Bitcoin

Inledning

Bitcoin är en digital, kryptografisk valuta som utvecklades under år 2008 och introducerades 3 januari 2009 av pseudonymen Satoshi Nakamoto. Ingen vet med säkerhet vem eller vilka som ligger bakom Bitcoinprotokollet men spekulationer har gjorts och senast i mars 2014 trodde man sig ha hittat personen i fråga, men så var egentligen inte fallet. Alla bakomliggande implementationer samt programkod finns fritt tillgänglig för allmänheten via GitHub(_ref_), där små uppdateringar till protokollet sker om en överväldigande majoritet anser att förändringen är till fördel för Bitcoins framtid.

https://github.com/bitcoin/

TODO:

Referat

I den här avhandlingen kommer jag att gå in på Bitcoins tekniska bakgrund i detalj. Först kommer lite information vad Bitcoin är och varför det är intressant. Sedan följer bakgrunder till kryptografi som har utvecklats under 70-, 80- och 90-talet, så som assymetrisk kryptering, elliptisk kurvkryptering och hashning. Andra delen av avhandlingen går in på Bitcoins struktur i detalj, bl.a. transaktioner, adresser och block. Jag kommer att illustrera och förklara hur transaktioner mellan plånböcker sker, hur adresser skapas från kryptografiska nycklar samt hur block blir till. Sen kommer vi in på blockkedjan och mining, som utgör en viktig grundpelare till säkerheten, anonymiteten och integriteten hos Bitcoin. Tredje delen i avhandlingen kommer jag att gå in på skillnader till andra elektroniska plånböcker, t.ex. PayPal. Där ingår korta beskrivningar hur betalningar sker och vilka protokoll som sköter om säkerhet och överföringar. Till sist ges en översikt över hur andra betalmedel kan garantera bl.a. säkerhet, och om överhuvudtaget Bitcoin kan matcha denna garanti.

1. Kryptografi

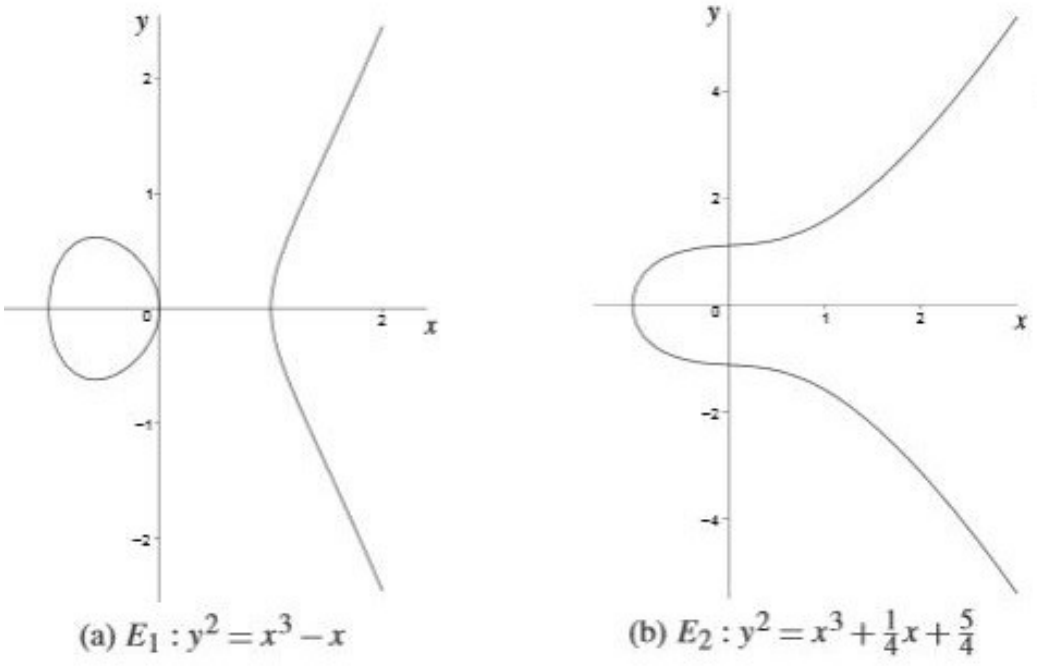
Med kryptering avses att ett meddelande sänt mellan två parter ska kunna göras obegripligt så att en tredje part inte ska kunna läsa meddelandet utan en kryptonyckel. Kryptonyckeln består av en förutbestämd algoritm, som krypterar meddelandet eller vice versa. Det finns två huvudkategorier vad gäller kryptering av elektroniska meddelanden:

- Symmetrisk kryptering, där samma kryptonyckel används för både kryptering och dekryptering.
- Assymetrisk kryptering, där två olika nycklar används för att utföra operationerna, publik nyckel (*public key*) och privat nyckel (*private key*).

