyttq9qy9qsq7lr0r6aze7xcrfc2lmwk0t
↪  gvsehq6juy47s25rfwprw4hp3ur83338r0s5706y6u67kxcl8m
↪  j2euhxkdt7ekd02w6dazcla7g3utg7gq5lxk6c
{
   "id": 1,
   "payment_hash":
   ↪  "424c77d10c34df567734908577cc5d01b553958aee157a2
   ↪  6e6d2d1e4b744efd3",
   "destination":
   ↪  "034f1e8200a7c82746f7b95ef2e5fcb57e747781971c37
   ↪  cdbe129b389df3db0d3e",
   "msatoshi": 50001002,
   "amount_msat": "50001002msat",
   "msatoshi_sent": 50001002,
   "amount_sent_msat": "50001002msat",
   "created_at": 1650627964,
   "status": "complete",
   "payment_preimage":
   ↪  "b9e4f7c8709fc50549cf3d9b8514c0eb7caed508dfaf4ec75345e68
   ↪  30b9a9167",
   "bolt11": "lnbcrt500u1p3x9rvkpp5gfx805gvxn04vae5jzzh0nzaqx64
   ↪  89v2ac2h5fhx6tg7fd6yalfsdqugeskkar4wfsjq7npypmx2cm
   ↪  9wf5sxqyjw5qcqp2sp50z44h63danmz3nyflqx7f2uutas30e4
   ↪  rfudkwktj2w3wv04qyttq9qy9qsq7lr0r6aze7xcrfc2lmwk0t
   ↪  gvsehq6juy47s25rfwprw4hp3ur83338r0s5706y6u67kxcl8m
   ↪  j2euhxkdt7ekd02w6dazcla7g3ut g7gq5lxk6c"
```

```
}
```


Z výstupu vidíme, kam byla platba zaslána, že jsme získali `payment_preimage` a platba proběhla úspěšně. Můžeme u Boba ze zajímavosti ještě zkontrolovat fakturu a jeho prostředky.


```
clightning@bob:/$ lightning-cli listinvoices
{
   "invoices": [
      {
         "label": "vecere",
         "bolt11":
         ↪  "lnbcrt500u1p3x9rvkpp5gfx805gvxn04vae5jzzh0nzaqx64
         ↪  89v2ac2h5fhx6tg7fd6yalfsdqugeskkar4wfsjq7npypmx2cm
         ↪  9wf5sxqyjw5qcqp2sp50z44h63danmz3nyflqx7f2uutas30e4
         ↪  rfudkwktj2w3wv04qyttq9qy9qsq7lr0r6aze7xcrfc2lmwk0t
         ↪  gvsehq6juy47s25rfwprw4hp3ur83338r0s5706y6u67kxcl8m
         ↪  j2euhxkdt7ekd02w6dazcla7g3ut g7gq5lxk6c",
         "payment_hash":
         ↪  "424c77d10c34df567734908577cc5d01b553958aee157a26e
         ↪  6d2d1e4b744efd3",
         "msatoshi": 50000000,
         "amount_msat": "50000000msat",
         "status": "paid",
         "pay_index": 1,
         "msatoshi_received": 50001002,
         "amount_received_msat": "50001002msat",
         "paid_at": 1650627964,
         "payment_preimage":
         ↪  "b9e4f7c8709fc50549cf3d9b8514c0eb7caed508dfaf4ec75
         ↪  345e6830b9a9167",
         "description": "Faktura za veceri",
         "expires_at": 1651230742
      }
   ]
}
clightning@bob:/$ lightning-cli listfunds
{
   "outputs": [],
   "channels": [
      {
         "peer_id":
         ↪  "02b3150bbeda79abc37d222cb2e62482c0320c9294db47dc8
         ↪  b6487f30357503960",
```

