Blockkedjan, Smarta kontrakt och Ethereum

Casper von Pfaler

Kandidatavhandling i datavetenskap

40548

Handledare: Mikhail Barash

Åbo Akademi

2018

1. Inledning

Varje dag skapas det mera data i världen. Den största andelen samlas och lagras av stora korporationer som t.ex. det sociala nätverket Facebook och sökmotorn Google. Samtidigt blir data allt mer värdefullt. Framsteg inom artificiell intelligens åstadkommer hela tiden nya sätt att utnyttja data. I dag är det alltså de stora företagen som vinner mest på utvecklingen. För att alla i framtiden skall kunna dra nytta av data krävs det innovativa lösningar. En av dessa lösningar, som kan komma att fungera som grunden för en mera decentraliserad framtid, är blockkedjan. Kortfattat kan en blockkedja beskrivas som en serie av block länkade till varandra i form av en kedja, med hjälp av kryptografiska bevis.

I ett system baserat på en blockkedja kan vi spara data på ett decentraliserat sätt. Ändringar i data på en blockkedja sker på basis av konsensus mellan deltagare. Blockkedjan förutsätter tillit till teknologin. Därför är det viktigt att förstå hur blockkedjan fungerar så att det faktiskt går att lita på den. Om blockkedjan möjliggör att förvara data på ett mera decentraliserat sätt, så är "smarta kontrakt" ett sätt att göra något med denna data. Ett smart kontrakt upprätthåller automatiskt kontraktets villkor och i samband med blockkedjan kan smarta kontrakt beskrivas som ett sätt att uttrycka regler för ändringar i data lagrat på kedjan. Samtidigt är det viktigt att förstå hur smarta kontrakt fungerar, så att vi fortsatt kan lita på ett system som består av en blockkedja och smarta kontrakt.

Kanske den mest grundläggande applikationen för en blockkedja är att fungera som ett system som definierar ägorätten till data på kedjan. Blockkedjan tillkom ursprungligen som en som grund för kryptovalutan Bitcoin, för att bestämma vem som äger vilka "bitcoins". Smarta kontrakt möjliggör mycket mer sofistikerade applikationer än att endast bestämma ägorätten till data. I princip går det med smarta kontrakt att bygga vilken som helst applikation på blockkedjan.

Syftet med den här essän är att ge läsaren en insikt i hur blockkedjan fungerar samt hur smarta kontrakt bygger på den. I avsnitt 2 beskrivs först blockkedjan och dess egenskaper. I följande avsnitt beskrivs sedan smarta kontrakt och Ethereum, en plattform för smarta kontrakt. Exempel på riktiga applikationer byggda med hjälp av en blockkedja och smarta kontrakt tas därefter upp i avsnitt 4. Bitcoin fungerar

genom essän som utgångspunkt för analysen av blockkedjan, och Ethereum som premiss för diskussionen gällande smarta kontrakt.

2. Blockkedjan

Det som i dag kallas för en blockkedja blev först introducerat i samband med Bitcoin [1], en decentraliserad digital valuta. Blockkedjan kan beskrivas som ett system för att åstadkomma konsensus i ett distribuerat nätverk. Detta kallas även för bysantinsk överenskommelse [2]. Det ursprungliga ändamålet för blockkedjan var att fungera som en öppen och distribuerad huvudbok, en lista över samtliga transaktioner på bitcoinnätverket. Med en transaktion avses att en användare sänder tillgångar åt en annan användare. Blockkedjan introducerades alltså som en del av s.k. *kryptovalutor*, och förknippas ännu starkt med dessa. Senare har dock blockkedjan också fått användning inom andra områden. Exempelvis finns det försök att motverka blodsdiamanter med hjälp av blockkedjan [3]. En annan applikation, som alla kan dra nytta av, är i form av ett distribuerat lagringssystem [4].

I det här avsnittet beskrivs först i stora drag hur en blockkedja fungerar. Efter det diskuteras hur konsensus uppnås på ett blockkedjenätverk, och några viktiga egenskaper som blockkedjan besitter tas upp. Till sist behandlas utmaningar och potentiella förbättringar som blockkedjan står inför.

2.1 Blockkedjans funktion

Utan en central auktoritet kan det vara svårt att åstadkomma konsensus. Som lösning på detta föreslår blockkedjan en struktur som använder kryptografiska bevis för att säkerställa transaktioner [1]. Kryptografi går ut på att säkra information. I samband med blockkedjan vill vi säkra integriteten för data lagrat på kedjan. Det är viktigt att det för varje transaktion kan påvisas vad den går ut på, vem som har sänt vad och till vem, samt när den inträffat. Blockkedjan åstadkommer ett system som uppfyller de ovannämnda kraven genom att decentraliserade noder tillsammans bygger upp en kedja av kryptografiskt länkade block. Blocken innehåller de enskilda transaktionerna och deras placering i kedjan beskriver när de inträffat.

För att bygga upp blockkedjan har varje nod som deltar en lokal kedja och jobbar på att lägga till ett nytt block i den. Detta diskuteras mera ingående i avsnitt 2.2. Om en nod lyckas lägga till ett block på sin lokala kedja, sänder den sedan blocket över

nätverket och om det nya blocket resulterar i en längre kedja än den lokala kedjan hos en mottagande nod, accepteras blocket av den noden. På så sätt förlängs den lokala kedjan hos noden som skapat ett nytt block, samt noder som mottar det nyligen skapade blocket. Den här processen gör att lokala kedjor förlängs med block som härstammar från olika noder. Det är viktigt att lägga märke till att en nod alltid arbetar på att förlänga den längsta möjliga lokala kedjan, vilket är en väsentlig aspekt för att blockkedjan skall vara säker [1].