Assymetrisk kryptering blev först beskrivet av Diffie, Hellman och Merkle år 1975 och senare utvecklat av Rivest, Shamir och Adleman år 1978 till vad som idag kallas RSA-kryptering. Benämningen RSA härstammar från utvecklarnas efternamn. Assymetrin i RSA-krypteringen baserar sig på ett problem, faktoriseringsproblemet, där man multiplicerar ihop primtalsfaktorer för att få ett stort tal. Detta kan liknas med att kryptera ett meddelande och dekrypteringen sker genom att faktorisera talet till sina primtalsfaktorer. Datorer i dagens läge kan lätt multiplicera tal med hundra tusentals siffror, men att faktorisera ett hundrasiffrigt tal kan visa sig vara betydligt svårare. En funktion som är av den här typen, d.v.s. lätt en väg och svårt andra vägen, kallas för en falluck-funktion. Det är ändå inte omöjligt för en tredje part att dekryptera meddelandet. Algoritmer vidareutvecklas hela tiden för att tackla sådana här problem och i takt med att hastigheten på datorberäkningar ökar krävs större RSA-nycklar för att inte lätt bli dekrypterade. RSA-kryptering är inte ideal för Bitcoin så därför har en effektivare krypteringsalgoritm som ändå bygger på samma assymetriska princip valts.

1.1 ECDSA

Elliptic Curve Digital Signature Alhorithm (ECDSA) är en variant av assymetrisk kryptering baserat på Elliptic Curve Cryptography (ECC) som föreslogs år 1985 av Koblitz och Miller. Det var inte förrän sent 90-tal algoritmen började användas globalt, och används numera av olika företag som behöver högeffektiva krypteringar för olika ändamål, t.ex. kreditkortsföretag. The Standards for Efficient Cryptography Group (SECG) skapades år 1998 och har nu medlemmar som t.ex. VISA, IBM, HP, Microsoft, Sun och NIST (National Institute of Standards and Technology). I September, 2000 släpptes ett dokument[ref] av SECG, innehållandes rekommenderade parametrar för algoritmer baserade på ECC. Detta dokument garanterar olika säkerhetsnivåer så att algoritmerna uppfyller vissa säkerhetskrav. Bitcoin använder sig av denna algoritm vid bl.a. generering av adresser.



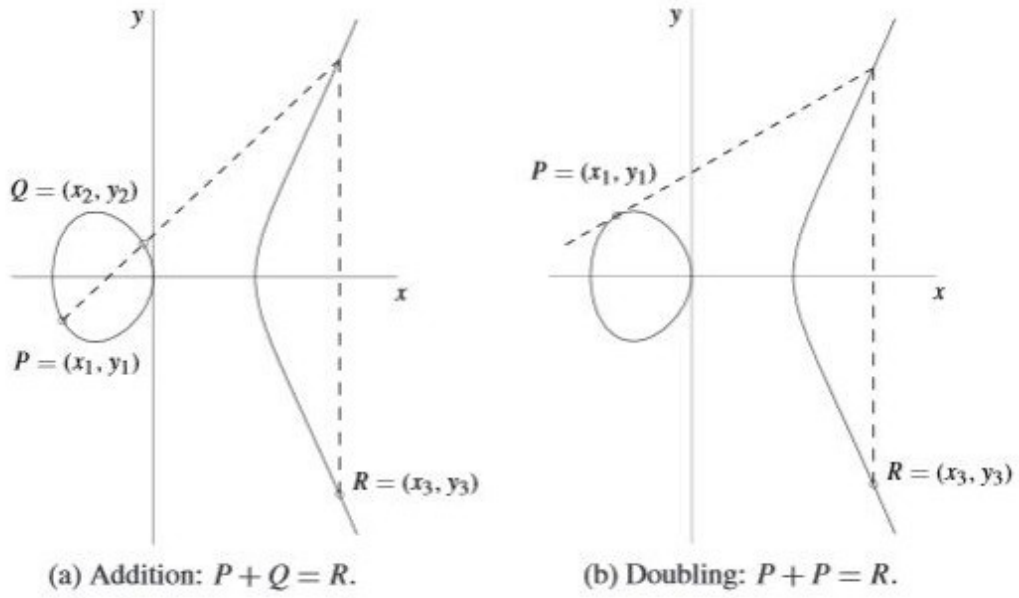
Figur 1: Exempel på elliptiska kurvor.

Algoritmen grundar sig på användningen av en elliptisk kurva, figur 1, som innehar vissa egenskaper. Bland annat ger kurvan upphov till ett diskret logaritmiskt problem som är lätt att lösa förutsatt man har tillgång till all information, men otroligt svår när man inte känner till en variabel, privata nyckeln. En annan egenskap är symmetrin vid x-axeln, d.v.s. varje punkt ovanför x-axeln har en motsvarande punkt under x-axeln som utnyttjas vid olika operationer. En elliptisk kurva ställs upp i ett tvådimensionellt fält där punkter