```
        "connected": true,
        "state": "CHANNELD_NORMAL",
        "short_channel_id": "108x1x1",
        "channel_sat": 50001,
        "our_amount_msat": "50001000msat",
        "channel_total_sat": 1000000,
        "amount_msat": "1000000000msat",
        "funding_txid":
        ↪  "499ff0ee406e3c6ad183d5f87fe244b28a2caa8a574ad980a
        ↪  34f08333125411f",
        "funding_output": 1
      }
   ]
}
```

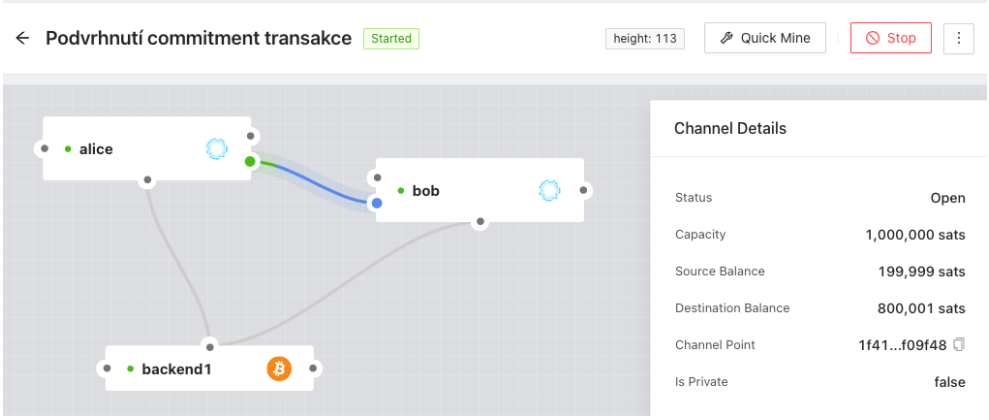
Bob nyní vlastní svoje první satoshi. Obě strany se dohodly na nové commitment transakci a ta předešlá byla zneplatněna odhalením odpovídajícího `per_commitment_secret`. Alice se rozhodne, že v tuto chvíli si chce vytvořit snapshot aktuálního stavu kanálu, tedy že na konci útoku bude disponovat 950 000 satoshi. Toho docílí tím, že si připraví lokálně aktuální commitment transakci na pozdější útok, pomocí příkazu, který není dostupný v c-lightning implementacích, které nebyly sestaveny ve vývojářském módu.

```
clightning@alice:/$ lightning-cli dev-sign-last-tx
↪  034f1e8200a7c82746f7b95ef2e5fcb57e747781971c37cdbe129b389df
↪  3db0d3e
{
    "tx":
    ↪  "020000000001011f41253133084fa380d94a578aaa2c8ab244e27f
    ↪  f8d583d16a3c6e40eef09f490100000000ad07e8800251c30000000
    ↪  000001600140ccd68e3e68eaa97d106ccf40753c72546f0f1bb377e
    ↪  0e00000000002200202d234fc6569a453b973d3829999ecff28fa44
    ↪  3ddab201929827e015c7e8cedde040047304402202794fb141acf9a
    ↪  7f85fa6b69ba03baf86122632d23b77ea9615f48df120b6bb102202
    ↪  88fb4cd2aa9f6b3574bdcaacd8d350d3ecc9e22157bc8fa48df94d0
    ↪  41289bf601473044022065ae33de3f15f49e0588f2ea7fab2edce48
    ↪  ea55b564d97be1097fc0c083375f602207f6c6e47802bc64630892d
    ↪  adca922294cde96bf988c2206c7fa14440ba606bed0147522102822
    ↪  13d9d8a15a78100ddc8fd22ce335ab00e4cab3471512e1d8f6737a2
    ↪  4047db2103a7ae3ebc9c2eb8d690768cb7398a249b7297af4321f9f
    ↪  1d05226bfd057ddbaea52ae85e5fc20"
}
```

Tuto podepsanou commitment transakci Alice ve vhodný čas vyšle do sítě.

Po této přípravě na útok následuje běžná operace uzlu, Alice přesouvá likviditu k Bobovi a případně se to samé děje i ve druhém směru. Na čase nezáleží, pouze na vhodném momentu pro útok.

Později nastane předpoklad útoku, a to, že Bobův uzel se stane v síti nedostupným. V tuto chvíli je jeho zůstatek 800 000 satoshi, zatímco Alice má 200 000.



Obrázek 4.2: Stav kanálu před útokem

Alice se rozhodne, že tento moment je vhodný pro útok a iniciuje ho odesláním připravené commitment transakce do sítě.

```
clightning@alice:/$ lightning-cli sendrawtransaction
↪  020000000001011f41253133084fa380d94a578aaa2c8ab244e27f
↪  f8d583d16a3c6e40eef09f490100000000ad07e8800251c30000000
↪  000001600140ccd68e3e68eaa97d106ccf40753c72546f0f1bb377e
↪  0e00000000002200202d234fc6569a453b973d3829999ecff28fa44
↪  3ddab201929827e015c7e8cedde040047304402202794fb141acf9a
↪  7f85fa6b69ba03baf86122632d23b77ea9615f48df120b6bb102202
↪  88fb4cd2aa9f6b3574bdcaacd8d350d3ecc9e22157bc8fa48df94d0
↪  41289bf601473044022065ae33de3f15f49e0588f2ea7fab2edce48
↪  ea55b564d97be1097fc0c083375f602207f6c6e47802bc64630892d
↪  adca922294cde96bf988c2206c7fa14440ba606bed0147522102822
↪  13d9d8a15a78100ddc8fd22ce335ab00e4cab3471512e1d8f6737a2
↪  4047db2103a7ae3ebc9c2eb8d690768cb7398a249b7297af4321f9f
↪  1d05226bfd057ddbaea52ae85e5fc20
{
   "success": true,
   "errmsg":
   ↪  "3d36c35f661c9d5244babaab7ff5919623f36a4e311d03f71d3078a
   ↪  953a616c8"
```

