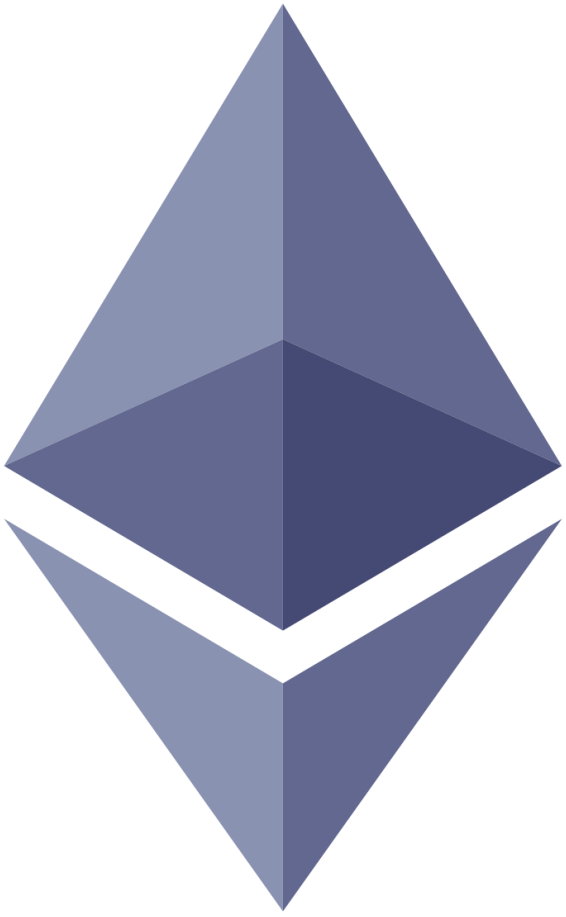
Ethereum Whitepaper

Autor: Vitalik Buterin, ethereum.org

Překlad: František Starý, kryptosvet.eu



Tento úvodní dokument byl v roce 2013 publikován zakladatelem společnosti Ethereum, Vitalikem Buterinem, a to před spuštěním projektu v roce 2015. Stejně jako mnoho komunitních open-source softwarových projektů se Ethereum od svého počátečního stavu vyvinulo, a proto tento dokument není aktuální.

Inteligentní smlouva nové generace a decentralizovaná platforma pro aplikace

Vývoj Bitcoinu Satoshim Nakamotem v roce 2009 byl často označován za radikální vývoj ve financích a měně, protože byl prvním příkladem digitálního aktiva, které současně nemá žádnou vnitřní hodnotu a centralizovaného emitenta, nebo správce. Další – pravděpodobněji důležitější - součástí Bitcoinového experimentu je však blockchainová technologie jako nástroj distribuovaného konsenzu, a pozornost se v poslední době začíná rapidně přesouvat k tomuto aspektu Bitcoinu. Mezi běžně uváděné alternativní využití technologie blockchainu patří používání digitálních aktiv na blockchainu k reprezentaci vlastních měn a finančních nástrojů (colored coins), vlastnictví základního fyzického zařízení (smart property), dále nepostradatelná aktiva, jako jsou názvy domén (namecoin), stejně jako složitější aplikace zahrnující přímé ovládání digitálních aktiv pomocí kódu implementujícího libovolná pravidla (smart contracts), nebo dokonce decentralizované autonomní organizace (DAO) založené na blockchainu. Ethereum zamýšlí poskytnout blockchain s vestavěným, plně rozvinutým programovacím jazykem Turing, jenž lze využít k vytváření „smluv“, které lze použít ke kódování libovolných funkcí stavového přechodu, což uživatelům umožňuje samostatně vytvářet jakýkoliv z výše popsaných systémů, stejně jako mnoho dalších, o kterých se nám ani nesnilo, to vše jednoduchým přepsáním logiky do několika řádků kódu.

Úvod do Bitcoinu a již existující koncepty

Koncept decentralizované digitální měny a alternativních aplikací, jako jsou registry nemovitostí existuje po celá desetiletí. Anonymní protokoly elektronické hotovosti (e-cash) z 80. a 90. let, které se většinou opíraly o kryptografický primitiv známý jako Chaumian Blind, poskytovaly měnu s vysokým stupněm soukromí, avšak nepodařilo se jim do značné míry získat trakci kvůli jejich spoléhání se na centralizovaného zprostředkovatele. V roce 1998 se „b-money" společnosti Wei Dai staly prvním návrhem, který zavedl myšlenku vytváření peněz prostřednictvím řešení složitých výpočetních hádanek a decentralizovaného konsenzu, avšak tento návrh nebyl dostatečně podrobný na to, aby bylo možné decentralizovaný konsenzus skutečně implementovat. V roce 2005 představil Hal Finney koncept opakovaně použitelných důkazů o práci (proof of work) - systém, který využívá nápady z b-money společně s výpočetně obtížnými hádankami „Haschcash" od Adama Backa k vytvoření konceptu kryptoměny. Dosáhnout ideálního stavu se jim ale ani v tomto případě nepodařilo, protože se systém spoléhal na důvěryhodné výpočty jako je backend. Decentralizovanou měnu poprvé uvedl do praxe Satoshi Nakamoto v roce 2009, když zkombinoval zavedená primitiva pro správu vlastnictví pomocí kryptografie veřejného klíče s konsensuálním algoritmem pro sledování toho, kdo mince vlastní. Tento algoritmus označujeme jako důkaz o práci (proof of work)

Mechanismus stojící za důkazem o práci byl průlomový, protože se mu podařilo vyřešit dva problémy současně. Zaprvé - poskytl jednoduchý a středně efektivní konsenzuální mechanismus, který umožnil uzlům v síti kolektivně se dohodnout na sadě kanonických aktualizací stavu hlavní účetní knihy Bitcoinu. Zadruhé – poskytl mechanismus umožňující volný vstup do procesu konsenzu, který řeší politický problém rozhodování o tom, kdo konsenzus ovlivní, a současně předchází útokům sybil (sybil attacks). Dělá to tím způsobem, že nahradí formální překážku účasti, např. požadavek na registraci jako jedinečná entita na konkrétním seznamu ekonomickou bariérou – váha jednoho uzlu v procesu konsenzuálního hlasování je přímo úměrná výpočetní síle, kterou uzel přináší. Od té doby byl navržen alternativní přístup, který se nazývá důkaz o vkladu (proof of stake), který vypočítává váhu uzlu jako proporcionální k měně jím držené, a nikoliv k jeho výpočetním zdrojům; diskuze o relativních výhodách těchto dvou přístupů je nad rámec tohoto dokumentu, avšak je třeba poznamenat, že oba tyto přístupy mohou být použity jako pomyslná páteř kryptoměny.

Bitcoin jako systém stavového přechodu

Z technického hlediska lze hlavní účetní knihu kryptoměny (ledger) považovat za přechodový systém, kde existuje „stav“ skládající se ze statusu vlastnictví všech stávajících Bitcoinů a „funkce stavového přechodu“, která převezme stav společně s transakcí, a odešle stav nový, který je označen za výsledek. Například ve standardním bankovním systému je stav účetní rozvaha, transakce je požadavek na přesunutí určitého počtu USD z účtu A na účet B. Funkce stavového přechodu snižuje hodnotu na účtu A o určitý počet USD (hodnotu transakce), zatímco hodnotu USD na účtu B o stejný počet zvýší. Pokud je na účtu A menší zůstatek než je třeba pro úspěšné provedení transakce, funkce stavového přechodu zobrazí error. Proto lze formálně definovat:

```
APPLY(S,TX) -> S' or ERROR
```

Ve výše definovaném bankovním systému:

```
APPLY({ Alice: $50, Bob: $50 },"send $20 from Alice to Bob") = { Alice: $30, Bob: $70 }
```

Avšak:

```
APPLY({ Alice: $50, Bob: $50 },"send $70 from Alice to Bob") = ERROR
```

Stav v Bitcoinu je kolekce všech mincí (technicky „neutracených transakčních výstupů“, nebo UTXO), které byly vytěženy a dosud utraceny, přičemž každý UTXO má nominální hodnotu a vlastníka (definovaného 20ti bajtovou adresou, která je v zásadě kryptografický veřejný klíč 1). Transakce obsahuje jeden nebo více vstupů, přičemž každý z nich obsahuje odkaz na již existující UTXO, kryptografický podpis vytvořený soukromým klíčem přidruženým k adrese vlastníka, a jeden nebo více výstupů, přičemž každý výstup obsahuje nový UTXO, který má být přidán do stavu.

Funkci stavového přechodu APPLY (S,TX)- > S' lze definovat zhruba takto:

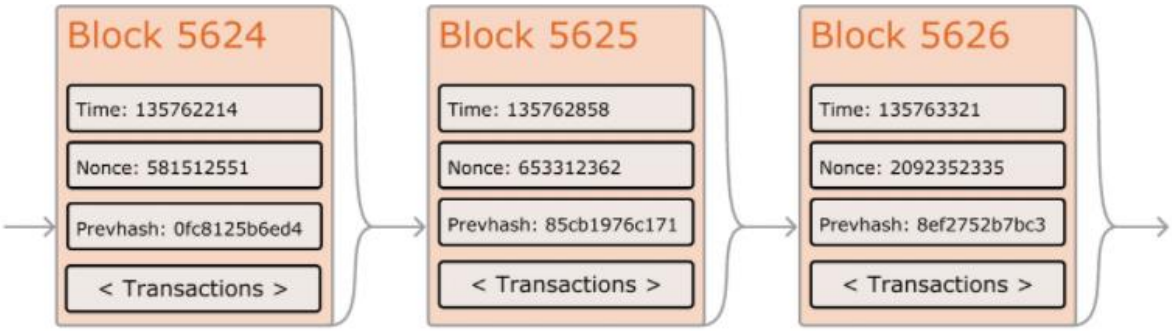
1. Pro každý vstup v TX:
    - Pokud odkazované UTXO není v S, poslat zpět error
    - Pokud zadaný podpis neodpovídá vlastníkovi UTXO, poslat zpět error.

2. Pokud je součet nominálních hodnot všech vstupních UTXO menší než součet nominálních hodnot všech výstupních UTXO, poslat zpět error

3. Poslat zpět S' tak, že všechny vstupní UTXO jsou odstraněné, a všechny výstupní UTXO přidané.

Úvodní část prvního kroku brání odesílatelům transakcí v utrácení neexistujících mincí, a druhá část jim zase brání v utrácení mincí jiných uživatelů. Druhý krok zase vynucuje zachování hodnoty. Aby se toto dalo využít k platbě, protokol je následující. Předpokládejme, že Alice chce Bobovi poslat 11,7 BTC. Nejprve Alice vyhledá sadu všech dostupných UTXO, které vlastní, a to nejméně 11,7 BTC. Realisticky nebude Alice schopna získat přesně 11,7 BTC (řekněme, že nejmenší hodnota kterou může získat je: 6+4+2 = 12 BTC). Následně Alice vytvoří transakci s těmito třemi vstupy (2,4,6) a dvěma výstupy. Prvním výstupem bude 11,7 BTC s Bobovou adresou (bude je vlastnit), a druhým výstupem bude 0,3 BTC (drobné), které bude vlastnit sama Alice.

Těžba



Pokud bychom měli přístup k důvěryhodným centralizovaným službám, implementace tohoto systému by byla triviální – mohlo by to být jednoduše kódováno přesně podle popisu za pomoci pevného disku centralizovaného serveru pro sledování stavu. S Bitcoiny se však snažíme vybudovat decentralizovaný měnový systém, takže budeme muset kombinovat systém stavového přechodu s konsensuálním systémem, abychom zajistili, že se všichni uživatelé dohodnou na pořadí transakcí. Proces decentralizovaného konsensu Bitcoinu vyžaduje, aby se uzly v síti neustále pokoušely vytvářet balíčky transakcí, zvané „bloky“. Síť je určena k vytvoření zhruba jednoho bloku každých deset minut, přičemž každý blok obsahuje časovou známku (timestamp), nonci, odkaz (hash) na předchozí blok a seznam všech transakcí, které od předchozího bloku proběhly. Postupem času tak vznikne trvalý a stále rostoucí „blockchain“, který se neustále aktualizuje a zobrazuje aktuální stav Bitcoinové účetní knihy.

Algoritmus který kontroluje, zda je blok platný, vyjádřený v tomto paradigmatu je následující:

1. Zkontroluje, zda je předchozí blok, na který aktuální blok odkazuje, platný.

2. Zkontroluje, zda je časová známka aktuálního bloku větší, než známka přechozího bloku, a menší, než 2 hodiny do budoucna.
3. Zkontroluje, zda je „proof of work“ bloku platný.
4. Stav S na konci předchozího bloku musí být roven 0.
5. Za předpokladu, že TX je seznam transakcí bloku s počtem transakcí n, pro všechna i v 0 ... n-1 je nastaveno jako S [i + 1] = APPLY (S [i], TX [i]) . Pokud některá aplikace zaregistruje chybu, ukončit a vrátit „false“.
6. Vrátit „true“ a zaregistrovat S[n] jako stav na konci tohoto bloku.