med koordinaterna (x, y) och parametrarna a, b satisfierar ekvationen

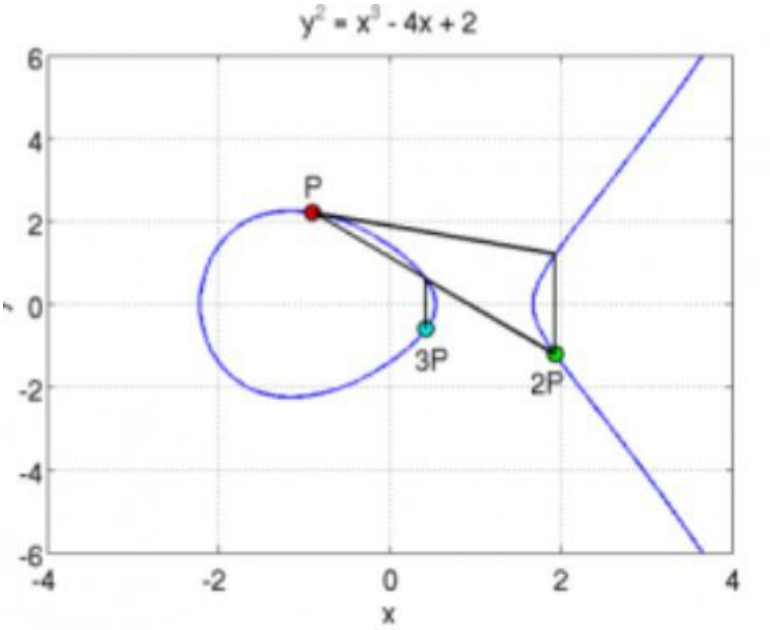
$$y^2 = x^3 + ax + b.$$

Kompositionen av punkter på kurvan följer en metod som kallas "tangent-och-korda"-metoden, där målet är att hitta en punkt R genom att räkna ut ekvationen $P + Q = R.$ För varje punkt P och Q på kurvan kan man dra en linje mellan dem, så att linjen skär kurvan i en tredje punkt. R fås sedan genom att dra en linje vertikalt mot dess motsvarande punkt på kurvan enligt figur 2, detta kallas EC-point addition. Ifall punkterna väljs så att $P = Q$ fås istället R genom att ta tangenten till punkten, $P + P = 2\mathrm{P} = R,$ som kallas EC-point dubbling. Ett undantag är ifall y-koordinaten råkar vara noll, då kommer tangenten att vara vertikal och kan således inte skära någon annan punkt på kurvan.



Figur 2: a) EC-point additon, b) EC-point dubbling

Det finns en tredje operation som är central för att förstå hur man utgående från den privata nyckeln kan räkna ut publika nyckeln. Operationen kallas EC-point multiplikation och går ut på att addera baspunkten P till sig själv k gånger, enligt formeln $kP = R.$ Om man till exempel vill räkna ut $3\mathrm{P} = R$ fås det genom att först konstatera att 3P är samma som $2\mathrm{P} + P.$ Eftersom vi vet 2P blir formeln $(P + P) + P = R,$ man tar alltså resultatet från föregående addition och adderar det till baspunkten för att få R, se figur 3.



Figur 3: Illustration hur 3P räknas ut med hjälp av EC-point multiplikation.

Det är EC-point multiplikationen som ger styrkan i hela ECDSA-algoritmen och är grunden till säkerheten som garanteras. Enligt formeln $kP = R$, fastän en tredje part vet utgångspunkten P och slutpunkten R, är det omöjligt att fastställa hur personen med privata nyckeln, k, har kommit fram till resultatet. Tredje parten kan alltså inte på annat sätt än att göra s.k. ”brute-force”-attacker veta hur många gånger man har gjort EC-point multiplikationen. Dessa attacker görs helt enkelt genom att prova olika k tills slutpunkten R nås. Med tillräckligt höga värden på olika kurvparametrar kan dessa attacker göras obefintliga, eftersom det i dagens läge skulle ta flera miljoner år av beräkningar innan algoritmen skulle knäckas.

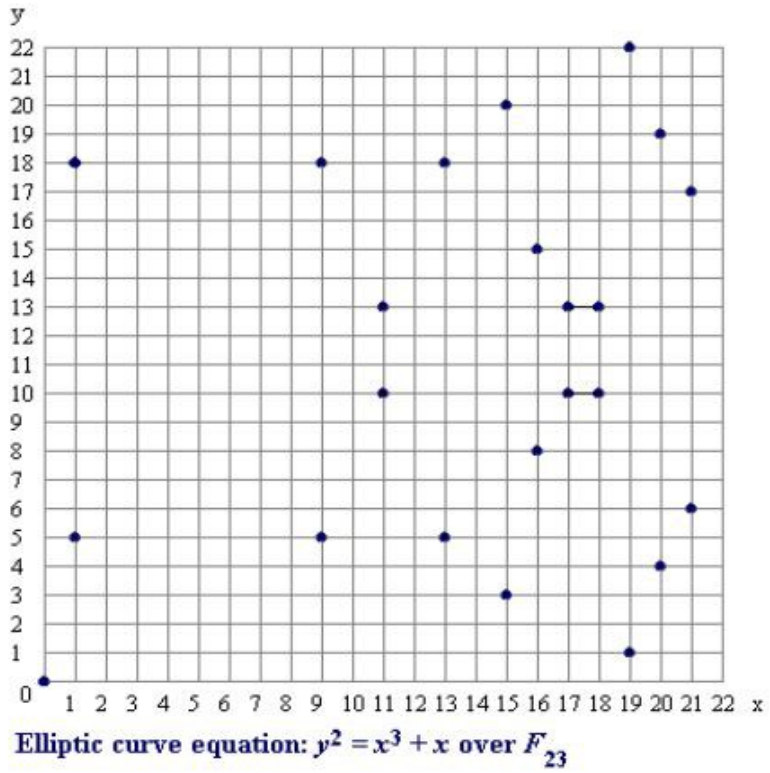
Med hjälp av dessa grundpelare kan ett kryptografiskt nyckelpar skapas enligt följande:

1. Alla som kommer att använda sig av krypteringen kommer överens om vilken kurva som ska användas, *CURVE.*
2. Bestäm gemensamt en baspunkt, G.
3. Välj i hemlighet ett godtyckligt tal som kommer att fungera som privat nyckel, k.
4. Räkna ut publik nyckel, Q, genom att utföra EC-point multiplikationen $Q = kG$.

En elliptisk kurva kan vara väldigt oprecis och ge fel svar när uträkningar sker, främst för att punkter nästan aldrig kan ges exakta, utan kräver avrundningar. För att råda bot på detta införs ett ändligt fält F_p, där p är ett primtal som avgränsar kurvan via modulo-operationen. På varje beräkning som görs tas *modulo* p som medför att varje punkt kommer att ligga inom intervallet $[0,\ p-1]$. Elliptiska kurvan bildas genom att välja parametrarna a och b i detta intervall och sedan hitta punkter som satisfierar ekvationen

$$y^2 \bmod p \ = \ (x^3+ax+b) \bmod p$$

Den grafiska representationen är inte längre en kurva, utan ser mera ut som slumpmässiga punkter placerade över ett rutnät, se figur 4. Symmetrin finns ändå kvar och EC-point-operationerna går ännu att rita med linjer. För att simulera *modulo* p säger man att linjen sveps runt fältet, den fortsätter alltså på motsatt sida när den når en kant. I praktiken ritas inte linjer ut utan datorer använder sig istället av avancerade algoritmer för beräkningarna. Dessa algoritmer gås inte in på här, utan kan studeras närmare i t.ex. [ref to 'Guide to ECC'].



Figur 4: Elliptisk kurva över ändligt fält

1.2 Hashing

Hashfunktioner tar inputdata av valfri längd och ger som resultat ett hashvärde av förutbestämd längd med hjälp av en komplex algoritm. Hashningen är en process som bara går att göra en väg, man kan alltså inte utgående från hashvärdet bestämma ursprungsdata. Man kan inte heller se några mönster i olika hashvärden, en liten förändring i inputdata ger ett helt annat hashvärde. Ytterligare är det väldigt osannolikt att två olika inputdata ger samma hashvärde, p.g.a. hur algoritmen är uppbyggd.

1.2.1 SHA-256

Secure Hash Algorithm (SHA) är en familj av hashfunktioner utvecklade av NSA under 90-talet. Bitcoin använder sig av SHA-256 algoritmen för hashning av adresser, transaktioner och block. Hashvärden för "Bitcoin" respektive "bitcoin" finns nedan i tabell 1. Notera hur stor skillnaden är mellan hashvärdena fastän inputdata ändrades minimalt.

Inputdata	**Resulterande SHA-256 hashvärde i hexadecimal form**
Bitcoin	b4056df6691f8dc72e56302ddad345d65fead3ead9299609a826e2344eb63aa4
bitcoin	6b88c087247aa2f07ee1c5956b8e1a9f4c7f892a70e324f1bb3d161e05ca107b

Tabell 1: Inputdata och dess hashvärde.

SHA-256 är en variant av SHA-2-standarden som använder sig av Merkle-Damgård konstruktion, först beskrivet år 1979. Som namnet anger fås ur SHA-256 ett hashvärde på 256 bitar. Teoretiskt sett kan inputdata inte vara större än $2^{64}-1$ bitar, men eftersom detta motsvarar ca 4 miljoner petabyte lär inte detta bli ett problem. Algoritmen delar upp inputdata i block med fixerad storlek, i det här fallet 512 bits, och lägger till padding samt kontrollbitar i slutet. Blocken körs en efter en genom logiska funktioner som manipulerar bitar, bl.a. med operationerna AND, OR, XOR, SHR och ROT. När ett block har passerat funktionen används resultatet i nästa blocks funktion, på så sätt länkas blocken samman. Hashvärdet fås av sista blockets resultat, och eftersom alla operationer är entydiga kommer hashvärdet alltid vara identiskt för samma inputdata. [figur?]

1.2.2 RIPEMD-160

RIPEMD-160 uppkom 1996 som respons mot SHA, eftersom flera ville ha alternativ till algoritmerna som regeringen hade skapat. Bitcoin utnyttjar denna hashmetod samt SHA-256 i samband med skapandet av öppna nyckeln. RIPEMD-160 ger ett kortare hashvärde och i kombination med SHA gör detta att adresserna blir lättare att hantera utan att påverka säkerheten. En annan teori är att NSA skulle ha lyckats skapa en bakdörr i SHA-algoritmerna, men detta anses högst osannolikt. På samma sätt som SHA utnyttjar RIPEMD-160 olika logiska operationer för att skapa ett hashvärde med 160 bitar.