```
}
```

Z výstupu příkazu je vidět, že propagace do sítě proběhla v pořádku. Jakmile se transakce potvrdí (zahrne do prvního bloku), tak můžeme v logovacích souborech kontejneru nalézt veškeré informace o výměně zpráv mezi uzlem Alice a sítí. Pozorováním těchto zpráv jsem postupně upravil útočníkovi implementaci, aby fungovala, jak se od ní očekává. Klíčovou částí je způsob uzavření kanálu. Bez úpravy uzavření vypadá následovně:

```
2022-04-21T11:36:35.338Z DEBUG
↪  033b75c3ffa801867be5985db49bf60e1f28c143613f0e2039b797361eb
e522866-onchaind-chan#3: Resolved
↪  FUNDING_TRANSACTION/FUNDING_OUTPUT by
↪  THEIR_REVOKED_UNILATERAL
```

Uzel Alice identifikuje, že podvádí, a potrestá jí ihned i bez Bobovy přítomnosti. Když si, ale Alice upraví implementaci podle přílohy C, tak logy nyní obsahují místo `THEIR_REVOKED_UNILATERAL` slovo `OUR_UNILATERAL` indikující, že dochází k Force Close a ne Protocol Breach.

```
2022-04-22T14:19:36.107Z DEBUG
↪  034f1e8200a7c82746f7b95ef2e5fcb57e747781971c37cdbe129b389df
3db0d3e-onchaind-chan#1: Resolved
↪  FUNDING_TRANSACTION/FUNDING_OUTPUT by OUR_UNILATERAL
```

Jakmile Alice vyšle transakci, tak už nemusí nic dělat, jen doufat, že se Bob do sítě mezitím nepřipojí a dostatečně rychle nesesynchronizuje. V našem testovacím prostředí prodlevu odsimulujeme vytěžením bloků. Po vytěžení několika bloků nalezneme v logovacích souborech tyto informace (dlouhé řetězce jsem pro přehlednost výpisu zkrátil):

```
2022-04-22T14:39:31.485Z DEBUG ...-chan#1: Got depth change
↪  1->401 for ...
2022-04-22T14:39:31.493Z DEBUG lightningd: sendrawtransaction:
↪  ...
2022-04-22T14:39:31.509Z DEBUG ...-onchaind-chan#1: Got new
↪  message WIRE_ONCHAIN_DEPTH
2022-04-22T14:39:31.509Z DEBUG ...-onchaind-chan#1: Sending 0
↪  missing htlc messages
2022-04-22T14:39:31.510Z DEBUG ...-onchaind-chan#1:
↪  FUNDING_TRANSACTION/FUNDING_OUTPUT->OUR_UNILATERAL depth
↪  401
2022-04-22T14:39:31.511Z DEBUG ...-onchaind-chan#1:
↪  OUR_UNILATERAL/OUTPUT_TO_THEM->SELF depth 401
```

```
2022-04-22T14:39:31.511Z DEBUG ...-onchaind-chan#1:
↪  Broadcasting OUR_DELAYED_RETURN_TO_WALLET
```

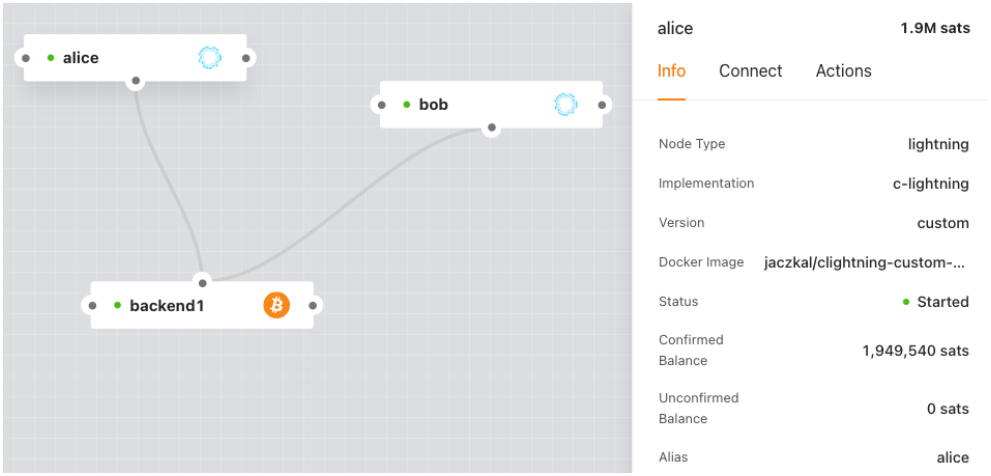
Z těchto řádků se dozvíme, že se naplnila lhůta 400 bloků a prostředky se uvolňují na zvolené peněženky. U Alice rázem vidíme nové UTXO.