V podstatě musí každá transakce v bloku poskytovat platný přechod stavu z toho, co bylo kanonickým stavem před provedením transakce do nějakého nového stavu. Všimněte si, že stav v bloku není nijak zakódován – je to čistě abstrakce, kterou si pamatuje ověřovací uzel, a lze ji (bezpečně) vypočítat pro jakýkoliv blok vycházející ze stavu geneze, a postupně aplikovat každou transakci v každém bloku. Dále si všimněte, že záleží na pořadí, ve kterém těžař zahrnuje transakce do bloku – pokud jsou v bloku dvě transakce A a B tak, že B utratí UTXO vytvořené A, blok bude platný pouze v případě, že A přijde před B. Jedinou podmínkou platnosti bloku obsaženou ve výše uvedeném seznamu, která se nenachází v jiných systémech, je požadavek na „důkaz o práci“ (proof of work). Přesná podmínka je, že dvojitý hash SHA256 každého bloku, který je považován ze 256ti bitové číslo, musí být menší, než dynamicky upravený cíl, který je v době psaní tohoto dokumentu přibližně 2187. Účelem je vytvoření výpočtově složitého bloku, což zabrání tomu, aby útočníci sibyl (sibyl attackers) přetvořili celý blockchain ve svůj prospěch. Protože SHA256 je navržen tak, aby byl zcela nepředvídatelnou pseudonáhodnou funkcí, jediným způsobem, jak vytvořit platný blok zůstává jednoduše pokus a omyl. Útočník by musel opakovaně zvyšovat nonci a sledovat, zda se nový hash shoduje.

Pro aktuální cíl, který je přibližně 2187, musí síť provést asi 269 pokusů, než se najde platný blok; obecně je cíl rekalibrován sítí každých 2016 bloků, takže průměrný nový blok je jedním z uzlů produkován každých deset minut. Za účelem kompenzace těžařů za tuto výpočetní práci je těžař každého bloku oprávněn zahrnout do něj transakci, která mu přidá 12,5 BTC, vytvořených z ničeho. Navíc, pokud má jakákoliv transakce vyšší celkovou nominální hodnotu na svých vstupech než na výstupech, rozdíl je také připsán těžaři jako „transakční poplatek“. Mimochodem, toto je také jediný mechanismus, který vydává BTC; stav geneze neobsahoval vůbec žádné mince.

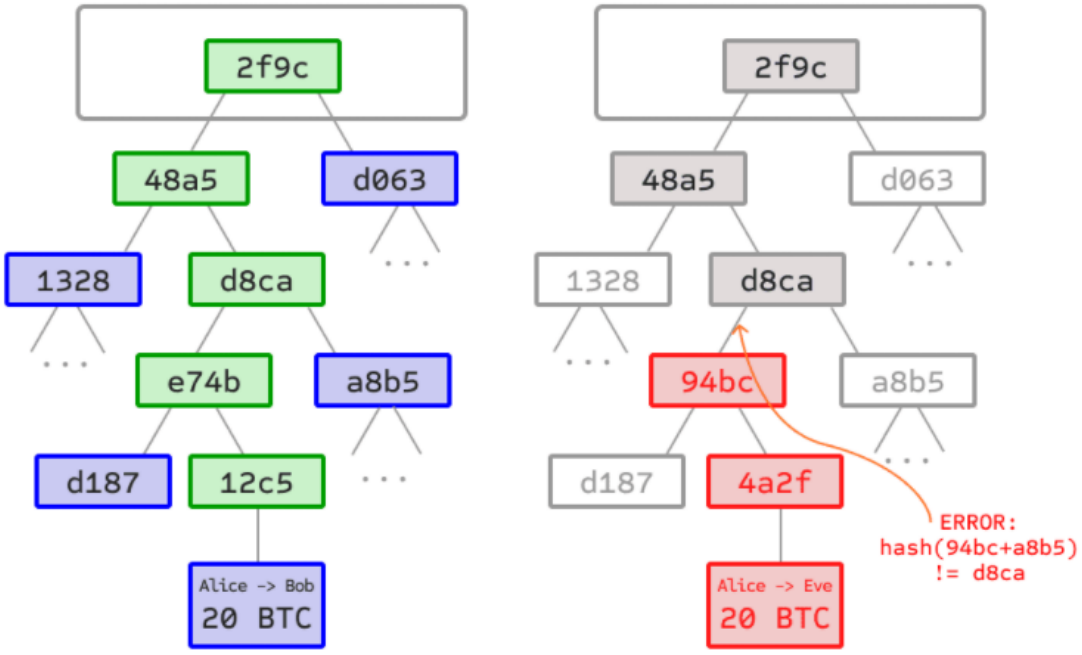
Abychom lépe porozuměli účelu těžby, podívejme se, co se stane v případě zlomyslného útočníka. Jelikož je známo, že kryptografie Bitcoinu je bezpečná, útočník se zaměří na část systému, která touto kryptografií není chráněna – pořadí transakcí. Strategie útočníka je jednoduchá:

1. Poslat obchodníkovi 100 BTC výměnou za nějaký produkt (nejlépe digitální zboží s rychlým doručením)
2. Počkat na dodání produktu
3. Produkovat další transakci, a poslat 100 BTC sám sobě
4. Pokusit se přesvědčit síť, že transakce, kterou provedl vůči sám sobě byla provedena první.

Jakmile proběhne první krok, po několika minutách zahrne některý z těžařů transakci do bloku, řekněme že jeho číslo je 270. Přibližně po jedné hodině bude po tomto bloku do řetězce přidáno dalších pět bloků, přičemž každý z nich nepřímo směřuje k transakci, a tím ji potvrzuje. V tomto okamžiku přijme obchodník platbu jako dokončenou a doručí produkt; protože předpokládáme, že se jedná o digitální produkt, doručení je okamžité. Útočník nyní vytvoří další transakci, a pošle sám sobě 100 BTC. Pokud tuto transakci útočník jednoduše vypustí, nebude zpracována; těžaři se pokusí spustit APPLY (S, TX) a všimnou si, že TX spotřebovává UTXO, které již není v požadovaném stavu. Místo toho

tedy útočník vytvoří „vidličku“ (fork) blockchainu tím, že začne těžit jinou verzi bloku 270, která ukazuje na stejný blok 269 jako na nadřazený, avšak s novou transakcí, namísto té staré. Protože se data bloku liší, bude vyžadován důkaz o práci (proof of work). Kromě toho má útočníkova verze bloku 270 jiný hash, takže původní bloky 271-275 na něj „neodkazují“; původní řetězec a útočníkův nový řetězec jsou tak zcela oddělené. Pravidlem je, že ve vidličce (fork), je nejdelší blockchain považován za ten platný, a tak budou legitimní těžaři pracovat na řetězci 275, zatímco samotný útočník pracuje na řetězci 270. Aby se útočníkovi podařilo vytvořit nejdelší řetězec, potřeboval by mít větší výpočetní výkon, než zbytek sítě dohromady. Kdyby se tak stalo, útočníkův řetězec by předběhl ten původní. (51% útok)

Stromy Merkle



Vlevo: K prokázání platnosti větvě stačí uvést pouze malý počet uzlů ve stromu Merkle.

Vpravo: Jakýkoliv pokus o změnu kterékoliv části stromu Merkle nakonec povede k nekonzistenci v nějakém místě řetězce.

Důležitou funkcí škálovatelnosti Bitcoinu je to, že blok je uložen ve víceúrovňové datové struktuře. „Hash“ bloku je ve skutečnosti pouze hash záhlaví bloku – zhruba 200 bajtová část dat obsahující časovou známku, nonci, hash předchozího bloku a kořenový hash struktury zvané strom Merkle, který ukládá všechny transakce v bloku. Strom Merkle je typ binárního stromu, který se skládá ze sady uzlů s velkým počtem listových uzlů (leaf nodes) ve spodní části stromu, které obsahují

podkladová data ze sady pokročilých uzlů (intermediate nodes), kde každý uzel je hash jeho dvou dětí, a nakonec jediný kořenový uzel, který je také vytvořený z hashů jeho dvou potomků, a který představuje pomyslný vrchol stromu. Účelem stromu Merkle je umožnit postupné doručování dat v bloku; uzel si může stáhnout pouze záhlaví bloku z jednoho zdroje a malou část stromu, která je pro něj relevantní z jiného zdroje, a stále si může být jistý, že všechna data jsou správná. Důvodem, proč to funguje je to, že hashe se šíří směrem vzhůru: pokud se uživatel s nekalými úmysly pokusí vyměnit falešnou transakci ze spodní části stromu Merkle, tato změna způsobí změnu bloku uvedeného nad blokem s falešnou transakcí, a tato změna způsobí další změnu bloku uvedeného nad tímto blokem. Nakonec se změní kořen stromu a hash bloku, což způsobí , že jej protokol zaregistruje jako úplně jiný blok s neplatným důkazem o práci.

Pro dlouhodobou udržitelnost je protokol stromu Merkle pravděpodobně nezbytný. „Plný uzel“ (full node) v Bitcoinové síti, který ukládá a zpracovává každý celý blok od dubna 2014 zabírá přibližně 15 GB místa na disku v Bitcoinové síti, a jeho velikost roste každý měsíc o více než 1 GB. V současné době je tato velikost přijatelná pro některé stolní počítače, nikoliv telefony, a v budoucnu se budou moci účastnit pouze firmy a fanoušci. Protokol známý jako „Zjednodušené ověřování plateb“ (SPV) umožňuje existenci další třídy uzlů, tzv. „lehkých uzlů“ (Light nodes), které stahují záhlaví bloků a následně stahují pouze větve spojené s transakcemi, které jsou pro ně relevantní. To umožňuje lehkým uzlům se silnou zárukou bezpečnosti určit, jaký je stav jakékoliv Bitcoinové transakce a jejich aktuální zůstatek, zatímco stahují pouze velmi malou část celého blockchainu.

Alternativní využití blockchainu

Nápad převzít základní myšlenku blockchainu, a aplikovat ji na jiné koncepty má také dlouhou historii. V roce 1998 přišel Nick Szabo s konceptem bezpečných vlastnických titulů s autoritou vlastníka, dokumentem popisujícím, jak „nový pokrok v technologii replikovaných databází“ umožní systémům na bázi blockchainu ukládat registr, čímž vytvoří propracovaný rámec zahrnující pojmy jako usedlost, nepříznivé držení nebo Gruzínská daň z pozemku. V té době bohužel nebyl k dispozici žádný efektivní replikovaný databázový systém, a proto nebyl protokol nikdy uveden do praxe. Po roce 2009 se však po rozvinutí decentralizovaného konsenzu Bitcoinu začala rychle objevovat řada alternativních aplikací.

- **Namecoin** – byl vytvořen v roce 2010 a lze ho nejlépe popsat jako decentralizovanou databázi pro registraci jmen. V decentralizovaných protokolech jako jsou Tor, Bitcoin a BitMessage musí existovat nějaký způsob identifikace účtů, aby s nimi mohli komunikovat ostatní lidé, avšak ve všech existující řešeních je jediným dostupným identifikátorem pseudonáhodný hash (jako: 1LW79wp57BqaHW1jL5TCiBCrhQYtHagUWy). V ideálním případě by člověk chtěl mít možnost vlastnit účet s názvem jako „george“. Problém však je, že pokud si jedna osoba může vytvořit účet s názvem „george“, může někdo jiný použít stejný proces k registraci „george“ také pro sebe, a vydávat se za původního vlastníka. Jediným řešením je first-to-file paradigma, kde první registr uspěje, a druhý selže – problém dokonale vhodný pro konsensuální protokol Bitcoinu. Namecoin je nejstarší a nejúspěšnější implementací systému registru jmen, která využívá myšlenku blockchainu.

- **Barevné mince (Colored coins) –** jejich účelem je sloužit jako protokol, který lidem umožňuje vytvářet vlastní digitální měny – nebo v jednom důležitém triviálním případě měny s jednou jednotkou, a to digitálními tokeny na Bitcoinovém blockchainu. V protokolu barevných mincí se nová měna „vydá“ přiřazením barvy konkrétnímu Bitcoinovému UTXO, a protokol rekurzivně definuje barvu ostatních UTXO tak, aby byla stejná jako barva vstupů utracených transakcí, která je vytvořila (v případě vstupů se smíšenými barvami platí některá zvláštní pravidla). To umožňuje uživatelům uchovávat peněženky obsahující pouze UTXO určité barvy a odesílat je podobně, jako běžné Bitcoiny, a dále skrz blockchain zpětně sledovat a určovat barvu jakéhokoliv UTXO, které dostávají.