1.2.3 Base58Check

Base58Check är ett schema för översättning från binär representation till ett ASCII strängformat, ett slags talsystem med bas 58. Det kan efterliknas Base64 som används bl.a. i vissa applikationer av e-post, XML och HTML, fast Bitcoins Base58Check innehåller flera inbakade metoder för error- och versionskontroll. Utöver detta fungerar det som vilket talsystem som helst, t.ex. går binära från 0 till 1 och hexadecimala från 0 till F. Base58Check går istället från 1 till z, där 1-9, A-Z och a-z ingår, se figur []. Notera att 0, O, I och l inte finns med, detta för att undvika problem med vissa teckensnitt, där dessa tecken kanske misstas för varandra. Omvandlingen fungerar naturligtvis båda vägarna, så en sträng i Base58Check-format kan lätt omvandlas tillbaka till sin binära eller hexadecimala representation. Bitcoinadresser, som används för att ta emot bitcoins, är i Base58Check-format, t.ex. 1CKtBR2xRUhThfoJFX2TQXQxQtMayiQLLX.

Dec	Base58	Dec	Base58	Dec	Base58	Dec	Base58	Dec	Base58	Dec	Base58
0	1	10	B	20	M	30	X	40	h	50	s
1	2	11	C	21	N	31	Y	41	i	51	t
2	3	12	D	22	P	32	Z	42	j	52	u
3	4	13	E	23	Q	33	a	43	k	53	v
4	5	14	F	24	R	34	b	44	m	54	w
5	6	15	G	25	S	35	c	45	n	55	x
6	7	16	H	26	T	36	d	46	o	56	y
7	8	17	J	27	U	37	e	47	p	57	z
8	9	18	K	28	V	38	f	48	q		
9	A	19	L	29	W	39	g	49	r		

Tabell 2: Översättningstabell mellan decimala talsystemet och Base58Check.

2. Bitcointeknik

2.1 Översikt

Bitcoin är en decentraliserad digital valuta som använder sig av ett underliggande P2P-nätverk (*eng.* p*eer-to-peer network*) för att förmedla information. Det här innebär att alla noder inom nätverket agerar som både server och klient, i motsats till den traditionella klient-server modellen där klienten skickar förfrågningar till en server. En av huvuduppgifterna som nätverket har är att upprätthålla en gemensam lista över transaktioner, som noder kan enas om utan att förlita sig på någon central entitet. När en nod tar emot en ny transaktion har noden som uppgift att verifiera dess autenticitet, och om den anses vara giltig skickar noden vidare transaktionen till sina grannar. Detta görs vid varje nod, så till sist vet hela nätverket om transaktionen. Så gott som var tionde minut sammanställs ett *block*, en lista på transaktioner som ännu inte finns i något block. Blocket länkas därefter ihop med föregående block, så att det bildar en kedja med resterande block, *blockkedjan*.

P2P-nätverket ger noder möjligheten att skicka olika meddelanden mellan varandra, främst information om enskilda transaktioner, block och blockkedjan. Noder kopplar upp sig till nätverket genom att starta en bitcoinklient och sedan länka ihop sig med grannar, d.v.s. närliggande noder. Normalt är en nod ihopkopplad med allt mellan 10 och 100 andra noder, man kan själv specificera max antalet uppkopplingar. Länkningen görs genom att noder skickar versionsmeddelanden mellan varandra för att se om deras klienter matchar och ifall de tar emot fler anslutningar. När ett par anslutningar har fastställts kan noden börja ta emot meddelanden och förutsatt att man har tillgång till blockkedjan, även verifiera nya transaktioner.

2.2 Plånböcker

Möjligheten att sända och ta emot bitcoin förutsätter att man har någon slags elektronisk plånbok. Den vanligaste i användning är BitcoinCore, som är en vidareutveckling på Satoshi Nakamoto's ursprungliga plånbok. Plånböcker i allmänhet måste på något sätt ha tillgång till blockkedjan, och BitcoinCore samt andra PC-baserade plånböcker gör detta genom att helt enkelt ladda ner blockkedjan. Plånböcker där man har en egen kopia av blockkedjan kallas för äkta plånböcker, eftersom dessa kan bidra till P2P-nätverket genom att vara en äkta

nod. När man för första gången ansluter till nätverket skickar man meddelanden till andra noder att man vill ladda ner blockkedjan. Eftersom blockkedjan byggs på var tionde minut måste man hela tiden uppdatera sin kopia så man har den senaste versionen. Just nu, efter snart fem år, har blockkedjan växt till en storlek på 16 GB och fortsätter öka varje dag.