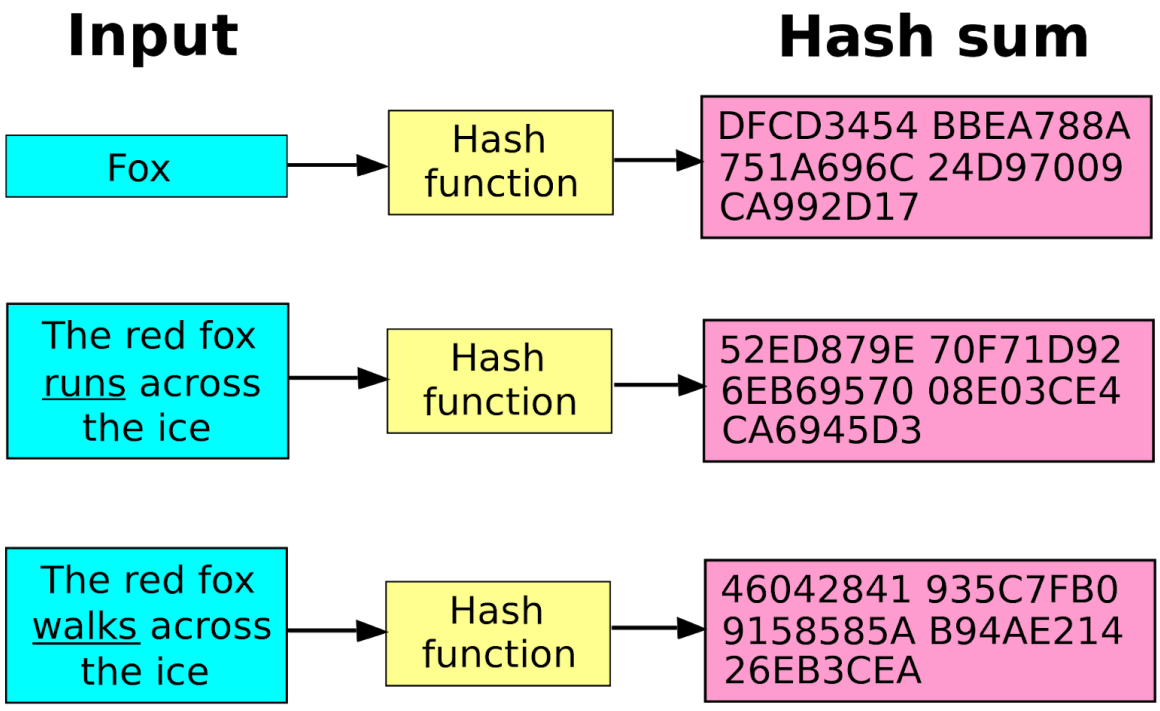
En transaktion har inkorporerats på blockkedjan då den förekommer i ett accepterat block. I praktiken skapar en nod på nätverket transaktionen, och lägger till den i blocket den försöker förlänga sin lokala kedja med. Transaktionen sänds även över nätverket till andra noder så att också de kan skapa block där transaktionen är med. Efter att en transaktion kommit med i ett block krävs det ännu att blocket accepteras av andra noder.

2.2 Hashvärden och konsensus på en blockkedja

I det här avsnittet diskuteras mekanismen som används för att åstadkomma konsensus på ett distribuerat nätverk. Problemet med konsensus kan illustreras med det klassiska exemplet på bysantinska generaler som försöker komma överens om de skall anfalla en stad [2]. Generalerna kan sinsemellan enbart kommunicera via meddelanden och beslutet görs på basis av en enkel majoritet. Vissa generaler kan dock vara förrädare. Därför behövs ett system som försäkrar att de lojala generalerna skapar konsensus sinsemellan. För att åstadkomma detta föreslår Bitcoin något som kallas “Proof of Work” (PoW) [1]. Enligt PoW måste en nod lösa ett kryptografiskt problem för att få skapa ett nytt block. Ett kryptografiskt problem är ett problem som används för autentisering. En korrekt lösning fungerar som autentisering. För problemet skall det gälla att den enda lönsamma taktiken är att utföra maximalt många beräkningar. Dessutom bör lösningen vara lätt att verifiera. Vidare är det bra om det går att reglera svårigheten på problemet, eftersom det då går att göra anpassningar beroende på hur mycket beräkningskraft som nätverket upptar.

Bitcoin föreslår att PoW skall gå ut på att leta efter ett “hashvärde” vilket har ett specifikt antal ledande nollor [1]. Ett hashvärde fås genom att konvertera ett digitalt

värde till en obegriplig form. Samma värde ger alltid upphov till samma hashvärde. En funktion som används för att konvertera normala värden till hashvärden kallas för en krypteringsfunktion. SHA-256 är krypteringsfunktionen som Bitcoin använder. Alla värden konverterade till hashvärden med SHA-256 är lika långa, 256 bits, vilket är en väsentlig egenskap för PoW. Om hashvärdena kunde vara hur korta som helst vore det inte möjligt att vara säker på om det går att öka på antalet nollor lösningen kräver. Att ändra på antalet ledande nollor som lösningen kräver är viktigt eftersom tiden som krävs för att hitta en lösning på PoW är exponentiellt relaterad till antalet ledande nollor [1]. Svårighetsgraden kan regleras genom att sänka och höja antalet ledande nollor lösningen kräver. Verifieringen av en lösning är enkel. Det är bara att se om en föreslagen lösning resulterar i rätt antal ledande nollor då den matas in i krypteringsfunktionen.