- **Meta mince (Metacoins) –** myšlenkou Metacoinu je poskytnout protokol, který žije na vrcholu Bitcoinu, a používá Bitcoinové transakce k ukládání transakcí Metacoinu, avšak má odlišnou funkci stavového přechodu, APPLY'. Protože protokol Metacoinu nemůže zabránit tomu, aby se neplatné transakce Metacoinu objevily v Bitcoinovém blockchainu, bylo přidáno pravidlo, že pokud APPLY' (S, TX) pošle zpět chybu, Metacoin použije protokol APPLY' (S, TX)= S. To poskytuje snadný mechanismus pro vytvoření libovolného protokolu kryptoměny, potenciálně s pokročilými funkcemi, které nelze implementovat uvnitř samotného Bitcoinu, které však disponují velmi nízkými náklady na vývoj, protože složitost těžby a vytváření sítí jsou již zpracovány Bitcoinovým protokolem. Metacoiny byly použity k implementaci některých tříd finančních smluv, registraci jmen a decentralizovaným výměnám.

Obecně tedy existují dva přístupy k budování konsensuálního protokolu: budování nezávislé sítě, a budování protokolu na vrcholu Bitcoinu. První přístup, i když poměrně úspěšný v případě aplikací, jako je Namecoin, je obtížně proveditelný; každá jednotlivá implementace potřebuje vytvořit nezávislý blockchain, a také vytvořit a otestovat veškerý nezbytný stavový přechod a síťový kód. Navíc předpovídáme, že sada aplikací pro decentralizovanou konsensuální technologii bude následovat distribuci zákonů a moci, kde by drtivá většina aplikací byla příliš malá na to, aby zvládla provozovat svůj vlastní blockchain, a poznamenáváme, že existují velké třídy decentralizovaných aplikací, zejména decentralizované autonomní organizace, které potřebují vzájemně komunikovat.

Přístup založený na Bitcoinu má zase tu chybu, že nezdědí funkce zjednodušeného ověřování plateb (SPV), kterými disponuje samotný Bitcoin. U Bitcoinu SPV funguje, protože může použít hloubku blockchainu jako proxy pro svou platnost; v určitém okamžiku, jakmile se předci transakce dostanou dostatečně daleko zpátky, lze s jistotou říci, že byli legitimní součástí stavu. Na druhou stranu meta-protokoly založené na blockchainu nemohou blockchain přinutit, aby nezahrnoval transakce, které nejsou v rámci jejich vlastních protokolů platné. Plně bezpečná implementace meta-protokolu SPV by tedy musela zpětně prohledávat celou cestu na začátek Bitcoinového blockchainu, aby zjistila, zda jsou určité transakce platné. V současné době se všechny „lehké“ implementace Bitcoinových meta-protokolů spoléhají na to, že data budou poskytnuta důvěryhodným serverem, což je pravděpodobně velmi neoptimální výsledek, zejména když jedním z hlavních účelů kryptoměny je eliminovat potřebu důvěry.

Skriptování

I bez jakýchkoliv rozšíření Bitcoinový protokol ve skutečnosti zjednodušuje verzi konceptu „inteligentních smluv.“ UTXO v Bitcoinu může být vlastněno nejen veřejným klíčem, ale také složitějším skriptem v jednoduchém stack-based programovacím jazyce. V tomto paradigmatu transakční výdaje, které UTXO musí poskytnout obsahují data splňující skript. Ve skutečnosti je dokonce i základní mechanismus vlastnictví veřejného klíče implementován pomocí skriptu: vstup vezme jako vstup eliptický podpis křivky, ověří jej proti transakci a adrese, která UTXO vlastní, a pokud je ověření úspěšné, vrátí zpátky 1. V opačném případě vrátí zpátky 0. Pro různé další případy použití existují další, složitější skripty. Například lze vytvořit skript, který vyžaduje podpisy dvou ze tří daných soukromých klíčů k ověření (multisig) - nastavení užitečné pro firemní účty, zabezpečené spořící účty, a v některých situacích pro „merchant escrow“. Skripty lze také použít k výplatě odměn za řešení výpočtových problémů a lze dokonce vytvořit skript, který říká něco jako: „tento Bitcoin UTXO patří vám, pokud můžete poskytnout SPV důkaz o tom, že jste mi zaslali dogecoinovou transakci této nominální hodnoty“, v zásadě umožňující decentralizovanou výměnu mezi kryptoměnami.

Skriptovací jazyk implementovaný v Bitcoinu má však několik zásadních omezení:

- **Nedostatek Turingovské úplnosti** – to znamená, že zatímco existuje velká podmnožina výpočtů, které skriptovací jazyk Bitcoinu podporuje, zdaleka nepodporuje všechny. Hlavní kategorií, která chybí, jsou smyčky. To je z důvodu, aby se zabránilo nekonečným smyčkám během ověřování transakce; teoreticky je to překonatelná překážka pro programátory skriptů, protože jakoukoliv smyčku lze simulovat jednoduchým rozsáhlým opakováním základního kódu s příkazem if, avšak to vede ke skriptům, které jsou prostorově velmi neefektivní. Například implementace alternativního algoritmu podpisu eliptické křivky by pravděpodobně vyžadovala 256 opakovaných kol násobení, která jsou jednotlivě zahrnuta v kódu.
- **Hodnotová slepota** – pro skript UTXO neexistuje žádný způsob, jak zajistit fine-grained kontrolu nad částkou, kterou lze vybrat. Například jedním mocným případem použití tzv. věštecké smlouvy (oracle contract) by byla zajišťovací smlouva, kde A a B vložily BTC v hodnotě 1000 USD, a po třiceti dnech skript pošle BTC v hodnotě 1000 USD straně A, a zbytek straně B. To by vyžadovalo, aby věštec určil hodnotu 1 BTC v USD, ale i tak jde o masivní zlepšení z hlediska důvěryhodnosti a požadavků na infrastrukturu oproti plně centralizovaným řešením, která jsou v současné době k dispozici. Protože jsou však UTXO all-or-nothing (všechno nebo nic), jediným způsobem, jak toho dosáhnout, je velmi neefektivní hack, který má mnoho UTXO různých nominálních hodnot (například jeden UTXO 2k za každé k až do 30) a výběr O, který vybere, která UTXO pošle straně A, a která straně B.
- **Nedostatek stavu** – UTXO může být buď utraceno, nebo neutraceno; neexistuje možnost pro vícestupňové smlouvy nebo skripty, které udržují jakýkoliv jiný vnitřní stav nad tento rámec. To ztěžuje uzavírání vícestupňových smluv opcí, decentralizovaných výměnných nabídek, nebo dvoustupňových kryptografických závazkových protokolů (jsou nutné pro bezpečné výpočetní odměny). Také to znamená, že UTXO lze použít pouze k vytvoření jednoduchých

jednorázových smluv, a nikoliv k vytvoření složitějších „stavových smluv“, jako jsou decentralizované organizace. To ztěžuje implementaci meta-protokolů. Binární stav v kombinaci s hodnotovou slepotou také znamená, že další důležité využití (výběrové limity) není k dispozici.

- **Blockchainová slepota** – UTXO jsou slepá vůči blockchainovým datům jako je nonce, časové razítko a hash předchozího bloku. To vážně omezuje použití na poli hazardních her a v několika dalších kategoriích tím, že skriptovací jazyk připravuje o potenciálně cenný zdroj náhodnosti.

Vidíme tedy tři přístupy k budování pokročilých aplikací na vrcholu kryptoměny: budování nového blockchainu, používání skriptování na vrcholu Bitcoinu a budování meta-protokolu na vrcholu Bitcoinu. Budování nového blockchainu umožňuje neomezenou svobodu při vytváření sady funkcí, avšak za cenu času vývoje, úsilí bootstrappingu a bezpečnosti. Použití skriptování je snadné implementovat a standardizovat, avšak jeho schopnosti jsou velmi omezené. Dále meta-protokoly, i když velmi snadné, trpí poruchami škálovatelnosti. S pomocí Etherea hodláme vybudovat alternativní rámec, který poskytuje ještě snadnější vývoj, silnější vlastnosti lehkých klientů (light clients), a současně umožňuje aplikacím sdílet ekonomické prostředí a zabezpečení blockchainu.

Ethereum

Záměrem Etherea je vytvořit alternativní protokol pro vytváření decentralizovaných aplikací, který by poskytoval odlišnou řadu kompromisů, o kterých se domníváme, že pro velkou třídu decentralizovaných aplikací budou velmi užitečné, se zvláštním důrazem na situace, při kterých je čas na rychlý vývoj, bezpečnost pro malé a zřídka kdy užívané aplikace, a schopnost různých aplikací velmi efektivně komunikovat. Ethereum to dělá tak, že v podstatě vytváří konečnou abstraktní základní vrstvu: blockchain s kompletním vestavěným programovacím jazykem Turing, který umožňuje komukoli psát chytré smlouvy a decentralizované aplikace, kde si může vytvořit libovolná pravidla pro vlastnictví, transakční formáty, a funkce stavového přechodu. Holá verze Namecoinu může být napsána ve dvou řádcích kódu a další protokoly, jako jsou měny a reputační systémy, mohou být zabudovány do dvaceti následujících řádků. Inteligentní smlouvy, kryptografické „boxy“ obsahující uzamčené hodnoty, které odemknou pouze po splnění určitých podmínek, lze také stavět na horní části platformy s mnohem větší silou, než jakou nabízí skriptování s Bitcoinem, kvůli přidaným schopnostem Turingovské úplnosti, povědomím o hodnotě, blockchainovém povědomí a stavu.

Filozofie

Návrh stojící za Ethereem má dodržovat následující postupy:

1. **Jednoduchost** – Protokol Ethereum by měl být co nejjednodušší, a to i za cenu většího úložiště dat, nebo časové neefektivity. Průměrný programátor by měl v ideálním případě být schopen sledovat a implementovat celou specifikaci, za účelem plného využití bezprecedentního demokratizačního potenciálu, který kryptoměna přináší, a poskytnout další vize Etherea jako protokolu, který je přístupný všem. Jakákoliv optimalizace zvyšující složitost by neměla být zahrnuta, pokud neposkytuje protokolu velmi podstatnou výhodu.
2. **Univerzálnost** – Základním aspektem Etherea je to, že nemá „funkce“. Místo nich Ethereum poskytuje kompletní interní skriptovací jazyk Turing, který programátor může použít k vytvoření libovolné inteligentní smlouvy nebo typu transakce, který lze matematicky definovat. Chtěli byste vytvořit svůj vlastní finanční derivát? S Ethereem můžete. Chcete si vytvořit vlastní měnu? Nastavte ji jako smlouvu na Ethereu. Chcete nastavit Daemon nebo Skynet v plném rozsahu? Možná budete potřebovat několik tisíc vzájemně propojených smluv, které budete muset štědře „krmit“ abyste dosáhli svého cíle, s Ethereem na dosah ruky vás však nic nezastaví.
3. **Modularita** – Části protokolu Ethereum by měly být navrženy tak, aby byly co nejvíce modulární a oddělitelné. V průběhu vývoje je naším cílem vytvořit program, ve kterém kdyby se na jednom místě ocitla malá modifikace, všechny funkce by fungovaly bez jakékoliv další úpravy. Inovace jako jsou Ethash (viz příloha Yellow paper, nebo článek wiki), nebo stromy Patricia (Yellow paper, wiki), jsou a měly by být implementovány jako samostatné knihovny s úplnými funkcemi. I když se používají na Ethereu, které nevyžaduje určité funkce, tyto funkce jsou stále použitelné i v jiných protokolech. Vývoj Etherea by měl být prováděn tak, aby byl přínosem pro celý ekosystém kryptoměn, a nejen pro Ethereum samotné.
4. **Ohebnost** – Podrobnosti protokolu Ethereum nejsou pevně dané. I když budeme nesmírně uvážliví ohledně úprav konstrukcí na vysoké úrovni, například s plánem horizontálního dělení nebo abstraktním provedením, bude tak možné učinit pouze s dostupností dat zakotvenou v konsenzu. Výpočtové testy v procesu vývoje nás později mohou vést ke zjištění, že určité úpravy, např. úpravy architektury protokolu nebo virtuálního počítače Ethereum (EVM) podstatně zlepší škálovatelnost nebo zabezpečení. Pokud se takové příležitosti naskytnou, rozhodně je využijeme.
5. **Žádná diskriminace nebo cenzura** – Protokol by se neměl pokoušet aktivně omezovat nebo bránit užívání konkrétních kategorií. Všechny regulační mechanismy v protokolu by měly být navrženy tak, aby přímo regulovaly poškození a nesnažily se bránit konkrétním nežádoucím aplikacím. Programátor může dokonce spustit skript nekonečné smyčky na vrcholu Etherea po tak dlouhou dobu, po jakou je ochoten platit transakční poplatek za výpočetní kroky.