```
clightning@alice:/$ lightning-cli listfunds
{
   "outputs": [
      {
         "txid":
         ↪  "499ff0ee406e3c6ad183d5f87fe244b28a2caa8a574ad980a3
         ↪  4f08333125411f",
         "output": 0,
         "value": 999846,
         "amount_msat": "999846000msat",
         "address":
         ↪  "bcrt1qpw3vskfa4c54v4wt23x8zj05dg5vdw3d2k9p28",
         "status": "confirmed",
         "blockheight": 108
      },
      {
         "txid":
         ↪  "215ca96c9ee82bb53f95e0d65d4136e7e4c7ddeda7f4e6e77f
         ↪  c00142216ba720",
         "output": 0,
         "value": 949694,
         "amount_msat": "949694000msat",
         "address":
         ↪  "bcrt1q5h5y5v6q0fmsswl2n3vsyjya7xy6gmx8jlfak5",
         "status": "confirmed",
         "blockheight": 515
      }
   ],
   "channels": []
}
```

V našem testovacím prostředí došlo k potvrzení transakce v 515 bloku. Tento výstup transakce odpovídá 950 000 satoshi, což je přesně tolik, kolik si Alice v minulosti vybrala. Nastartujeme Bobův uzel, ať vidíme jak dopadl on, což je poměrně zřejmé, protože platby jsou v tuto chvíli skutečně nevratné (zapsané na Bitcoin blockchainu, vlastněné Bitcoin adresami).

```
clightning@bob:/$ lightning-cli listfunds
{
   "outputs": [
      {
         "txid":
         ↪  "3d36c35f661c9d5244babaab7ff5919623f36a4e311d03f71
         ↪  d3078a953a616c8",
         "output": 0,
         "value": 50001,
         "amount_msat": "50001000msat",
         "address":
         ↪  "bcrt1qpnxk3clx364f05gxen6qw578y4r0pudmscm4kr",
         "status": "confirmed",
         "blockheight": 114
      }
   ],
   "channels": []
}
```

Bob má místo 800 000 satoshi pouze 50 000, které byly za tu první fakturu. Alice ho ve výsledku okradla o 750 000 satoshi.



Obrázek 4.3: Stav uzlů po útoku

4.1.3 Doporučení

Na základě výše uvedených informací je zřejmé, že v případě, kdy se k potenciálnímu útočníkovi dostane nebezpečná implementace LN, tak útok není

nikterak složitý a přitom má fatální následky v případě, že se naplní předpoklad útoku.

Existuje však řešení, tzv. Watchtower, které mohou střežit LN uzel a kdyby došlo k jeho výpadku, tak v případě, že by útočník chtěl daný uzel okrást, tak daný Watchtower pokus o útok odhalí a uplatní se penalizační mechanismus.

Watchtower

Pro účely demonstrace obrany si tentokrát připravíme 3 uzly. Jedním bude útočník (Carol), který bude používat naši nebezpečnou implementaci c-lightningu, druhým oběť s aktivovanou obranou (Bob) s implementací LND (ve verzi 0.12.1-beta). LND jsem zvolil poněvadž ze všech implementací je nejvíce používaný (v roce 2022 bylo 87 % uzlů sítě právě LND [25]), čímž poskytnu konkrétní návod na obranu pro více uživatelů, a zároveň jsem vyzkoušel, že je stejně zranitelný jako c-lightning implementace na které byl útok demonstrován v předchozí sekci. Posledním uzlem je Watchtower (Alice), který používá stejnou verzi LND jako oběť, pouze jsem upravil Docker Image, aby tam bylo možné nastavit veřejné porty.

Do Docker Image jsem přidal `ufw` (Firewall pro Ubuntu/Debian) a sadu síťových nástrojů, které jsou užitečné při testování propojení.

```
RUN apt update
RUN apt install -y ufw iputils-ping net-tools iproute2
```

Implementace na kontejnerech je pak třeba spustit s dodatečnými parametry. U Boba přidáme parametr `--wtclient.active`, indikující aktivaci Watchtower klienta. Alice bude potřebovat parametry dva a to `--watchtower.active`, indikující aktivaci Watchtower serveru a `--watchtower.listen=0.0.0.0:9911`, kde řekneme, komu všemu bude server naslouchat. V našem případě mu můžeme dovolit, že bude naslouchat všem adresám, které o tuto službu stojí. Nyní je potřeba ještě Alici povolit port 9911 (pokud tomu tak není). To učiníme pomocí programu `ufw` a následně můžeme programem `netstat` zkontrolovat otevřené porty.