Bitcoins framtidsvisioner är att vara den första världsvalutan, man ska alltså ha möjlighet att använda bitcoin till vad som helst, men framförallt var som helst. Plånböcker har därför utvecklats som applikationer till mobiltelefoner där blockkedjan inte behöver laddas ner. Mobiltelefoner har vanligtvis inte så stor förvaringskapacitet, detta har lösts genom att applikationen kopplas till en central kopia av blockkedjan via internet. Det här medför förstås att viss pålitlighet måste etableras mellan dig och organisationen som handhåller blockkedjan. Förstås har vissa mobiltelefoner eller smarttelefoner möjlighet att lagra hela blockkedjan, och har då fria händer att göra det om de vill.

Det finns en tredje grupp av plånböcker, nämligen webb-plånböcker. Kännetecknande för dessa är att plånboken är tillgänglig via en webbplats, ungefär som internetbanker. Ännu mer försiktighet måste iakttas, eftersom webbsidor på ett eller annat sätt kan utsättas för attacker. Det rekommenderas därför att man inte använder webb-plånböcker som en plats för långvarig förvaring, utan istället förflyttar sina bitcoins till en lämpligare plånbok. Börser som används för växling av fiatpengar mot bitcoins kan ses som en slags webb-plånbok, eftersom ens bitcoins bara finns på deras webbsida tills man flyttar dem till en annan plånbok.

Plånböcker är i grund och botten en lista på bitcoinadresser som används för att ta emot betalningar, medan privata nycklarna motsvarar ett lösenord för att auktorisera betalningar. Därför är det extremt viktigt att hålla privata nycklar hemliga, om de hamnar i fel händer kan en tredje part få tillgång till adressens bitcoinbalans. Varje plånbok kan i praktiken ha ett oändligt antal adresser, t.ex. kan man generera en ny adress för varje transaktion. Som standard brukar därför plånböcker generera ett färdigt antal adresser som lagras i plånbokens *wallet.dat*-fil. För backup behöver man därför bara kopiera denna fil, eftersom den också innehåller adressernas privata nycklar.

2.3 Adresser och nycklar

Bitcoin använder sig av en 256-bits Koblitz kurva, beskriven i SECG's dokument som *secp256k1*. Kurvan E ges enligt ekvationen

$$E: y^2 = x^3 + 7,$$

där fältet F_p har värdet $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1.$ Eftersom privat nyckeln väljs inom intervallet $[0, p-1]$ finns det ungefär $1{,}158\,x\,10^{77}$ olika nycklar. Som jämförelse har det beräknats att i synliga universum finns mellan $10^{78} - 10^{80}$ atomer så chansen att någon skulle generera samma nyckel är oändligt liten, förutsatt att man själv har valt nyckeln slumpmässigt. Självklart är nyckeln mera utsatt säkerhetsmässigt om man inte låter en slumptalsgenerator generera den, utan istället väljer t.ex. nyckeln med värdet 5. Varje privat nyckel har en korresponderande publik nyckel, och då automatiskt en motsvarande bitcoinadress. Privata nyckeln är oftast gömd för användaren och plånboken sköter om nyckelförvaringen. Normalt sätt är privata nyckeln ett stort tal angivet i hexadecimal form, t.ex. så här:

C569E438E8E7B91A5A13B30E33E43246F603BE23F4EBAFB00008BD150B7F26DA

Plånböcker har också möjligheten att importera privata nycklar, och då används samma tillvägagångssätt som vid skapande av nya nycklar, d.v.s. användning av ECDSA- och hashningsförfarandet beskrivet i kapitel 1.

2.3.1 Hur en bitcoinadress skapas

Nedan ges en kort beskrivning hur bitcoinadressen fås utgående från privata nyckeln. I representationen av adressen avgränsas olika delar med symbolen ” | ” och hela adressen med ” [ ] ”.

1. Ta en privat nyckel som väljs slumpmässigt inom intervallet för fältet.
2. Kör den genom ECDSA-algoritmen och skapa publika nyckeln utgående från koordinaterna i resultatet,
   [1 byte 0x04 | 32 byte X-koordinat | 32 byte Y-koordinat].
3. Hasha resultatet från 2. först med SHA256 och sedan med RIPEMD-160 samt lägg ett ID-nummer för nätverket i mest signifikanta position,
   [1 byte 0x00 | 20 byte ripemd160(sha256([resultat från 2.]))].

4. Skapa en checksumma för kontroll av error genom att hasha resultatet från 3. med SHA-256 två gånger. Fyra första byte från checksumman läggs i minst signifikant position i resultatet från 3.

   [21 byte resultat från 3. | 4 byte sha256(sha256([resultat från 3.]))].
5. Använd Base58Check för att koda om adressen till bitcoins standardformat, det här är slutgiltiga bitcoinadressen. En typisk adress skulle se ut på det här viset:

   1CKtBR2xRUhThfoJFX2TQXQxQtMayiQLLX.

2.4 Transaktioner

På en abstrakt nivå fungerar transaktioner genom att överföra bitcoins från ett antal ursprungsadresser (*input*) till en eller högst två destinationsadresser (*output*). En transaktion kan alltså ha flera olika input, det här har att göra med hur bitcoin implementerar äganderätten av mynt. Värt att notera är att bitcoins kan delas upp i mindre valörer, hela mynt behöver inte användas vid transaktioner. Minsta myntenheten kallas för *satoshi*, namngett efter grundaren, som motsvarar $0{,}000\ 000\ 01\ BTC$. Andra vanliga enheter är:

- millibitcoin, mBTC som motsvaras av $0{,}001\ BTC$,
- och microbitcoin, µBTC som motsvaras av $0{,}000\ 001\ BTC$.