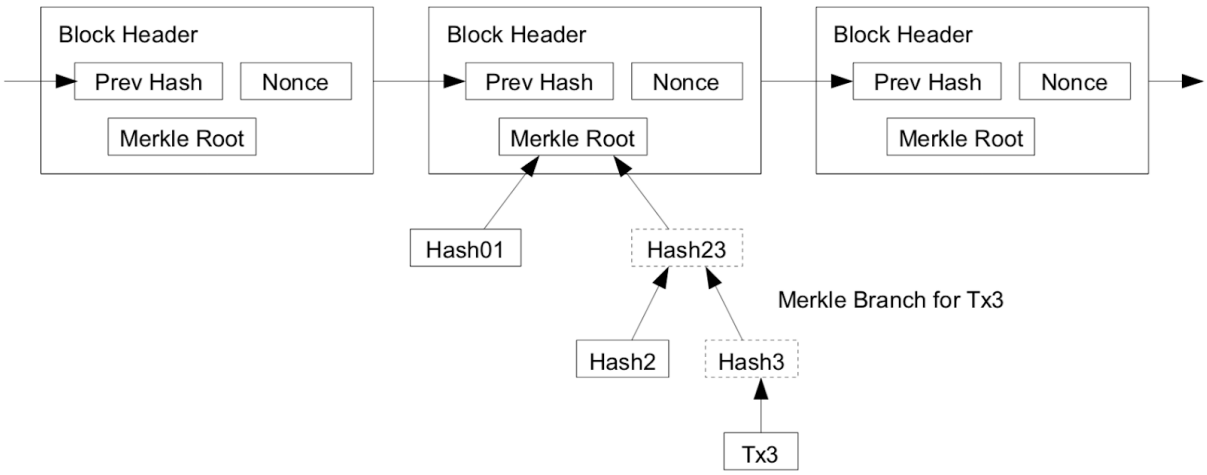


Figur 1: Exempel på värden konverterade till hashvärden (Källa: https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/Hash_function_long.svg/2000px-Hash_function_long.svg.png)

Utöver de ovannämnda egenskaperna bör det kryptografiska problemet inkorporera lösningen till motsvarande problem i det föregående blocket, samt transaktionerna

som skall finnas i det nya blocket. Det är på det här sättet som blocken bildar en kedja. Varje block har en referens till föregående block i form av lösningen till PoW för det blocket. I praktiken inkorporeras meddelanden i krypteringen med hjälp av Merkle-träd, som krypterar data på ett sådant sätt att det effektivt kan verifieras om data förändrats [5]. Merkle-träd är en datastruktur där egentliga värden endast lagras i trädets lövnoder. Andra noder håller värdet som fås genom att sammanslå de krypterade värdena av nodens barn. Om värdet som finns i en lövnod ändras kommer på så sätt också värdet för rotnoden att ändras vilket betyder att det genast från rotnoden kan upptäckas om värdet hos en lövnod ändrat. I figur 2 illustreras ett litet Merkle-träd.

För att lösa PoW för ett nytt block använder vi den motsvarande lösningen från det tidigare blocket samt nya transaktioner, representerade i Merkle-träd form. Målet är att komma till en korrekt lösning för PoW med dessa två värden. I problemet från det förra stycket skulle det här betyda att vi får rätt antal ledande nollor då vi konverterar kombinationen av de två värdena med SHA-256. För att det skall vara möjligt att göra flera gissningar lägger vi till en siffra i kombinationen. Siffran kallas för “nonce” och inkrementeras alltid efter ett misslyckat försök, tills en korrekt lösning hittas.



Figur 2: Uppbyggnaden av en blockkedja. (Källa: http://bitcoin.org/bitcoin.pdf)

PoW är numera inte det enda protokollet som används för att skapa konsensus på en blockkedja. Det mest framträdande alternativet är “Proof of Stake” (PoS) [6], som

använder delaktighet i stället för arbete då det kommer till de kryptografiska bevisen. Den här uppsatsen kommer att begränsa sig till PoW som mekanismen för att skapa konsensus.

Det krävs ett visst incitament för att noder som deltar i en blockkedja skall vara uppmuntrade att utföra arbetet som PoW förutsätter. Upplägget som Bitcoin lägger fram för hur detta skall ske baserar sig på belöningar.. På bitcoinnätverket finns det ett visst antal mynt i cirkulation vid varje given tidpunkt. Dessa mynt antas ha ett ekonomiskt värde, och därför förmodas det att deltagare i nätverket strävar efter att få flera mynt. På basis av dessa antaganden delar bitcoinnätverket ut nya mynt åt den noden som hittar PoW för nästa block. Utöver detta kräver transaktionerna också en betalning åt noderna som arbetar för att få med transaktionen i nätverket. Ett liknande system med mynt används i praktiken på alla blockkedjor.