```
root@alice:/$ ufw allow 9911
Rules updated
Rules updated (v6)
lnd@alice:/$ netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
↪  State       PID/Program name
tcp        0      0 0.0.0.0:10009           0.0.0.0:*
↪  LISTEN      1/qemu-x86_64
tcp        0      0 127.0.0.11:33485        0.0.0.0:*
↪  LISTEN      -
```

```
tcp        0      0 0.0.0.0:8080            0.0.0.0:*
↪  LISTEN      1/qemu-x86_64
tcp6       0      0 :::9911                 :::*
↪  LISTEN      1/qemu-x86_64
tcp6       0      0 :::9735                 :::*
↪  LISTEN      1/qemu-x86_64
udp        0      0 127.0.0.11:36863        0.0.0.0:*
↪  -
```

Jakmile máme server připraven, tak zjistíme důležitá data od Alice pro propojení. Potřebujeme IP adresu, veřejný klíč Watchtoweru a port. IP adresu a port známe, stačí zjistit veřejný klíč pro Watchtower (pozor, je odlišný od veřejného klíče uzlu).

```
lnd@alice:/$ lncli tower info
{
    "pubkey":
    ↪  "030729e3a9178ed621e24e9e726644742813b1ed628646eba01ce6
    ↪  1cb278cef729",
    "listeners": [
        "[::]:9911"
    ],
    "uris": [
    ]
}
```

Nyní můžeme zaregistrovat Watchtower u Boba. K tomu nám poslouží třída příkazů pro Watchtower klienta, `wtclient`.

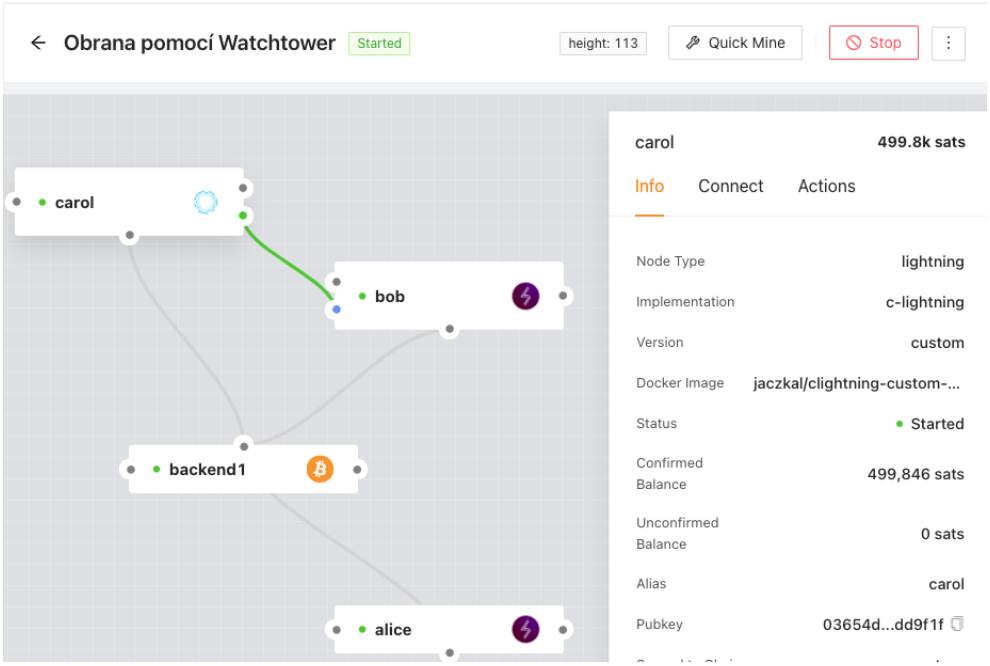
```
lnd@bob:/$ lncli wtclient add
↪  030729e3a9178ed621e24e9e726644742813b1ed628646eba01ce61cb27
↪  8cef729@172.29.0.4:9911
{

}
lnd@bob:/$ lncli wtclient towers
{
    "towers": [
        {
            "pubkey":
            ↪  "030729e3a9178ed621e24e9e726644742813b1ed628646
            ↪  eba01ce61cb278cef729",
            "addresses": [
```

```
                "172.29.0.4:9911"
            ],
            "active_session_candidate": true,
            "num_sessions": 0,
            "sessions": [
            ]
        }
    ]
}
```

Připojení Watchtoweru je v tuto chvíli hotové. Zdali je zapojen správně lze zjistit z logovacích souborů. Watchtower logy mají identifikátor `WTWT` a Watchtower klient logy mají `WTCL`. Ve chvíli, kdy na klientovi spatříme podobný řádek jako níže uvedený, tak se zpravidla jedná o úspěšné navázání komunikace, pokud následuje i odpovídající tok zpráv po dobu životnosti tohoto propojení.