2.4.1 Output

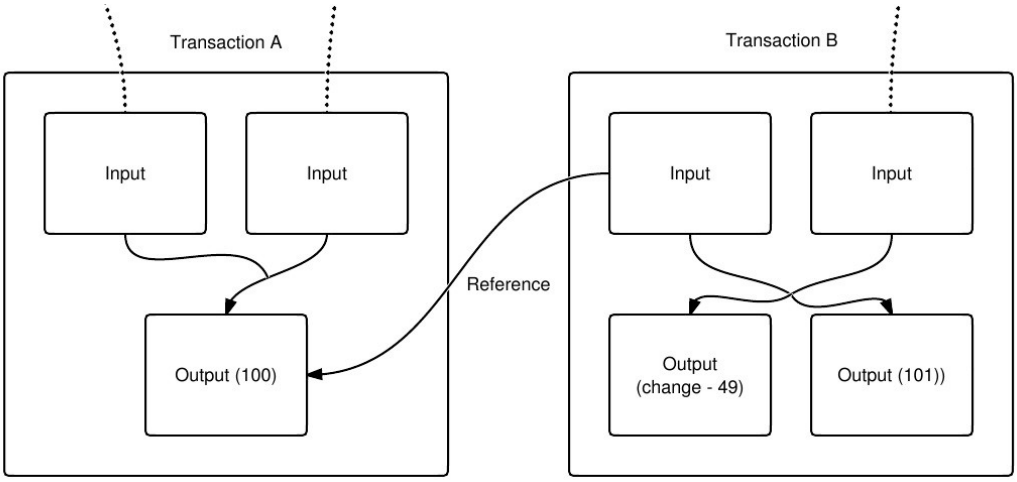
Varje transaktion bestämmer via ett script hur dess output blir spenderbar. Scriptet är implementerat i ett stackbaserat programmeringsspråk, närmast lik 1970-talets Forth. Det som kännetecknar Forth är att man kan använda det som högnivåspråk lika lätt som ett Assembler-liknande språk. Scriptet skapar en lista på instruktioner som anger hur nästa person kan överta myntens äganderättighet. Scriptet för en typisk transaktion omfattar att mottagaren måste bevisa två saker:

1. Att hashvärdet för ens publik nyckel motsvarar destinationsadressen inbakad i transaktionen.
2. Genom att skapa en signatur visa att man har tillgång till publik nyckelns korresponderande privat nyckel.

Med hjälp av scriptet kan avancerade användare definiera vilka parametrar som måste uppfyllas för tillgång till mynten i transaktionen. Till exempel kan man ge krav på att två privata nycklar måste anges, att ingen privat nyckel alls behövs

eller att transaktionen är låst tills en viss tidpunkt. De här modifierade scripten bildar kryptografiskt-bindande s.k. *kontrakt* som kan användas i framtiden för olika lån- och förmedlingstjänster samt byteshandel av elektronisk mjukvara.

2.4.2 Input



Ett sätt för att garantera att den som försöker spendera bitcoins faktiskt äger dessa har implementerats genom att länka ihop transaktionerna som en kedja. I varje transaktions input anges en referens till den transaktion som mynten senast spenderades i, specifikt till den transaktionens output [fig]. Det här betyder att man i varje transaktion kan följa referensen bakåt till föregående transaktion och verifiera att mynten verkligen just nu ägs av personen som försöker spendera dem. I praktiken verkställs det här genom att man i transaktionens input undertecknar föregående transaktions output med sin privat nyckel, och i kombination med publik nyckeln kan transaktionen därför verifieras.

Varje output kan bara spenderas en gång genom att referera den i en annan transaktions input. Summan av alla input måste vara större eller lika med output. Eftersom input måste refereras helt och hållet betyder det här att summan av alla input oftast är större än output, och därför anges en returadress för växel i output. Ifall ingen returadress anges tolkas överflödigt input som en transaktionsavgift.

Medan en output inte är spenderad flaggas den som en UTXO (Unspent Transaction Output) och kommer att vara synlig i plånböcker som innehar äganderättigheten. Det här betyder att en plånbok i själva verket är en lista på

icke-spenderade transaktioner man äger, och har rätt att spendera med tillhörande privat nyckel.