2.3 Viktiga egenskaper hos en blockkedja

För att få en djupare förståelse om hur och varför blockkedjan fungerar, d.v.s. hur den uppnår konsensus på ett distribuerat och säkert sätt, är det värt att begrunda egenskaperna “common prefix” (CP), “chain growth” (CG) och “chain quality” (CQ) introducerade i [7]. De baserar sig på parametrar som t.ex. det totala antalet noder, och antalet oärliga noder av dessa, som deltar i ett blockkedjenätverk. Ärliga noder försöker inte förfalska meddelanden, eller på något sätt förhindra att blockkedjan fungerar som den skall. Oärliga noder har däremot som mål, att attackera kedjan på olika sätt för att skapa en fördel åt sig själv gentemot de ärliga noderna. Exemplevis kan oärliga noder inom kryptovalutor försöka använda samma tillgångar flera gånger. Analysen i [7] är uppdelad i omgångar och en annan viktig parameter som det är värt att nämna är sannolikheten att åtminstone en ärlig nod åstadkommer PoW under en given omgång.

Med det ovannämnda i åtanke kan vi informellt beskriva Chain Growth, Common Prefix och Chain Quality. ***Chain Growth*** betyder att den lokala kedjan hos en ärlig nod växer i åtminstone samma takt som frekvensen på lyckade omgångar. Med en lyckad omgång avses en omgång där åtminstone en ärlig nod lyckades skapa ett nytt

block och sända det över nätverket. Ju mera lyckade omgångar vi har desto snabbare kommer ärliga transaktioner att komma med i blockkedjan.

Common Prefix säger att om vi i en blockkedja har en majoritet av ärliga noder, så kommer de lokala kedjorna för dessa noder att dela ett stort gemensamt prefix. Vidare kan vi säga att då en ärlig nod eliminerar *k* block från sin lokala kedja, så ökar chansen att den resulterande kedjan är ett prefix i en annan ärlig nods lokala kedja, exponentiellt i takt med att *k* stiger. Det gemensamma prefixet är den huvudsakliga egenskapen för att påvisa överenskommelse inom nätverket. Den viktiga insikten som **CP** kristalliserar är att en given transaktion blir mer och mer säker då allt fler block läggs till blocket där transaktionen först inkorporerats.

Till sist säger ***Chain Quality***, med samma antagande om en ärlig majoritet som **CP** förutsätter, att det garanterat finns block producerade av ärliga noder i de lokala kedjorna hos ärliga noder. Andelen block som åstadkommits av oärliga noder kan emellertid vara större än andelen beräkningskraft som de besitter. Det är i sammanhang med sannolikheten, att åtminstone en ärlig nod hittar PoW under en omgång, som **CQ** får sin mening. Då sannolikheten blir mindre får vi bättre **CQ** [7]. Vi kan alltså säga att en liten sannolikhet för att åtminstone en ärlig nod hittar PoW resulterar i att transaktioner som härstammar från ärliga noder har en större chans att komma med i blockkedjan.

2.4 Utmaningar och förbättringar

Den ursprungliga utmaningen, som Bitcoin strävade att lösa med hjälp av blockkedjan, är något som kallas för “double-spending” [1]. Med double-spending avses att man använder samma tillgångar i flera olika transaktioner. För att en digital valuta som Bitcoin skall vara pålitlig bör risken för double-spending minimeras. I det förra avsnittet antogs en ärlig majoritet av noder och i en sådan situation är risken för double-spending liten. Ett system som krävde en mindre andel ärliga noder skulle vara en eftertraktad förbättring.

Blockkedjan står också inför andra utmaningar än double-spending. En av de största utmaningarna är energianvändningen. För tillfället använder Bitcoin ungefär lika

mycket energi som Uzbekistan [8]. En möjlig lösning på detta problem nämndes också tidigare i form av PoS. Flera blockkedjor grundar sig redan i dag på PoS. Det är möjligt att vi i framtiden kommer att se PoW helt övergiven på grund av attraktivare alternativ.

En annan nämnvärd utmaning som blockkedjor står inför är antalet transaktioner de kan klara av. Bitcoin klarar av ungefär två transaktioner per sekund [11], vilket i praktiken betyder att det kan ta väldigt länge för en transaktion att komma med in i ett block. “Lightning Network” [12] erbjuder en möjlig lösning genom att låta transaktioner ske utanför blockkedjan på ett nätverk av s.k. *mikrobetalnings kanaler*. Det skulle emellertid vid behov vara möjligt att upprätthålla transaktioner via den egentliga blockkedjan.

Resten av denna uppsats kommer att vara ägnad åt smarta kontrakt, en av de viktigaste utvidgningarna av blockkedjan. Vad de är, hur de är implementerade och vad de möjliggör. Fokus kommer speciellt att ligga på Ethereum, en plattform för s.k. decentraliserade applikationer. Decentraliserade applikationer liknar andra applikationer som fungerar över nätet. Men de har ingen server-klient arkitektur [13], utan det finns endast klienter.

3. Smarta kontrakt och Ethereum

Idén med smarta kontrakt är att på ett formaliserat sätt uttrycka kontrakt i hård- och mjukvara [14]. Mera konkret kan vi tänka oss att ett smart kontrakt, med hjälp av kod, bestämmer vad som händer ifall kontraktet upprätthålls, samt följderna om kontraktet bryts. I [14] används ett exempel där äganderätten till bilar grundar sig på smarta kontrakt. I ett sådant system kan man tänka sig att du endast kan använda en bil ifall du med hjälp av de smarta kontrakten kan bevisa att det faktiskt är du som äger bilen. Mera sofistikerad funktionalitet kunde också vara inbyggt i kontrakten. Ifall du har ett lån för din bil och inte lyckas betala tillbaka det på utsatt tid kunde ett smart kontrakt automatiskt överföra bilens äganderätt till långivaren eller kanske bara hindra dig från att använda bilen tills du betalat en viss del av din skuld.