```
2022-04-24 14:59:47.669 [INF] WTCL: (legacy) Acquired new
↪  session with
↪  id=036570919f21bf5f970617409af4793c604a562be5c0f7a2358c94ef
↪  d0fade39c8
```

Watchtower je nasazen, přejdeme k útoku. Carol disponuje 1 milionem satoshi, Bob a Alice žádné nemají. Carol otevře kanál k Bobovi o kapacitě 500 000 satoshi a ihned si uloží aktuální commitment transakci, aby měla po zaplaceních Bobových faktur opět 1 milion.



Obrázek 4.4: Rozestavění uzlů

Po proplacení několika faktur nastane stav kanálu, že Carol má 300 000 satoshi a Bob má 200 000 satoshi a jeho uzel vypadnul ze sítě. Carol stačí výdělek 200 000 satoshi, takže se rozhodne, že vyšle zneplatněnou commitment transakci. Jakmile se vytěží blok v kterém je obsažena (blok 114), tak můžeme náhlednout do Watchtoweru, že mu tato informace neunikla (dlouhé řetězce jsem pro přehlednost výpisu zkrátil).

```
2022-04-24 15:16:45.576 [INF] NTFN: New block: height=114,
↪  sha=...
2022-04-24 15:16:45.581 [INF] UTXN: Attempting to graduate
↪  height=114: num_kids=0, num_babies=0
2022-04-24 15:16:45.594 [INF] WTWR: Found 1 breach in
↪  (height=114, hash=...)
2022-04-24 15:16:45.595 [INF] WTWR: Dispatching punisher for
↪  client ..., breach-txid=...
2022-04-24 15:16:45.604 [INF] CRTR: Block ... (height=114)
↪  closed 1 channels
2022-04-24 15:16:45.611 [INF] WTWR: Publishing justice
↪  transaction for client=... with txid=...
2022-04-24 15:16:45.612 [INF] LNWL: Inserting unconfirmed
↪  transaction ...
```

```
2022-04-24 15:16:45.638 [INF] WTWR: Punishment for client ...
↪  with breach-txid=... dispatched
```

Z výpisu je vidět, že Watchtower odhalil Protocol Breach. Pro jistotu se ale budeme držet scénaře útoku a necháme vytěžit např. 500 bloků než zapneme Boba. Ještě před zapnutím je ale vidět, že Carol nic nedostala ba naopak, zůstatek kanálu ji zmizel.

```
clightning@carol:/$ lightning-cli listfunds
{
   "outputs": [
      {
         "txid":
         ↪  "41806b275117e13f03d032910107e4a8c95f634192109e5c7
         ↪  d63e0d770d27ccf",
         "output": 0,
         "value": 499846,
         "amount_msat": "499846000msat",
         "address":
         ↪  "bcrt1qm5ewnwvuhlnsyekewxatc5fxn0px3lgynek8z7",
         "status": "confirmed",
         "blockheight": 108
      }
   ],
   "channels": []
}
```

Zatímco když se podíváme na stav Bobovy peněženky, tak ta se má o poznání lépe.