Účty Ethereum

V Ethereu je stav tvořen objekty, které nazýváme „účty“, přičemž každý takový účet disponuje 20ti bajtovou adresou, a přechody stavu jsou přímé přechody hodnoty společně s informacemi mezi účty. Účet Ethereum obsahuje čtyři pole:

- **Nonce** – čítač používaný k zajištění toho, že každá transakce může být zpracována pouze jednou
- **Aktuální zůstatek ETH na účtu**
- **Kód smlouvy účtu, je-li k dispozici**
- **Úložiště účtu** (ve výchozím nastavení – prázdné)

„Ether“ je hlavní interní kryptopalivo Etherea, a používá se k placení transakčních poplatků. Obecně existují dva typy účtů: **účty externě vlastněné,** které jsou ovládány soukromými klíči, a **smluvní účty**. Externě vlastněný účet nemá žádný kód a zprávy z něj je možné externě odesílat vytvořením a podepsáním transakce; u smluvního účtu, pokaždé, když účet obdrží zprávu, aktivuje se kód účtu, který umožní číst a zapisovat do interního úložiště, a odesílat další zprávy, nebo postupně vytvářet smlouvy.

Pamatujte si, že „smlouvy“ v Ethereu by neměly být vnímány jako něco, co by mělo být „splněno“; jsou spíš něco jako „autonomní agenti“ žijící uvnitř prostředí Etherea, a vždy provádějí konkrétní část kódu, když jsou k tomu donuceni. Mají přímou kontrolu nad vlastním Etherovým zůstatkem a vlastním klíčem/úložištěm hodnot pro sledování trvalých proměnných.

Zprávy a transakce

Termín „transakce“ se v Ethereu používá k označení podepsaného datového balíčku ukládajícího zprávu, která má být odeslána z externě vlastněného účtu.

Tyto transakce obsahují:

- **Příjemce zprávy**
- **Podpis, který identifikuje odesílatele zprávy**
- **Množství Etheru, které se má převést od odesílatele k příjemci**
- **Nepovinné datové pole**
- **Hodnotu STARTGAS, představující maximální počet výpočetních kroků, které provedená transakce zvládne**
- **Hodnotu GASPRICE, která představuje poplatek za výpočetní krok, hrazený odesílatelem**

První tři body jsou aplikovatelné v jakékoliv kryptoměně. Datové pole nemá ve výchozím nastavení žádnou funkci, avšak EVM disponuje operačním kódem, který může smlouva použít pro přístup k datům; například, pokud smlouva funguje jako služba registrace domény na blockchainu, může

vyžadovat interpretaci dat, která jí jsou předávána. Obsahovala by dvě pole, přičemž první pole je doména k registraci a druhé pole je IP adresa, na kterou bude doména registrována. Smlouva by tyto hodnoty přečetla z dat zprávy, a vhodně je umístil do úložiště.

Pole STARTGAS a GASPRICE jsou zásadní pro model „anti-denial of service“ společnosti Ethereum. Aby se zabránilo náhodným, nebo úmyslně vytvořeným nekonečným smyčkám (nebo jinému výpočetnímu plýtvání kódu), je u každé transakce vyžadováno nastavení limitu počtu výpočetních kroků, které může kód použít. Základní jednotkou tohoto výpočtu je „gas“; jeden výpočetní krok obvykle stojí 1 gas, avšak některé výpočetně složitější operace stojí více, protože zvyšují množství dat, která musí být uložena jako součást stavu. Za každý bajt v transakčních datech je také účtován poplatek ve výši 5 gas. Záměrem tohoto systému poplatků je požadovat, aby za každý zdroj, který útočník spotřebuje musel zaplatit (včetně výpočtu, šířky pásma a úložiště); proto každá transakce vedoucí k použití většího množství těchto zdrojů musí mít gasový poplatek úměrný přírůstku zdrojů.

Zprávy

Smlouvy mají schopnost posílat zprávy jiným smlouvám. Zprávy jsou virtuální objekty, které nejsou nikdy serializovány, a existují pouze v prostředí Ethereum.

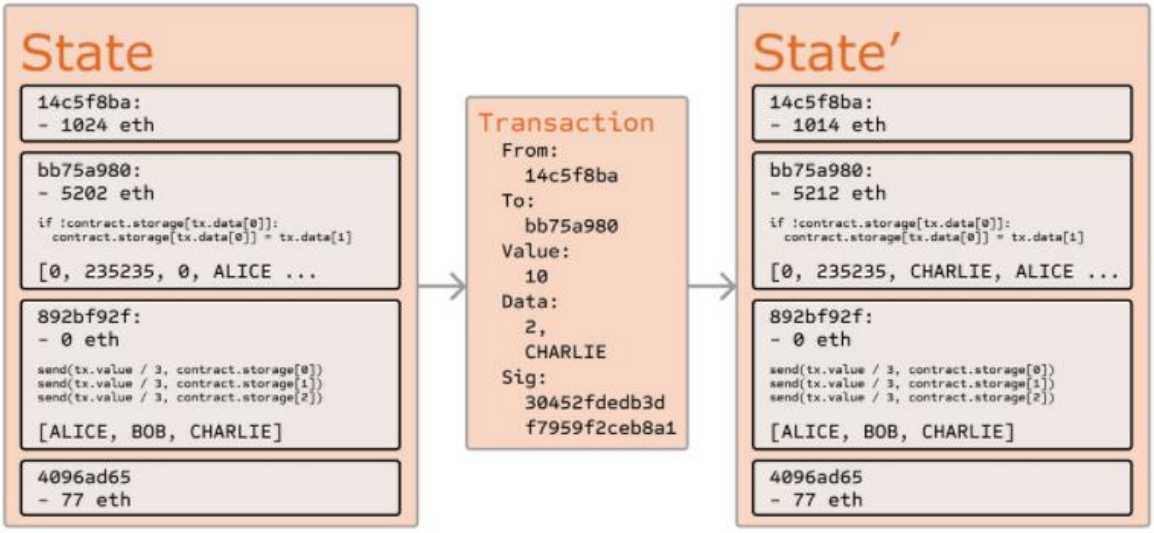
Každá zpráva obsahuje:

- **Odesílatele zprávy** (implicitní)
- **Příjemce zprávy**
- **Množství Etheru přenášeného společně se zprávou**
- **Nepovinné datové pole**
- **Hodnotu STARTGAS**

Zpráva je v podstatě jako transakce s tím rozdílem, že je vytvořena smlouvou, a nikoliv externím aktérem. Zpráva je vytvořena, když smlouva provádějící kód spustí operační kód CALL. Stejně jako transakce, vede zpráva k účtu příjemce, na kterém běží jeho vlastní kód. Smlouvy tak mohou navazovat „vztahy“ s jinými smlouvami přesně tak, jakým to mohou dělat externí aktéři.

Pamatujte, že povolení na gas přiřazené k transakci nebo smlouvě se vztahuje na celkový gas spotřebovaný touto transakcí a všech jejích dílčích procesů. Například pokud externí aktér A pošle druhému externímu aktérovi B transakci s 1000 gas, a B spotřebuje 600 gas před odesláním zprávy externímu aktérovi C (toto interní provedení spotřebuje před návratem 300 gas), aktér B pak může před spuštěním utratit dalších 100 gas, než dojde „palivo“ (celkově 1000 gas).

Funkce stavového přechodu Etherea



Funkci stavového přechodu Etherea, APPLY (S, TX) - >S' lze definovat takto:

1. Zkontroluje, zda je transakce dobře vytvořená (tj. Má správný počet hodnot), podpis je platný a nonce příjemce odpovídá nonci na účtu odesílatele. Pokud ne, vrátit zpět error.
2. Vypočítat transakční poplatek, jako - STARTGAS*GASPRICE, z podpisu určit adresu odeslání. Pokud není k dispozici dostatečný Etherový zůstatek, vrátit zpátky error.
3. Inicializovat GAS=STARTGAS, a odebrat určité množství gas na bajt, aby se uhradil poplatek za bajty v transakci.
4. Převést hodnotu transakce z účtu odesílatele na účet příjemce.
5. Pokud přijímací účet zatím neexistuje, vytvořit ho. Je-li přijímacím účtem smlouva, spouštět její kód, dokud nedojde k dokončení transakce, nebo vyčerpání gas.
6. Pokud se převod hodnoty nezdařil, protože odesílatel neměl dostatek peněz, nebo došlo k vyčerpání gas, obnovit všechny změny stavu kromě uhrazených poplatků, které se přidají na účet těžaře.
7. V opačném případě vrátit poplatky za veškerý zbývající gas odesílateli a poplatky za spotřebovaný gas těžaři.

Předpokládejme, že kód smlouvy je:

```
if !self.storage[calldataload(0)]:
    self.storage[calldataload(0)] = calldataload(32)
```

Pamatujte, že ve skutečnosti je kód smlouvy napsán v nízko-úrovňovém kódu EVM; kvůli srozumitelnosti je tento příklad napsán v Serpentu, jednom z našich vysokoúrovňových jazyků, který lze zkompilovat do kódu EVM. Předpokládejme, že úložiště smlouvy je ze začátku prázdné, a transakce má hodnotu 10 ETH, 2000 gas, 0,001 GASPRICE a 64 bajtů dat, přičemž 0-31 bajtů představuje číslo 2, 32-63 bajtů představuje hodnotu řetězce CHARLIE. Proces funkce přechodového stavu je v tomto případě následující:

1. Zkontrolovat, zda je transakce platná a správně vytvořená.
2. Zkontrolovat, zda má odesílatel transakce alespoň 2000 * 0,001 = 2 ETH. Pokud ano, odečíst 2 ETH.
3. Inicializovat gas = 2000; za předpokladu, že má transakce 170 bajtů a poplatek za 1 bajt činí 5 gas, odečíst 850 gas. (zbývá 1150 gas)
4. Odečíst z účtu odesílatele dalších 10 ETH a přidat je na účet smlouvy.
5. Spustit kód. V tomto případě je to triviální: zkontroluje se, zda je používáno úložiště smlouvy na indexu 2, pokud to není pravda, nastavit úložiště na indexu 2 na hodnotu CHARLIE. Předpokládejme, že to vyžaduje 187 gas, takže zbývající počet gas bude 1150 – 187 = 963.
6. Přidat 963 * 0,001 = 0,963 ETH zpět na učet odesílatele a vrátit zpět výsledný stav.

Pamatujte, že zprávy a transakce jsou rovnocenné, pokud se jedná o reverty: dojde-li ke spuštění zprávy, tato zpráva a všechno ostatní spuštěné tímto procesem se vrátí zpět (až na rodičovské spuštění – „parent execution“, to se vrátit nemusí). Znamená to, že pro smlouvu je bezpečné „volat“ jinou smlouvu. Například kdyby smlouva A volala smlouvu B pomocí gasu G, provedení smlouvy A by spotřebovalo maximálně G gas. Existuje operační kód CREATE, který vytváří smlouvu; jeho mechanika je obecně podobná kódu CALL s tím rozdílem, že výstup procesu určuje kód nově vytvořené smlouvy.

Provedení kódu

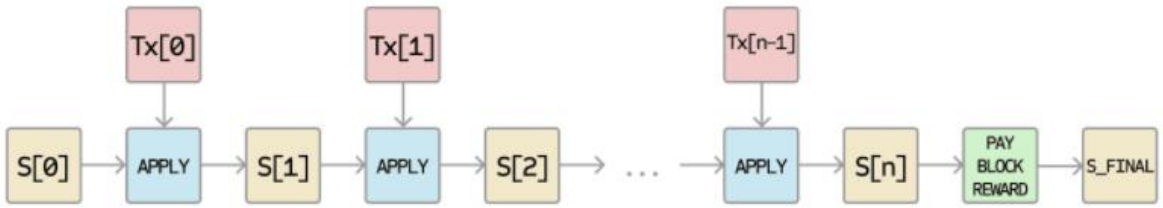
Kód v kontraktech Etherea je napsán v nízko-úrovňovém stack-based bytecode jazyce, který se označuje jako „kód virtuálního počítače Ethereum“, nebo „kód EVM“. Kód se skládá z řady bajtů, ve kterém každý bajt představuje určitou operaci. Obecně platí, že proces kódu je nekonečná smyčka, která spočívá v opakovaném provádění operace na aktuálním počitadle programu (začíná na nule), a následném zvyšování počitadla programu o jednu, dokud není dosaženo kódu, chyby, STOPu, nebo není detekován příkaz RETURN. Tyto operace mají přístup ke třem typům prostoru, do nichž jsou ukládána data:

- **Stack**, kontejner typu last-in-first-out, do kterého lze tlačit a zasouvat hodnoty
- **Paměť**, nekonečně rozšiřitelné bajtové pole
- **Dlouhodobé úložiště smlouvy, úložiště klíčů/hodnot**. Na rozdíl od stacku a paměti, které se po ukončení výpočtu resetují, úložiště dlouhodobě přetrvává.