Smarta kontrakt har alltså inte vuxit fram ur blockkedjan, utan blockkedjan råkar vara ett system som det går att bygga säkra och decentraliserade smarta kontrakt på. Decentraliseringen är en viktig aspekt då den möjliggör att skapa kontrakt som fungerar utan mellanhänder.

För implementeringen av smarta kontrakt på blockkedjan kan det löna sig att tänka på en blockkedja som ett system av tillståndsförändringar [9]. Det här fungerar även för den enklaste typen av blockkedja som endast stöder transaktioner från en användare till en annan. En transaktion ger då upphov till ett nytt tillstånd i.o.m. att tillgångar har bytt ägare. Smarta kontrakt är ett sätt att åstadkomma mera sofistikerade tillståndsförändringar. Om vi programmatiskt kan uttrycka regler för när och varför tillståndsförändringar i en blockkedja skall ske, och göra det på ett säkert sätt, har vi ett robust system för smarta kontrakt.

Bitcoin, och dess blockkedja, stöder i princip smarta kontrakt. Som tidigare beskrivet behövs det egentligen bara en blockkedja som stöder transaktioner. I praktiken är det här vilken som helst blockkedja. Bitcoin har även stöd för ett skriptspråk, men det är väldigt begränsat. Viktiga egenskaper som saknas är exempelvis loopar och internt tillstånd för skript [9]. På grund av detta är det svårt och opraktiskt att använda Bitcoin som ett bakgrundssystem för smarta kontrakt.

För att det skall vara praktiskt att skriva smarta kontrakt på en blockkedja krävs alltså något annat än Bitcoin. Här stiger Ethereum in i bilden. Ethereums blockkedja fungerar i stora drag på samma sätt som Bitcoins, med den väsentliga skillnaden att Ethereum har ett komplett programmeringsspråk inbyggt i sin blockkedja. Sofistikerade smarta kontrakt kan på så sätt bli skrivna med det här programmeringsspråket på Ethereums blockkedja. De sofistikerade kontrakten möjliggör byggandet av ännu mera sofistikerade applikationer. Målet är att Ethereum skall fungera som en plattform för s.k. decentraliserade applikationer [9].

3.1 Tillstånd i Ethereum

Tidigare nämndes det att varje blockkedja har någon form av mynt som antas ha ekonomiskt värde inbyggt. Mynten på Ethereums blockkedja kallas för Ether. En stor del av tillståndet på ethereums blockkedja beskriver fördelningen av Ether. Det övriga tillståndet på Ethereums blockkedja utgörs av data sparat i smarta kontrakt.

På Ethereums blockkedja sägs tillståndet vara uppbyggt av "konton" [9], av vilka det finns två olika slag, externt ägda konton samt kontraktskonton. De förstnämnda kontrolleras privat, d.v.s. av personer eller organisationer. Det är i ett sådant konto du, som deltagare i nätverket, har dina tillgångar (Ether) i. Tillgångar kan även finnas på kontraktskonton.

Kontraktskonton skiljer sig från externt ägda konton i och med att de kontrolleras av kod. Då ett kontraktskonto mottar en transaktion reagerar det alltid med att köra sin kod. I praktiken är det alltså kontraktets kod som äger kontot och tillgångarna som kontot besitter. Ett kontraktskonto kan också lagra data i sitt kontrakt. I princip kan ett kontrakt innehålla data för att beskriva vad som helst. Den här egenskapen att ha vilken som helst data lagrad på blockkedjan är viktig för att Ethereum skall kunna fungera som en plattform för decentraliserade applikationer, vilka alla kan kräva data av olika slag.

3.2 Transaktioner på Ethereums blockkedja

Transaktioner på Ethereum liknar transaktioner på andra blockkedjor t.ex. Bitcoin. De bör innehålla en mottagare, sändarens signatur samt mängden tillgångar som transaktionen omfattar, vilka alla är normala saker att inkludera i transaktioner på blockkedjor. Utöver det ovannämnda kräver Ethereum även att transaktioner innehåller ytterligare information i form av STARTGAS och GASPRICE [9], vilka är tal som tillsammans bestämmer hur många beräkningar ett kontrakt får utföra. STARTGAS beskriver hur många beräkningar ett kontrakt som mottar transaktionen är berättigad att göra, och GASPRICE beskriver hur mycket ether sändaren är beredd att betala för en beräkning. Dessa krävs eftersom det kostar att utföra beräkningar på Ethereum nätverket. På grund av att man måste specificera betalningen redan i transaktionen kan man även motverka s.k. “denial of service” attacker [9].

Ethereum stöder också något som kallas meddelanden. Dessa är egentligen samma sak som transaktioner med den skillnaden att de är sända från kontraktskonton i stället för privata konton. Genom att på så sätt sända meddelanden kan flera kontrakt fungera tillsammans. En viktig detalj är att ett meddelande kan ge upphov till beräkningar i ett annat kontrakt som mottar meddelandet. Dessa beräkningar tas med i den totala mängden tillåtna beräkningar, som bestäms av STARTGAS och GASPRICE i transaktionen som startade exekveringen av det första kontraktet.

Utöver data som transaktioner och meddelanden kräver för att vara valida kan de innehålla extra data i valfri form. Denna data kan sedan användas i exekveringen av de smarta kontrakten. Den här möjligheten att skicka data, utöver det som en transaktion kräver, är en viktig detalj för att möjliggöra komplexa smarta kontrakt såväl som decentraliserade applikationer.