```
lnd@bob:/$ lncli walletbalance
{
    "total_balance": "498602",
    "confirmed_balance": "498602",
    "unconfirmed_balance": "0"
}
```

Bob se z minulé zkušenosti s Alicí[23] poučil a při nové zkušenosti s Carol se už okrást nenechal. Ba naopak si přivydělal, poněvadž kdyby uzavřel kanál klasickým způsobem, tak má jen 200 000 satoshi, zatímco takto získal kapacitu celého kanálu.

[23] Tehdy útočník, v tomto případě mu naopak pomáhala.

Závěr

Hlavním cílem této práce bylo seznámení se s Lightning Network a představení jeho bezpečnostních zranitelností, které nalezneme ve třetí kapitole. Bylo zde popsáno celkem 8 známých zranitelností a z toho jedna byla detailně předvedena v kapitole čtvrté. Tato zranitelnost byla zvolena k demonstraci, poněvadž proti ní existuje účinná obrana a vyžaduje dodatečné netriviální kroky k útoku, načež to samé nelze tvrdit o ostatních zranitelnostech. Většina z nich je stále proveditelná i na nejnovějších implementacích a z toho plyne na základě této práce obecné doporučení pro uživatele:

- vždy využívat pomocný uzel Watchtower,
- sledovat novinky ohledně protokolu a udržovat svůj uzel aktualizovaný a synchronizovaný,
- nevkládat do kanálů více prostředků než je uživatel ochoten ztratit,
- neotevírat kanály s neznámými uzly.

Dodržováním těchto bodů se minimalizuje risk ztráty prostředků či jiné újmy způsobené protokolem. Lightning Network je stále v rané fázi vývoje, avšak počet jeho uživatelů stále přibývá a to mimo akcelerace vývoje způsobí i větší zájem útočníků. Oproti Bitcoinu se jedná o komplexnější protokol a je pravděpodobné, že s vyšší adopcí bude LN čelit více útokům různých druhů. Je to částečně způsobené i tím, že uživatele jsou nuceni podnikat dodatečné netriviální kroky pro triviální úkony jako je platba v obchodě nebo na internetu. Ve chvíli, kdy by se ale snažila nějaká třetí strana řešit za uživatele tyto věci, jako je dostatek likvidních otevřených kanálů s důvěryhodnými a vhodně propojenými uzly, uzel který hlídá jejich hlavní uzel, atd., vyvstává otázka zdali v tomto případě jsme nevyřešili škálovatelnost za cenu decentralizace, kde vzniknou na trhu velké společnosti vlastnící většinu sítě. Síť pak může mít tendenci se shlukovat do různých podsítí vlastněných těmito společnostmi, které si můžou například určovat vysoké směrovací poplatky,

případně budou aplikace pro pohodlí uživatelů postavené tak, že uživatele nebudou vlastnit své klíče, atd. Pravděpodobně jako v běžném světě technologií si uživatelé budou volit cestu k LN na škále pohodlí versus bezpečnost a soukromí, a od toho se bude částečně vyvíjet i podoba oné sítě.

Literatura

[1] Bitcoin: A Peer-to-Peer Electronic Cash System [online].

[2] Antonopoulos, A. M.; Osuntokun, O.; Pickhardt, R.: Mastering the Lightning Network. 2021. Dostupné z: https://github.com/lnbook/lnbook

[3] Harris, J.; Zohar, A.: Flood amp; Loot: A Systemic Attack On The Lightning Network. 2020, doi:10.48550/ARXIV.2006.08513. Dostupné z: https://arxiv.org/abs/2006.08513

[4] Lin, Q.; Li, C.; Zhao, X.; aj.: Measuring Decentralization in Bitcoin and Ethereum using Multiple Metrics and Granularities. In *2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*, 2021, s. 80–87, doi:10.1109/ICDEW53142.2021.00022.

[5] Henriques, I.; Sadorsky, P.: Can Bitcoin Replace Gold in an Investment Portfolio? *Journal of Risk and Financial Management*, ročník 11, č. 3, 2018, ISSN 1911-8074, doi:10.3390/jrfm11030048. Dostupné z: https://www.mdpi.com/1911-8074/11/3/48

[6] Gkillas, K.; Longin, F.: Is Bitcoin the new digital gold? Evidence from extreme price movements in financial markets. *Evidence From Extreme Price Movements in Financial Markets (January 18, 2019)*, 2019.

[7] Antonopoulos, A.: *Mastering Bitcoin: Transaction*. O'Reilly, 2014, ISBN 9781449374044. Dostupné z: https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch05.html

[8] Official Bitcoin wiki: Transaction [online]. 2022. Dostupné z: https://en.bitcoin.it/wiki/Transaction

[9] Official Bitcoin wiki: Script [online]. 2022. Dostupné z: https://en.bitcoin.it/wiki/Script

[10] Bistarelli, S.; Mercanti, I.; Santini, F.: An Analysis of Non-standard Bitcoin Transactions. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2018, s. 93–96, doi:10.1109/CVCBT.2018.00016.

[11] Official Bitcoin wiki: Segregated Witness [online]. 2022. Dostupné z: https://en.bitcoin.it/wiki/Segregated_Witness

[12] Segregated Witness proposal [online]. 2021. Dostupné z: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

[13] Perez-Sol, C.; Delgado-Segura, S.; Herrera-Joancomart, J.; aj.: Analysis of the SegWit adoption in Bitcoin [online].

[14] Decker, C.; Wattenhofer, R.: Bitcoin Transaction Malleability and MtGox. 2014: s. 313–326.