2.4.3 Coinbase transaktioner

Det finns en speciell slags transaktion som sker ungefär var tionde minut, en *coinbase transaktion*. Den utges av nätverket automatiskt i samband med att ett nytt block har fästs vid blockkedjan. Noden som lyckats hitta blocket lägger sin bitcoinadress som output i transaktionen och får äganderätten till mynten. Coinbase transaktioner är sättet som nya mynt kommer i omlopp och fungerar delvist som motivation för noder att skapa block. Transaktionen får inte något input eftersom det inte går att referera mynten från en annan transaktion. Enligt design ger coinbase transaktioner ut 50 BTC per block, som halveras efter 210.000 block. Med ett block var tionde minut, motsvarar detta en halvering vart fjärde år. Första halveringen till 25 BTC skedde i december 2012, och nästa halvering sker år 2016. Enligt den här principen kommer det sista mynten i cirkulation år 2140 och då är totala antalet mynt i omlopp 21 miljoner BTC. Just nu i april 2014 har drygt 12,5 miljoner BTC getts ut av nätverket, knappt 60% av totala antalet mynt som kommer att skapas.

När ett transaktion har skapats sänds den ut på P2P-nätverket och en tidsstämpel läggs till som anger när transaktionen har skett. Närliggande noder tar emot transaktionen, verifierar den och ifall den anses vara giltig lagras den temporärt i nodens minne. Genom att kolla på sändarens publik nyckel kan transaktionens autenticitet lätt kontrolleras, man vet alltså att det faktiskt är sändaren som har undertecknat transaktionen med sin privata nyckel.

https://en.bitcoin.it/wiki/Contracts
https://en.bitcoin.it/wiki/Script
https://en.bitcoin.it/wiki/Transactions

2.5 Block och mining

Eftersom varje nod lagrar sin egen lista på transaktioner i dess minne kan småningom varje nod ha små oundvikliga skillnader i sina listor. Till exempel kan noden ta emot en transaktion som kräver input från en annan transaktion som ännu inte hunnit fram till noden, då kan inte noden acceptera transaktionen. En

annan situation som kan uppstå är att två eller flera transaktioner försöker spendera samma mynt flera gånger, det här kallas för *double spending*-problemet. Notera att varje nod ändå sparar ogiltiga transaktioner så att ifall de mottas flera gånger inte behöver kollas varje gång.

Ett system för att upprätthålla en gemensam transaktionshistorik behövs därför och det är implementerat i form av block. Ett block är en lista på verifierade transaktioner som alla noder i nätverket kan vara överens om. Skapandet av block och framförallt länkningen till blockkedjan är den största innovationen i bitcoinprotokollet och orsaken till varför bitcoin har lyckats där så många andra har misslyckats. Systemet är baserat på Hashcash[ref.], ett förslag på hur man kunde motverka skräppost genom att med varje e-mail även kräva ett s.k. *proof-of-work* (sv. bevis-av-arbete).

Bitcoins proof-of-work är ett sätt att bevisa att ett block har skapats genom att tillräckligt hög beräkningskraft har använts. Att göra beräkningar för att försöka åstadkomma ett proof-of-work kallas för *mining* (sv. gruvdrift) och noder i nätverket som tävlar om att göra det snabbast kallas för *miners* (sv. gruvarbetare). Miners samlar upp lösa transaktioner i nätverket som inte ännu tillhör något block och sparar dem i ett eget temporärt block. Därefter försöker varje miner självständigt hasha blocket så att hashvärdet är lägre än ett specifikt mål, *target.*

TODO: mera om target, nonce, blockchain, difficulty, mining, merkle tree, PoW

[] http://www.hashcash.org/papers/hashcash.pdf
https://en.bitcoin.it/wiki/Proof_of_work

3. TODO

Ungefärlig litteraturlista

TODO: Formatting

[] IEEE - Protocols for Public Key Cryptosystems, Merkle, 1980

[] Hankerson, Menezes, Vanstone: Guide to Elliptic Curve Cryptography (Springer, 2004)

[] Standards for Efficient Cryptography (SEC), Certicom Research, 2000

[] CRYPTO '85 - Use of Elliptic Curves in Cryptography, Miller, 1985

[] Mathematics of Computation - Elliptic Curve Cryptosystems, Koblitz, 1985

[] Bitcoin: A Peer-to-Peer Electronic Cash System, Nakamoto, 2008

[] IEEE - Bitcoin and The Age of Bespoke Silicon, Taylor, 2013

[] ACM - Double-Spending Fast Payments in Bitcoin, Karame, Androulaki, Capkun, 2012

[] IEEE - Information Propagation in the Bitcoin Network, Decker, Wattenhofer, 2013

[] ACM - On Bitcoin and Red Balloons, Babaioff, Dobzinski, Oren, Zohar, 2012

[] https://en.bitcoin.it/wiki/Category:Technical

[] ACM - A Fistful of Bitcoins: Characterizing Payments Among Men with No Names, Meiklejohn, Pomarole, Jordan, Levchenko, McCoy, Voelker, Savage, 2013

[] IEEE - Have a Snack, Pay with Bitcoins, Bamert, Decker, Elsen, Wattenhofer, Welten, 2013

[] IEEE - Perforamce Comparison of Executing Fast Transactions in Bitcoin Network Using Verifiable Code Execution, Singh, Surathkal, Arora, Chandavarkar, Agrawal, 2013

[] Financial Cryptography and Data Security, Sadeghi, (Springer, 2013)

[] CRYPTO '88 - Untraceable Electronic Cash, Chaum, Fiat, Naor, 1988