3.3 Ethereums tillståndsförändringar

Nu när vi sett hur transaktioner på Ethereum-nätverket fungerar, samt att nätverkets tillstånd är uppbyggt av s.k. konton, kan vi beskriva hur tillståndsförändringar sker [9]. Smarta kontrakt är en viktig del av tillståndsförändringar. Kontrakten exekveras i respons till transaktioner, som i sig själv redan är tillståndsförändringar, och kontrakt kan också ge upphov till vidare tillståndsförändringar.

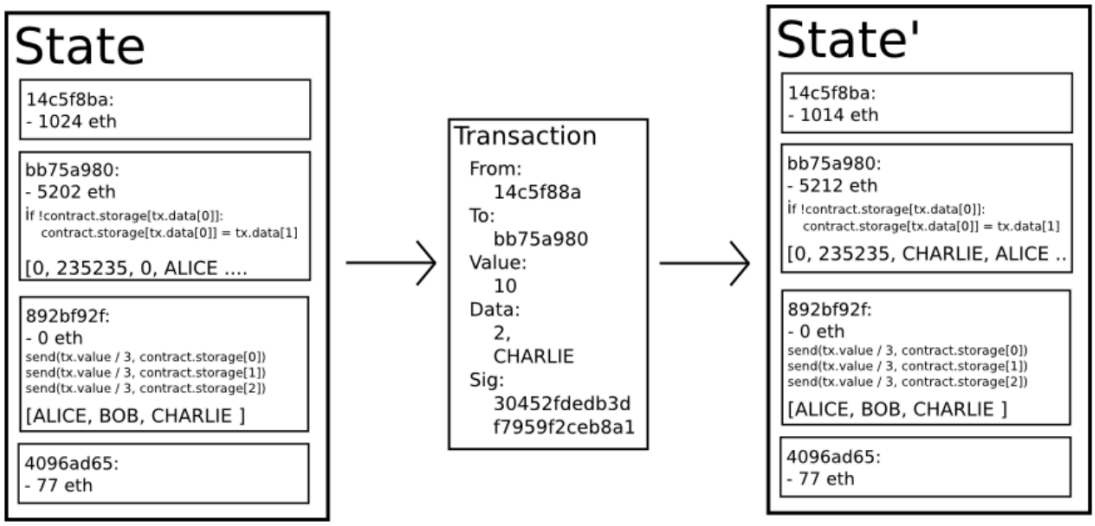
En tillståndsförändring härstammar alltså alltid från en transaktion, och det första som händer då en transaktion sänds är att transaktionen verifieras. Om transaktionen inte är korrekt formad uppstår ett fel och processen avbryts. I så fall sker egentligen inte transaktionen och därför heller ingen tillståndsförändring.

Efter verifieringen beräknas transaktionens kostnad enligt STARTGAS gånger GASPRICE. Det här är hur mycket GAS (specialmått på tillgångar) transaktionen tillåter ett kontrakt att använda. Tillgångar som motsvarar summan subtraheras från sändarens konto. I det här skedet måste sändaren naturligtvis ha tillräckligt med tillgångar. Om så inte är fallet uppstår igen ett fel och processen avbryts. Det här leder till samma resultat som om verifieringen hade misslyckats. Om tillgångarna är tillgängliga ökas sändarens “nonce” värde, som används för att se till att transaktioner sker endast en gång. Transaktionen skickas sedan till den angivna mottagaradressen.

Då en transaktion mottas används genast en del av de tillåtna beräkningarna som betalning för transaktionens datamängd. Om transaktionen mottas av ett privat konto, så är tillståndsförändringen klar i och med det. I det andra fallet att transaktionen mottagits av ett kontraktskonto, exekveras kontraktets kod.

Då koden på ett kontraktskonto exekveras är det någon som står för beräkningen. Vi kallar dem som utför beräkningar för “miners”. Transaktionskostnader erhålls av miners som belöning för beräkning. Det är under exekveringen som möjliga vidare tillståndsförändringar sker. Ett kontrakt kan här modifiera sitt eget tillstånd och det modifierade tillståndet blir sparat på blockkedjan.

Om GAS tar slut under exekveringen av ett kontrakt så vidhåller minern betalning för den GAS som använts, men alla andra tillståndsförändringar återtas. Ifall koden exekveras lyckat sänds tillgångar motsvarande den återstående andelen GAS tillbaka åt sändaren och minern behåller resten. Ifall exekveringen av ett kontrakt påbörjas får vi alltid en tillståndsförändring, i form av att minern blir belönad på basis av utförda beräkningar. Tillståndsförändringarna sparas i blockkedjan i samma stil som transaktioner.



Figur 3: Tillståndsförändring i Ethereum. (Källa: https://github.com/ethereum/wiki/wiki/White-Paper)

Ethereum fungerar alltså som en plattform för smarta kontrakt genom en kombination av blockkedjan och ett globalt tillstånd som kan modifieras med hjälp av ett turingkomplett programmeringsspråk. Ett språk är turingkomplett om man kan skriva vilket som helst program i det.

3.4 Solidity

Programmeringsspråket som kan användas för att skapa smarta kontrakt på Ethereums blockkedja kallas *Solidity*. Språket beskrivs som "kontraktorienterat" [15], och är alltså konstruerat för det specifika ändamålet att skriva smarta kontrakt. Semantiskt tar Solidity efter existerande programmeringsspråk, speciellt "objektorienterade" språk. C++, Python och JavaScript nämns specifikt som källor för inspiration [15].