Kód může také přistupovat k hodnotě, odesílateli, a datům příchozí zprávy stejně, jako k datům záhlaví bloku, a vrátit zpět bajtové pole dat jako výstup.

Formální model provedení kódu EVM je překvapivě jednoduchý. Zatímco běží virtuální počítač Ethereum (EVM), jeho plný výpočetní stav může být definován n-ticí (block_state, transakce, zpráva, kód, paměť, stack, pc, plyn), kde block_state je globální stav obsahující všechny účty a zahrnuje zůstatky a úložiště. Na začátku každého kola procesu je aktuální příkaz nalezen převzetím pc-tého bajtu kódu (nebo 0, pokud pc>=len(kód)), a každý příkaz má svou vlastní definici, pokud jde o to, jak definuje n-tice. Například ADD vyhodí ze stacku dvě položky a předá dál jejich součet, sníží plyn o 1 a zvýší pc o 1. SSTORE vyhodí vrchní dvě položky stacku , a vloží druhou položku do úložiště smlouvy do indexu specifikovaného první položkou. Ačkoliv existuje mnoho způsobů, jak specifikovat spuštění EVM pomocí kompilace just-in-time, základní implementaci Etherea lze provést v několika stovkách řádků kódu.

Blockchain a těžba



Blockchain Etherea je v mnoha ohledech podobný tomu Bitcoinovému, i když s určitými rozdíly. Hlavním rozdílem, pokud jde o blockchainovou architekturu je, že na rozdíl od Bitcoinu (který obsahuje pouze kopii seznamu transakcí), bloky Ethereum obsahují kopii seznamu transakcí i nejnovějšího stavu. Kromě toho jsou v bloku uloženy další dvě hodnoty, číslo bloku a obtížnost. Algoritmus ověření základního bloku Etherea je následující:

1. Zkontrolovat, zda předchozí odkazovaný blok existuje a je platný.
2. Zkontrolovat, zda je časová známka bloku větší než ta, která patří předchozímu bloku, avšak je menší než 15 minut do budoucna.
3. Zkontrolovat, zda je počet bloků, obtížnost, kořen transakce, root a limit plynu (různé koncepty specifické pro nízko úroveň Etherem) platný.

4. Zkontrolovat, zda je proof of work platný.
5. Na konci předchozího bloku musí být S[0]
6. TX je seznam bloků s transakcemi Pro všechna i v 0…n-1 nastavit S[i + 1] = APPLY (S[i],TX [i]). Pokud některá aplikace vrátí zpět error, nebo pokud celkový gas do té doby spotřebovaný v bloku překročí GASLIMIT, vrátit zpět error.
7. S_FINAL = S[n], přidat odměnu těžaři
8. Zkontrolovat, zda se kořen stromu Merkle stavu S_FINAL rovná konečnému stavu kořene uvedeného v záhlaví bloku. Pokud ano, blok je platný. V opačném případě není.

Tento přístup se na první pohled může zdát být vysoce neefektivním, protože musí ukládat celkový stav s každým novým blokem, avšak ve skutečnosti by účinnost měla být srovnatelná s tou Bitcoinovou. Důvodem je to, že stav je uložen ve stromové struktuře, a s každým novým blokem je třeba změnit jen malou část stromu. Obecně by tedy mezi dvěma sousedními bloky měla být drtivá většina stromu totožná, a proto lze data uložit jednou a odkazovat na ně dvakrát pomocí ukazatelů (tj. Hashů podstromů). K tomu se používá speciální druh stromu známý jako „strom Patricia“, včetně modifikace konceptu stromu Merkle, která umožňuje efektivní vkládání a následné mazání uzlů. Protože všechny informace o stavu jsou součástí posledního bloku, není nutné ukládat celou historii blockchainu – tato strategie by musela být vypočítána tak, aby poskytovala 5-20x větší úsporu místa.

Běžně kladenou otázkou je, kde je prováděn kód smlouvy, jde-li o fyzický hardware. Odpověď je jednoduchá: Proces provádění smluvního kódu je součástí funkce stavového přechodu, která je zahrnuta do algoritmu ověření bloku. Takže pokud je do bloku B přidána transakce, provedení kódu vytvořené touto transakcí projde všemi uzly, které tento blok ověřují. A to nyní, i v budoucnu.

Použití

Obecně existují na vrcholu Etherea tři kategorie aplikací. První kategorií jsou finanční aplikace, které uživatelům poskytují výkonnější způsob správy a uzavírání smluv s využitím jejich financí. To zahrnuje dílčí měny, finanční deriváty, zajišťovací smlouvy, spořící účty, závěti, a dokonce i některé třídy pracovních smluv v plném rozsahu. Druhou kategorií jsou semi-finanční aplikace, kde se jedná o peníze, avšak je zde také těžká nepeněžní stránka aktuálního dění; dokonalým příkladem je samo-prosazující se odměna za řešení výpočetních problémů. A konečně existují aplikace, jako je online hlasování a decentralizovaná správa, kde o finance vůbec nejde.

Systémy tokenů

Systémy tokenů mají mnoho aplikací od sub-měn, představujících aktiva od USD nebo zlata až po akcie společností, individuálních tokenů představujících inteligentní majetek, a dokonce i systémů tokenů bez vazby na konvenční hodnotu, používané jako bodové systémy pro pobídky. Tokenové systémy se v Ethereu implementují překvapivě snadno. Klíčovým bodem k jejich pochopení je to, že měna je v zásadě databáze s jednou operací: Odečtěte X jednotek měny z A a předejte je B s tím, že A měla před transakcí dostatečný zůstatek a schválila transakci. Vše potřebné k fungování tokenového systému je implementace této logiky do smlouvy.

Základní kód pro implementaci tokenového systému v Serpentu vypadá takto:

```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

Jedná se v zásadě o doslovnou implementaci funkce stavového přechodu do „bankovního systému“. Tato implementace je dále popsána v tomto dokumentu. Je třeba přidat několik dalších řádků kódu, které zajistí první krok distribuce měnových jednotek, a několik dalších okrajových případů.
V ideálním případě by byla přidána funkce, která umožní ostatním smlouvám dotazovat se na zůstatek adresy. To je však vše. Teoreticky mohou tokenové systémy založené na Ethereu, fungující jako dílčí měny, potenciálně zahrnovat další vlastnost, kterou on-chain Bitcoin-based meta-měny postrádají: schopnost platit transakční poplatky přímo v této měně. Způsob, jakým by to bylo implementováno je, že smlouva by udržovala rovnováhu ETH, díky které by vrátila ETH použitý k úhradě poplatků odesílateli a doplnila by to tím, že by sbírala vnitřní měnové jednotky, které by přijímala, a prodávala je v neustále probíhajících aukcích. Uživatelé by tedy museli aktivovat své účty pomocí Etheru, který by byl znovu použitelný, protože by jim ho smlouva pokaždé navrátila.

Finanční deriváty a měny stabilní hodnoty

Finanční deriváty jsou nejběžnějším využitím inteligentní smlouvy, a jsou velmi jednoduše implementovány do kódu. Hlavní výzvou při provádění finančních smluv je, že většina z nich vyžaduje odkaz na externí cenovou kalkulaci; například velmi žádoucí je inteligentní smlouva, která zajišťuje uživatele proti volatilitě Etheru (nebo jiné kryptoměny) vůči USD, ale to vyžaduje, aby smlouva věděla, jaká je hodnota ETH/USD. Nejjednodušší způsob, jak toho docílit, je prostřednictvím smlouvy o „datovém kanálu“ (data feed) udržované konkrétní stranou (např. NASDAQ) navržené tak, aby tato

strana měla možnost smlouvu podle potřeby aktualizovat, a aby poskytovala rozhraní, které by umožňovalo odesílat zprávu jiným smlouvám a získat zpět odpověď, která by stanovila cenu.

Vzhledem k této kritické složce by zajišťovací smlouva vypadala takto:

1. Počkat, až strana A zadá vstup 1000 ETH.
2. Počkat, až strana B zadá vstup 1000 ETH.
3. Zaznamenat hodnotu 1000 ETH v USD, vypočtenou dotazem na smlouvu o přenosu dat, do úložiště, například $x.
4. Po třiceti dnech nechat stranu A nebo stranu B „znovu aktivovat“ smlouvu, aby bylo možné odeslat ETH (v hodnotě $x)(počítáno opětovným dotazem na smlouvu o přenosu dat, aby se získala nová cena) straně A, a zbytek vrátit straně B.

Taková smlouva by měla významný potenciál na trhu s kryptoměnami. Jedním z hlavních problémů uváděných v souvislosti s kryptoměnami je jejich volatilita; ačkoliv mnoho uživatelů a obchodníků může požadovat bezpečnost a pohodlí při práci s kryptografickými aktivy, nemusí chtít čelit vyhlídce na ztrátu 23% hodnoty svých prostředků za jeden jediný den. Doposud nejběžněji navrhovaným řešením byla aktiva zajištěná emitentem; myšlenka spočívá v tom, že emitent vytvoří dílčí měnu, ve které má právo vydávat a odvolávat jednotky, a následně poskytovat jednu jednotku měny každému, kdo mu offline poskytne jednu jednotku specifikovaného podkladového aktiva (např. zlato, USD). Emitent poté slibuje, že poskytne jednu jednotku kryptoměny každému, kdo pošle zpět jednu jednotku podkladového aktiva. Tento mechanismus umožňuje, aby bylo jakékoliv ne-kryptografické aktivum „pozvednuto“ na kryptografické za předpokladu, že emitentovi lze důvěřovat.

V praxi však emitenti nejsou vždy důvěryhodní, a v některých případech je bankovní infrastruktura příliš slabá nebo nepřátelská na to, aby takové služby mohly existovat. Finanční deriváty nabízejí alternativu. Místo toho, aby jeden emitent poskytoval finanční prostředky pro zálohování aktiva, tuto roli hraje decentralizovaný trh spekulantů, který sází na to, že cena kryptografického referenčního aktiva (např. ETH) vzroste. Na rozdíl od emitentů nemají spekulanti možnost neplnění své strany dohody, protože zajišťovací smlouva drží jejich prostředky v úschově. Všimněte si, že tento přístup není plně decentralizovaný, protože pro zajištění ceníku je stále třeba důvěryhodný zdroj, i když je pravděpodobné, že i přesto se jedná o výrazné zlepšení, pokud jde o snížení požadavků na infrastrukturu (na rozdíl od emitenta vydávání ceníku nevyžaduje žádné licence a lze jej klasifikovat jako svobodu projevu) a snížení možnosti podvodu.

Systém identity a reputace

Nejstarší alternativní kryptoměna ze všech, Namecoin, se pokusila použít blockchain podobný Bitcoinu k vytvoření registru jmen, kde si uživatelé mohou svá jména registrovat ve veřejné databázi spolu s ostatními daty. Hlavní citovaný případ použití je pro systém DNS mapující názvy domén jako „bitcoin.org“ (nebo v případě Namecoinu bitcoin.bit) na IP adresu. Další případy použití zahrnují ověřování e-mailů a potenciálně pokročilejší systémy reputace.

Zde je základní smlouva o poskytnutí registru jmen podobného Namecoinu na Ethereu:

```
def register(name, value):
    if !self.storage[name]:
        self.storage[name] = value
```

Smlouva je velmi jednoduchá; vše je v databázi sítě Ethereum, do které lze jména přidávat, nikoliv však měnit nebo odebírat. Kdokoliv si může zaregistrovat jméno pomocí určité hodnoty, které pak zůstane v registru už navždy. Sofistikovanější smlouva o registru jmen bude mít také „klauzuli funkce“, která bude umožňovat ostatním smlouvám ji vyhledat, a také mechanismus pro vlastníka jména (prvního zaregistrovaného člověka s tímto jménem), který bude měnit data, nebo přenášet vlastnictví.

Decentralizované úložiště souborů

Za posledních několik let se objevila řada populárních start-upů pro ukládání souborů, z nichž nejznámější je Dropbox, který umožňuje uživatelům nahrát zálohu jejich pevného disku. Tato záloha je uživateli za měsíční poplatek volně přístupná. V tomto okamžiku je však trh s úložišti souborů velmi předražený, zejména na úrovni 20-200 GB, pro kterou neplatí bezplatné kvóty ani podnikové slevy jsou měsíční ceny takové, že platíte více, než kolik byste platili za provoz pevného disku po celý měsíc. Smlouvy Ethereum umožňují vývoj decentralizovaného ekosystému pro ukládání souborů, kde si jednotliví uživatelé mohou vydělat malé množství peněz pronajmutím vlastních pevných disků ostatním uživatelům, a nevyužitý prostor lze využít k dalšímu snížení nákladů na ukládání souborů.

Klíčovým základem takového protokolu by bylo něco, co jsme nezvali „decentralizovaná smlouva Dropbox“. Tato smlouva funguje následovně. Uživatel rozdělí požadovaná data do bloků, které zašifruje, aby chránil své soukromí, a následně z něj vytvoří strom Merkle. Dále je uzavřena smlouva s pravidlem, že každých n bloků smlouva vybere náhodný index ze stromu Merkle (pomocí předchozího hashe bloku přístupného z kódu smlouvy – zdroj náhodnosti), a do první entity uloží x ETH, která dodá transakci se zjednodušeným dokladem o ověření platby jako doklad o vlastnictví bloku v daném indexu stromu. Pokud si chce uživatel svůj soubor znovu stáhnout, může k jeho obnovení použít protokol kanálu mikroplatby (1 szabo/32 kilobajtů); cenově nejefektivnějším přístupem je, že uživatel nezveřejňuje transakci až do konce, a místo toho každých 32 kilobajtů nahradí transakci její o něco lukrativnější verzí se stejnou nonce.

Důležitým rysem protokolu je, že i když se na první pohled může zdát, že uživatel důvěřuje mnoha náhodným uzlům, že se nerozhodnou zapomenout jeho soubor, lze toto riziko téměř eliminovat rozdělením souboru na mnoho částí pomocí tajného sdílení a sledování smluv, aby bylo vidět, že každý kus souboru je stále v držení některého z uzlů. Pokud smlouva stále vyplácí peníze, poskytuje tak kryptografický důkaz, že někdo zvenčí stále soubor ukládá.

Decentralizované autonomní organizace

Obecným pojmem „decentralizované autonomní organizace“ se označují virtuální entity, které mají určitý počet členů nebo akcionářů. Tito členové mají s 67% většinou právo utrácet finanční prostředky entity a měnit její kód. Členové společně rozhodují o tom, jak budou rozděleny finanční prostředky organizace. Metody přidělování finančních prostředků DAO se mohou pohybovat od odměn a platů, až po mnohem exotičtější mechanismy, jako je interní měna, která slouží jako odměna za práci. Doposud se většina rozhovorů s tématem DAO týkala „kapitalistického“ modelu „decentralizované autonomní korporace (DAC)“ s akcionáři, kteří přijímají dividendy a obchodovatelné akcie; alternativa označovaná jako „decentralizovaná autonomní komunita“ by měla stejný podíl všech členů na rozhodování a vyžadovala by, aby 67% stávajících členů souhlasilo s přidáním, nebo odebráním jednoho z nich. Požadavek na to, aby jedna osoba mohla mít pouze jedno členství by pak musela skupina kolektivně prosadit.

Obecný přehled toho, jak kódovat DAO je následující. Nejjednodušší design je jednoduše část samočinně se měnícího kódu, který se změní, pokud se dvě třetiny členů na něčem dohodnou. I když je kód teoreticky neměnný, dá se to snadno obejít tím, že bude kód rozdělen na jednotlivé část, a každý z nich bude v samostatné smlouvě. Adresa, na kterou budou smlouvy „volat“ by musela být také uložena v upravitelném úložišti. V jednoduché implementaci takové smlouvy by existovaly tři typy transakcí, které by se lišily údaji v nich poskytnutých:

- [0,i,K,V] zaregistrovat návrh s indexem i, aby byla změněna adresa v indexu úložiště K na hodnotu V
- [1,i] zaregistrovat hlas ve prospěch návrhu i
- [2,1] dokončit návrh i, pokud bylo dostatečné množství hlasů pro

Smlouva by pak měla klauzuli pro každou z nich. To by zachovalo záznam všech změn v otevřeném úložišti spolu se seznamem členů, kteří se hlasování účastnili. Měl by také existovat seznam všech platných členů. Kdyby se jakákoliv změna úložiště dostala ke dvěma třetinám členů, kteří jsou pro, mohla by tuto změnu provést finalizující transakce. Sofistikovanější kostra by také měla vestavěnou hlasovací schopnost pro funkce jako je odeslání transakce, přidání nebo odebrání členů, a mohla by dokonce zajistit delegování hlasování ve stylu „Liquid Democracy“ (tj. Člen může ustanovit jiného člena za svého zástupce, který za něj bude hlasovat. Úkol je tranzitivní, takže pokud člen A ustanoví člena B zástupcem, a člen B určí člena C, člen C určí, jak bude člen A hlasovat). Tento design by umožnil DAO organicky růst jako decentralizovaná komunita, což by lidem nakonec umožnilo delegovat a filtrovat členy. Specialisté, na rozdíl od „současného systému“ mohou snadno začít a přestat existovat podle toho, jak v průběhu času členové mění své uspořádání.

Alternativní model je pro decentralizovanou společnost, kde jakýkoliv účet může mít 0 nebo více akcií, a aby mohl rozhodovat, potřeboval by mít dvě třetiny všech akcií. Kompletní kostra systému by zahrnovala funkčnost správy aktiv, schopnost učinit nabídku pro nákup a prodej, a schopnost nabídky přijímat (nejlépe s mechanismem párování objednávek uvnitř smlouvy). Delegace by také existovala ve stylu Liquid Democracy, čímž by se zevšeobecnil koncept „správní rady“.

Další využití

1. **Spořicí peněženky** – Předpokládejme, že Alice chce udržet své prostředky v bezpečí, ale obává se, že ztratí svůj soukromý klíč, nebo se stane cílem kybernetického útoku. Smlouvu s bankou tedy uzavře takto:
   Samotná Alice může vybrat maximálně 1% prostředků denně. Samotný Bob může vybrat maximálně 1% finančních prostředků za den, ale Alice má právo Bobovi tuto pravomoc odebrat. Alice společně s Bobem mají právo vybrat si kolik jen chtějí. Normálně Alici stačí 1% denně, a pokud si chce Alice vybrat více, může se obrátit na Boba. Dojde-li k hacknutí Alicina klíče, rozběhne se k Bobovi, aby přesunul prostředky na novou smlouvu. Pokud ztratí klíč, Bob nakonec peníze získá (bude dál vybírat 1% denně). Pokud se Bob ukáže jako nedůvěryhodný, může mu Alice odebrat svěřená privilegia.

2. **Pojištění úrody** – Smlouva o finančních derivátech může být snadno uzavřena pomocí datového kanálu počasí namísto jakéhokoliv cenového indexu. Pokud zemědělec v Iowě nakoupí derivát, který se vyplácí nepřímo na základě srážek v Iowě, pak v případě sucha dostane zemědělec peníze, a pokud bude deště dostatek, zemědělec bude šťastný, protože se úrodě bude dařit. To lze obecně rozšířit na pojištění katastrof.

3. **Decentralizovaný zdroj dat** – U rozdílových finančních smluv může být ve skutečnosti možné decentralizovat zdroj dat pomocí protokolu s názvem SchellingCoin, který funguje následovně: všechny strany N vloží do systému hodnotu (např. cena ETH/USD). Hodnoty jsou seřazeny a každý mezi 25. a 75. percentilem dostane jako odměnu jeden token. Každý má motivaci poskytnout odpověď, kterou poskytnou všichni ostatní, a jedinou hodnotou, na které se může velký počet hráčů realisticky shodnout, je zřejmá výchozí hodnota: pravda. Tím se vytvoří decentralizovaný protokol, který může teoreticky poskytnout libovolný počet hodnot, včetně ceny ETH/USD, teploty v Berlíně, nebo dokonce výsledek konkrétního tvrdého výpočtu.

4. **Inteligentní více-podpisová úschova** – Bitcoin umožňuje transakční smlouvy s více podpisy, kde například tři z daných pěti klíčů mohou utrácet finanční prostředky. Ethereum však umožňuje ještě větší granularitu; například tři z pěti mohou utratit až 10% za den, a zbylí dva mohou za den utratit 0.5%. Navíc je to asynchronní – dvě strany mohou zaregistrovat své podpisy na blockchainu v různých časech, a poslední podpis transakci automaticky odešle.

5. **Cloudové výpočty** – Technologie EVM může být také použita k vytvoření ověřitelného výpočetního prostředí, což uživatelům umožňuje požádat ostatní o provedení výpočtu a poté volitelně požádat o důkazy, že výpočty v určitých náhodných kontrolních bodech byly provedeny správně. To umožňuje vytvořit trh cloudových výpočtů, kde se může zúčastnit jakýkoliv uživatel se svým stolním počítačem, notebookem nebo specializovaným serverem.

K zajištění důvěryhodnosti systému lze použít namátkovou kontrolu společně s bezpečnostními depozity (tj. Uzly nemohou se ziskem podvádět). Takový systém však nemusí být vhodný pro všechny úkoly: úkoly, které například vyžadují vysokou úroveň mezi-procesové komunikace nelze snadno provést na velkém cloudu uzlů. Jiné úkoly jsou však mnohem snazší paralelizovat; Na takovou platformu lze snad implementovat projekty jako SETI @ home, folding @ home a genetické algoritmy.

6. **Peer to peer hazardní hry** – Na blockchainu Ethereum lze implementovat libovolný počet protokolů hazardních her typu peer to peer, jako je Frank Stajano, nebo Cyberdice od Richarda Claytona. Nejjednodušší protokol o hazardních hrách je ve skutečnosti jednoduchá smlouva o rozdílu v hashi dalšího bloku, a odtud lze vytvářet další pokročilejší protokoly, které vytvářejí hazardní služby s téměř nulovými poplatky.

7. **Predikční trhy** – Predikční trhy jako Oracle nebo SchellingCoin jsou snadno implementovatelné, a mohou se ukázat být první mainstreamovou aplikací futarchie, jako protokolu správy pro decentralizované organizace.

8. **Decentralizované tržiště na blockchainu** – využívají systém identity a reputace jako základnu

Upravená implementace GHOST

Protokol „Greedy Heaviest Observed Subtree“ (GHOST) je inovace, kterou poprvé představili Yonatan Sompolinsky a Aviv Zohar v prosinci roku 2013. Blockchainy s rychlou dobou potvrzení v současné době trpí sníženou bezpečností kvůli vysokému „zatuchlému“ kurzu – protože šíření bloků po síti trvá určitou dobu, pokud těžař A vytěží blok, a poté těžař B náhodně vytěží další blok, než se blok těžaře A dostane k těžaři B, blok těžaře B bude zamítnut a nepřispěje k zabezpečení sítě. Krom toho existuje problém s centralizací: pokud je těžař A těžební fond s 30% hashpower a B má 10% hashpower, A bude pod rizikem produkce „zatuchlého bloku“ 70% času (od ostatních 30% času vyprodukoval A poslední blok, a tak okamžitě získá data o těžbě), zatímco u B bude časové riziko celých 90%. Pokud je tedy blokový interval dostatečně krátký na to, aby byla zastaralá rychlost vysoká, bude A díky své velikosti podstatně efektivnější. Díky kombinaci těchto dvou efektů blockchainy, které rychle produkují bloky velmi pravděpodobně povedou k tomu, že jeden těžařský fond bude mít dostatečně velké procento síťového hardware, aby měl defacto kontrolu nad procesem těžby.

Jak popsali Sompolinsky a Zohar, GHOST řeší problém ztráty zabezpečení sítě zahrnutím zastaralých bloků do výpočtu toho, který blok má největší celkový důkaz o podpoře práce. Přidává se nejen rodič a další předkové bloku, ale také zastaralí potomci předka bloku (v Ethereu jim říkáme strýcové). Abychom vyřešili druhou otázku předpojatosti centralizace, půjdeme nad rámec protokolu, který popsali Sompolinsky a Zohar, a také poskytneme blokové odměny: zastaralý blok obdrží 87,5% své základní odměny a synovec, který zahrnuje zastaralý blok dostane zbývajících 12,5%. Transakční poplatky se však strýcům nebudou přiznávat.

Ethereum implementuje zjednodušenou verzi GHOST, která má pouze šest bodů. Konkrétně je definována takto:

- Blok musí specifikovat rodiče a 0 nebo více strýců
- Strýc zahrnutý v bloku B musí být přímým potomkem předka k-té generace B, kde 2<= k<=7.
- Nemůže to být předchůdce B
- Strýc musí být platnou hlavičkou bloku, ale nemusí to být dříve ověřený nebo platný blok
- Strýc musí být odlišný od všech strýců zahrnutých v předchozích blocích a všech ostatních strýců zahrnutých ve stejném bloku (bez dvojitého zahrnutí)
- Za každého strýce U v bloku B dostane jeho těžař k jeho coinbase odměně dalších 3,125% a těžař B dostane 93.75% standardní coinbase odměny.

Tato zjednodušená verze GHOST se strýci, které lze zahrnout pouze do sedmi generací byla použita ze dvou důvodů. Prvním důvodem je, že původní GHOST by zahrnoval až moc komplikací ve výpočtu o tom, kteří strýcové jsou platní pro určitý blok. Tím druhým je, že neomezený GHOST s Ethereovou kompenzací odstraňuje motivaci těžaře těžit v hlavním řetězci, a nikoliv v řetězci útočníka.

Poplatky

Protože každá transakce publikovaná do blockchainu ukládá síti náklady na její stažení a ověření, je zapotřebí, aby se zabránilo zneužití jednoho nebo více regulačních mechanismů, obvykle zahrnujících transakční poplatky. Výchozí přístup, který se používá v Bitcoinu, je mít čistě dobrovolné poplatky a spoléhat na to, že těžaři budou fungovat jako pomyslní „vrátní“, a budou stanovovat dynamická minima. Tento přístup byl Bitcoinovou komunitou přijat velmi příznivě, a to zejména proto, že je „tržní“, což umožňuje, aby cenu určovala poptávka a nabídka mezi těžaři a odesílateli transakcí. Problém této argumentace však spočívá v tom, že zpracování transakcí není tržní; i když je intuitivně atraktivní považovat zpracování transakcí za službu, kterou těžař nabízí odesílateli transakce, ve skutečnosti bude každou těžařovu transakci muset zpracovat každý uzel v síti, takže drtivou většinu nákladů na transakci nese třetí strana, nikoliv těžař, který se rozhoduje, zda ji zahrne, či ne. Proto je velmi pravděpodobné, že nastanou problémy.

Jak se však ukázalo, tato chyba v tržním mechanismu se magicky ruší, když je dán konkrétní nepřenosný předpoklad. Argument je následující:

1. Transakce vede k operacím „k“ a nabízí odměnu kR jakémukoliv horníkovi, který ji zahrne. R je nastaveno odesílatelem a „k“ a „R“ jsou předem (zhruba) pro těžaře viditelné.
2. Operace má náklady na zpracování C do libovolného uzlu (tj. Všechny uzly mají stejnou účinnost.)
3. Existuje množství n těžebních uzlů, každý s přesně stejným výkonem zpracování (tj. celkem 1/N)
4. Neexistují žádné úplné uzly, které nejsou těžební

Těžař by byl ochotný transakci zpracovat za předpokladu, že by očekávaná odměna byla vyšší, než náklady. Očekávaná odměna je tedy kR/N, protože miner má šanci 1/N na zpracování dalšího bloku a náklady těžaře jsou jednoduše kC. Těžaři proto budou zahrnovat transakce, kde kR/N > Kc nebo R>NC. Všimněte si, že R je poplatek za operaci poskytovaný odesílatelem, a je tedy spodní hranicí výhody, kterou odesílatel získá z transakce, a NC je cena celé sítě společně se zpracováním operace. Těžaři proto mají motivaci zahrnout pouze ty transakce, jejichž celková výhoda převyšuje náklady.

Ve skutečnosti však existuje několik důležitých odchylek od těchto předpokladů:

1. Těžař za zpracování transakce platí vyšší náklady než ostatní ověřovací uzly, protože delší ověřovací čas zpožďuje šíření bloku a zvyšuje tak šanci, že se blok stane „zatuchlým“.
2. Existují neuzavřené plné uzly.
3. Distribuce těžební energie může v praxi skončit radikálně nerovnoměrně.
4. Existují spekulanti, političtí nepřátelé a blázni, jejichž užitná funkce zahrnuje poškození sítě, a mohou chytře uzavírat smlouvy, jejichž náklady jsou mnohem nižší než náklady placené jinými ověřovacími uzly.

(1)poskytuje těžaři tendenci zahrnovat méně transakcí, (2) zvyšuje NC; tyto dva efekty se tedy alespoň částečně navzájem ruší. Jak?(3) a (4) jsou hlavní otázkou; k jejich vyřešení jednoduše zavedeme „floating cap“: žádný blok nemůže mít více operací než BLK_LIMIT_FACTOR násobek dlouhodobého exponenciálního klouzavého průměru. Konkrétně:

```
blk.oplimit = floor((blk.parent.oplimit \* (EMAFACTOR - 1) +
floor(parent.opcount \* BLK\_LIMIT\_FACTOR)) / EMA\_FACTOR)
```

BLK_LIMIT_FACTOR a EMA_FACTOR jsou konstanty, které budou prozatím nastaveny na 65536 a 1,5, ale po analýze budou pravděpodobně změněny.

V Bitcoinu existuje další faktor, který demotivuje velké rozměry bloků: bude jim déle trvat, než se rozšíří, a proto mají vyšší pravděpodobnost stát se zatuchlými. V Ethereu může rozšíření bloků s vysokou spotřebou gas trvat déle, protože jsou fyzicky větší a protože trvá déle, než se zpracují stavové přechody k ověření. Toto demotivující zpoždění je významným faktorem v Bitcoinu, avšak v Ethereu již méně, a to díky protokolu GHOST: spoléhá se na limity regulovaných bloků, proto poskytuje stabilnější základní linii.

Výpočet a Turingovská úplnost

Důležité je poznamenat, že EVM je Turing-complete; to znamená, že kód EVM dokáže rozluštit jakýkoliv výpočet, který lze myslitelně provést, a to včetně nekonečných smyček. Kód EVM umožňuje smyčkování dvěma způsoby. Za prvé, existuje příkaz JUMP, který umožňuje přeskočit zpět na předchozí místo v kódu, a instrukce JUMPI k podmíněnému skákání, což umožňují příkazy jako while x<27:x=x*2. Za druhé, smlouvy mohou vyvolat další smlouvy, což potenciálně umožňuje opakování rekurze. To přirozeně vede k problému: uživatelé se zlými úmysly mohou v podstatě vyšachovat těžaře a plné uzly tím, že je donutí vstoupit do nekonečné smyčky. Tento problém nazýváme problémem zastavení: v obecném případě neexistuje způsob, jak určit, zda se daný program někdy zastaví.

Jak je popsáno v části o stavovém přechodu, naše řešení funguje tak, že vyžaduje, aby transakce nastavila maximální počet výpočetních kroků, které je povoleno provést, a pokud provedení trvá déle, výpočet se vrátí zpět, avšak poplatky jsou stále účtovány. Zprávy fungují stejným způsobem. Abychom ukázali motivaci našeho řešení, zvažte následující příklady:

- Útočník vytvoří smlouvu, která spustí nekonečnou smyčku, a poté odešle transakci aktivující tuto smyčku těžaři. Těžař zpracuje transakci spuštěním nekonečné smyčky a počká, až jí dojde gas. I když se v realizaci vyčerpá gas a zastaví se v polovině, transakce je stále platná a těžař si za každý výpočetní krok stále nárokuje poplatek od útočníka.
- Útočník vytvoří velmi dlouhou nekonečnou smyčku s úmyslem donutit těžaře, aby udržoval výpočetní techniku v chodu tak dlouho, že v době, kdy výpočet skončí, vyjde několik dalších bloků a nebude možné, aby těžař zahrnul poplatek. Útočník však bude muset zadat hodnotu pro STARTGAS omezující počet výpočtových kroků, které může provést, takže těžař bude předem vědět, že výpočet bude trvat příliš velký počet kroků.
- Útočník vidí smlouvu s kódem v nějaké formě, jako send (A,contract.storage [A]); contract.storage [A]=0, a odešle transakci s dostatečným množstvím gas k provedením prvního kroku, ale nikoliv už druhého (tj. provedení výběru, ale nenechání zůstatku klesnout). Autor smlouvy se nemusí starat o ochranu před takovými útoky, protože pokud se proces zastaví v polovině změn, změny se vrátí zpět.
- Finanční smlouva funguje na základě mediánu devíti vlastních datových zdrojů, aby se minimalizovalo riziko. Útočník převezme jeden z datových kanálů, který je navržen tak, aby jej bylo možné upravit pomocí mechanismu volání proměnné určité adresy, který je popsán v části DAO, a převede jej na spuštění nekonečné smyčky, čímž se pokusí vynutit jakékoliv pokusy o získání prostředků finanční smlouvy, aby došel gas. Finanční smlouva však může ve zprávě stanovit limit gas, aby se tomuto problému předešlo.

Alternativou k Turingovské úplnosti je Turingovská neúplnost., kde JUMP a JUMPI neexistují, a v call stacku může v daném okamžiku existovat pouze jedna kopie každé smlouvy. U tohoto systému není popsaný systém poplatků a nejistoty ohledně účinnosti našeho řešení nemusí být nutné, protože náklady na provedení smlouvy by byly výše omezeny jeho velikostí. Turingovská neúplnost navíc není ani tak velkým omezením; ze všech smluvních příkladů, které jsme vytvořili interně, zatím pouze jeden vyžadoval smyčku a dokonce i tuto smyčku bylo možné odstranit provedením 26ti opakování jednorázového kódu. Vzhledem k závažným důsledkům Turingovské úplnosti a omezeným výhodám, proč prostě nepoužívat Turingovskou neúplnost? Ve skutečnosti však ani ta není zdaleka čistým řešením problému. Chcete-li zjistit proč, zvažte následující smlouvy:

```
C0: call(C1); call(C1);
C1: call(C2); call(C2);
C2: call(C3); call(C3);
...
C49: call(C50); call(C50);
C50: (run one step of a program and record the change in storage)
```

Nyní odešleme transakci do A. Tudíž máme v 51 transakcích smlouvu, která trvá 250 výpočetních kroků. Těžaři by se mohli pokusit detekovat takové logické bomby předem tím, že u každé smlouvy udrží hodnotu určující maximální počet výpočetních kroků, které může podniknout, a vypočítá to pro smlouvy rekurzivně volající jiné smlouvy, ale to by vyžadovalo, aby těžaři zakázali smlouvy, které vytvářejí další smlouvy (protože vytvoření a provedení všech výše uvedených smluv lze snadno sloučit do jediné). Dalším problematickým bodem je, že pole adresy zprávy je proměnná, takže obecně nemusí být ani možné předem zjistit, které další smlouvy daná smlouva následně zavolá. Celkově tedy máme překvapivý závěr: Turingovská úplnost je snadno ovladatelná a její nedostatek je obtížně zvládnutelný, pokud neexistují přesně stejné kontroly. Proč tedy nenechat protokol pracovat s Turingovskou úplností?

Měna a emise

Síť Ethereum zahrnuje vlastní vestavěnou měnu Ether, která slouží dvojímu účelu: poskytuje primární vrstvu likvidity, která umožňuje efektivní výměnu mezi různými typy digitálních aktiv, a co více, poskytuje mechanismus pro placení transakčních poplatků. Pro pohodlí a vyhnutí se budoucím hádkám (viz aktuální debata mBTC/uBTC/Satoshi v Bitcoinech) bude měna značena takto:

1: wei

10^{12}: szabo

10^{15}: finney

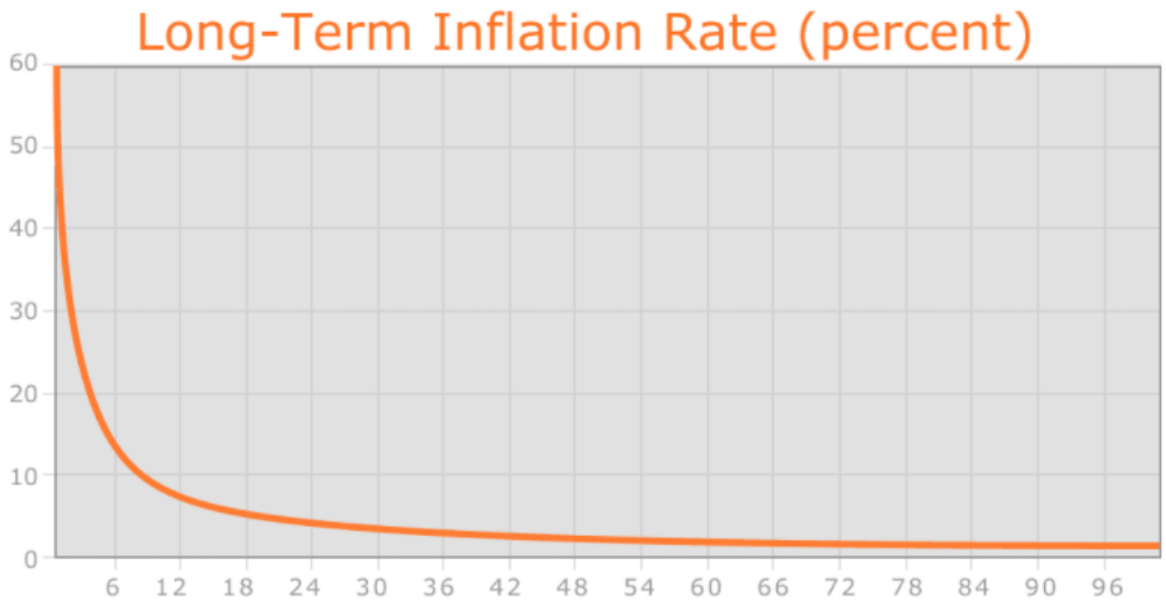
10^{18}: ether

Mělo by to být bráno jako rozšířená verze pojmů jako „dolar a cent“, nebo „Bitcoin a Satoshi“.
V blízké budoucnosti očekáváme, že Ether bude používán pro běžné transakce, a „szabo“ a „wei“ pro technické diskuze o poplatcích a implementaci protokolu; zbývající označení mohou být užitečná později, a neměla by být v tomto okamžiku rozšířena mezi klienty.

Emisní model bude následující:

- Ether bude vydán do prodeje za cenu 1000-2000 ETH/BTC, což je mechanismus určený k financování společnosti Ethereum a placení za vývoj, který s úspěchem využívají i jiné platformy jako Mastercoin a NXT. Lidé, kteří nakoupí dříve budou těžit z větších slev. BTC získané z prodeje budou použity výhradně k výplatě mezd a odměn vývojářům, a budou investovány do různých neziskových projektů v ekosystému Ethereum.
- 0,099x celková prodaná částka bude přidělena organizaci za účele kompenzace prvních investorů, a platby výdajů denominovaných v ETH před blokem genesis.
- 0,99x celková prodaná částka bude udržována jako dlouhodobá rezerva
- 0,26x celková prodaná částka bude přidělena těžařům každý rok

Dlouhodobá míra růstu nabídky (%)



Navzdory lineárnímu vydávání měn, stejně jako v případě Bitcoinu, se tempo růstu nabídky v průběhu času přiklání k nule

Dvěma hlavními možnostmi ve výše uvedeném modelu jsou (1) existence a velikost nadačního fondu a (2) existence trvale rostoucí lineární nabídky, na rozdíl od omezené nabídky, jako v Bitcoinu. Odůvodnění nadačního fondu je následující. Pokud by nadační fond neexistoval a lineární emise by se snížila na 0,217x aby byla zajištěna stejná míra inflace, pak by bylo celkové množství ETH o 16,5% nižší, a každý ETH by tak bylo o 19,8% cennější. V rovnovážném stavu by bylo při prodeji zakoupeno o 19,8% ETH více, a tak by každá jednotka byla opět stejně cenná jako dříve. Organizace by také měla 1,198x více BTC, které by se rozdělily do dvou částí: původní BTC a dodatečné 0,198x. Proto je tato situace přesně ekvivalentní s dotací, avšak s jedním důležitým rozdílem: organizace drží pouze BTC, a proto není motivována podporovat hodnotu ETH.

Trvalý lineární model růstu nabídky snižuje riziko toho, co někteří považují za nadměrnou koncentraci bohatství v Bitcoinech, a dává jednotlivcům žijícím v současnosti a budoucnosti spravedlivou šanci na získání jednotek ETH, přičemž si zároveň zachovává silnou motivaci k získání a držení ETH, protože „míra růstu nabídky“ jako procento má v průběhu času stále tendenci směřovat k nule. Rovněž předpokládáme, že z důvodu občasné ztráty mincí kvůli neopatrnosti, smrti atd., lze tuto ztrátu modelovat jako procento z celkové nabídky za rok, která se nakonec stabilizuje na hodnotě rovnající se roční emisi vydělené ztrátovostí (např. při ztrátové míře 1%, jakmile zásoba dosáhne 26x, bude vytěženo 0,26x a 0,26x ztraceno každý rok, což vytvoří rovnováhu).

Je pravděpodobné, že v budoucnosti Ethereum pravděpodobně přejde z důvodu bezpečnosti na model proof of stake, čímž sníží emisní požadavek o hodnotu mezi nulou a 0,05x ročně. V případě, že organizace Ethereum ztratí financování nebo z jakéhokoliv jiného důvodu zanikne, necháváme otevřenou „sociální smlouvu“: kdokoliv má právo vytvořit budoucí kandidátskou verzi Etherea, přičemž jedinou podmínkou je, že množství Etheru musí být nanejvýš 60 102 216 * (1,198 +0,26*n), kde n je počet let po genezi. Tvůrci mohou volně prodávat nebo jinak přiřazovat část, nebo všechny rozdíly mezi rozšířením nabídky poháněným PoS a maximálním rozšířením této nabídky. Upgrady kandidátů, které nebudou v souladu se sociální smlouvou, budou moct být oprávněně rozdány kompatibilním verzím.

Centralizace těžby

Algoritmus pro těžbu BTC funguje tak, že těžaři počítají SHA256 na mírně upravených verzích záhlaví bloku znovu a znovu, dokud jeden uzel neposkytne verzi, jejíž hash je menší než cíl (aktuálně kolem 2192). Tento algoritmus je však zranitelný vůči dvěma formám centralizace. Nejprve v těžebním systému začaly dominovat ASIC (aktivně specifické integrované obvody), počítačové komponenty určené přímo pro těžbu, a tudíž tisíckrát efektivnější. To znamená, že těžba BTC už není vysoce decentralizovaným rovnostářským úsilím, k jehož účasti je zapotřebí milionů dolarů. Zadruhé, většina Bitcoinových těžařů ve skutečnosti neprovádí lokální ověření bloku; místo toho spoléhají na centralizovaný fond těžby, který poskytuje záhlaví bloků. Tento problém je prokazatelně horší: v době psaní tohoto dokumentu tři hlavní těžební fondy nepřímo kontrolují zhruba 50% zpracovatelského výkonu Bitcoinové sítě, i když je tento problém zmírněn skutečností, že těžaři mohou přejít na jiné těžební fondy, pokud by došlo k pokusu o 51% útok ze strany fondu.

Současným záměrem Etherea je použít těžební algoritmus, kde jsou těžaři povinni načíst náhodná stavová data, vypočítat některé náhodně vybrané transakce posledních n bloků v blockchainu, a vrátit zpět hash výsledku. To disponuje dvěma důležitými výhodami. Zaprvé, smlouvy na Ethereu mohou zahrnovat jakýkoliv druh výpočtu, takže ASIC Ethereum by byl v podstatě ASIC pro obecný výpočet – tj. lepší CPU. Zadruhé, těžba vyžaduje přístup k celému blockchainu, což nutí těžaře ukládat celý blockchain, a přinejmenším jsou schopni ověřit každou transakci. To odstraňuje potřebu centralizovaných těžebních fondů; i když těžební fondy mohou stále hrát legitimní roli při vyrovnávání náhodnosti distribuce odměn, tuto funkci mohou stejně efektivně obsluhovat i peer to peer fondy bez centrální kontroly.

Tento model je nevyzkoušený, a mohou nastat potíže při vyhýbání se určitým chytrým optimalizacím při provádění smlouvy jako algoritmu těžby. Jedním ze zajímavých rysů tohoto algoritmu je však to, že umožňuje komukoliv „otrávit studnu“ zavedením velkého počtu smluv do blockchainu speciálně navrženého pro stymie určitých ASIC. Existují ekonomické pobídky pro výrobce ASIC, aby tento trik použili k útoku. Řešení, které vyvíjíme, je tedy v konečném důsledku spíše adaptivním ekonomickým lidským řešením, než čistě technickým řešením.

Škálovatelnost

Velkou obavou je otázka škálovatelnosti Etherea. Stejně jako Bitcoin, i Ethereum disponuje chybou, ve které musí každou transakci zpracovat každý uzel v síti. U Bitcoinu se velikost jeho blockhainu v současné době pohybuje okolo 15 GB, a roste přibližně o 1 MB každou hodinu. Pokud by Bitcoinová síť zpracovávala 2 000 Visa transakcí za sekundu, velikost blockchainu by o 1 MB vzrostla za tři sekundy (1 GB za hodinu, 8 TB za rok). Ethereum by pravděpodobně prokazovalo podobný model, který by se zhoršoval se skutečností, že na blockchainu Etherea bude mnoho aplikací namísto pouhé měny, jako je tomu u Bitcoinu. Zmírňuje to však skutečnost, že plné uzly Etherea ukládají pouze stav, namísto celé historie blockchainu.

Problém s tak obrovským blockchainem je riziko centralizace. Pokud se velikost blockchainu zvýší řekněme na 100 TB, pak by plné uzly provozovaly pouze velmi malý počet velkých byznysů, přičemž všichni běžní uživatelé by používali lehké uzly SPV. V takové situaci vyvstává potenciální obava, že by se plné uzly mohly spojit dohromady a souhlasit s tím, že budou podvádět nějakým výnosným způsobem (např. změna blokové odměny, připsání BTC). Lehké uzly by to nemohly včas zjistit. Samozřejmě by pravděpodobně existoval alespoň jeden poctivý plný uzel, a informace o podvodu by po několika hodinách unikly prostřednictvím kanálů, jako je například Reddit, avšak v tom okamžiku už by bylo pozdě. Oprava problému by závisela pouze na uživatelích. V případě Bitcoinu je to v současné době problém, ale existuje úprava blockchainu navržená Peterem Toddem, která tento problém dokáže zmírnit.

V blízké budoucnosti použije Ethereum k řešení tohoto problému dvě další strategie. Zaprvé, kvůli těžařským algoritmům bude každý z těžařů přinucen být plným uzlem, čímž se vytvoří dolní hranice počtu plných uzlů. Zadruhé, po zpracování každé transakce zahrneme do blockchainu kořen stromu přechodného stavu. I když je ověření bloku centralizované a existuje alespoň jeden poctivý plný uzel, lze problém s centralizací obejít pomocí ověřovacího protokolu. Pokud těžař publikuje neplatný blok, tento blok je buď špatně naformátovaný, nebo je stav S [n] nesprávný. Protože je známo, že stav S[0] je správný, musí existovat nějaký první stav S[i], který je nesprávný (správný je S[i-1]). Ověřovací uzel poskytne index spolu s „důkazem neplatnosti“ sestávajícím z podmnožiny uzlů stromu Patricia, které potřebují zpracovat APPLY (S [i-1], TX [i] - > S [i]). Uzly by byly schopny použít tyto uzly Patricia ke spuštění této části výpočtu a následně zjistit, že generovaný stav S[i] neodpovídá poskytnutému stavu.

Další, sofistikovanější útok by zahrnoval škodlivé těžaře publikující neúplné bloky (úplné informace proto neexistují), a proto by nebylo možné určit, zda jsou bloky platné. Řešením je protokol challenge-response: ověřovací uzly vysílají „challenge“ ve formě indexů cílových transakcí, a po přijetí bude lehký uzel považovat blok za nedůvěryhodný, dokud jiný uzel, ať už těžař, nebo jiný ověřovatel neposkytne podmnožinu uzlů Patricia jako důkaz platnosti.

Závěr

Protokol Ethereum byl původně koncipován jako upgradovaná verze kryptoměny poskytující pokročilé funkce, jako jsou on-blockchain escrow, limity pro výběr, finanční smlouvy, trhy s hazardními hrami a podobně, prostřednictvím vysoce zobecněného programovacího jazyka. Protokol Ethereum by nepodporoval žádnou z aplikací přímo, avšak existence kompletního programovacího jazyka Turing znamená, že teoreticky lze vytvořit libovolné smlouvy pro jakýkoliv typ transakce nebo aplikace. Co je však na Ethereu ještě zajímavější je fakt, že se pohybuje daleko za hranicí pouhé měny. Protokoly týkající se decentralizovaného ukládání souborů, decentralizovaného výpočtu a trhů s decentralizovanou predikcí, mají potenciál podstatně zvýšit efektivitu výpočetního průmyslu, a poskytnout masivní podporu dalším protokolům typu peer to peer přidáním ekonomické vstvy. Konečně zde také mohou existovat aplikace, které nemají s penězi nic společného.

Koncept funkce libovolného stavového přechodu implementovaný protokolem Ethereum poskytuje platformu s jedinečným potenciálem; místo toho, aby byl uzavřeným jednoúčelovým protokolem určeným pro konkrétní řadu aplikací v oblasti ukládání dat, hazardních her nebo financí, je Ethereum otevřeno záměrně a věříme, že je vhodné pro to, aby sloužilo jako základ pro obrovský počet finančních i nefinančních protokolů v nadcházejících letech.