[15] Kenny, L.: The Blockchain Scalability Problem & the Race for Visa-Like Transaction Speed [online]. leden 2019. Dostupné z: https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44

[16] Security and reliability [online]. 2021. Dostupné z: https://usa.visa.com/run-your-business/small-business-tools/retail.html

[17] Decker, C.; Wattenhofer, R.: Information Propagation in the Bitcoin Network [online]. Dostupné z: https://tik-old.ee.ethz.ch/file//49318d3f56c1d525aabf7fda78b23fc0/P2P2013_041.pdf

[18] Wang, S.; Yuan, Y.; Wang, X.; aj.: An Overview of Smart Contract: Architecture, Applications, and Future Trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, s. 108–113, doi:10.1109/IVS.2018.8500488.

[19] Poon, J.; Dryja, T.: Lightning Network whitepaper 0.5.9 [online]. 2021. Dostupné z: https://lightning.network/lightning-network-paper.pdf

[20] Wirdum, A. V.: The History of Lightning: From Brainstorm To Beta [online]. 2021. Dostupné z: https://bitcoinmagazine.com/technical/history-lightning-brainstorm-beta

[21] Poon, J.; Dryja, T.: Lightning Network whitepaper 0.5 [online]. 2021. Dostupné z: https://web.archive.org/web/20150301065655/http://lightning.network/lightning-network-paper-DRAFT-0.5.pdf

[22] BOLT 2: Peer Protocol for Channel Management [online]. 2022. Dostupné z: https://github.com/lightning/bolts/blob/master/02-peer-protocol.md

[23] BOLT 3: Bitcoin Transaction and Script Formats [online]. 2022. Dostupné z: https://github.com/lightning/bolts/blob/master/03-transactions.md

[24] Lin, J.-H.; Primicerio, K.; Squartini, T.; aj.: Lightning Network: a second path towards centralisation of the Bitcoin economy. *New Journal of Physics*, ročník 22, 08 2020, doi:10.1088/1367-2630/aba062.

[25] Zabka, P.; Foerster, K.-T.; Schmid, S.; aj.: Empirical evaluation of nodes and channels of the lightning network. *Pervasive and Mobile Computing*, ročník 83, 2022: str. 101584.

[26] Pickhardt, R.; Tikhomirov, S.; Biryukov, A.; aj.: Security and Privacy of Lightning Network Payments with Uncertain Channel Balances. 2021, doi:10.48550/ARXIV.2103.08576. Dostupné z: https://arxiv.org/abs/2103.08576

[27] BOLT 11: Invoice Protocol for Lightning Payments [online]. 2022. Dostupné z: https://github.com/lightning/bolts/blob/master/11-payment-encoding.md

[28] Mizrahi, A.; Zohar, A.: Congestion Attacks in Payment Channel Networks. *CoRR*, ročník abs/2002.06564, 2020, 2002.06564. Dostupné z: https://arxiv.org/abs/2002.06564

[29] Mazumdar, S.; Banerjee, P.; Ruj, S.: Griefing-Penalty: Countermeasure for Griefing Attack in Lightning Network. 2020, doi:10.48550/ARXIV.2005.09327. Dostupné z: https://arxiv.org/abs/2005.09327

[30] Heilman, E.; Kendler, A.; Zohar, A.; aj.: Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In *24th USENIX Security Symposium (USENIX Security 15)*, Washington, D.C.: USENIX Association, Srpen 2015, ISBN 978-1-939133-11-3, s. 129–144. Dostupné z: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman

[31] Riard, A.; Naumenko, G.: Time-Dilation Attacks on the Lightning Network. *CoRR*, ročník abs/2006.01418, 2020, 2006.01418. Dostupné z: https://arxiv.org/abs/2006.01418

[32] Bitcoin Optech: Eclipse attacks [online]. 2022. Dostupné z: https://bitcoinops.org/en/topics/eclipse-attacks/

[33] Antonopoulos, A.: *Mastering Bitcoin: Chapter 6. The Bitcoin Network*. O'Reilly, 2014, ISBN 9781449374044. Dostupné z: https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch06.html

[34] Lightning-dev - Pinning : The Good, The Bad, The Ugly [online]. 2022. Dostupné z: https://lists.linuxfoundation.org/pipermail/lightning-dev/2020-June/002758.html

[35] Reckless_Satoshi: Stealing Sats from the Lightning Network Custodial Services [online]. 2022. Dostupné z: https://lightningnetwork.plus/posts/37

[36] CVE Details: CVE-2021-41592 [online]. 2022. Dostupné z: https://www.cvedetails.com/cve/CVE-2021-41592/

[37] Tikhomirov, S.; Pickhardt, R.; Biryukov, A.; aj.: Probing Channel Balances in the Lightning Network. *CoRR*, ročník abs/2004.00333, 2020, 2004.00333. Dostupné z: https://arxiv.org/abs/2004.00333

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

DoS Denial of Service

HTLC Hash Time-Locked Contract

LN Lightning Network

P2P Peer-to-peer

UTXO Unspent Transaction Output

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl.zip` ..... zašifrovaný adresář ve formátu ZIP
    - `appendix-c.tex` ..... zdrojová forma Přílohy C ve formátu LaTeX
    - `impl` ..... zdrojové kódy implementace
      - `lightning` ..... zdrojové kódy implementace c-lightning
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `appendix-c.zip` ..... zašifrovaný adresář ve formátu ZIP
    - `appendix-c.pdf` ..... text Přílohy C ve formátu PDF