All soliditykod skrivs i form av kontrakt, vilka ungefär motsvarar klasser bekanta från objektorienterade språk. En ännu bättre jämförelse är kanske mellan kontrakt och singletons [16] d.v.s klasser, som det bara kan finnas en instans av. Kontrakt i Solidity kan innehålla tillståndsvariabler och funktioner. Det är tillståndsvariablerna i soliditykontrakt tillsammans med fördelningen av Ether som utgör tillståndet för Ethereums blockkedja. Funktioner används för att ändra på tillståndsvariabler.

```
pragma solidity ^0.4.0;

contract Bounty {

    uint256 reward;
    address creator;

    function Bounty() payable public {
        reward = msg.value;
        creator = msg.sender;
    }

    function award (address awardee) payable public {
        if (msg.sender == creator && awardee != creator) {
            awardee.transfer(reward);
        }
    }
}
```

Figur 4: Exempel på ett smart kontrakt skrivet i solidity.

Figur 4 visar ett väldigt enkelt smart kontrakt skrivet i Solidity. Med kontraktet kan man skapa en form av belöning. Skaparen kan tilldela belöningen åt något annat konto men kan inte återta belöningen åt sig själv. I kontraktet finns tillståndsvariablerna “reward” och “creator”, funktionerna “award” och “Bounty”. “Bounty” är speciell på grund av att den fungerar som konstruktor till kontraktet, metoden som åberopas då kontraktet initieras.

Exemplet i figur 4 är väldigt enkelt och illustrerar i stort sett endast vilka delar som ett kontrakt består av. En mera användbart variant av ett “Bounty”-kontrakt skulle låta skaparen definiera ett problem och sedan dela ut belöningen automatiskt åt den som löser problemet.

4. Applikationer baserade på Ethereum

Det finns flera orsaker att bygga en applikation med en plattform typ Ethereum. Det är kanske viktigt att applikationen aldrig är offline, eller så skall ingen ha ensam kontroll över applikationens data. I [9] identifieras tre olika kategorier av applikationer som går att bygga på blockkedjan; finansiella, semi-finansiella samt decentraliserade. Man kan tyda det här som att blockkedjan och smarta kontrakt kan fungera som basis för nästan vilken som helst applikation. Applikationer som är byggda på en blockkedja kallas för decentraliserade applikationer och står i kontrast till andra webbaserade applikationer. Exempel på applikationer som inte är byggda på en blockkedja och förlitar sig på en central auktoritet är sociala nätverk som Facebook och Reddit.

På Ethereum har det redan byggts applikationer av flera olika slag, exempelvis CryptoKitties [17] vilket är ett samlarspel. I de följande avsnitten presenteras ett par av dessa applikationer. Syftet är att konkretisera vad man kan bygga på en blockkedja och att illustrera vilka fördelar det finns med att basera en applikation på smarta kontrakt.

4.1 CryptoKitties

Det redan tidigare nämnda samlarspelet CryptoKitties är ett roligt exempel på en decentraliserad applikation. Det är fråga om att föda upp och samla digitala katter. Varje katt är unik och vem som äger vilken katt finns sparat på Ethereums blockkedja. En digital katt kan säljas och köpas, precis som med andra samlarobjekt. Dessutom kan det genom avel skapas helt nya katter.

Det finns flera fördelar med att implementera en applikation som CryptoKitties på Ethereum. För det första är Ethereums decentraliserade arkitektur attraktiv. Utan en central auktoritet har varje kattägare själv direkt åtkomst till sina katter. På så sätt behöver ingen lita på någon annan för att ens katter skall vara i säkert förvar, det enda som krävs är att lita på teknologin. Att Ethereum i praktiken har en valuta i form av Ether inbyggd är också något CryptoKitties kan dra nytta av. Inköp och försäljning av katter kan göras med Ether i stället för mera klassiska valutor.

4.2 Gnosis

Ethereum är emellertid inte bara en plattform för spel. Gnosis [18] är ett bra exempel på detta. Det är fråga om en plattform för prediktionsmarknader. I en prediktionsmarknad försöker deltagare förutspå händelser och blir belönade om de har rätt. På Gnosis marknad kan du köpa andelar i ett utfall av en händelse. Du blir sedan belönad på basis av hur många andelar du innehar om utfallet du investerat i är korrekt. Då ett stort antal människor deltar i en marknad av den här typen kan man bättre förutsäga sannolikheten för utfall i framtida händelser. Det går sedan att använda dessa sannolikheter för att göra beslut på förhand.

I [18] argumenteras det att en opartisk platform är väsentlig för att en prediktionsmarknad skall fungera rättvist. Utan en sådan plattform har stora spelare betydligt mera inflytande än små aktörer. Det här kan ses som den huvudsakliga orsaken till att Ethereum är en bra plattform att bygga en prediktionsmarknad på. Med smarta kontrakt är det möjligt att bygga genomskinliga regler för när och hur deltagare på marknaden blir belönade. Ethereum säkerställer även att det inte är en enda entitet som äger data genererat på marknaden. Det finns tillgänglig för alla på den öppna blockkedjan.

5. Avslutning

En blockkedja kan uppfattas som en serie av block länkade till varandra i form av en kedja, med hjälp av kryptografiska bevis. Blockkedjan fungerar som en öppen och distribuerad huvudbok. Deltagare i en blockkedja kan utföra transaktioner säkert utan en centraliserad auktoritet d.v.s decentraliserat. Transaktionerna sparas på blockkedjan. För att ett sådant system skall fungera krävs det att deltagare kan lita på teknologin. Genom att förstå hur en blockkedja fungerar, d.v.s. hur transaktioner utförs och lagras, är det möjligt att stärka tilliten till teknologin. Blocken innehåller alla transaktioner som skett. De kryptografiska bevisen garanterar transaktioners integritet och skapas med hjälp av ett konsensusprotokoll (PoW), som krävs för att blockkedjan skall vara decentraliserad. En mera formell bild av blockkedjan fås genom att begrunda dess grundläggande egenskaper: Common Prefix, Chain Quality och Chain Growth. De utgör den matematiska grunden som kan användas för att beskriva hur blockkedjan fungerar. Egenskaperna hjälper också till med att förstå under vilka förhållanden en blockkedja är pålitlig. För att få en djupare förståelse för dessa egenskaper refereras läsaren till [6].

Blockkedjan var först tänkt att användas i samband med digitala valutor, specifikt Bitcoin. Men det finns också andra områden som kan dra nytta av systemet som blockkedjan erbjuder. Exempelvis kan man tänka sig ett decentraliserat filsystem, vilket i dag existerar i form av FileCoin [4]. För att det skall vara praktiskt att använda blockkedjan för att skapa decentraliserade applikationer, i stil med FileCoin, krävs det att det går att göra mera än att blott och bart sända och motta transaktioner. Det som behövs är ett sätt att programmatiskt uttrycka regler för tillståndsförändringar på blockkedjan. Sådana regler kallas för smarta kontrakt.

Ethereum fungerar redan i dag som en plattform för att bygga decentraliserade applikationer. Att skriva en decentraliserad applikation på Ethereum är inte mycket svårare än att skriva en annan webbapplikation. Ethereums programmeringsspråk, Solidity, liknar i hög grad andra språk som exemplevis JavaScript. I praktiken gör Ethereum det möjligt för alla som kan grunderna till programmering att bygga decentraliserade applikationer. I framtiden antas allt flera applikationer välja Ethereum, eller någon annan blockkedja, som bakgrundssystem. Detta eftersom decentraliserade applikationer gör det möjligt att kringgå mellanhänder.

För att decentraliserade applikationer i framtiden skall vara lika tillgängliga som traditionella webbapplikationer finns det ännu en god bit att gå. Ett av problemen, som den här essän vill råda bot på, är en otillräcklig förståelse för hur blockkedjan fungerar. Andra problem inkluderar transaktionsfrekvens och energianvändning. Dessutom är värdesättningen av mynten som används för betalning på olika blockkedjor väldigt instabil vilket kan vara avskräckande. Genom att lösa de här problemen tas ett steg närmare ett mera öppet internet.

6. Referenser

[1] Nakamoto, S. (2009), 'Bitcoin: A Peer-to-Peer Electronic Cash System'. https://bitcoin.org/bitcoin.pdf

[2] Lamport, L.; Shostak, R. & Pease, M. (1982), 'The byzantine generals problem', *ACM Transactions on Programming Languages and Systems* **4** , 382--401. https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

[3] Volpicelli, G. (2017). How the blockchain is helping stop the spread of conflict diamonds. *Wired*. http://www.wired.co.uk/article/blockchain-conflict-diamonds-everledger

[4] Protocol Labs. (2017), Filecoin: A Decentralized storage network. https://filecoin.io/filecoin.pdf

[5] Merkle, R. C. (1988), A Digital Signature Based on a Conventional Encryption Function, in 'CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology' , Springer-Verlag, London, UK , pp. 369--378. https://people.eecs.berkeley.edu/~raluca/cs261-f15/readings/merkle.pdf

[6] Kiayias, A.; Konstantinou, I.; Russell, A.; David, B. & Oliynykov, R. (2016), 'A Provably Secure Proof-of-Stake Blockchain Protocol.', *IACR Cryptology ePrint Archive* **2016** , 889. https://eprint.iacr.org/2016/889.pdf

[7] Garay, J. A.; Kiayias, A. & Leonardos, N. (2014), 'The Bitcoin Backbone Protocol: Analysis and Applications.', *IACR Cryptology ePrint Archive* **2014** , 765. https://eprint.iacr.org/2014/765.pdf

[8] digiconomist.net, *Bitcoin Energy Consumption Index*. https://digiconomist.net/bitcoin-energy-consumption

[9] github.com, (2014) *A Next-Generation Smart Contract and Decentralized Application Platform*. https://github.com/ethereum/wiki/wiki/White-Paper

[10] github.com, (2016) *Proof of Stake FAQ*. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[11] blockchain.info, *Transaction Rate*. https://blockchain.info/sv/charts/transactions-per-second

[12] Poon, J.; Dryja, T. (2016), The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf

[13] Cormack, M. (2001). Peer-to-Peer vs. Client-Server. w3.org, https://www.w3.org/2001/04/w3c-p2p/comparison

[14] Szabo, N. (1997). The Idea of Smart Contracts. fon.hum.uva.nl. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html

[15] readthedocs.io, (2017) *Solidity*. https://solidity.readthedocs.io/en/latest/

[16] Gamma et al. (1995). Design patterns: elements of reusable object-oriented software. Reading, Mass, Addison-Wesley.

[17] cryptokitties.co, CryptoKitties. https://www.cryptokitties.co

[18] gnosis.pm, Gnosis. https://gnosis.pm/resources/default/pdf/gnosis-whitepaper-DEC2017.pdf